

LA BIBLIA



Ian Gilfillan

MySQL

INCLUYE
CD-ROM



ANAYA
MULTIMEDIA

Índice

Introducción	33
Razones para comprar este libro.....	34
¿Qué queda fuera del alcance de este libro?	35
¿Qué necesita?	35
¿Cómo utilizar este libro?	36
Parte I. Uso de MySQL	37
1. Guía rápida de MySQL	39
Comprensión de los fundamentos de MySQL	40
¿Qué es una base de datos?	41
Terminología sobre bases de datos	41
Conexión a una base de datos	42
Creación y uso de nuestra primera base de datos	45
Caso en el que acabemos de instalar MySQL	45
Si un administrador necesita concederle permiso	46
Uso de la base de datos	46
Creación de una tabla	48
Listado de las tablas de una base de datos con SHOW TABLES	49
Análisis de la estructura de las tablas con DESCRIBE	50
Inserción de nuevos registros en una tabla	50
Inserción de datos dentro de una instrucción INSERT	52
Inserción de grandes cantidades de datos desde un archivo de texto con LOAD DATA	52

Recuperación de información de una tabla	53
Orden en el que MySQL procesa las condiciones	54
Correspondencia de patrones	56
Ordenacion	57
Limitación del numero de resultados	58
Devolución del valor maximo con MAX()	60
Recuperacion de registros distintos	61
Como contar	62
Como recuperar la media, el minimo y el total con AVG() , MIN() y SUM()	62
Realización de calculos en una consulta	63
Eliminacion de registros	64
Como cambiar los registros de una tabla	64
Eliminación de tablas y bases de datos	65
Como modificar la estructura de la tabla	66
Como agregar una columna	66
Modificación de una definición de columna	66
Como cambiar el nombre de una columna	67
Como eliminar una columna	68
Uso de las funciones de fecha	68
Como especificar el formato de fecha	69
Recuperación de la fecha y la hora actual	70
Creación de consultas mas avanzadas	71
Como aplicar un nuevo encabezado a una columna con AS	71
Combinación de columnas con CONCAT	71
Como buscar el dia del año	72
Como trabajar con varias tablas	72
Combinacion de dos o mas tablas	74
Realización de calculos con fechas	77
Agrupacion de una consulta	79
Resumen	81
2. Tipos de datos y tipos de tabla	85
Análisis de los distintos tipos de columna	86
Tipos de columna numericos	86
Tipos de columnas de cadena.....	91
Tipos de columna de fecha y hora	97
Opciones de MySQL	98
Análisis de los distintos tipos de tablas	104
Tablas ISAM	104
Tablas MyISAM	105
Tablas estaticas	105
Tablas dinamicas	106
Tablas comprimidas	107

Tablas MERGE.....	109
Tablas HEAP	113
Tablas InnoDB	115
Tablas DBD	116
Resumen	117
3. SQL Avanzado	119
Operadores	120
Operadores logicos	120
Operadores aritmeticos	121
Operadores de comparacion.....	123
Como usar LIKE en equivalencias de patron de SQL	127
Expresiones regulares	128
Operadores bit a bit	133
Combinaciones avanzadas	138
Combinaciones internas	140
Combinaciones por la izquierda (o combinaciones externas por la izquierda)	140
Combinaciones por la derecha (o combinaciones externas por la derecha)	143
Combinaciones externas completas	143
Combinaciones naturales y la palabra clave USING	144
Recuperación de los datos encontrados en una tabla pero no en la otra	146
Combinación de resultados con UNION	147
Subselecciones	149
Como escribir subselecciones como combinaciones	149
Como agregar registros a una tabla desde otras tablas con INSERT SELECT ...	151
Mas sobre la agregacion de registros	152
Mas sobre cómo eliminar registros (DELETE y TRUNCATE).....	153
Variable de usuario	153
Ejecucion de instrucciones SQL almacenadas en archivos	157
Redireccionamiento de la salida hacia un archivo	159
Como usar los archivos desde la linea de comandos MySQL	160
Transacciones y bloqueos	160
Las transacciones en las tablas InnoDB	161
Lecturas coherentes.....	163
Lectura de bloqueos para actualizaciones.....	164
Bloqueos de lectura en modo compartido	167
Confirmaciones automaticas	168
Transacciones en tablas DBD.	171
Otros comportamientos transaccionales	173
Bloqueo de tablas	175
Como evitar los bloqueos de tabla	181
Niveles de transacción	181
Resumen	182

4. Índices y optimización de consultas.....	185
Comprension de los indices	186
Creación de un índice	187
Creación de una clave primaria	187
Creación de un índice primario	190
Creación de un índice de texto completo	191
Uso de los indices de texto completo	192
Palabras ruido	193
Relevancia	194
Busquedas booleanas de texto completo	195
Creación de un índice unico	201
Creación de indices sobre una parte de un campo	202
Como utilizar un campo de incremento automatico	202
Creación de un campo de incremento automatico	203
Inserción de registros que contienen un campo de incremento automatico	204
Como recuperar y reiniciar el valor de incremento automatico	205
Más allá de los límites	209
Problemas con LAST-INSERT-ID()	209
indices de varias columnas y campos de incremento automatico	211
Eliminación o modificación de un índice	213
Tipos de tabla e indices	214
Uso eficaz de los indices	215
Donde utilizar los indices	215
Selección de indices	218
Uso del sistema de prefijacion mas a la izquierda	218
Como utiliza MySQL los indices con EXPLAIN	220
Realización de cálculos en una consulta	225
Uso de EXPLAIN con el sistema de prefijos a la izquierda	227
Optimización de las selecciones	229
Como ayudar al optimizador de MySQL con ANALYZE	234
Optimización de las instrucciones SELECT y seguridad	236
Evaluación del rendimiento de las funciones	237
Optimización de actualizaciones, eliminaciones e inserciones	238
Resumen	240
5. Programación con MySQL	243
Uso de buenas tecnicas de programacion de bases de datos	244
Uso de conexiones permanentes	245
Como lograr codigo portable y sencillo de mantener	245
La conexión	246
Consultas de base de datos	249
¿Cuánto trabajo deberia realizar el servidor de la base de datos?	251
Las fases del desarrollo de aplicaciones	255

Fase 1: análisis de las necesidades	255
Determinación de las necesidades del usuario	256
Determinación de tecnología necesaria	256
Fase 2: Diseiio de la aplicacion	257
Modelado	257
Uso de pseudocodigo	257
Codificación	257
Fase 4: fase de pruebas e implementación	260
Resumen	260
6. Como ampliar las funciones de MySQL.....	263
Funciones definidas por el usuario	264
Funciones UDF estandar	268
La funcion init	268
La funcion principal	270
La funcion deinit	272
Creación de una UDF estandar de ejemplo	272
Análisis de las funciones agregadas	274
Creación de una UDF agregada de ejemplo	275
Resolución de problemas de UDF	278
Resumen	278
Parte II. Diseiio de una base de datos	281
7. Comprensión de las bases de datos relacionales	283
Análisis de los modelos anteriores a las base de datos	284
Modelo jerarquico de base de datos	284
Modelo de base de datos en red	286
Modelo de base de datos relacional	286
Terminos basicos	287
Claves de tabla	290
Claves externas	290
Introducción a las vistas	292
Resumen	295
8. Normalización de bases de datos	297
Concepto de normalización	298
Primera forma normal	305
Segunda forma normal	306
Tercera forma normal	307
Forma normal de Boyce-Codd	309
Cuarta forma normal	312
Quinta forma normal y otras formas	314
Concepto de desnormalizacion	316
Resumen	318

9. Diseño de bases de datos	321
Ciclo de vida de las bases de datos	322
Fase 1: Análisis	323
Fase 2: Diseño	324
Diseño conceptual	325
Diseño lógico y físico	329
Fase 3: Implementación	333
Fase 4: Pruebas	333
Fase 5: Puesta en marcha	334
Fase 6: Mantenimiento	335
Un ejemplo del mundo real: un sistema de seguimiento de publicaciones	336
Fase 1 de la base de datos de Poet's Circle: Análisis	336
Fase 2 de la base de datos de Poet's Circle: Diseño	337
Fase 2 de la base de datos Poet's Circle: Implementación	341
Fase 4 a 6 de la base de datos Poet's Circle: Prueba, puesta en marcha y mantenimiento	342
Control de simultaneidad mediante transacciones	343
Atomicidad	343
Coherencia	343
Aislamiento	344
Durabilidad	344
Resumen	344
Parte III. Administracibn de MySQL	347
10. Administración básica	349
Uso de MySQL como administrador	350
Como iniciar y cerrar MySQL	352
Como iniciar y cerrar MySQL en Unix	352
Como iniciar MySQL automáticamente al arrancar el sistema	354
Como evitar problemas comunes al iniciar MySQL en Unix	355
Como iniciar y cerrar MySQL en Windows	355
Como iniciar MySQL automáticamente	356
Como evitar problemas comunes al iniciar MySQL en Windows	358
Configuración de MySQL	359
Registro	363
El archivo de errores	363
El registro de consultas	364
El registro de actualización binario	365
El registro de consultas lentas	368
Rotación de registros	369
Optimización, análisis, comprobacion y reparación de tablas	372
Optimización de tablas	373
Optimización de tablas con la instrucción OPTIMIZE	373

Optimización de tablas con mysqlcheck	374
Optimización de tablas con myisamchk	374
Análisis de tablas	375
Análisis de tablas con ANALYZE TABLE	376
Análisis de las tablas con mysqlcheck	377
Análisis de tablas con myisamchk	377
Comprobacion de tablas	378
Comprobacion de las tablas con CHECK TABLES	379
Comprobacion de tablas con mysqlcheck	380
Comprobacion de tablas con myisamchk	381
Reparación de tablas	383
Reparación tablas de tipo diferente a MyISAM	384
Reparación de tablas con REPAIR TABLE	384
Reparación de las tablas con mysqlcheck	386
Reparación de tablas con myisamchk	386
Como usar mysqlcheck	388
Uso de myisamchk	391
Resumen	396
11. Copias de seguridad de bases de datos	399
Volcados de seguridad de tablas MyISAM con BACKUP	400
Uso de BACKUP en Unix	400
Uso de BACKUP con Windows	403
Restauracion de tablas MyISAM con RESTORE	404
Volcados de seguridad de tablas de MyISAM mediante la copia de archivos directamente	406
Realización de volcados con mysqldump	409
Restauracion de una base de datos volcada con mysqldump	411
Copias de seguridad con SELECT INTO	416
Restauracion de una tabla con LOAD DATA	419
¿Qué ocurriría si algo sale mal?	420
Uso de LOAD DATA con opciones	421
Aspectos de seguridad relacionados con LOAD DATA LOCAL	426
Uso de mysqlimport en lugar de LOAD DATA	426
Uso de mysqlhotcopy para realizar copias de seguridad	429
Uso del registro de actualización binario para restablecer la base de datos a su posición mas reciente	431
Copia de seguridad y restauracion de tablas InnoDB	436
Duplicación como medio de realizar copias de seguridad	438
Resumen	439
12. Duplicación de base de datos	441
Que es la duplicación	441

Configuración de duplicación	443
Opciones de duplicación	444
Comandos de duplicación	448
Dificultades de la duplicación	450
Duplicación de una base de datos	450
Duplicación con un registro binario activo en el principal	457
Eliminación de registros binarios antiguos del servidor principal e inicio de la operación	459
Como evitar un exceso de actualizaciones	462
Como evitar errores clave	464
Resumen	467
13. Configuración y optimización de MySQL	471
Optimización de las variables mysql	472
Optirnizacion de table_cache	477
Optirnizacion de key-buffer-size	478
Control de un elevado numero de conexiones	479
Optirnizacion de las variables delayed_queue_size y back-log	484
Optirnizacion de la variable sort-buffer	484
Configuración de tablas InnoDB	485
Presentación de las opciones mysql	485
Descripción de las variables mysql	491
Análisis de todas las variables de estado	503
Cambio de valores de variables con el servidor en funcionamiento	508
Mejoras en el hardware para acelerar el servidor	511
Memoria	511
Discos	512
CPU	512
Uso de análisis comparativos	512
Ejecucion de MySQL en modo ANSI	527
Uso de distintos lenguajes en MySQL	528
Como mostrar mensajes de error en otro idioma	528
Utilización de un conjunto de caracteres diferente	529
Como añadir un conjunto de caracteres propio	529
Resumen	533
14. Seguridad de bases de datos	535
Seguridad al conectarse	536
Gestión de usuarios y permisos	537
La base de datos mysql	537
Campos de las tablas	538
Como examina MySQL permisos para conceder el acceso	543
Como completar las tablas de permiso	544

Que hacer si no puede conectarse o no tiene permisos	564
Que hacer si la tabla de usuarios se daña	565
Otras opciones de GRANT	567
Estrategia para gestionar usuarios de forma segura	570
Como evitar la concesion de privilegios peligrosos	571
Conexiones SSL	572
Seguridad de aplicaciones	574
Seguridad del sistema	575
Problemas de seguridad relacionados con LOAD DATA LOCAL	575
Resumen	576
15. Instalación de MySQL	579
Instalacion de una distribución fuente o binaria	580
Instalacion de MySQL en Windows	581
Instalacion de una distribución binaria en Windows	582
Instalacion de MySQL como servicio en Windows NT/2000/XP	583
Instalacion de MySQL en Unix	584
Instalacion de una distribución binaria (tar) en Unix	584
Instalacion de una distribución binaria (rpm) en Unix	587
Instalacion desde codigo fuente en Unix	588
Compilación optima de MySQL	590
Instalacion de varios servidores en el mismo equipo	591
Como evitar problemas de instalacion comunes	594
Problemas al iniciar msyqld	594
Problemas de compilacion	595
Problemas de Windows	596
Actualización de MySQL 3.x a MySQL 4	597
Resumen	599
16. Multiples unidades	601
Significado de RAID	601
RAID 0	602
RAID 1	603
RAID 2 y RAID 3	603
RAID 4	604
RAID 5	604
RAID 10	605
RAID 0+1	605
Otros tipos de RAID	606
Uso de enlaces simbolicos	606
Vinculacion simbolica de bases de datos	607
Vinculacion simbolica de tablas	609
Resumen	611

Apéndices	613
A. Guía de referencia de la sintaxis de MySQL	615
ALTER	616
ANALYZE TABLE	617
BACKUP TABLE	617
BEGIN	617
CHECK TABLE	617
COMMIT	618
CREATE	618
DELETE	621
DESC	621
DESCRIBE	621
DO	622
DROP	622
EXPLAIN	622
FLUSH	623
GRANT	623
INSERT	625
JOIN	626
KILL	626
LOAD DATA INFILE	626
LOCK TABLES	627
OPTIMIZE	628
RENAME	628
REPAIR TABLE	628
REPLACE	629
RESET	629
RESTORE TABLE	629
REVOKE	629
ROLLBACK	630
SELECT	630
SET	632
SET TRANSACTION	635
SHOW	635
TRUNCATE	635
UNION	636
UNLOCK TABLES	636
UPDATE	636
USE	636
B. Funciones y operadores de MySQL	639
Operadores lógicos	639

AND, &&	639
OR, 	640
NOT, !	640
Operadores aritmeticos	641
+	641
-	641
*	642
/	642
%	642
Operadores de comparacion	642
=	643
!=, <>	643
>	644
<	644
>=	645
<=	645
<=>	645
IS NULL	646
BETWEEN	646
LIKE	647
IN	647
REGEXP, RLIKE	647
Operadores de bits	650
&	650
.....	650
<<	651
>>	651
Funciones de fecha y hora	651
ADDDATE	652
CURDATE	652
CURRENT-DATE	652
CURRENT-TIME	653
CURRENT_TIMESTAMP	653
CURTIME	653
DATE_ADD	653
DATE_FORMAT	654
DATE_SUB	656
DAYNAME	656
DAYOFMONTH	656
DAYOFWEEK	657
DAYOFYEAR	657
EXTRACT	657
FROM_DAYS	658
FROM-UNIXTIME	658

HOUR	658
MINUTE	659
MONTH	659
MONTHNAME	659
NOW	660
PERIOD-ADD	660
PERIOD_DIFF	660
QUARTER	661
SEC_TO-TIME	661
SECOND	661
SUBDATE	662
SYSDATE	662
TIME_FORMAT	662
TIME-TO-SEC	662
TO_DAYS	662
UNIX_TIMESTAMP	663
WEEK	663
WEEKDAY	664
YEAR	664
YEARWEEK	665
Funciones de cadena	665
ASCII	665
BIN	666
BIT-LENGTH	666
CHAR	667
CHAR-LENGTH	667
CARÁCTER_LENGTH	667
CONCAT	667
CONCAT_WS	668
CONV	669
ELT	670
EXPORT-SET	670
FIELD	671
FIND_IN_SET	671
HEX	672
INSERT	673
INSTR	673
LCASE	674
LEFT	674
LENGTH	674
LOAD-FILE	675
LOCATE	675
LOWER	676

LPAD	676
LTRIM	677
MAKE_SET	677
OCT	677
OCTET-LENGTH.....	678
ORD	678
POSITION	679
QUOTE	679
REPEAT	679
REPLACE	680
REVERSE	680
RIGHT	680
RPAD	681
RTRIM	681
SOUNDEX	682
SPACE	682
SUBSTRING	682
SUBSTRING-INDEX	683
TRIM	684
UCASE	684
UPPER	684
Funciones numericas	685
ABS	685
ACOS	685
ASIN	686
ATAN	686
ATAN2	686
CEILING	686
COS	687
COT	687
DEGREES	687
EXP	688
FLOOR	688
FORMAT	689
GREATEST	689
LEAST	690
LN	690
LOG	690
LOG10	691
LOG2	691
MOD	691
PI	692
POW	692

POWER	692
RADIANS	692
RAND	693
ROUND	693
SIGN	694
SIN	694
SQRT.....	695
TAN.....	695
TRUNCATE	695
Funciones agregadas	696
AVG.....	696
BIT_AND.....	696
BIT_OR	697
COUNT.....	697
MAX	697
MIN	698
STD	698
STDDEV	698
SUM	698
Otras funciones	699
AES_DECRYPT	699
AES_ENCRYPT	699
BENCHMARK	699
CASE	699
CAST	701
CONNECTION-ID	702
CONVERT	702
DATABASE	702
DECODE.....	703
DES_DECRYPT.....	703
DES_ENCRYPT	703
ENCODE	704
ENCRYPT.....	704
FOUND_ROWS	704
GET-LOCK	705
IF	705
IFNULL	706
INET_ATON.....	707
INET_NTOA	707
IS_FREE-LOCK	707
LAST-INSERT-ID	708
MASTER-POS-WAIT	708
MD5	709

NULLIF	709
PASSWORD	709
RELEASE-LOCK	710
SESSION-USER	710
SHA	711
SHA1	711
SYSTEM-USER	711
USER	711
VERSION	711
C. API PHP	715
Opciones de configuración PHP	715
Funciones MySQL PHP	716
mysql_affected_rows	716
mysql_change_user	717
mysql_client_encoding	717
mysql_close	718
mysql_connect	718
mysql_create_db	719
mysql_data_seek	719
mysql_db_name	720
mysql_db_query	720
mysql_drop_db	721
mysql_errno	721
mysql_error	721
mysql_escape_string	722
mysql_fetch_array	723
mysql_fetch_assoc	723
mysql_fetch_field	724
mysql_fetch_lengths	725
mysql_fetch_object	726
mysql_fetch_row	726
mysql_field_flags	727
mysql_field_len	728
mysql_field_name	728
mysql_field_seek	729
mysql_field_table	729
mysql_field_type	730
mysql_free_result	730
mysql_get_client_info	731
mysql_get_host_info	731
mysql_get_proto_info	731
mysql_get_server_info	732

mysql_info.....	732
mysql_insert_id.....	732
mysql_list_dbs	733
mysql_list_fields	733
mysql_list_processes	734
mysql_list_tables	735
mysql_num_fields	735
mysql_num_rows.....	735
mysqlqconnect.....	736
mysqlqing	737
mysql_query	737
mysql_real_escape_string	738
mysql_result	738
mysql_select_db	739
mysql_stat	739
mysql_tablename	740
mysql_thread_id	740
mysql_unbuffered_query	741
D. DBI Perl.....	743
Metodos de la clase DBI	744
available-drivers	744
connect	744
connect_cached	745
data_sources	746
trace.....	746
Metodos DBI comunes a todos los identificadores	746
err	747
errstr	747
func	747
set_err	747
state	748
trace.....	748
trace_msg	748
Funciones de utilidad DBI	748
hash	748
looks_like_number	748
neat	749
neat_list	749
Metodos de identificadores de base de datos	749
begin_work	749
column_info.....	749
commit.....	749

disconnect	750
do	750
foreign_key_info	750
get-info	751
ping	751
prepare	751
prepare_cached	751
primary_key	752
primary_key_info	752
quote	752
quote_identifier	752
rollback	752
selectall_arrayref	752
selectall_hashref	753
selectcol_arrayref	753
selectrow_array	753
selectrow_arrayref	754
selectrow_hashref()	754
table_info	754
tables	754
type_info	754
Metodos de procesamiento de instrucciones	755
bind_col	755
bind-columns	755
bindqaram	755
bindqaram_array	757
bind_param_inout	757
dump_results	757
execute	758
execute_array	758
fetch	759
fetchall_arrayref	759
fetchall_hashref	760
fetchrow_array	760
fetchrow_arrayref	760
fetchrow_hashref	760
finish	761
rows	761
Atributos DBI comunes a todos los identificadores	761
Active	761
ActiveKids	762
CachedKids	762
ChopBlanks	762
CompatMode	762

FreeHashKeyName	762
HandleError	762
InactiveDestroy	763
Kids	763
LongReadLen	763
LongTruncOK	763
PrintError	763
private-*	764
Profile	764
RaiseError	764
ShowErrorStatement	764
Taint	765
Warn	765
Atributos de identificadores de base de datos	765
AutoCommit	765
Driver	765
Name	765
RowCacheSize	766
Statement	766
Atributos de identificadores de instrucciones	766
CursorName	766
NAME	766
NAME_hash	766
NAME_lc	767
NAME_lc_hash	767
NAME_uc	767
NAME_uc_hash	767
NULLABLE	767
NUM_OF_FIELDS	768
NUM_OF_PARAMS	768
ParamValues	768
PRECISION	768
RowsInCache	768
SCALE	768
Statement	768
TYPE	769
Atributos dinamicos	769
err	769
errstr	769
lasth	769
rows	769
state	769
Breve ejemplo de DBI Perl	770

E. API de base de datos Phyton	773
Atributos.....	773
Atributos de modulo	773
APILEVEL	774
CONV	774
PARAMSTYLE	774
THREADSAFETY	774
Atributos de cursor.....	774
ARRAYSIZE	774
DESCRIPTION	774
ROWCOUNT	775
Metodos	775
Metodos de modulo	775
Metodos de conexión	776
BEGIN	776
CLOSE	776
COMMIT	776
CURSOR	776
ROLLBACK	776
Metodos de cursor	776
CLOSE	777
EXECUTE	777
EXECUTEMANY	777
FETCHALL.....	777
FETCHMANY	778
FETCHNONE.....	778
INSERT_ID	778
NEXTSET, SETINPUTSIZES y SETOUTPUTSIZES	778
Breve ejemplo de Phyton	778
F. API Java	781
Metodos generales	781
getBundle	781
getConnection	782
getString	783
Metodos de conexión	783
clearWarnings	783
close.....	783
commit.....	784
createStatement	784
getAutoCommit.....	784
getMetaData	784
getTransactionIsolation	784

getTypeMap	784
isClosed	785
isReadOnly	785
nativeSQL	785
prepareStatement	785
rollback	785
setAutoCommit	785
setReadOnly	785
setTransactionIsolation	786
setTypeMap	786
Metodos de instrucciones y de instrucciones preparadas	786
addBatch	786
clearBatch	786
clearWarnings	786
close	787
execute	787
executeBatch	787
executeQuery	787
executeUpdate	787
getConnection	787
getFetchSize	788
getMaxFieldSize	788
getMaxRows	788
getMoreResults	788
getQueryTimeout	788
getResultSet	788
getResultType	789
getUpdateCount	789
setXXX	789
setCursorName	790
setEscapeProcessing	790
setFetchSize	790
setMaxFieldSize	790
setMaxRows	791
setQueryTimeout	791
Metodos ResultSet	791
absolute	791
afterLast	791
beforeFirst	791
cancelRowUpdates	791
close	792
deleteRow	792
findColumn	792

first	792
getXXX	792
getCursorName	793
getFetchSize	793
getMetaData	793
getRow()	793
getStatement	794
getType	794
getWarnings	794
insertRow	794
isAfterLast	794
isBeforeFirst	794
isFirst	794
isLast	795
last	795
moveToCurrentRow	795
moveToInsertRow	795
next	795
previous	796
refreshRow	796
relative	796
rowDeleted	796
rowInserted	796
rowUpdated	796
setFetchSize	797
updateXXX	797
updateRow	798
wasNull	798
Metodos ResultSetMetaData	798
getColumnCount	798
getColumnDisplaySize	798
getColumnName	798
getColumnType	798
getColumnTypeName	799
getPrecision	799
getScale	799
getTableName	799
isAutoIncrement	799
isCaseSensitive	799
isDefinitelyWritable	799
isNullable	800
isReadOnly	800
isSearchable	800

isSigned	800
isWritable	800
Metodos SQLException	800
getErrorCode	800
getMessage	801
getNextException	801
getSQLState	801
printStackTrace	801
setNextException	801
Metodos Warning	801
getNextWarning	801
setNextWarning	801
Breve ejemplo de Java	802
G. API C	805
Tipos de datos del API C	805
my_ulonglong	805
MYSQL	806
MYSQL_FIELD	806
MYSQL_FIELD_OFFSET	808
MYSQL_RES	808
MYSQL_ROW	808
Funciones del API C	809
mysql_affected_rows	809
mysql_change_user	809
mysql_character_set_name	809
mysql_close	810
mysql_connect	810
mysql_create_db	810
mysql_data_seek	810
mysql_debug	810
mysql_drop_db	811
mysql_dump_debug_info	811
mysql_eof	811
mysql_errno	811
mysql_error	811
mysql_escape_string	811
mysql_fetch_field	812
mysql_fetch_field_direct	812
mysql_fetch_fields	812
mysql_fetch_lengths	812
mysql_fetch_row	813
mysql_field_count	813

mysql_field_seek	814
mysql_field_tell	814
mysql_free_result	814
mysql_get_client_info	814
mysql_get_host_info	814
mysql_get_proto_info	814
mysql_get_server_info	815
mysql_info	815
mysql_init	815
mysql_insert_id	815
mysql_kill	816
mysql_list_dbs	816
mysql_list_fields	816
mysql_list_processes	817
mysql_list_tables	817
mysql_num_fields	817
mysql_num_rows	818
mysql_options	818
mysql_ping	819
mysql_query	819
mysql_real_connect	819
mysql_real_escape_string	821
mysql_real_query	821
mysql_reload	821
mysql_row_seek	821
mysql_row_tell	822
mysql_select_db	822
mysql_shutdown	822
mysql_stat	822
mysql_store_result	823
mysql_thread_id	823
mysql_use_result	823
Breve ejemplo del API C	824
H. ODBC y .NET	827
Origenes de datos	828
Configuración de un origen de datos en Windows	828
Configuración de un origen de datos en Unix	829
Configuración de opciones de conexión	829
Exportación de datos desde Microsoft Access a MySQL	830
Uso de ODBC	831
Ejemplo de VB.NET	831
Ejemplo de C#.NET	833

Ejemplo de VB ADO	834
Ejemplo de VB RDO	835
Ejemplo de VB DAO	837
Funciones MyODBC	838
SQLAllocConnect	838
SQLAllocEnv	838
SQLAllocHandle	839
SQLAllocStmt	839
SQLBindParameter	839
SQLBulkOperations	840
SQLCancel	840
SQLCloseCursor	840
SQLColAttribute	840
SQLColAttributes	841
SQLColumnPrivileges	841
SQLColumns	841
SQLConnect	841
SQLDataSources	841
SQLDescribeCol	842
SQLDescribeParam	842
SQLDisconnect	843
SQLDriverConnect	843
SQLDrivers	843
SQLEndTran	844
SQLError	844
SQLExecDirect	844
SQLExecute	844
SQLExtendedFetch	844
SQLFetch	844
SQLFetchScroll	844
SQLFreeConnect	845
SQLFreeEnv	845
SQLFreeHandle	845
SQLFreeStmt	845
SQLForeignKeys	846
SQLGetConnectAttr	846
SQLGetConnectOption	848
SQLGetCursorName	848
SQLGetDiagField	848
SQLGetDiagRec	850
SQLGetEnvAttr	851
SQLGetFunctions	852
SQLGetInfo	852

SQLGetStmtAttr	852
SQLGetStmtOption	856
SQLGetTypeInfo	856
SQLNativeSql	857
SQLNumParams	857
SQLNumResultCols	857
SQLParamData	857
SQLPrepare	858
SQLPrimaryKeys	858
SQLPutData	858
SQLRowCount	858
SQLSetConnectAttr	858
SQLSetConnectOption	859
SQLSetCursorName	859
SQLSetEnvAttr	859
SQLSetPos	859
SQLSetScrollOptions	860
SQLSetStmtAttr	860
SQLSetStmtOption	861
SQLSpecialColumns	861
SQLStatistics	862
SQLTablePrivileges	862
SQLTables	862
SQLTransact	862
I. Contenido del CD-ROM	865
Indice alfabético	867

Introducción

MySQL HA CRECIDO. Lo que durante un tiempo se consideró como un sencillo juguete para su uso en sitios Web, se ha convertido en la actualidad en una solución viable y de misión crítica para la administración de datos. Antes, MySQL se consideraba como la **opción** ideal para sitios Web; sin embargo, ahora incorpora muchas de las funciones necesarias para otros **entornos** y conserva su gran velocidad. MySQL **supera** desde **hace** tiempo a muchas soluciones comerciales en velocidad y dispone de un sistema de permisos elegante y potente, y ahora, **además**, la versión 4 incluye el motor de almacenamiento **InnoDB** compatible con ACID.

MySQL 4 es rápido, dispone de funciones de volcado online e incorpora una gran **cantidad** de funciones nuevas. Son pocas las razones para desechar MySQL como solución de base de datos. MySQL AB, la **compañía** responsable del desarrollo de MySQL, dispone de un sistema de asistencia eficiente y a un precio razonable, y, como ocurre con la mayor parte de las comunidades de código abierto, encontrará una gran **cantidad** de ayuda en la Web. Las funciones estándar no incluidas todavía en MySQL (como las vistas y los procedimientos almacenados) están en fase de desarrollo y es posible que estén disponibles para cuando lea estas **líneas**.

Son muchas las razones para escoger MySQL como solución de misión crítica para la administración de datos.

- **Coste:** El coste de MySQL es gratuito para la mayor parte de los usos y su servicio de asistencia resulta economico.
- **Asistencia:** MySQL AB ofrece contratos de asistencia a precios razonables y existe una nutrida y activa comunidad MySQL.
- **Velocidad:** MySQL es mucho mas rapido que la mayor parte de sus rivales.
- **Funcionalidad:** MySQL dispone de muchas de las funciones que exigen los desarrolladores profesionales, como compatibilidad completa con ACID, compatibilidad para la mayor parte de SQL ANSI, volcados online, duplication, funciones SSL e **integración** con la mayor parte de los entornos de programacion. Asi mismo, se desarrolla y actualiza de **forma** mucho mas rapida que muchos de sus rivales, por lo que practicamente todas las funciones estandar de MySQL todavia no estan en fase de desarrollo.
- **Portabilidad:** MySQL se ejecuta en la inmensa **mayoría** de sistemas operativos y, la mayor parte de los **casos**, los datos se pueden transferir de un sistema a otro sin dificultad.
- **Facilidad de uso:** MySQL resulta facil de utilizar y de administrar. Gran parte de las viejas bases de datos presentan problemas por utilizar sistemas obsoletos, lo que complica innecesariamente las **tareas** de administracion. Las herramientas de MySQL son potentes y flexibles, sin sacrificar su capacidad de uso.

Razones para comprar este libro

Este libro va dirigido a desarrolladores, administradores de bases de datos (DBA) y usuarios de MySQL. Aborda los siguientes temas:

- Exploración del lenguaje de **consulta** estructurado (SQL) en funcion de la **implementación** de MySQL.
- Comprension y uso de tipos de datos y de tablas.
- **Optimización** de sus consultas e indices.
- Volcados de bases de datos.
- Administracion de usuarios y seguridad.
- Administracion y **configuración** de MySQL (y optimizacion de la configuration para potenciar el rendimiento).
- **Duplicación** de MySQL en varios servidores.
- Comprension del **diseño** y la **normalización** de bases de datos y **análisis** de un **completo ejemplo práctico**. El conocimiento de estos temas resulta fun-

damental si se tiene la **intención** de utilizar MySQL en aplicaciones profesionales.

- Programacion con MySQL.
- Desarrollo de extensiones **propias** en MySQL.
- Instalacion de MySQL.

En los apendices del libro se incluyen los siguientes elementos:

- Una guia de referencia completa a MySQL.
- Guia de referencia **sobre** funciones y metodos de PHP, Perl, C, Java, Python y ODBC para interactuar con MySQL.

¿Qué queda fuera del alcance de este libro?

MySQL es un **concepto inmenso**, y en este libro se presenta **todo** el material necesario para ayudarle a convertirse en un **experto** administrador de bases de datos y desarrollador de MySQL. Sin embargo, como no se puede explicar **todo**, no se abordan los siguientes temas:

- Como programar. Este libro le ayudara a programar con MySQL, **pero no enseña** a programar desde cero.
- MySQL incrustado.
- Un **análisis completo** de como compilar e instalar bibliotecas. La labor de desarrollar extensiones exige ciertos conocimientos **sobre** la compilacion y la instalacion de bibliotecas en la plataforma de desarrollo seleccionada. Aunque este tema se explica, el libro no puede abordar todas las configuraciones posibles de las distintas plataformas, por lo que si tiene **pensado** abordar este nivel avanzado de desarrollo, necesitara una buena fuente de **información sobre** su plataforma.

¿Qué necesita?

Necesitara los siguientes elementos para seguir los ejemplos de este libro:

- Una copia de, o acceso a, un cliente y un servidor MySQL. Puede descargar la version actual del sitio de MySQL: www.mysql.com.
- Un sistema en el que instalar MySQL (si no dispone de acceso a alguno todavia). Puede instalar MySQL en **su** equipo de sobremesa, **pero** es mas habitual ejecutarlo en un servidor dedicado para aplicaciones complejas.

- Si tiene **pensado** desarrollar aplicaciones con MySQL, es posible que necesite descargar **los** ultimos controladores o interfaces de programacion de aplicaciones (API) de su **entorno** de desarrollo. MySQL integra lo mejor de PHP, Perl, Java, C, C++ y Python, pero puede utilizarlo en cualquier **entorno** de programacion, como .NET a traves del sistema de conectividad abierta de base de datos (ODBC). Visite el sitio Web de MySQL (www.mysql.com) para descargar las versiones mas actuales de los controladores.

¿Cómo utilizar este libro?

Este libro se divide en cuatro partes. Si tiene poca **experiencia** con bases de datos, le aconsejamos comenzar por el primer capitulo de la **primera** parte, que presenta el mundo de SQL a los usuarios noveles. Los lectores **experimentados** en el uso de otros sistemas de gestion de bases de datos pueden echar un rapido vistazo al capitulo 1 para familiarizarse con la forma de trabajar de MySQL, antes de pasar a examinar los tipos de datos y tablas de esta plataforma que se analizan en el capitulo 2. Los lectores con nivel intermedio pueden empezar en el capitulo 3 y capitulo 4, dedicados a aspectos de SQL avanzados, indices y temas de optimizacion. Los lectores que deseen utilizar un lenguaje de programacion con MySQL deberian leer el capitulo 5 y consultar el apendice relacionado con el lenguaje de programacion que utilicen. El capitulo 6 va dirigido a los lectores que ya conocen MySQL y que desean incorporar sus nuevas funciones.

Los lectores sin conocimiento formal **sobre** el **diseño** de bases de datos pueden aprovechar **los** contenidos de la segunda parte de este libro, en la que se analizan distintos aspectos **sobre** el diseño de base de datos a **menudo** ignorados y que **resultan** necesarios para desarrollar bases de datos a gran escala.

Todos los lectores que deseen administrar MySQL se beneficiaran de la parte **III** en la que se analizan conceptos avanzados **sobre** la optimizacion de bases de datos de alto rendimiento. Tambien se explican **los** temas de volcado, **duplicacion**, seguridad e instalacion.

Por ultimo, deberia consultar los apendices cuando tenga **dudas** sobre el SQL de MySQL y sus funciones y operadores, asi como **sobre las** funciones y **los** metodos de base de datos utilizados por los lenguajes de programacion mas **populares**.

NOTA: Los listados que aparecen en el libro son una representación impresa del código fuente que encontrará en el CD-ROM que acompaña a esta obra, especialmente creado para facilitar el aprendizaje al lector y como complemento práctico a su lectura.

Parte I

Uso

de MySQL

1

Guia rapida de MySQL

Asi que ha conseguido una copia de este libro. Puede que muchos de los lectores ya dispongan de conocimientos **sobre** MySQL y que deseen sumergirse en las turbias aguas de la **duplicación** de base de datos y de la **optimización** de variables de servidor. Si ya dispone de conocimientos avanzados **sobre** MySQL, puede saltarse este capitulo. Los principiantes no **deben** preocuparse. En este libro se abordan todos **los** elementos necesarios para empezar a utilizar MySQL y para convertirse en un usuario avanzado.

MySQL es la base de datos de codigo abierto mas popular del mundo. *Codigo abierto* significa que **todo** el mundo puede acceder **al** *codigo fuente*, es decir, **al** codigo de programacion de MySQL. **Todo** el mundo puede contribuir para incluir elementos, arreglar problemas, realizar mejoras o sugerir optimizaciones. Y asi ocurre. MySQL ha **pasado** de ser una "pequeia" base de datos a una completa herramienta y ha conseguido superar a una gran **cantidad** de bases de datos comerciales (lo que ha asustado a la mayor parte de **los** proveedores comerciales de bases de datos). Por lo **tanto**, su rapido desarrollo se debe a la **contribución** de mucha gente **al** proyecto, asi **como** a la **dedicación** del equipo de MySQL.

A diferencia de **los** proyectos propietarios, en **los** que el codigo fuente es desarrollado por un numero reducido de personas y se protege atentamente, **los** proyectos de codigo abierto no excluyen a nadie interesado en aportar ideas, si disponen de **los** conocimientos necesarios. En el aiiio 2000, cuando MySQL con-

taba con **sólo** cuatro **años** de existencia, Michael "MONTY" Widenius, el **funda-** dor de MySQL, predijo grandes avances para MySQL durante la **primera** con- vencion **sobre** bases de datos de código abierto. En aquel entonces, muchos proveedores de base de datos se burlaron de sus palabras. Hoy en día ya han desaparecido varios.

La versión 3 de MySQL logró hacerse con el dominio de la gama baja del mercado de Internet. Con el lanzamiento de la versión 4, este **producto** se dirige ahora a una base de clientes mucho más amplia. MySQL **hace** su entrada en el mercado de las bases de datos en un **momento** en el que Apache es el **producto** de código abierto dominante en el mercado de servidores Web y en el que la presencia de varios sistemas operativos de código abierto (como Linux y **FreeBSD**) es **cada** día más notable en el mercado de servidores.

En este capítulo se abordan **los** siguientes temas:

- Conceptos y terminología esenciales relacionados con bases de datos
- **Conexión** y desconexión a un servidor MySQL
- **Creación** y eliminación de bases de datos
- Agregación de datos a una tabla
- **Recuperación** y eliminación de datos desde una tabla
- Comprensión de las funciones estadísticas y de fecha **básicas**
- **Combinación** de varias tablas

Comprensión de los fundamentos de MySQL

MySQL es un *sistema de administración de bases de datos relacional* (RDBMS). Se trata de un programa capaz de almacenar una **enorme cantidad** de datos de gran **variedad** y de distribuirlos para cubrir las necesidades de cualquier **tipo de organización**, desde **pequeños** establecimientos comerciales a grandes empresas y organismos administrativos. MySQL compite con sistemas RDBMS propietarios conocidos, como Oracle, SQL Server y **DB2**.

MySQL incluye todos **los** elementos necesarios para instalar el programa, preparar diferentes niveles de acceso de usuario, administrar el sistema y proteger y **hacer** volcados de datos. Puede desarrollar sus **propias** aplicaciones de base de datos en la mayor parte de **los** lenguajes de programación utilizados en la **actua-** lidad y ejecutarlos en casi todos **los** sistemas operativos, incluyendo algunos de **los** que probablemente no ha oído nunca hablar. MySQL utiliza el lenguaje de **consulta** estructurado (SQL). Se trata del lenguaje utilizado por todas las bases de relacionales, que presentaremos en una **sección** posterior. Este lenguaje permite crear bases de datos, así como agregar, manipular y recuperar datos en función de criterios específicos.

Pero nos estamos adelantando. En este capítulo, se analizan brevemente los conceptos **relativos a las bases de datos relacionales**. Aprenderemos que se entiende exactamente por una base de datos relacional y como **funciona, además** de comentar terminología clave. Armados con esta **información**, podremos crear una sencilla base de datos y trabajar con sus datos.

¿Qué es una base de datos?

Una base de datos, en su **definición** mas sencilla, **es** una colección de **archivos** relacionados. Imagine un archivo (ya sea en **formato de papel** o electrónico) que contenga **los pedidos de ventas** de una tienda. También existirá otro archivo de productos, en el que se incluyen los registros **sobre existencias**. Para completar un pedido, necesitará **buscar el producto** en el archivo de pedidos y **los niveles de existencias** relativos a dicho **producto** en el archivo de productos. Una base de datos y el software que **controla** la base de datos, denominado **sistemn de administración de base de datos** (DBMS), le ayudará a realizar estas **tareas**. La mayor parte de **las bases de datos** actuales son de tipo **relacional**. Se denominan así porque utilizan tablas de datos relacionadas por un campo en común. Por ejemplo, la tabla 1.1 muestra la tabla Product y la tabla 1.2 muestra la tabla Invoice. Como puede observar, la **relación** entre **las** dos tablas se establece a **partir del** campo `stock_code`. Dos tablas cualesquiera se pueden relacionar utilizando un campo común.

Tabla 1.1. La tabla Product

Stock_code	Description	Prize
A416	Clavos, caja	0,14 dolares
C923	Chincheta, caja	0,08 dolares

Tabla 1.2. La tabla Invoice

Invoice_code	Invoice_line	Stock_code	Quantity
3804	1	A416	10
3804	2	C923	15

Terminología sobre bases de datos

Examinemos **más de cerca** las dos tablas anteriores para comprobar como se organizan:

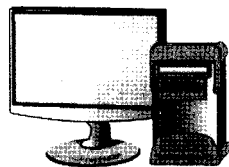
- Cada tabla se compone de una gran **cantidad** de *filas* y *columnas*

- Cada fila contiene datos sobre una sola *entidad* (como un producto o un pedido). Estos datos se conocen como *registros*. Por ejemplo, la primera fila de la tabla 1.1 es un registro: describe el producto A416, que es una caja de clavos que cuesta 14 centimos de dolar. Por lo tanto, el termino *fila* y el termino *registro* son intercambiables.
- Cada columna contiene datos relacionados con el registro, llamados *atributos*. Ejemplos de atributos son la *cantidad* de un articulo vendido o el precio de un producto. Los atributos en referencia a una tabla de base de datos se denominan *campos*. Por ejemplo, los datos de la columna Description de la tabla 1.1 son campos. Por lo tanto, el termino *atributo* y el termino *campo* son intercambiables.

Dado este tipo de estructura, la base de datos nos brinda una forma de manipular los datos: SQL. SQL es una potente herramienta para realizar búsquedas sobre registros o realizar cambios. Practicamente todos los DBMS utilizan este lenguaje, aunque la mayoría ha agregado sus propios elementos, lo que significa que al estudiar SQL en este capítulo y en los siguientes, se explicaran características específicas de MySQL. La mayor parte de los conceptos que se expliquen, se pueden utilizar en otras bases de datos relacionales, como PostgreSQL, Oracle, Sybase o SQL Server. Sin embargo, tras comprobar las ventajas de MySQL, es probable que no desee cambiar.

Conexión a una base de datos

El equipo en el que se ejecuta MySQL y que almacena los datos se denomina *servidor MySQL*. Para establecer una *conexión* a este servidor, dispone de varias opciones de instalacion. En primer lugar, puede instalar el cliente y el servidor MySQL en su equipo de escritorio, como ilustra la figura 1.1. En segundo lugar, puede instalar el cliente MySQL en su equipo de sobremesa y el servidor MySQL en otro equipo al que se estableciera la conexion, como se ilustra en la figura 1.2. Por ultimo, su equipo de sobremesa puede ser cualquier ordenador que se conecte a otro equipo con un cliente MySQL instalado, que a su vez se conectara al servidor MySQL, situado en el mismo equipo o en otro, como muestra la figura 1.3.



Servidor y cliente MySQL

Figura 1.1. Nuestro equipo tiene instalado el cliente y el servidor MySQL

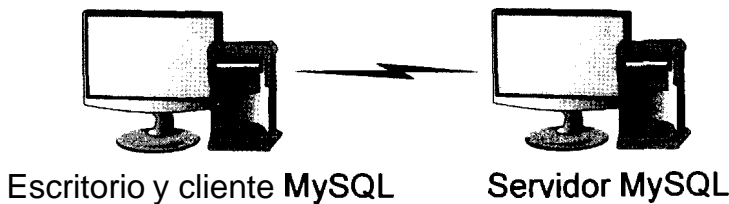


Figura 1.2. Nuestro equipo tiene instalado el cliente MySQL. El servidor MySQL se encuentra instalado en otro equipo al que se conecta el nuestro.

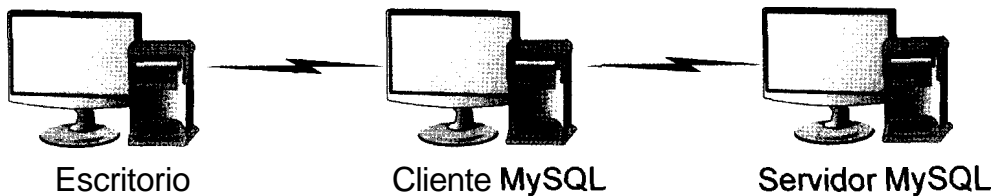


Figura 1.3. En este caso, nuestra terminal puede ser cualquier equipo capaz de conectarse a otro, ya que ni siquiera lleva instalado el cliente MySQL.

Si el cliente MySQL no se encuentra instalado en su equipo de sobremesa y necesita conectarse a un segundo equipo para utilizar el cliente MySQL, es probable que necesite utilizar Telnet o un cliente Secure Shell (SSH) para realizar la conexión.

Para ello, basta con abrir el programa Telnet, introducir el nombre del anfitrión, un nombre de usuario y una contraseña. Si no tiene claro cómo hacerlo, consulte al administrador de su sistema.

Tras registrarse en el equipo en el que está instalado el programa cliente MySQL, la operación de conexión al servidor resulta sencilla:

- En un equipo Unix (por ejemplo, Linux o FreeBSD), ejecute el siguiente comando desde la línea de comandos de su intérprete:

```
% mysql -h nombre del anfitrión -u nombre de usuario -p contraseña
nombre de la base de datos
```

- En un equipo Windows, ejecute el mismo comando desde la línea de comandos:

```
% mysql -h nombre del anfitrión -u nombre de usuario -p contraseña
nombre de la base de datos
```

El símbolo % indica el símbolo de comando del intérprete de comandos. Es probable que su equipo utilice otro símbolo (por ejemplo, `c: \>` en Windows o `$` en algunos intérpretes de comandos de Unix). La `-h` y la `-u` pueden aparecer seguidas de un espacio (también puede eliminar el espacio). La `-p` debe ir seguida inmediatamente de la contraseña, sin espacios intercalados.

Tras establecer la conexión, aparecerá el símbolo de comandos `mysql>`, como ocurre en la mayor parte de las distribuciones. No necesita escribir esta secuencia ya que se generará automáticamente.

Si aparece un símbolo de comando ligeramente diferente, no se preocupe y escriba el texto en negrita. Ésta es la convención que se utilizará a lo largo de todo el libro.

TRUCO: Existe una forma más segura de introducir la contraseña, que es aconsejable utilizar en un entorno multiusuario. Escriba `-p` sin introducir la contraseña. Cuando MySQL se inicie, le solicitará la contraseña, que podrá introducir sin que aparezca en pantalla. De esta forma evitará que alguien pueda verla al escribirla.

El nombre del anfitrión será el nombre del equipo en el que se aloja el servidor (por ejemplo, `www.sybex.com` o una dirección IP como `196.30.168.20`). No necesita utilizar este parámetro si ya está registrado en el servidor (en otras palabras, si el cliente y el servidor MySQL se encuentran instalados en el mismo equipo). El administrador le asignará el nombre de usuario y la contraseña (se trata de la contraseña y el nombre de usuario de MySQL, que son diferentes a los utilizados para el equipo cliente). Algunos equipos inseguros no requieren el uso de un nombre de usuario o contraseña.

TRUCO: El administrador del sistema puede dificultar el trabajo si no ha instalado MySQL en la ruta predeterminada. Por lo tanto, al escribir el comando `mysql`, es posible que reciba un error `command not found` (Unix) o un error `bad command or file name` (en Windows) a pesar de que estar seguros de haber instalado MySQL. En este caso deberá introducir la ruta completa hasta el cliente MySQL (por ejemplo, `/usr/local/build/mysql/bin/mysql`, o en Windows, algo así como `C:\mysql\bin\mysql`). Pregúntele a su administrador cuál es la ruta correcta si se le presenta este problema.

Para desconectarse, basta con escribir `QUIT` de la siguiente forma:

```
mysql> QUIT  
Bye
```

También puede escribir `EXIT` o pulsar **Control-D**.

NOTA: MySQL no distingue entre mayúsculas y minúsculas aquí. Puede utilizar `QUIT`, `quit` o incluso `QUIT`, si lo desea.

Creación y uso de nuestra primera base de datos

En las siguientes secciones se describe como crear una base de datos y como realizar consultas **sobre ella**. Asumiremos que ha establecido una **conexión al servidor MySQL** y que dispone de permisos para utilizar una base de datos. De lo contrario, solicite dichos permisos a su administrador. Si denominamos a esta base de datos como `firstdb`, pida a su administrador que **Cree y le conceda permiso absoluto de acceso a dicha base de datos únicamente**. De esta forma evitara problemas relacionados con permisos posteriormente, **además de ahorrarle un ataque al corazon a su administrador si ocurriera alguna catastrofe con las bases de datos existentes**. Si nuestro administrador no tiene claro que es lo que tiene que hacer o si hemos instalado MySQL nosotros mismos, deberemos utilizar uno de **los dos conjuntos de comandos** que se indican a continuacion para poder empezar a trabajar con la base de datos. Recuerde que sólo debe **introducir el texto en negrita**.

Caso en el que acabemos de instalar MySQL

En primer lugar, establezca una **conexión** a la base de datos MySQL como **raiz**. Como acaba de empezar, todavia no dispondra de una contraseiia raiz, **razon por la que lo primero que tiene que hacer es asignar una contraseiia al usuario raiz**.

```
% mysql -u root mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 15 to server version: 4.0.2-alpha-Max

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> SET PASSWORD=PASSWORD('g00r002b');
Query OK, 0 rows affected (0.00 sec)
```

Por cuestiones de sencillez, vamos a utilizar la misma contraseiia para el usuario raiz, `g00f002b`, que para **el usuario que vamos a crear, guru2b**.

A continuacion, tendremos que crear la base de datos `firstdb` con la que vamos a trabajar.

```
mysql> CREATE DATABASE firstdb;
Query OK, 1 row affected (0.01 sec)
```

Por ultimo, es necesario crear el usuario con el que vamos a trabajar, `guru2b`, con la contraseiia `g00r002b`, y concederle **permiso completo de acceso a la base de datos firstdb**:

```
mysql> GRANT ALL ON firstdb.* to guru2b@localhost
```



```
IDENTIFIED BY 'g00r002b';
Query OK, 0 rows affected (0.01 sec)
mysql> exit
Bye
```

Si un administrador necesita concederle permiso

En primer lugar, el administrador tendrá que establecer la conexión a la base de datos MySQL como usuario *raíz* (o como cualquier otro usuario que disponga de permisos para conceder permiso a otro).

```
% mysql -u root -p mysql
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 15 to server version: 4.0.2-alpha-Max
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

A **continuación**, su administrador tendrá que crear la base de datos `firstdb` con la que vamos a trabajar:

```
mysql> CREATE DATABASE firstdb;
Query OK, 1 row affected (0.01 sec)
```

Por último, es necesario crear el usuario con el que vamos a trabajar, `guru2b`, con la contraseña `g00r002b`, y concederle acceso **completo** a la base de datos `firstdb`. Fíjese en que se asume que la conexión a la base de datos se establecerá desde `localhost` (es decir, que el cliente y el servidor de la base de datos están instalados en el mismo equipo). Si no fuera así, su administrador tendrá que sustituir `localhost` por el nombre del equipo pertinente:

```
mysql> GRANT ALL ON firstdb.* to guru2b@localhost
IDENTIFIED BY 'g00r002b';
Query OK, 0 rows affected (0.01 sec)
mysql> exit
Bye
```

`guru2b` es su nombre de usuario para acceder a MySQL y el que utilizaremos a lo largo de **todo** el libro, y `g00r002b`, es la contraseña. Puede utilizar, o le puede asignar, otro nombre de usuario. En un capítulo posterior, analizaremos el tema de la concesión de permisos.

Uso de la base de datos

Si no ha trabajado antes con SQL o MySQL, esta es su oportunidad de **ponerse manos a la obra**. Le aconsejamos realizar los ejemplos que se incluyen a **continuación** en el **orden** en el que se presentan. Ahora bien, el verdadero proceso de aprendizaje consiste en dejar el libro a un **lado** y escribir otras consultas. Por lo **tanto**, le aconsejamos que experimente. Utilice variaciones que le parezcan que

pueden funcionar. No tenga miedo de cometer **errores** en esta fase, ya que son la mejor **forma** de aprender.

Los datos con los que **estamos** trabajando no son importantes. Es mejor **eliminar** ahora las bases de datos de ejemplo de **forma** accidental que **millones** de registros vitales dentro de un **año**.

Comenzaremos por crear una tabla dentro de nuestra base de datos de **ejemplo** y la **rellenaremos** con datos.

Tras crear varias tablas y completarlas, explicaremos como realizar consultas **sobre ellas**. En primer lugar, estableceremos una **conexión** a la tabla recién **creada** utilizando el siguiente comando:

```
% mysql -u guru2b -pg00r002b firstdb
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 15 to server version: 4.0.2-alpha-Max

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

Si **los permisos** no se han establecido correctamente, obtendra un error como el siguiente:

```
ERROR 1044: Access denied for user: 'guru2be@localhost' to
database 'firstdb'
```

Si así fuera, necesitaremos (nosotros o nuestro administrador) revisar **los pasos** de las dos secciones anteriores.

Todas estas cuestiones relacionadas con **los permisos** pueden parecer un poco complicadas, **pero resultan** de gran utilidad. En el futuro necesitara restringir el acceso a sus datos y la concesion de permisos es la **forma** de conseguirlo.

Tambien puede establecer la **conexión** sin especificar una base de datos, de la siguiente **forma**:

```
% mysql -u guru2b -pg00r002b guru2b
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 15 to server version: 4.0.2-alpha-Max

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

A **continuación**, si queremos estar seguros de utilizar la base de datos **correcta**, tendremos que indicarselo a MySQL.

Para **ello**, utilice la siguiente **instrucción**:

```
mysql> USE firstdb
Database changed
```

Puede establecer la **conexión** a su base de datos de dos **formas**: especificando la base de datos **al** establecer la **conexión** o posteriormente cuando este **conectado**. En el futuro, cuando tenga que utilizar mas de una base de datos en su **sistema**, descubrira que resulta mucho mas sencillo cambiar de bases de datos utilizando la instruccion **USE**.

Creación de una tabla

Tras conectarse a la base de datos, es probable que desee introducir datos. Para ello, vamos a crear una base de datos que puede hacer el seguimiento de un equipo de ventas. Como ya aprendimos, las bases de datos, se componen de una gran cantidad de tablas y, para empezar, crearemos una tabla que contenga datos sobre los comerciales. Almacenaremos sus nombres, números de identificación y comisiones. Para crear una tabla, también vamos a utilizar el comando CREATE, pero necesitaremos especificar TABLE en lugar de DATABASE, así como algunos elementos adicionales. Introduzca la siguiente instrucción CREATE:

```
mysql> CREATE TABLE sales-rep(  
    employee-number INT,  
    surname VARCHAR(40),  
    first-name VARCHAR(30),  
    commission TINYINT  
);  
Query OK, 0 rows affected (0.00 sec)
```

ADVERTENCIA: No olvide introducir el punto y coma al final de la línea. Todos los comandos de MySQL deben terminar en un punto y coma. Su olvido es la principal razón de la frustración de los principiantes. Además, si no introduce el punto y coma y pulsa Intro, tendrá que hacerlo antes de volver a pulsar Intro. MySQL acepta comandos distribuidos en varias líneas.

No necesita introducir la instrucción de la forma en la que aparece impresa en el ejemplo. Aquí se ha dividido la instrucción en varias líneas para facilitar su lectura, pero es probable que le resulte más sencillo introducir el comando en una sola. Así mismo, puede variar el uso de mayúsculas y minúsculas del ejemplo, sin que ello afecte a su funcionamiento. A lo largo de este libro, utilizaremos mayúsculas para representar palabras clave de MySQL y minúsculas para representar nombres seleccionados. Por ejemplo, podríamos haber introducido la siguiente secuencia:

```
mysql> create table SALES-REPRESENTATIVE(  
    EMPLOYEE-NO int,  
    SURNAME varchar(40),  
    FIRST_NAME varchar(30),  
    COMMISSION tinyint  
);
```

sin problemas. Sin embargo, si utilizamos el siguiente fragmento:

```
mysql> CREATE TABLES sales-rep(  
    employee-number INT,
```

```

    surname VARCHAR(40),
    first_name VARCHAR(30),
    commission TINYINT
);

```

se generaria este error:

```

ERROR 1064: You have an error in your SQL syntax near
'TABLES sales_reps(employee_number INT,surname
VARCHAR(40),first_name VARCHAR(30) ' at line 1

```

porque se ha escrito erroneamente TABLE. Por lo tanto, al escribir texto en mayusculas tenga cuidado de no introducir errores; puede cambiar el texto en minusculas sin problemas (siempre y cuando se haga de forma uniforme y se utilicen los mismos nombres de principio a fin).

Puede que se este preguntando por el significado de los terminos INT, VARCHAR y TINYINT que aparecen tras los nombres de los campos. Es lo que se denominan *tipos de datos* o *tipos de columna*. INT equivale a *entero*, un numero sin decimales cuyo valor oscila entre -2.147.483.648 y 2.147.483.647. Es aproximadamente a un tercio de la poblacion mundial, por lo que resultara suficiente para el equipo de ventas, por mucho que crezca. VARCHAR equivale a *caracter de longitud variable*. El numero entre parentesis indica la longitud maxima de la cadena de caracteres. Una cantidad de 30 y 40 caracteres resultara suficiente para el nombre y el apellido de los comerciales, respectivamente. Y TINYINT equivale a *entero pequeño*, por lo general un numero sin decimales cuyo valor oscila entre -128 y 127. El campo commission indica un valor de porcentaje y, como nadie puede ganar mas del 100 por cien, basta con utilizar un numero entero pequeiio. En el siguiente capitulo se analizaran detalladamente los distintos tipos de columna y cuando utilizarlos.

Listado de las tablas de una base de datos con SHOW TABLES

Ahora que ya tenemos una tabla, podemos confirmar su existencia con SHOW TABLES:

```

mysql> SHOW TABLES;
+-----+
| Tables_in_firstdb |
+-----+
| sales_rep          |
+-----+
1 row in set (0.00 sec)

```

SHOW TABLES muestra todas las tablas existentes en la base de datos actual. En el caso de nuestra tabla firstdb solo tenemos una: sales_rep. Por lo tanto, a menos que tenga un problema de memoria, este comando no resulta de gran utilidad en este momento. Sin embargo, en las bases de datos de gran tamaño compuestas por una gran cantidad de tablas, este comando resultara de utili-

dad para recordar el nombre de aquella tabla que creamos hace dos meses y cuyo nombre hemos olvidado. También puede ocurrir que tengamos que trabajar sobre una base de datos que no hayamos creado. En este caso, el comando `SHOW TABLES` resultará de gran valor.

Análisis de la estructura de las tablas con DESCRIBE

`DESCRIBE` es el comando que muestra la estructura de la tabla. Para comprobar que MySQL ha creado la tabla correctamente, escriba lo siguiente:

```
mysql> DESCRIBE sales_rep;
+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| employee-number | int(11)       | YES  |     | NULL    |      |
| surname        | varchar(40)   | YES  |     | NULL    |      |
| first-name     | varchar(30)   | YES  |     | NULL    |      |
| commission     | tinyint(4)    | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)
```

Esta tabla incluye todo tipo de columnas con las que no está todavía familiarizado. Fijese por el momento en la columna `Field` y en la columna `Type`. En el siguiente capítulo se explicarán el resto de los encabezados. Los nombres de los campos son los mismos que los introducidos y los dos campos `VARCHAR` tienen el tamaño que les asignamos. Como observará, se ha asignado un tamaño a los campos `INT` y `TINYINT`, aunque no especificamos ninguno al crearlos. Recuerde que el valor de un campo `TINYINT` oscila entre -128 y 127 de manera predeterminada (cuatro caracteres incluyendo el signo menos) y que el valor de un campo `INT` oscila entre -2.147.483.648 y 2.147.483.647 (11 caracteres incluyendo el signo menos), por lo que la misteriosa asignación de tamaño se corresponde con la longitud en pantalla.

Inserción de nuevos registros en una tabla

Ahora que ya tenemos una tabla, procederemos a introducir algunos datos en ella. Supongamos que tenemos tres comerciales (como muestra la tabla 1.3).

Tabla 1.3. Tabla Sales Reps

Employee_number	Surname	First_Name	Commission
1	Rive	Sol	10
2	Gordimer	Charlene	15
3	Serote	Mike	10

Para introducir estos datos en la tabla, se utiliza la instrucción SQL INSERT para crear un registro, de la siguiente forma:

```
mysql> INSERT INTO
sales_rep(employee_number,surname,first_name,commission)
VALUES(1,'Rive','Sol',10);
mysql> INSERT INTO
sales_rep(employee_number,surname,first_name,commission)
VALUES(2,'Gordimer','Charlene',15);
mysql> INSERT INTO
sales_rep(employee_number,surname,first_name,commission)
VALUES(3,'Serote','Mike',10);
```

NOTA: Es necesario **encerrar** el valor del campo de **cadena** (un campo de carácter VARCHAR) entre **comillas sencillas**; en el caso de los campos **numéricos** (commission, employee_number) no es necesario realizar esta operación. **Asegúrese de aplicar comillas a valores pertinentes y de abrirlas y cerrarlas correctamente (todos los valores que tengan una comilla de apertura deben llevar una comilla de cierre)**, ya que **al empezar a trabajar con SQL** suele olvidarse.

Existe otra forma mas sencilla de introducir datos con la instrucción INSERT, como se muestra en la siguiente secuencia:

```
mysql> INSERT INTO sales_rep VALUES(1,'Rive','Sol',10);
mysql> INSERT INTO sales_rep VALUES(2,'Gordimer','Charlene',15);
mysql> INSERT INTO sales_rep VALUES(3,'Serote','Mike',10);
```

Si introduce los comandos de esta forma, debe incluir los campos en el mismo orden en el que se define en la base de datos. No puede utilizar la siguiente secuencia:

```
mysql> INSERT INTO sales_rep VALUES(1,'Sol','Rive',10);
Query OK, 1 row affected (0.00 sec)
```

Aunque esta secuencia funciona en apariencia, los datos se han introducido en el orden incorrecto, ya que Sol sera el apellido y Rive el nombre.

TRUCO: Es aconsejable que se acostumbre a utilizar la **instrucción INSERT completa especialmente si tiene pensada realizar consultas utilizando un lenguaje de programación**. En primer lugar, **el uso de este formato reduce las posibilidades de error (en la instrucción no se podía apreciar el apellido y el nombre se habían introducido en el orden correcto)** y, en **segundo lugar, potencia la flexibilidad de sus programas**. En un capítulo posterior se **analiza este concepto de forma** mas extensa.

Insercion de datos dentro de una instruccion INSERT

Otra forma mas sencilla de utilizar el comando INSERT para introducir los datos de una sola vez consiste en separar los registros mediante comas, como se ilustra a continuación:

```
mysql> INSERT INTO sales-rep
(employee_number,surname,first_name,commission)
VALUES
(1,'Rive','Sol',10),(2,'Gordimer','Charlene',15),
(3,'Serote','Mike',10);
Query OK, 3 rows affected (0.05 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

Este método reduce la cantidad de código que resulta necesario escribir y el servidor procesa la secuencia de manera mas rapida.

Insercion de grandes cantidades de datos desde un archivo de texto con LOAD DATA

Una ultima forma de insertar datos (y la mejor en caso de que se necesiten introducir grandes cantidades de datos a la vez) consiste en utilizar la instruccion LOAD DATA, de la siguiente forma:

```
mysql> LOAD DATA LOCAL INFILE "sales-rep.sql" INTO TABLE
sales-rep;
```

El formato del archivo de datos debe ser correcto, sin excepciones. En este caso, en el que estamos utilizando los valores predeterminados, el archivo de texto incluye cada registro en una nueva linea y cada campo se separa por medio de un tabulador. Si asumimos que el caracter `\t` representa un tabulador y que cada linea termina en un caracter de nueva linea, el archivo presentaria este aspecto:

```
1\tRive\tSol\t10
2\tGordimer\tCharlene\t15
3\tSerote\tMike\t10
```

Se utiliza una **variación** de esta secuencia para restaurar copias de seguridad (comentadas en un capítulo posterior). Esta instruccion resulta incluso mas eficaz que una instruccion INSERT para introducir varios registros. La palabra clave LOCAL indica al servidor que el archivo se encuentra incluido en el equipo cliente (el equipo desde el que se establece la conexion). Si se omite, MySQL buscara el archivo en el servidor de la base de datos. De manera predeterminada, LOAD DATA asume que los valores se encuentran en el mismo orden que en la **definición** de la tabla, que cada campo esta separado por un tabulador y que cada registro se incluye en una linea.

Recuperación de información de una tabla

La operación de extraer información de una tabla resulta sencilla. Para ello, puede utilizar el potente comando SELECT, como se muestra en el siguiente ejemplo:

```
mysql> SELECT commission FROM sales-rep WHERE surname='Gordimer';
+-----+
| commission |
+-----+
|          15 |
+-----+
1 row in set (0.01 sec)
```

La instrucción SELECT consta de varias partes. La primera, inmediatamente después del comando SELECT, es la lista de campos. Puede recuperar otros campos, en lugar de recuperar únicamente el campo commission, de la siguiente forma:

```
mysql> SELECT commission, employee-number FROM
      sales-rep WHERE surname='Gordimer';
+-----+-----+
| commission | employee-number |
+-----+-----+
|          15 |                2 |
+-----+-----+
1 row in set (0.00 sec)
```

También puede utilizar el carácter comodín (*) para devolver todos los campos, de la siguiente forma:

```
mysql> SELECT * FROM sales-rep WHERE surname='Gordimer';
+-----+-----+-----+-----+
| employee-number | surname | first-name | commission |
+-----+-----+-----+-----+
|                2 | Gordimer | Charlene   |          15 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

El uso del comodín * indica todos los campos de la tabla. Por lo tanto, en el ejemplo anterior, se recuperan los cuatro campos, en el mismo orden en el que se recogen en la estructura de la tabla.

La parte de la instrucción SELECT situada tras el término WHERE se denomina cláusula WHERE. Esta cláusula es muy flexible y contiene una gran cantidad de condiciones de distinto tipo. Examine el siguiente ejemplo:

```
mysql> SELECT * FROM sales-rep WHERE commission>10
      OR surname='Rive' AND first_name='Sol';
+-----+-----+-----+-----+
| employee-number | surname | first-name | commission |
```



```

+-----+-----+-----+
|           1 | Rive       | Sol         |           10 |
|           2 | Gordimer  | Charlene   |           15 |
+-----+-----+-----+
2 rows in set (0.00 sec)

```

La comprensión de los operadores AND y OR es fundamental para utilizar SQL correctamente. Las condiciones situadas a ambos lados de la palabra clave AND **deben** ser verdad para que el **conjunto** sea verdadero. En el **caso** de una **instrucción** OR basta con que una de las condiciones sea verdadera. En la tabla 1.4 se recoge la tabla de verdad de los operadores AND/OR.

Tabla 1.4. Tabla de verdad AND/OR

Palabra clave	Cálido	Húmedo	Global
AND	Verdadero	Verdadero	Verdadero, calido AND humedo
AND	Verdadero	Falso	Falso, no calido AND no humedo, puesto que no es humedo.
AND	Falso	Verdadero	Falso, no calido AND no humedo, puesto que no es calido.
AND	Falso	Falso	Falso, no calido AND no húmedo, puesto que ninguna de las condiciones es verdadera.
OR	Verdadero	Verdadero	Verdadero, calido OR humedo, puesto que ambas son verdaderas.
OR	Verdadero	Falso	Verdadero, calido OR humedo, puesto que es calido.
OR	Falso	Verdadero	Verdadero, calido OR humedo puesto que es húmedo.
OR	Falso	Falso	Falso, no calido OR humedo, puesto que ninguna de las dos condiciones es valida.

Orden en el que MySQL procesa las condiciones

El siguiente ejemplo muestra una trampa en la resulta sencillo caer. Suponga que nuestro jefe nos pide una lista de los empleados cuyo apellido sea Rive y cuyo nombre sea Sol o que su comision supere el 10 por ciento. Podríamos construir la siguiente consulta:

```

mysql> SELECT * FROM sales_rep WHERE surname='Rive'
AND first_name='Sol' OR commission>10;

```

```

+-----+-----+-----+
| employee-number | surname | first-name | commission |
+-----+-----+-----+
|           1 | Rive   | Sol        |          10 |
|           2 | Gordimer | Charlene   |          15 |
+-----+-----+-----+
2 rows in set (0.00 sec)

```

Puede que este sea el resultado que estábamos buscando. Pero puede que nuestro jefe se **refiriera** a otra cosa: que el **empleado** tenga **como** apellido Rive y, dentro de estos registros, que su nombre sea Sol o que tenga una comisión superior **al** 10%.

En este caso, el segundo registro devuelto por la consulta no sería pertinente porque aunque su porcentaje es superior **al** 10%, no se llama Sol. La **construcción** AND implica que **ambas** cláusulas **deben** ser verdaderas. En este caso, la consulta presentaría este aspecto:

```

mysql> SELECT * FROM sales-rep WHERE surname='Rive'
AND (first-name='Sol' OR commission>10);
+-----+-----+-----+
| employee-number | surname | first-name | commission |
+-----+-----+-----+
|           1 | Rive   | Sol        |          10 |
+-----+-----+-----+
1 row in set (0.00 sec)

```

Fíjese en los **parentesis** utilizados en la consulta. Cuando se aplican varias condiciones, es fundamental conocer el **orden** en el que **deben** procesarse. ¿Qué va primero, la parte OR o la parte AND? Por **regla** general, es probable que las instrucciones **orales** que reciba **sean** poco claras, **pero** este ejemplo muestra la importancia de determinar con claridad los registros que se desean recuperar antes de implementar la consulta.

En ocasiones este **tipo** de errores no se descubren nunca. A **menudo** se suelen achacar a los ordenadores **pero** en **realidad** la culpa es de una persona, por lo general la responsable de **diseñar** la consulta.

En un **capítulo** posterior se recoge una lista de operadores y su **orden** de **prioridad**. Es aconsejable utilizar los **parentesis** para determinar el **orden** de **preferencia** dentro de sus consultas. En algunos libros y algunas personas asumen que se conoce el **orden** de prioridad.

Por ejemplo, puede que en la escuela haya aprendido que $1 + 1 * 3 = 4$, no 6, porque sabe que la **multiplicación** se realiza antes que la **operación** de suma. Lo mismo se **aplica** al operador AND, que tiene preferencia **sobre** OR. Pero puede que **no todo** el mundo sea consciente de estas reglas, por lo que el uso de **paréntesis** ayudara a dejar claro que lo que queremos decir es $1 + (1 * 3)$. Incluso después de muchos **años** de programación, muchos profesionales no conocemos el **orden** de prioridad **completo** de todos los operadores y probablemente nunca lo sepamos.

Correspondencia de patrones

A continuación examinaremos algunos elementos adicionales de la instrucción SELECT. Imagine que queremos recuperar los datos de Mike Serote. Sencillo, ya que bastara con utilizar la siguiente consulta:

```
mysql> SELECT * FROM sales-rep WHERE surname='Serote' and
first-name='Mike';
+-----+-----+-----+
| employee-number | surname | first-name | commission |
+-----+-----+-----+
|                3 | Serote  | Mike       |           10 |
+-----+-----+-----+
```

Pero, ¿qué ocurriría si ha olvidado como se escribe *Serote*? ¿Era *Serotte* o *Serota*? Es posible que necesite probar varias consultas antes de lograr el resultado deseado o, si no logra acordarse de como se escribe correctamente, puede que nunca lo consiga. Puede probar simplemente con *Mike*, pero recuerde que es posible que la base de datos conste de miles de registros. Afortunadamente, existe un método mejor de solucionar este problema. MySQL permite utilizar la instrucción LIKE. Si se acuerda de que el apellido comienza por Sero, puede utilizar la siguiente secuencia:

```
mysql> SELECT * FROM sales-rep WHERE surname LIKE 'Sero%';
+-----+-----+-----+
| employee-number | surname | first-name | commission |
+-----+-----+-----+
|                3 | Serote  | Mike       |           10 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Fijese en el simbolo 8. Se trata de un comodin parecido al simbolo *, pero específicamente diseñado para su uso dentro de la condición SELECT. Significa 0 o mas caracteres.

Por lo tanto, esta instruccion devolvera todas las permutaciones consideradas anteriormente. Puede utilizar el comodin cuantas veces desee, como en el siguiente ejemplo:

```
mysql> SELECT * FROM sales-rep WHERE surname LIKE '%e%';
+-----+-----+-----+
| employee-number | surname | first-name | commission |
+-----+-----+-----+
|                1 | Rive   | Sol        |           10 |
|                2 | Gordimer | Charlene  |           15 |
|                3 | Serote  | Mike       |           10 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

En este caso se recuperan todos los registros, porque se estan buscando todos los nombres que contengan una e.

Este resultado es diferente al de la siguiente consulta en la que sólo se buscan apellidos que comiencen por una *e* :

```
mysql> SELECT * FROM sales_rep WHERE surname LIKE 'e%';  
Empty set (0.00 sec)
```

También puede utilizar una consulta como la siguiente, en la que se buscan apellidos que contengan una *e* en alguna parte de su nombre y que terminen en una *e* :

```
mysql> SELECT * FROM sales_rep WHERE surname LIKE '%ete';  
+-----+-----+-----+-----+  
| employee-number | surname | first-name | commission |  
+-----+-----+-----+-----+  
|                | 3 | Serote   | Mike       |          10 |  
+-----+-----+-----+-----+
```

A continuación agregaremos unos cuantos registros más a la tabla para poder probar consultas más complejas. Agregue los siguientes dos registros.

```
mysql> INSERT INTO sales_rep values (4,'Rive' , 'Mongane' ,10);  
mysql> INSERT INTO sales_rep values (5,'Smith' , 'Mike' ,12);
```

Ordenación

Existe otra cláusula útil y de uso habitual que permite la ordenación de los resultados. Una lista alfabética de empleados resulta de utilidad y puede recurrir a la cláusula `ORDER BY` para generarla.

```
mysql> SELECT * FROM sales_rep ORDER BY surname;  
+-----+-----+-----+-----+  
| employee-number | surname | first-name | commission |  
+-----+-----+-----+-----+  
|                | 2 | Gordimer | Charlene   |          15 |  
|                | 1 | Rive     | Sol        |          10 |  
|                | 4 | Rive     | Mongane    |          10 |  
|                | 3 | Serote   | Mike       |          10 |  
|                | 5 | Smith    | Mike       |          12 |  
+-----+-----+-----+-----+  
5 rows in set (0.00 sec)
```

Como habrá observado, la lista no es correcta si desea ordenarla por nombres ya que Sol Rive aparecen antes que Mongane Rive. Para corregir este problema, necesitara ordenar la lista por los nombres cuando los apellidos de dos registros coincidan. Para ello, utilice la siguiente instrucción:

```
mysql> SELECT * FROM sales_rep ORDER BY surname,first_name;  
+-----+-----+-----+-----+  
| employee-number | surname | first-name | commission |  
+-----+-----+-----+-----+  
|                | 2 | Gordimer | Charlene   |          15 |  
|                | 4 | Rive     | Mongane    |          10 |
```

```

|          1 | Rive      | Sol      |          10 |
|          3 | Serote    | Mike     |          10 |
|          5 | Smith     | Mike     |          12 |
+-----+-----+-----+
5 rows in set (0.00 sec)

```

Ahora el pedido es correcto. Para ordenar la lista de registros de **forma inversa** (en **orden descendente**), se utiliza la **palabra clave DESC**.

La siguiente **consulta** devuelve todos los registros segun la comision asignada, de mayor a **menor**:

```

mysql> SELECT * FROM sales-rep ORDER BY commission DESC;
+-----+-----+-----+
| employee-number | surname | first-name | commission |
+-----+-----+-----+
|          2 | Gordimer | Charlene   |          15 |
|          5 | Smith    | Mike       |          12 |
|          1 | Rive     | Sol        |          10 |
|          4 | Rive     | Mongane    |          10 |
|          3 | Serote   | Mike       |          10 |
+-----+-----+-----+
5 rows in set (0.00 sec)

```

De nuevo, puede ordenar los registros de los tres empleados que **tienen** asignado un 10 por ciento de comision.

Para **ello**, puede utilizar la **palabra clave ASC**. Aunque no resulta estrictamente necesario porque se trata del **orden** aplicado de **manera** predeterminada, el uso de esta **palabra clave** aporta mayor claridad:

```

mysql> SELECT * FROM sales-rep ORDER BY commission DESC,
surname ASC,first_name ASC;
+-----+-----+-----+
| employee-number | surname | first-name | commission |
+-----+-----+-----+
|          2 | Gordimer | Charlene   |          15 |
|          5 | Smith    | Mike       |          12 |
|          4 | Rive     | Mongane    |          10 |
|          1 | Rive     | Sol        |          10 |
|          3 | Serote   | Mike       |          10 |
+-----+-----+-----+
5 rows in set (0.01 sec)

```

Limitación del número de resultados

Hasta el momento siempre se ha obtenido el numero **completo** de resultados que satisfacen las condiciones establecidas. Sin embargo, en una base de datos real, puede que se trate de varios miles de registros y que no queramos verlos todos a la vez. Para **ello**, MySQL **permite** utilizar la clausula **LIMIT**. Se trata de una clausula **no convencional** de SQL que, por **tanto**, no podra utilizar de la misma **forma** en todas las bases de datos, **pero** resulta de gran potencia y utilidad en MySQL.

Si sólo desea buscar el empleado con la mayor comisión (suponiendo que solo sea uno, como en nuestro conjunto de datos de ejemplo), puede ejecutar la siguiente consulta:

```
mysql> SELECT first_name,surname,commission FROM sales-rep
ORDER BY commission DESC;
+-----+-----+-----+
| first-name | surname | commission |
+-----+-----+-----+
| Charlene   | Gordimer |          15 |
| Mike       | Smith    |          12 |
| Sol        | Rive     |          10 |
| Mike       | Serote   |          10 |
| Mongane    | Rive     |          10 |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

El empleado que estamos buscando es Charlene Gordimer. Sin embargo, LIMIT permite devolver únicamente dicho registro, de la siguiente manera:

```
mysql> SELECT first_name,surname,commission FROM sales-rep
ORDER BY commission DESC LIMIT 1;
+-----+-----+-----+
| first-name | surname | commission |
+-----+-----+-----+
| Charlene   | Gordimer |          15 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Si se incluye un solo número detrás de la cláusula LIMIT, éste determinará el número de filas que se devuelven.

NOTA: LIMIT 0 no devuelve registros. Puede que no le parezca un comando de gran utilidad, pero es una buena forma de probar una consulta en bases de datos de gran tamaño sin ejecutarlas.

La cláusula LIMIT no solo permite devolver un número limitado de registros a partir de la parte superior o inferior de la base de datos. También puede establecer el *desplazamiento* que utilizar, es decir desde que resultado comenzar la operación de limitación. Si se incluyen dos números tras la cláusula LIMIT, el primero es el desplazamiento y el segundo es el límite de fila. El siguiente ejemplo devuelve el segundo registro, en orden descendente.

```
mysql> SELECT first_name,surname,commission FROM
sales-rep ORDER BY commission DESC LIMIT 1,1;
+-----+-----+-----+
| first-name | surname | commission |
+-----+-----+-----+
| Mike       | Smith    |          12 |
+-----+-----+-----+
```

```
+-----+-----+-----+
1 row in set (0.00 sec)
```

El desplazamiento predeterminado es 0 (los ordenadores empiezan a contar siempre en el 0) por lo que si se especifica un desplazamiento de 1, la búsqueda comenzara en el segundo registro. Para comprobarlo, ejecute la siguiente consulta:

```
mysql> SELECT first-name,surname,commission FROM sales-rep
ORDER BY commission DESC LIMIT 0,1;
+-----+-----+-----+
| first-name | surname | commission |
+-----+-----+-----+
| Charlene  | Gordimer |          15 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

LIMIT 1 equivale a utilizar LIMIT 0, 1, ya que se asume 0 como valor predeterminado si no se especifica nada.

Pero ¿cómo recuperar el tercer, el cuarto y el quinto registro en orden descendente por el campo de comisión?

```
mysql> SELECT first name,surname,commission FROM sales-rep
ORDER BY commission DESC LIMIT 2,3;
+-----+-----+-----+
| first_name | surname | commission |
+-----+-----+-----+
| Sol        | Rive    |          10 |
| Mike       | Serote  |          10 |
| Monqane    | Rive    |          10 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

El desplazamiento es 2 (recuerde que el desplazamiento empieza en 0, por lo que 2 es el tercer registro) y el 3 es el número de registros que recuperar.

NOTA: El comando LIMIT se suele utilizar en motores de búsqueda que ejecuten MySQL, por ejemplo, para 10 resultados por página. Los resultados de la primera página utilizarán LIMIT 0,10, los de la segunda utilizarán LIMIT 10, 10, etc.

Devolución del valor máximo con MAX()

MySQL consta de una gran cantidad de funciones que permiten ajustar las consultas. No es necesario aprenderse todas ellas. Nadie se las sabe. Pero una vez conocida su existencia puede intentar averiguar si existe una para realizar una tarea deseada. Con el tiempo descubriremos que ha inmemorizado las más habituales. Además, este método exige mucho menos esfuerzo que aprenderlas de

memoria. La primera función que vamos a analizar es la función `MAX()`. Utilizaremos esta función para recuperar la comisión más alta asignada a un comercial:

```
mysql> SELECT MAX(commission) from sales-rep;
+-----+
| MAX(commission) |
+-----+
|                15 |
+-----+
1 row in set (0.00 sec)
```

Fíjese en los paréntesis al utilizar las funciones. Las funciones se aplican a todos los elementos incluidos en su interior. A lo largo de este libro, se utilizan los paréntesis para indicar que se trata de una función y como utilizarla; por ejemplo, `MAX()`.

ADVERTENCIA: Preste atención al uso de espacios en las consultas. En la mayor parte de los casos, su uso no ocasionará ningún error, pero al trabajar con funciones (las funciones se suelen identificar por el hecho de que necesitan incluir elementos entre paréntesis), debe prestar especial atención. Si ha colocado un espacio tras la palabra `COUNT`, MySQL devolverá un error de sintaxis.

Recuperación de registros distintos

Es posible que no desee obtener resultados duplicados. Examine la siguiente consulta:

```
mysql> SELECT surname FROM sales-rep ORDER BY surname;
+-----+
| surname |
+-----+
| Gordimer |
| Rive    |
| Rive    |
| Serote  |
| Smith   |
+-----+
5 rows in set (0.00 sec)
```

Esta consulta es correcta, pero puede que no desee recuperar apellidos repetidos, como en el caso de Rive en los registros correspondientes a los nombres Mongane y Sol, sino solamente una vez. La solución consiste en utilizar la instrucción `DISTINCT`, de la siguiente forma:

```
mysql> SELECT DISTINCT surname FROM sales-rep ORDER BY surname;
+-----+
| surname |
```



```

+-----+
| Gordimer |
| Rive     |
| Serote   |
| Smith    |
+-----+
4 rows in set (0.00 sec)

```

Como contar

Como puede observar por los resultados de los ejemplos utilizados hasta ahora, MySQL muestra el numero de filas, como `4 rows in set`. En ocasiones, sólo necesitaremos saber el numero de resultados y no los contenidos de los registros. Para ello se utilizará la función `COUNT()`.

```

mysql> SELECT COUNT(surname) FROM sales-rep;
+-----+
| COUNT(surname) |
+-----+|          5 |
+-----+
1 row in set (0.01 sec)

```

No importa demasiado el campo que se cuente en el ejemplo anterior, ya que la tabla consta de tantos apellidos como nombres. Obtendriamos el mismo resultado si realizaramos la siguiente consulta:

```

mysql> SELECT COUNT(*) FROM sales-rep;
+-----+
| COUNT(*) |
+-----+
|          5 |
+-----+
1 row in set (0.00 sec)

```

Para contar el numero de apellidos distintos que contiene la tabla, se combinan las instrucciones `COUNT()` y `DISTINCT`, de la forma siguiente:

```

mysql> SELECT COUNT(DISTINCT surname) FROM sales-rep;
+-----+
| COUNT(DISTINCT surname) |
+-----+
|          4 |
+-----+
1 row in set (0.00 sec)

```

Como recuperar la media, el minimo y el total con `AVG()`, `MIN()` y `SUM()`

Estas funciones se utilizan de la misma forma que `MAX()`. Dentro de los parntesis se incluye el campo con el que se desee trabajar. Por ejemplo, para obtener la comision media, se utiliza la siguiente consulta:

```

mysql> SELECT AVG(commission) FROM sales-rep;

```

```

+-----+
| AVG(commission) |
+-----+|          11.4000 |
+-----+
1 row in set (0.00 sec)

```

Y para descubrir la comision mas baja asignada a los comerciales, se utiliza la siguiente consulta:

```

mysql> SELECT MIN(commission) FROM sales-rep;
+-----+
| MIN(commission) |
+-----+
|          10 |
+-----+
1 row in set (0.00 sec)

```

SUM() funciona de **manera** similar. No es muy probable que le encuentre un uso a la **operación** de hallar el total de las comisiones como se muestra en el ejemplo, **pero** le ayudara a hacerse una idea de su funcionamiento.

```

mysql> SELECT SUM(commission) from sales-rep;
+-----+
| SUM(commission) |
+-----+
|          57 |
+-----+
1 row in set (0.00 sec)

```

Realización de calculos en una consulta

SQL le **permite** realizar calculos en las consultas. Examine la siguiente **instruccion** como ejemplo:

```

mysql> SELECT 1+1;
+---+
| 1+1 |
+---+
|    2 |
+---+
1 row in set (0.00 sec)

```

Obviamente Csta no es la razon mas importante para utilizar MySQL. No hay peligro de que las escuelas **adopten** MySQL para que lo utilicen los **alumnos** en los exámenes de matematicas. Sin embargo, la posibilidad de realizar calculos dentro de una consulta resulta util. Por ejemplo, utilice la siguiente **instrucción** si desea saber la comision que se llevaran los comerciales si se **incrementa** su porcentaje en un uno por ciento:

```

mysql> SELECT first-name,surname,commission + 1 FROM sales-rep;
+-----+-----+-----+
| first-name | surname | commission + 1 |

```

Sol	Rive	11
Charlene	Gordimer	16
Mike	Serote	11
Mongane	Rive	11
Mike	Smith	12

5 rows in set (0.00 sec)

Eliminación de registros

Para eliminar un registro, MySQL utiliza la instrucción DELETE. Esta instrucción es parecida a la instrucción SELECT, con la salvedad de que como se elimina el registro completo no es necesario especificar ninguna columna. Tan sólo necesitamos indicar el nombre de la tabla y la condición. Por ejemplo, si Mike Smith se despide, se utilizaría la siguiente instrucción para eliminarlo de la tabla:

```
mysql> DELETE FROM sales_rep WHERE employee_number = 5;
Query OK, 1 row affected (0.00 sec)
```

También podemos utilizar el nombre y el apellido como condición para eliminar registros, y en este caso también funcionaría. Sin embargo, en las bases de datos del mundo real, se utiliza un campo exclusivo para identificar a la persona correcta. En un capítulo posterior se abordará el tema de los índices. Por el momento, recuerde que el campo exclusivo es `employee_number` y es conveniente utilizarlo (Mike Smith es un nombre bastante común). En las secciones dedicadas a los índices estableceremos el campo `employee_number` como exclusivo dentro de la estructura de la base de datos.

ADVERTENCIA: Recuerde utilizar condiciones con la instrucción DELETE. Si introduce la instrucción `DELETE FROM sales_rep;` sin más, se eliminarán todos los registros de la tabla. No existe una opción para deshacer esta acción y, como todavía no hemos explicado cómo realizar una copia de seguridad de los datos, la situación resultará &&.

Como cambiar los registros de una tabla

Ya se ha explicado como agregar registros utilizando la instrucción INSERT, como eliminarlos utilizando DELETE y como recuperarlos utilizando SELECT. Todo lo que nos queda por aprender es como modificar los registros existentes. Supongamos que Sol Rive ha vendido un cargamento inmenso de arena a los habitantes del desierto de Namibia y que en recompensa se le ha aumentado su comisión a un 12 por ciento.

Para reflejar correctamente esta nueva circunstancia, se utiliza la instrucción UPDATE de la siguiente forma:

```
mysql> UPDATE sales-rep SET commission = 12 WHERE
employee_number=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

ADVERTENCIA: Tenga cuidado de nuevo al aplicar una condición. Sin la cláusula WHERE, actualizará la comisión de todos los comerciales a un 12 por ciento.

INSERT, SELECT, UPDATE y DELETE constituyen las cuatro instrucciones de uso más habitual para manipular datos. Estas cuatro instrucciones forman parte del Lenguaje de manipulación de datos (DML) de SQL. Con ellas, dispondrá de toda la munición necesaria para modificar los datos de sus registros. En el siguiente capítulo se examinarán consultas más avanzadas.

Eliminación de tablas y bases de datos

También existen instrucciones para definir la estructura de los datos y estas forman parte del Lenguaje de definición de datos de SQL (DDL). Ya hemos visto una (la instrucción CREATE) que se utiliza para crear bases de datos y, tras ello, las tablas y las estructuras dentro de las bases de datos. Como en el caso de los datos, también puede eliminar o modificar las tablas. A continuación, crearemos una tabla y la eliminaremos:

```
mysql> CREATE TABLE commission (id INT);
Query OK, 0 rows affected (0.01 sec)
mysql> DROP TABLE commission;
Query OK, 0 rows affected (0.00 sec)
```

ADVERTENCIA: La tabla y todos sus datos desaparecerán sin ningún aviso ni notificación. Por lo tanto, tenga cuidado con esta instrucción.

Puede hacer lo mismo con una base de datos:

```
mysql> CREATE DATABASE shortlived;
Query OK, 1 row affected (0.01 sec)
mysql> DROP DATABASE shortlived;
Query OK, 0 rows affected (0.00 sec)
```

Ya se habrá hecho una idea de por qué resultan tan importantes los permisos. Si concede a todo el mundo un poder semejante, el resultado puede ser desastroso. En un capítulo posterior se explica cómo evitar catástrofes de este tipo.

Como modificar la estructura de la tabla

La última instrucción DDL, ALTER, permite cambiar la estructura de las tablas. Puede agregar columnas, modificar definiciones, cambiar el nombre de las tablas y eliminar columnas.

Cómo agregar una columna

Suponga que necesita crear una columna en la tabla `sales_rep` para almacenar la fecha en la que contrató a los comerciales. UPDATE no serviría, ya que esta instrucción sólo modifica los datos, no la estructura. Para realizar este cambio, es necesario utilizar la instrucción ALTER:

```
mysql> ALTER TABLE sales_rep ADD date_joined DATE;
Query OK, 4 rows affected (0.01 sec)
Records: 4 Duplicates: 0 Warnings: 0
```

TRUCO: DATE es un tipo de columna que almacena datos en formato año-mes-día (AAAA-MM-DD). Si está acostumbrado a introducir las fechas de otras formas, como en el formato estadounidense (MM/DD/AAAA), necesitará realizar una serie de ajustes:

Pero además, se nos pide otro requisito. (Aunque la mayor parte de los cambios resultan fáciles de realizar, es aconsejable determinar el diseño de la base de datos correctamente desde el principio ya que algunos cambios pueden tener consecuencias poco deseables. En un capítulo posterior, se aborda el tema del diseño de base de datos.) En concreto, se nos pide que almacenemos el año de nacimiento de los comerciales para poder analizar la distribución de edad de la plantilla. Para ello, puede utilizar el tipo de columna YEAR que incluye MySQL. Agregue la siguiente columna:

```
mysql> ALTER TABLE sales_rep ADD year_born YEAR;
Query OK, 4 rows affected (0.02 sec)
Records: 4 Duplicates: 0 Warnings: 0
```

Modificación de una definición de columna

Pero segundos después de agregar el año, a nuestro jefe se le ocurre una idea mejor. ¿Por qué no almacenar la fecha de nacimiento completa de los comerciales? De esta forma se seguirá almacenando el año, pero además la compañía podrá sorprender a sus comerciales con un regalo por sus cumpleaños. Utilice la siguiente secuencia para modificar la definición de columna:

```
mysql> ALTER TABLE sales_rep CHANGE year_born birthday
DATE;
Query OK, 4 rows affected (0.03 sec)
Records: 4 Duplicates: 0 Warnings: 0
```

Tras la cláusula `CHANGE` se incluye el nombre de la antigua columna seguido del nombre de la nueva columna y de su **definición**. Para cambiar la definición, pero no el nombre de la columna, basta con mantener el nombre anterior, como se indica a continuación:

```
mysql> ALTER TABLE nombre_de_tabla CHANGE antiguo_nombre
antiguo_nombre nueva_definición_de_columna;
```

También puede utilizar la cláusula `MODIFY`, sin que resulte necesario repetir el nombre, de la siguiente forma:

```
mysql> ALTER TABLE nombre_de_tabla MODIFY antiguo_nombre
nueva_definición_de_columna;
```

Como cambiar el nombre de una columna

Una mañana a su jefe deja de gustarle el nombre utilizado para designar a los comerciales y le pide que sustituya *sales rep* por *cash-flow enhancers* y que se añada una nueva columna para recoger el valor de la contribución de los comerciales al bienestar de la empresa. Para complacerle, decidimos agregar un nuevo campo en primer lugar:

```
mysql> ALTER TABLE sales_rep ADD enhancement_value int;
Query OK, 4 rows affected (0.05 sec)
Records: 4 Duplicates: 0 Warnings: 0
```

y, a continuación, cambiamos el nombre de la tabla. Para ello, utilizamos la instrucción `RENAME` dentro de la instrucción `ALTER` de la siguiente forma:

```
mysql> ALTER TABLE sales_rep RENAME cash_flow_specialist;
Query OK, 4 rows affected (0.00 sec)
Records: 4 Duplicates: 0 Warnings: 0
```

Al día siguiente, su jefe aparece un poco avergonzado de la decisión tomada el día anterior y decidimos cambiar el nombre de la tabla y eliminar la nueva columna, antes de nadie lo note:

```
mysql> ALTER TABLE cash_flow_specialist RENAME TO
sales_rep;
Query OK, 4 rows affected (0.00 sec)
Records: 4 Duplicates: 0 Warnings: 0
```

NOTA: Observe la diferencia entre las instrucciones `ALTER NAME`; tras la segunda instrucción `RENAME` se ha introducido `TO`. Ambas instrucciones son idénticas en cuanto a su función. Existen varios casos en los que **MySQL** dispone de más de una forma de realizar una acción. De hecho, podemos cambiar el nombre de una tabla de otra forma con la instrucción `RENAME antiguo_nombre_de_tabla TO nuevo_nombre_de_tabla`. La función de estas opciones es proporcionar compatibilidad con otras bases de datos o con el estándar `SQL ANSI`.

Como eliminar una columna

Para eliminar la columna `enhancement_value`, utilizaremos la instrucción `ALTER . . . DROP` de la siguiente forma:

```
mysql> ALTER TABLE sales-rep DROP enhancement-value;
Query OK, 4 rows affected (0.06 sec)Records: 4 Duplicates: 0
Warnings: 0
```

Uso de las funciones de fecha

Tras agregar un par de columnas de fecha, vamos a examinar algunas funciones de fecha de MySQL. La estructura de la tabla presenta este aspecto:

```
mysql> DESCRIBE sales-rep;
+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| employee-number | int(11)       | YES  |     | NULL    |      |
| surname         | varchar(40)   | YES  |     | NULL    |      |
| first-name      | varchar(30)   | YES  |     | NULL    |      |
| commission      | tinyint(4)    | YES  |     | NULL    |      |
| date-joined     | date          | YES  |     | NULL    |      |
| birthday        | date          | YES  |     | NULL    |      |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

Si realizamos una consulta que devuelva los valores `date_joined` y `birthday`, obtendremos los siguientes valores:

```
mysql> SELECT date_joined,birthday FROM sales-rep;
+-----+-----+
| date-joined | birthday |
+-----+-----+
| NULL        | NULL     |
| NULL        | NULL     |
| NULL        | NULL     |
| NULL        | NULL     |
+-----+-----+
4 rows in set (0.00 sec)
```

Los valores NULL indican que nunca se ha introducido nada en estos campos. Habrá observado los encabezados Null que se devuelven al describir una tabla. La opción predeterminada es YES, que permite que el campo este vacío. Sin embargo, puede que necesitemos especificar que el campo no contenga un valor NULL (en un capítulo posterior aprenderemos a hacerlo). El uso de valores NULL suele afectar a los resultados de las consultas y tienen sus particularidades, que se analizarán en capítulos posteriores. Para estar seguro de no utilizar valores NULL, actualice los registros de los comerciales de la siguiente forma:

```
mysql> UPDATE sales-rep SET date-joined =
```

```

'2000-02-15', birthday='1976-03-18'
WHERE employee_number=1;
mysql> UPDATE sales-rep SET date-joined =
'1998-07-09', birthday='1958-11-30'
WHERE employee_number=2;
mysql> UPDATE sales-rep SET date-joined =
2001-05-14', birthday='1971-06-18'
WHERE employee_number=3;
mysql> UPDATE sales-rep SET date-joined =
'2002-11-23', birthday='1982-01-04'
WHERE employee_number=4;

```

Existe una gran **cantidad** de Ctiles funciones de fecha. Aqui **sólo** se muestran un pequeño conjunto. En capítulos posteriores encontrara mas **información sobre** las funciones de fecha.

Como especificar el formato de fecha

MySQL permite devolver las fechas en un **formato** especial, en lugar de utilizar el **formato** estandar AAAA-MM-DD. Para devolver **los cumpleaños** de toda la **plantilla** en **formato MM/DD/AAAA**, utilice la **función DATE_FORMAT()**, de la siguiente **forma**:

```

mysql> SELECT DATE_FORMAT(date_joined, '%m/%d/%Y')
FROM sales-rep WHERE employee_number=1;
+-----+
| date-format(date-joined, '%m/%d/%Y') |
+-----+
| 02/15/2000 |
+-----+

```

La **parte** incluida entre **comillas** simples tras la **columna date_joined** se denomina cadena de **formato**. Dentro de la **función** se utiliza un **especificador** para establecer el **formato** exacto deseado. %m devuelve el mes (01-12), %d devuelve el día (01-31) y %y devuelve el año en **formato** de cuatro dígitos. Existe una gran **cantidad** de **especificadores** (en un capítulo posterior se suministra la lista completa). A **continuación**, exarminaremos algunos ejemplos:

```

mysql> SELECT DATE_FORMAT(date-joined, '%W %M %e %y')
FROM sales-rep WHERE employee_number=1;
+-----+
| DATE_FORMAT(date_joined, '%W %M %e %y') |
+-----+
| Tuesday February 15 00 |
+-----+

```

%w devuelve el nombre del día de la semana, %M devuelve el nombre del mes, %e devuelve el día (1-31) y %y devuelve el año en **formato** de dos dígitos. Fíjese en que %d también devuelve el día (01-31), **pero** es diferente a %e ya que incluye ceros a la izquierda.

En la siguiente consulta, %a es el nombre del día de la semana en formato abreviado, %D es el día del mes con el sufijo adjunto, %b es el nombre del mes en formato abreviado y %Y es el año en formato de cuatro dígitos:

```
mysql> SELECT DATE_FORMAT(date_joined, '%a %D %b, %Y')
FROM sales-rep WHERE employee_number=1;
+-----+
| DATE-FORMAT(date_joined, '%a %D %b, %Y') |
+-----+
| Tue 15th Feb, 2000 |
```

NOTA: Puede agregar cualquier carácter deseado a la cadena de formato. En los ejemplos anteriores se ha utilizado una barra invertida (/) y una coma (,). Puede agregar cualquier secuencia de texto deseada para aplicar formato a la fecha si lo desea.

Recuperación de la fecha y la hora actual

Para determinar la fecha actual, según el servidor, puede utilizar la función CURRENT_DATE(). También existe otra función, NOW(), que devuelve la hora:

```
mysql> SELECT NOW(), CURRENT_DATE();
+-----+-----+
| NOW() | CURRENT_DATE() |
+-----+-----+
| 2002-04-07 18:32:31 | 2002-04-07 |
+-----+-----+
1 row in set (0.00 sec)
```

NOTA: Now() devuelve la fecha y la hora. Existe un tipo de columna llamado DATETIME que permite almacenar datos en el mismo formato (AAAA-MM-DD HH:MM:SS) en nuestras tablas.

Puede aplicar otras convenciones al campo birthday al recuperar los datos. Si le preocupa no poder recuperar el año por haber sustituido el campo del año por la fecha de nacimiento, puede utilizar la función YEAR() de la siguiente forma:

```
mysql> SELECT YEAR(birthday) FROM sales-rep;
+-----+
| YEAR(birthday) |
+-----+
| 1976 |
| 1958 |
| 1982 |
| 1971 |
```

```
-----+
4 rows in set (0.00 sec)
```

MySQL incluye otras **funciones** para recuperar una parte específica de la fecha, como `MONTH()` y `DAYOFMONTH()`:

```
mysql> SELECT MONTH(birthday),DAYOFMONTH(birthday) FROM sales_rep;
+-----+-----+
| MONTH(birthday) | DAYOFMONTH(birthday) |
+-----+-----+
|                3 |                    18 |
|                11 |                    30 |
|                 1 |                     4 |
|                 6 |                    18 |
+-----+-----+
4 rows in set (0.00 sec)
```

Creación de consultas mas avanzadas

Llegados a este **punto** de la **explicación**, debería sentirse cómodo trabajando con las consultas **básicas**. En el mundo real, la mayor parte de las consultas suelen resultar **bastante** simples, como las realizadas hasta ahora. **Además**, cuanto mejor diseñadas estén sus bases de datos, más sencillas resultaran las consultas. Sin embargo, **existen** situaciones en la que necesitara más (el **caso** más habitual es la unión de dos o más tablas; este **tipo** de consulta se denomina *combinación*).

Como aplicar un nuevo encabezado a una columna con AS

Las consultas anteriores no resultaban muy sencillas de leer o de entender. A **continuación**, modificaremos la consulta anterior ordenando los valores devueltos por los meses e incluyendo los nombres de los comerciales en los resultados. También se introducen alias con la **palabra** clave `AS` para asignar otro nombre a una columna:

```
mysql> SELECT surname,first_name,MONTH(birthday)
AS month,DAYOFMONTH(birthday) AS day FROM sales_rep
ORDER BY month;
+-----+-----+-----+-----+
| surname | first-name | month | day |
+-----+-----+-----+-----+
| Rive    | Mongane   | 1     | 4   |
| Rive    | Sol       | 3     | 18  |
| Serote  | Mike      | 6     | 18  |
| Gordimer | Charlene  | 11    | 30  |
+-----+-----+-----+-----+
4 rows in set (0.01 sec)
```

Combinación de columnas con CONCAT

En ocasiones puede que desee mostrar el nombre de la persona en un solo campo de resultados, en lugar de separar el nombre y el apellido en dos campos.

Puede combinar los resultados de las columnas, utilizando la función CONCAT () (que equivale a concatenar), de la siguiente forma:

```
mysql> SELECT CONCAT(first-name, ' ',surname)
AS name,MONTH(birthday) AS month,DAYOFMONTH(birthday)
AS day FROM sales-rep ORDER BY month;
+-----+-----+
| name | month | day |
+-----+-----+
| Mongane Rive | 1 | 4 |
| Sol Rive | 3 | 18 |
| Mike Serote | 6 | 18 |
| Charlene Gordimer | 11 | 30 |
+-----+-----+
4 rows in set (0.00 sec)
```

NOTA: Fijese en el espacio utilizado dentro de CONCAT (). Como en el caso de los especificadores de fecha, puede utilizar cualquier carácter para dar formato a CONCAT ().

Como buscar el día del año

Para buscar el día del año (de 1 al 366) en el que Sol Rive se unió a la compañía, utilice la siguiente secuencia:

```
mysql> SELECT DAYOFYEAR(date-joined) FROM sales-rep
WHERE employee_number=1;
+-----+
| DAYOFYEAR(date-joined) |
+-----+
| 46 |
+-----+
```

Como trabajar con varias tablas

El verdadero potencial de las bases de datos relacionales reside en la posibilidad de establecer relaciones entre las tablas.

Hasta ahora solo hemos trabajado con una tabla para familiarizarnos con la sintaxis de SQL. La mayor parte de las aplicaciones del mundo real constan de varias tablas, por lo que necesitaremos aprender a trabajar en estas situaciones.

En primer lugar, vamos a agregar dos nuevas tablas a la base de datos. La tabla 1.5 contendrá los datos de los clientes (un identificador de cliente, un nombre y un apellido) y la tabla 1.6 contendrá los datos de venta (un identificador de cliente, un identificador de comercial, el valor de las ventas en dolares y un código exclusivo para la venta).

Tabla 1.5. La tabla Customer

ID	FIRST NAME	SURNAME
1	Yvonne	Clegg
2	Johnny	Chaka-Chaka
3	Winston	Powers
4	Patricia	Mankunku

Tabla 1.6. La tala Sales

CODE	SALES_REP	CUSTOMER	VALUE
1	1	1	2000
2	4	3	250
3	2	3	500
4	1	4	450
5	3	1	3800
6	1	2	500

¿Puede crear estas tablas? A continuación se incluyen las instrucciones usadas:

```
mysql> CREATE TABLE customer (  
  id int,  
  first-name varchar(30),  
  surname varchar(40)  
);  
Query OK, 0 rows affected (0.00 sec)  
mysql> CREATE TABLE sales (  
  code int,  
  sales-rep int,  
  customer int,  
  value int  
);  
Query OK, 0 rows affected (0.00 sec)  
mysql> INSERT INTO customer(id, first-name, surname) VALUES  
  (1, 'Yvonne', 'Clegg'),  
  (2, 'Johnny', 'Chaka-Chaka'),  
  (3, 'Winston', 'Powers'),  
  (4, 'Patricia', 'Mankunku');  
Query OK, 4 rows affected (0.00 sec)  
Records: 4 Duplicates: 0 Warnings: 0  
mysql> INSERT INTO sales(code, sales-rep, customer, value) VALUES  
  (1, 1, 1, 2000),
```

```
(2,4,3,250),
(3,2,3,500),
(4,1,4,450),
(5,3,1,3800),
(6,1,2,500);
Query OK, 6 rows affected (0.00 sec)
Records: 6 Duplicates: 0 Warnings: 0
```

Combinación de dos o mas tablas

Como puede ver, aqui se utiliza el numero del comercial y el identificador del cliente de la tabla de **ventas**. Si examina el primer registro de **ventas**, observara que se compone de `sales_rep 1`, que, al examinar la tabla `sales_rep`, verá que se corresponde con **Sol Rive**. El proceso manual de examinar la **relación** entre las dos tablas es el mismo que el que realiza MySQL, siempre que se le indique que **relación** utilizar. A **continuación**, escribiremos una consulta que recupere toda la **información** desde el primer registro de **ventas** así como el nombre del representante de **ventas**.

```
mysql> SELECT sales_rep,customer,value,first_name,surname
FROM sales,sales_rep WHERE code=1 AND
sales_rep.employee_number=sales.sales_rep;
+-----+-----+-----+-----+
| sales_rep | customer | value | first_name | surname |
+-----+-----+-----+-----+
| 1 | 1 | 2000 | Sol | Rive |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

La **primera** parte de la consulta, tras el comando `SELECT`, **incluye** los campos que deseamos recuperar. La **operación** resulta **bastante** sencilla ya que **todo** consiste en indicar los campos que deseamos de las dos tablas.

La segunda parte, tras `FROM`, indica a MySQL que tablas utilizar. En este **caso**, son dos: la tabla `sales` y la tabla `sales_rep`.

La tercera parte, tras `WHERE`, contiene la **condición** `code=1`, que devuelve el primer registro de la tabla de **ventas**. La siguiente parte es la **sección** que convierte a esta consulta en un **vínculo**. Éste es el lugar en el que se indica a MySQL que campos vincular o entre que campos se relacionan las tablas. La **relación** entre la tabla `sales` y la tabla `sales_rep` se establece entre el campo `employee_number` de la tabla `sales_rep` y el campo `sales_rep` de la tabla `sales`. Por lo **tanto**, como en el **campo** `sales_rep` aparece un 1, debe **buscar** el **empleado** con dicho numero en la tabla `sales_rep`.

Vamos a **probar** otra consulta. En esta ocasion queremos **recuperar** todas las **ventas** realizadas por Sol Rive (con numero de empleada 1). Vamos a examinar el proceso de pensamiento subyacente a la **construcción** de esta consulta:

- ¿Qué tablas necesitamos? Claramente, la tabla `sales_rep` y la tabla `sales`, las cuales ya **forma** parten de la consulta `FROM sales_rep,sales`.

- ¿Qué campos necesitamos? Necesitamos toda la **información** de **ventas**. Por lo **tanto**, la **lista** de campos se **convierte** en `SELECT code , customer , value .`
- Y **finalmente** ¿cuáles son las condiciones? La **primera** es que **sólo** necesitamos los resultados de Sol Rive y la **segunda** consiste en especificar la **relación** que se establece entre el campo `sales_rep` de la tabla `sales` y el campo `employee_number` de la tabla `sales_rep`. Por lo **tanto**, las condiciones son **las siguientes**: `WHERE first_name='Sol' and surname='Rive' AND sales.sales_rep = sales_rep.employee_number.`

La consulta final presenta este aspecto:

```
mysql> SELECT code,customer,value FROM sales_rep,sales
  WHERE first_name='Sol' AND surname='Rive' AND
    sales.sales_rep = sales_rep.employee_number;
+-----+-----+
| code | customer | value |
+-----+-----+
|  1  |         1 | 2000 |
|  4  |         4 |  450 |
|  6  |         2 |  500 |
+-----+-----+
3 rows in set (0.00 sec)
```

Fíjese en la notación de la **condición** de la relación: `sales.sales_rep` o `sales_rep.employee_number`. Al especificar el nombre de la **tabla**, a **continuación** un **punto** y **después** el nombre del archivo **hace** que las consultas **resulten** mas claras y es el **método** obligatorio cuando se utilizan los mismos **nombres** para identificar tablas diferentes. También puede utilizar esta notación en la lista de campos. Por **ejemplo**, la consulta anterior se puede escribir de la **siguiente** forma:

```
mysql> SELECT code,customer,value FROM sales,
  sales_rep WHERE first_name='Sol' AND surname='Rive'
  AND sales_rep = employee_number;
+-----+-----+
| code | customer | value |
+-----+-----+
|  1  |         1 | 2000 |
|  4  |         4 |  450 |
|  6  |         2 |  500 |
+-----+-----+
3 rows in set (0.00 sec)
```

sin utilizar los nombres de las tablas delante de los nombres de archivo porque los campos de las diferentes tablas utilizan nombres **exclusivos**. También **podríamos** haber escrito la consulta de la siguiente forma:

```
mysql> SELECT sales.code,sales.customer,sales.value
  FROM sales,sales_rep WHERE sales_rep.first_name='Sol'
  AND sales_rep.surname='Rive' AND sales.sales_rep =
```

```

    sales_rep.employee_number;
+-----+-----+
| code | customer | value |
+-----+-----+
|    1 |         1 | 2000 |
|    4 |         4 |  450 |
|    6 |         2 |  500 |
+-----+-----+
3 rows in set (0.00 sec)

```

En ambos casos se obtienen los mismos resultados.

Para mostrar qué ocurre cuando se utilizan nombres de campo iguales, vamos a modificar el campo `sales_rep` de la tabla de `ventas` y vamos a denominarlo `employee_number`. No se inquiete, lo volveremos a modificar antes de que nadie se entere:

```

mysql> ALTER TABLE sales CHANGE sales_rep
      employee_number int;
Query OK, 6 rows affected (0.00 sec)
Records: 6 Duplicates: 0 Warnings: 0

```

A continuación, vamos a intentar realizar de nuevo la unión, una vez corregido el nombre pero sin utilizar el punto para especificar los nombres de las tablas:

```

mysql> SELECT code,customer,value FROM sales_rep,sales
      WHERE first_name='Sol' AND surname='Rive'AND employee_number =
      employee_number;
ERROR 1052: Columnn: 'employee_number' in where clause is ambiguous

```

Leyendo la consulta es probable que se de cuenta de que no resulta clara. Por lo tanto necesitamos utilizar los nombres de las tablas cada vez que hagamos referencia a uno de los campos `employee_number`:

```

mysql> SELECT code,customer,value FROM sales_rep,sales
      WHERE sales_rep.employee_number=1 AND sales_rep.employee_number =
      sales.employee_number;
+-----+-----+
| code | customer | value |
+-----+-----+
|    1 |         1 | 2000 |
|    4 |         4 |  450 |
|    6 |         2 |  500 |
+-----+-----+
3 rows in set (0.00 sec)

```

TRUCO: Podríamos haber utilizado `sales.employee_number` en lugar de `sales_rep.employee_number` dentro de la cláusula `WHERE`, pero es mejor utilizar la tabla más pequeña porque así se reduce el trabajo asignado a MySQL al responder a la consulta. En un capítulo posterior aprenderemos más sobre la optimización de consultas.

Antes de continuar, vamos a cambiar el nombre del campo sustituyendolo por el antiguo:

```
mysql> ALTER TABLE sales CHANGE employee-number sales-rep INT;
Query OK, 6 rows affected (0.00 sec)
Records: 6 Duplicates: 0 Warnings: 0
```

Realización de calculos con fechas

La operacion de realizar calculos con fechas resulta relativamente sencilla. En la siguiente seccion practicaremos con la edad de alguna de las personas en funcion de su fecha de nacimiento, pero en primer lugar vamos a realizar un cálculo mas sencillo. Para determinar el numero de años que median entre la fecha actual y la fecha de nacimiento de una persona, se utilizan las funciones YEAR() y NOW():

```
mysql> SELECT YEAR(NOW()) - YEAR(birthday) FROM sales-rep;
+-----+
| YEAR(NOW()) - YEAR(birthday) |
+-----+
|                               |
|                               |
|                               |
|                               |
+-----+
4 rows in set (0.00 sec)
```

NOTA: También puede utilizar CURRENT_DATE() en lugar de NOW() ya que ambas devuelven el mismo resultado.

La consulta anterior no devuelve la edad, solo la diferencia en años. No tiene en cuenta los días ni los meses. En esta seccion se describe como calcular la edad de una persona, tarea que puede resultar un tanto complicada si no tiene experiencia. Pero no se desanime. Tras practicar con una serie de consultas básicas, le resultara muy sencillo.

Debemos restar los años como hemos hecho anteriormente pero además debemos restar otro año si no ha transcurrido uno entero. Una persona nacida el 10 de diciembre de 2002 no tendrá un año en enero de 2003, sino que tendrá que esperar a diciembre de 2003. Una buena forma de realizar esta operacion consiste en tomar los componentes MM-DD de los dos campos de datos (la fecha actual y la fecha de nacimiento) y compararlos. Si el actual es mayor, habrá transcurrido un año, con lo que puede mantener el cálculo de los años sin modificar. Si la parte MM-DD es menor que la fecha de nacimiento; no habrá transcurrido un año entero y debe restar un año al cálculo de los años. Este proceso puede resultar un tanto complicado y existen algunas formas bastante complejas de realizar los

calculos decimales, pero MySQL facilita la operación porque devuelve 1 si la expresión verdadera y 0 si resulta falsa.

```
mysql> SELECT YEAR(NOW()) > YEAR(birthday) FROM
sales-rep WHERE employee_number=1;
+-----+
| YEAR(NOW()) > YEAR(birthday) |
+-----+
|                               1 |
+-----+
1 row in set (0.00 sec)
mysql> SELECT YEAR(NOW()) < YEAR(birthday) FROM
sales-rep WHERE employee_number=1;
+-----+
| YEAR(NOW()) < YEAR(birthday) |
+-----+
|                               0 |
+-----+
1 row in set (0.00 sec)
```

El año actual es mayor que el año del cumpleaños del empleado 1. Esta afirmación es verdadera y se le asigna el valor 1. El año actual es menor que el año de nacimiento. Esto es falso y se le asigna el valor 0.

A continuación necesitamos una forma rápida de devolver el componente MM-DD de la fecha. Para ello, es aconsejable utilizar la función de cadena RIGHT().

```
mysql> SELECT RIGHT(CURRENT_DATE,5),RIGHT(birthday,5) FROM
sales-rep;
+-----+-----+
| RIGHT(CURRENT_DATE,5) | RIGHT(birthday,5) |
+-----+-----+
| 04-06                | 03-18              |
| 04-06                | 11-30              |
| 04-06                | 01-04              |
| 04-06                | 06-18              |
+-----+-----+
4 rows in set (0.00 sec)
```

El 5 incluido dentro de la función RIGHT() hace referencia al número de caracteres situados a la derecha de la cadena que devuelve la función. La cadena completa es 2002-04-06 y los cinco caracteres situados más a la derecha son 04-06 (incluido el guion). Por lo tanto, ahora ya disponemos de todos los componentes para realizar el cálculo de la fecha:

```
mysql> SELECT surname, first_name, (YEAR(CURRENT_DATE) -
YEAR(birthday)) - (RIGHT(CURRENT_DATE,5) < RIGHT(birthday,5))
AS age FROM sales-rep;
+-----+-----+-----+
| surname | first-name | age |
+-----+-----+-----+
| Rive    | Sol        | 26  |
```

```

| Gordimer | Charlene | 43 |
| Rive     | Mongane  | 20 |
| Serote   | Mike     | 30 |
+-----+-----+
4 rows in set (0.00 sec)

```

Sus resultados puede que no coincidan con estos de manera exacta por el paso del tiempo y es posible que este utilizando una fecha posterior.

ADVERTENCIA: Tenga cuidado con los paréntesis al realizar un cálculo tan complejo. Debe cerrar cada paréntesis que abra en el lugar correcto.

¿Se le ocurre un caso en el que la consulta anterior sobre la edad no funcione? Si el año actual coincide con el año de nacimiento, obtendrá -1 como respuesta. Tras examinar los capítulos posteriores, pruebe a desarrollar una forma propia de calcular la edad. Existen muchas posibilidades, tantas como voces pidiendo a MySQL que desarrolle una función especial.

Agrupación de una consulta

Tras desarrollar una tabla de ventas, vamos a aplicar la función SUM() a un mejor uso que el que le dimos anteriormente para calcular el valor total de las ventas:

```

mysql> SELECT SUM(value) FROM sales;
+-----+
| SUM(value) |
+-----+
|          7500 |
+-----+
1 row in set (0.00 sec)

```

A continuación, queremos calcular las ventas totales de cada comercial. Para realizar esta tarea manualmente, necesitamos agrupar la tabla de ventas en función de los comerciales. Necesitaríamos colocar todas las ventas realizadas por el comercial 1, hallar el total y repetir la misma operación con el comercial número 2. SQL dispone de la instrucción GROUP BY, que MySQL utiliza de la misma forma:

```

mysql> SELECT sales-rep,SUM(value) FROM sales GROUP BY
sales-rep;
+-----+-----+
| sales-rep | SUM(value) |
+-----+-----+
|          1 |          2950 |
|          2 |           500 |
|          3 |          3800 |

```

```

|          4 |          250 |
+-----+-----+

```

Si prueba a realizar la misma consulta sin agrupar las **ventas**, obtendra un error:

```

mysql> SELECT sales-rep,SUM(value) FROM sales;
ERROR 1140: Mixing of GROUP columns
(MIN(),MAX(),COUNT()...) with no GROUP columns
is illegal if there is no GROUP BY clause

```

Esta consulta no tiene mucho **sentido**, ya que intenta combinar un campo de **resumen**, `SUM()`, con un campo normal. ¿Qué esperamos? ¿La suma de todos los valores repetidos junto a cada comercial?

Tambien puede ordenar el resultado de una consulta agrupada. Para **recuperar las ventas totales** de cada comercial desde la mayor a la **menor**, basta con agregar la **instrucción ORDER BY**:

```

mysql> SELECT sales-rep,SUM(value) AS sum FROM sales
GROUP BY sales-rep ORDER BY sum desc;
+-----+-----+
| sales-rep | sum |
+-----+-----+
|          3 | 3800 |
|          1 | 2950 |
|          2 | 500 |
|          4 | 250 |
+-----+-----+

```

A continuacion, vamos a realizar una consulta mas compleja utilizando varios de los **conceptos aprendidos**. Vamos a recuperar el nombre de los comerciales que **hayan** obtenido los **peores resultados de ventas**. En primer lugar, tendremos que devolver un numero de empleado. Puede que obtenga un numero diferente al ejecutar la consulta, ya que hay tres personas que sólo han realizado una venta. No importa el que devuelva por ahora. La consulta presentara este aspecto:

```

mysql> SELECT sales-rep,COUNT(*) as count from sales
GROUP BY sales-rep ORDER BY count LIMIT 1;
+-----+-----+
| sales-rep | count |
+-----+-----+
|          4 |      1 |
+-----+-----+
1 row in set (0.00 sec)

```

¿Puede ir mas alla y establecer un **vínculo** para recuperar el nombre del comercial 4? Si es capaz de realizar esta **operación**, y al comenzar este libro no habia trabajado nunca con bases de datos, esta en muy buen camino para **convertirse en un experto**. A continuacion, se incluye la consulta:

```

mysql> SELECT first_name,surname,sales-rep,COUNT(*) AS

```

```

count from sales,sales-rep WHERE sales_rep=employee_number
GROUP BY sales_rep,first_name,surname ORDER BY count
LIMIT 1;
+-----+-----+-----+-----+
| first-name | surname | sales-rep | count |
+-----+-----+-----+-----+
| Mongane   | Rive    |           | 4     |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

Resumen

MySQL es un sistema de administracion de base de datos relacional. Lógicamente, los datos se estructuran en tablas, que se relacionan entre si por un campo comun. Las tablas se componen de filas (o registros) y los registros se componen de **columnas** (o campos). Los campos pueden ser de diferente tipo: numericos, de cadena o de **tipo** fecha. (Este capitulo se limita a presentar SQL. A lo largo del libro ira desarrollando su habilidad con este lenguaje.)

El servidor MySQL es el lugar en el que se almacenan los datos y sobre él se ejecutan las consultas. Para establecer una **conexión** al servidor MySQL, necesita el cliente MySQL. Éste puede estar instalado en el mismo equipo que el servidor o en un equipo remoto.

El potencial de un sistema de administracion de bases de datos **procede** de su capacidad para estructurar datos y recuperarlos en funcion de una gran **variedad** de requisitos especificos. El estandar de la industria para manipular y definir datos es SQL. Sus comandos mas importantes son los siguientes:

- La instruccion **CREATE** crea bases de datos y tablas dentro de la base de datos.
- La instruccion **INSERT** coloca registros en una tabla.
- La instruccion **SELECT** devuelve los resultados de una columna.
- La instruccion **UPDATE** modifica los datos de una tabla
- La instruccion **ALTER** cambia la estructura de una tabla, utilizando **cláusulas** como **ADD** para agregar una nueva columna, **CHANGE** para cambiar el nombre o **definición** de una columna existente, **RENAME** para cambiar el nombre de una tabla o **DROP** para eliminar una tabla.

Las funciones incrementan el potencial de MySQL. Las funciones se **caracterizan** por ir seguidas de parentesis. MySQL incorpora una gran **cantidad** de funciones (maticas, como **SUM()** para calcular el total de un conjunto, de fecha y hora, como **YEAR()** para extraer la **porción** del año de una fecha, y funciones de cadena, como **RIGHT()** para extraer parte de una cadena que comience por el **lado** derecho de dicha cadena).

Armados con esta **información básica**, podemos abordar temas fundamentales **sobre** la estructuración de datos, continuar con el estudio de elementos más avanzados de SQL y analizar **los** distintos tipos de tablas que utiliza **MySQL** para las diferentes clases de soluciones.

2

Tipos de datos y tipos de tabla

Como ya sabemos, MySQL utiliza varios tipos de tablas. El tipo de tabla predeterminado es MyISAM que esta optimizado para la velocidad del comando `SELECT`.

La mayor parte de los sitios Web utilizan esta tabla, ya que estos sitios suelen utilizar la instrucción `SELECT` mucho mas que las instrucciones `INSERT` o `UPDATE`.

En este capitulo examinaremos los distintos tipos de tablas en detalle. En el capitulo anterior se examinaron brevemente varios tipos de datos.

En este, exploraremos los tipos de datos disponibles y aprenderemos a utilizarlos.

En este capitulo se abordan los siguientes temas:

- Tipos de columna numericos, de cadena y de fecha
- Las opciones de linea de comandos de MySQL
- Operadores lógicos, aritmeticos, comparativos y bit a bit
- Examen de las opciones para establecer conexiones a MySQL
- Estudio de los tipos de tablas

Análisis de los distintos tipos de columna

Para usar MySQL de forma efectiva es importante comprender los distintos bloques de construcción disponibles. Las listas de correo de MySQL estan llenas de peticiones de ayuda y a menudo la solución consiste sencillamente en utilizar otro tipo de columna o de tabla o en realizar un examen mas detenido de sus funciones. En este capitulo, analizaremos en primer lugar los distintos tipos de columna y posteriormente examinaremos los tipos de tablas disponibles en MySQL.

Existen tres tipos fundamentales de columnas en MySQL: numericas, de cadena y de fecha. Aunque existen muchos otros tipos especificos de columna, que no tardaremos en ver, todos ellos se pueden clasificar dentro de los tres tipos mencionados. Por regla general, deberia seleccionar el tipo de columna de menor tamaño, ya que de esta forma se ahorra espacio y se logra una mayor velocidad de acceso y actualización. Sin embargo, si se selecciona un tipo de columna demasiado pequeño puede dar como resultado la perdida de datos o que se recorten al introducirlos. Por lo tanto, hay que escoger el tipo que englobe todos los posibles casos. En la siguiente sección se estudia cada tipo de manera detallada.

NOTA: Los nombres de columna no discriminan nunca entre mayúsculas y minúsculas, por lo que `SELECT campo1 FROM tabla` es igual que `SELECT CamP01 FROM tabla`. Sin embargo, tenga presente que los nombres de tabla y de base de datos si distinguen entre mayúsculas y minúsculas, NO lo hacen de manera predeterminada en Windows, pero si en la mayor parte de las versiones de Unix, a excepción de MacOS X.

Tipos de columna numericos

Las columnas numericas estan diseñadas para almacenar todo tipo de datos numericos, como precios, edades o cantidades. Existen dos tipos principales de tipos numericos: tipos enteros (numeros enteros sin decimales ni partes fraccionales) y tipos de coma flotante.

Todos los tipos numericos permiten dos opciones: UNSIGNED y ZEROFILL. UNSIGNED no permite el uso de numeros negativos (extiende el rango positivo del tipo de los tipos enteros) y ZEROFILL rellena el valor con ceros en lugar de los espacios habituales, además de asignar el tipo UNSIGNED de manera predeterminada. Por ejemplo:

```
mysql> CREATE TABLE test1(id TINYINT ZEROFILL);  
Query OK, 0 rows affected (0.32 sec)
```

```
mysql> INSERT INTO test1 VALUES(3);  
Query OK, 1 row affected (0.16 sec)
```

```
mysql> INSERT INTO test1 VALUES (-1)
```

```
Query OK, 1 row affected (0.16 sec)
```

```
mysql> INSERT INTO test1 VALUES (256)
```

```
Query OK, 1 row affected (0.16 sec)
```

```
mysql> SELECT * from test1;
```

```
+----+ | id |
```

```
+----+
```

```
| 003 |
```

```
| 000 |
```

```
| 255 |
```

```
+----+
```

```
3 rows in set (0.00 sec)
```

Fijásc en que como el campo es UNSIGNED, el número negativo se ajusta para adaptarlo a la parte inferior del rango, y como 256 supera el máximo del rango, se ajusta a 255, el valor máximo permitido.

NOTA: Al realizar una consulta sobre un tipo de columna numérico, no es necesario utilizar comillas para encerrar los valores.

En la tabla 2.1 se recogen los tipos de valores numericos disponibles en MySQL.

Tabla 2.1. Tipos numericos

Tipo	Descripción
TINYINT [(M)] [UNSIGNED] [ZEROFILL]	Un entero pequeño; de -128 a 127 (SIGNED), de 0 a 255 (UNSIGNED); requiere 1 byte de espacio de almacenamiento.
BIT	Sinonimo de TINYINT (1).
BOOL	Otro sinonimo de TINYINT (1).
SMALLINT [(M)] [UNSIGNED] [ZEROFILL]	Un entero pequeño; de -32.768 a 32.767 (SIGNED); de 0 a 65,535 (UNSIGNED); requiere 2 bytes de espacio de almacenamiento.
MEDIUMINT [(M)] [UNSIGNED] [ZEROFILL]	Un entero de tamaño medio, de -8.388.608 a 8.388.607 (SIGNED); de 0 a 16.777.215 (UNSIGNED); requiere 3 bytes de espacio de almacenamiento.
INT [(M)] [UNSIGNED] [ZEROFILL]	Un entero; de -2.147.483.648 a 2.147.483.647 (SIGNED); de 0 a 4.294.967.295 (UNSIGNED); requiere 4 bytes de espacio de almacenamiento.

Tipo	Descripción
INTEGER	Sinonimo de INT
BIGINT[(M)] [UNSIGNED] [ZEROFILL]	Un entero grande; de -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807 (SIGNED); de 0 a 18.446.744.073.709.551.615 (UNSIGNED); requiere 8 bytes de espacio de almacenamiento . En las reglas incluidas tras esta tabla se exponen algunas consideraciones importantes sobre el uso de BIGINT.
FLOAT(precision) [UNSIGNED] [ZEROFILL]	Un numero de coma flotante. Se asigna una precision <=24 a los numeros de coma flotante de precision simple. Una precision de entre 25 y 53 se asigna a los numeros coma flotante de precision doble . FLOAT (x) consta del mismo rango que los tipos FLOAT y DOUBLE correspondiente, pero el tamaño y el número de los decimales no estan definidos . En las versiones de MySQL anteriores a la 3.23, no se trataba de un verdadero valor de coma flotante y siempre llevaba dos decimales . Este hecho puede originar problemas inesperados como que todos los cálculos de MySQL se realicen con precisión doble .
FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]	Un numero decimal pequeño o de precision simple. Su valor oscila entre 3,402823466E+38 y -1,175494351E-38 , 0 y de 1,175494351E-38 a 3,402823466E+38 . Con UNSIGNED , el rango positivo sigue siendo el mismo, pero no se admiten los numeros negativos . M indica el ancho total que se muestra y D indica el numero de decimales . FLOAT sin argumentos o FLOAT (X) , donde X <=24 equivale a un número de coma flotante de precision simple. FLOAT (X) , donde X se situa entre 25 y 53 equivale a un numero de coma flotante de precision simple. Requiere 4 bytes de espacio de almacenamiento (precision simple).
DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]	Un numero de coma flotante de precision doble . Oscila entre -1,7976931348623157E+308 y -2,2250738585072014E-308 , 0 y de 2,2250738585072014E-308 a 1,7976931348623157E+308 . Como en el caso de FLOAT , UNSIGNED no tocara el rango positi-

Tipo	Descripción
DOUBLE [(M, D)] [UNSIGNED] [ZEROFILL] (Cont.)	vo, pero no permitira el uso de numeros negativos. M se utiliza para indicar el ancho total que se muestra y D el numero de decimales . DOUBLE sin argumentos o FLOAT (X), donde $25 \leq X \leq 53$, equivale a un numero de coma flotante de precision double . Requiere 8 bytes de espacio de almacenamiento.
DOUBLE PRECISION [(M, D)] [UNSIGNED] [ZEROFILL]	Sinonimo de DOUBLE.
REAL [(M, D)] [UNSIGNED] [ZEROFILL]	Otro sinonimo de DOUBLE.
DECIMAL [(M[, D])] [UNSIGNED] [ZEROFILL]	Un numero decimal almacenado como una cadena, con un byte de espacio para cada carácter . Oscila entre $-1,7976931348623157E+308$ y $-2,2250738585072014E-308$, 0 y $2,2250738585072014E-308$ a $1,7976931348623157E+308$. M se utiliza para indicar el numero total de digitos (excluyendo el signo y el punto decimal, salvo en las versiones anteriores a la 3.23). D indica el número de decimales. Debe ser siempre inferior al valor de M. El valor predeterminado de D es 0 si se omite. A diferencia de los tipos numericos , M y D pueden limitar el rango de valores permitidos . Con UNSIGNED, los valores negativos no se permiten.
DEC [(M[, D])] [UNSIGNED] [ZEROFILL]	Sinónimo de DECIMAL.
NUMERIC [(M[, D])] [UNSIGNED] [ZEROFILL]	Otro sinonimo de DECIMAL.

Utilice las siguientes directrices a la hora de escoger el tipo numerico:

- **Seleccione** el tipo mas **pequeño** susceptible de aplicacion (TINYINT en lugar de INT si el valor no es mayor a 127 firmado).
- Para numeros enteros, seleccione el tipo entero. (Recuerde que las monedas tambien se pueden almacenar como numeros enteros; por ejemplo, se pueden almacenar en **forma** de centimos en lugar de en unidades con centimos.) Tambien podrian almacenarse como tipo DECIMAL.

- Para los casos en los que se necesite una mayor precisión, utilice los tipos enteros en lugar de los tipos de coma flotante (los errores de redondeo afectan a los números de coma flotante).

El valor M de la tabla 2.1 suele resultar **confuso**. Si se le asigna un valor superior a lo que **admite**, el **tipo** no permitira superar dicho limite. Por ejemplo:

```
mysql> CREATE TABLE test2(id TINYINT(10));
Query OK, 0 rows affected (0.32 sec)

mysql> INSERT INTO test2(id) VALUES(100000000);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT id FROM test2;
+----+
| id  |
+----+
| 127 |
+----+
1 row in set (0.00 sec)
```

Aunque la cifra insertada tiene **menos** de 10 dígitos (dado que se trata de un tipo TINYINT firmado), su valor **positivo** máximo se **limita** a 127.

La especificación del ancho **opcional** rellena de ceros la **representación** de los valores cuyo ancho sea inferior **al** especificado para la columna, sin restringir, con la **excepción** de los campos de tipo DECIMAL, el **rango** de valores que se pueden almacenar en la columna o el número de dígitos que se mostraran para los valores cuyo ancho **supere** el especificado para la columna.

Sin embargo, si intenta restringir un **tipo** a un límite inferior **al** permitido, el valor no se recortara. No se restringira el **rango** que se puede almacenar ni el número de dígitos representados. Por ejemplo:

```
mysql> CREATE TABLE test3(id INT(1));
Query OK, 0 rows affected (0.32 sec)

mysql> INSERT INTO test3(id) VALUES(42432432);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT id FROM test3;
+-----+
| id      |
+-----+
| 42432432 |
+-----+
1 row in set (0.16 sec)
```

La especificación del ancho se suele utilizar con **zerofill** porque resulta sencillo ver los resultados:

```
mysql> CREATE TABLE test4(id INT(3) ZEROFILL,id2 INT ZEROFILL);
```

```
Query OK, 0 rows affected (0.32 sec)
```

```
mysql> INSERT INTO test4(id,id2) VALUES (22,22);  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT * FROM test4;
```

```
+-----+-----+  
| id   | id2      |  
+-----+-----+  
|  22  | 000000022 |  
+-----+-----+  
1 row in set (0.22 sec)
```

El efecto de la especificación del ancho en `id` limita a tres los caracteres que se representan, aunque el campo `id2` utilice un tipo `INT` predeterminado(10) sin firmar.

Tipos de columnas de cadena

Las columnas de cadena se utilizan para almacenar todo tipo de datos compuestos de caracteres como nombres, direcciones o artículos de periódico. La tabla 2.2 describe los tipos de cadena disponibles para MySQL

Tabla 2.2. Tipos de cadena

Tipo	Descripción
[NATIONAL] CHAR(M) [BINARY]	Caracter. Una cadena de longitud fija, con relleno de espacios a la derecha para la longitud especificada. De 0 a 255 caracteres (de 1 a 255 en versiones de MySQL anteriores a la 3.23). Los espacios en blanco se eliminan al recuperar el valor.
CHAR	Sinonimo de CHAR(1).
[NATIONAL] VARCHAR(M) [BINARY]	Caracter de longitud variable. Una cadena de longitud variable, cuyos espacios en blanco se eliminan al almacenar el valor (se trata de un fallo que puede co-ger desprevenido a aquellos lectores acostumbrados a utilizar otros DBMS, en los que no ocurre lo mismo). De 0 a 255 caracteres (de 1 a 255 en versiones de MySQL anteriores a la 4.0.2).

Tipo	Descripción
TINYBLOB	Objeto binario grande pequeño. El numero de caracteres maximo es de 255 ($2^8 - 1$). Requiere una longitud de almacenamiento de + 1 bytes. Igual que TINYTEXT, con la salvedad de que la busqueda discrimina entre mayusculas y minusculas. Es aconsejable utilizar VARCHAR BINARY en la mayor parte de las situaciones porque resulta mas rapido.
TINYTEXT	El numero de caracteres maximo es de 255 ($2^8 - 1$). Requiere una longitud de almacenamiento de + 1 bytes. Igual que TINYBLOB con la salvedad de que la busqueda no discrimina entre mayusculas y minusculas. Es aconsejable utilizar VARCHAR en la mayor parte de las situaciones porque resulta mas rapido.
BLOB	Objeto binario grande. Maximo de 65.535 caracteres ($2^{16} - 1$). Requiere una longitud de almacenamiento de + 2 bytes. Igual que TEXT, con la salvedad de que la busqueda discrimina entre mayusculas y minusculas.
TEXT	Maximo de 65.535 caracteres ($2^{16} - 1$). Requiere una longitud de almacenamiento de + 2 bytes. Igual que BLOB, con la salvedad de que la búsqueda no discrimina entre mayusculas y minusculas.
MEDIUMBLOB	Objeto binario grande de tamaño medio. Maximo de 16.777.215 caracteres ($2^{24} - 1$). Requiere una longitud de almacenamiento de + 3 bytes. Igual que MEDIUMTEXT con la salvedad de que la búsqueda discrimina entre mayusculas y minusculas.
MEDIUMTEXT	Maximo de 16.777.215 caracteres ($2^{24} - 1$). Requiere una longitud de almacenamiento de + 3 bytes. Igual que MEDIUMBLOB, con la salvedad de que la búsqueda no discrimina entre mayúsculas y minusculas.
LONGBLOB	Objeto binario grande de gran tamaño . Maximo de 4.294.967.295 caracteres ($2^{32} - 1$). Requiere una longitud de almacenamiento de + 4 bytes. Igual que LONGTEXT, con la salvedad de que la búsqueda discrimina entre mayusculas y minusculas. Fijese en que debido a las restricciones externas existe un límite de 16MB por fila de comunicacion paquete/tabla .
LONGTEXT	Maximo de 4.294.967.295 caracteres ($2^{32} - 1$). Requiere una longitud de almacenamiento de + 4

Tipo	Descripción
	bytes. Igual que LONGBLOB, con la salvedad de que la búsqueda no discrimina entre mayúsculas y minúsculas. Fíjese en que debido a las restricciones externas existe límite de 16MB por fila de comunicación paquete/tabla .
ENUM('valor1','valor2',...)	Enumeración. Solo puede tener uno de los valores especificados, NULL o "". Valores máximos de 65.535.
SET('valor1', 'valor2',...)	Un conjunto. Puede contener de cero a 64 valores de la lista especificada.

Utilice las siguientes directrices a la hora de decidir qué tipo de cadena seleccionar:

- No almacene nunca números en **columnas** de cadena. Resulta mucho más **eficaz** hacerlo en **columnas** de tipo numérico. Cada dígito incluido en una cadena ocupa un byte de espacio, en contraposición a un campo numérico, que los almacena en bits. Así mismo, la ordenación de números almacenados en **columnas** de cadena puede generar resultados incoherentes.
- Para lograr mayor velocidad, utilice **columnas fijas**, como CHAR .
- Para ahorrar espacio, utilice **columnas dinámicas**, como VARCHAR .
- Para **limitar** los contenidos de una **columna** a una **opción**, utilice ENUM.
- Para permitir más de una entrada en una **columna**, seleccione SET
- Si desea **buscar** texto sin discriminar entre mayúsculas y minúsculas, utilice TEXT.
- Si desea **buscar texto** discriminando entre mayúsculas y minúsculas, utilice BLOB.
- Para imágenes y otros objetos binarios, almacénelos en el sistema de **archivos** en lugar de directamente en la base de datos.

De **manera** predeterminada, **las** búsquedas **sobre** CHAR y VARCHAR se realizan sin discriminar entre mayúsculas y minúsculas a **menos** que utilice la **palabra clave** BINARY. Por ejemplo:

```
mysql> CREATE TABLE test5(first_name CHAR(10));
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO test5(first_name) VALUES ('Nkosi');
Query OK, 1 row affected (0.06 sec)

mysql> SELECT first_name FROM test5 WHERE first_name='nkosi';
```

```

+-----+
| first_name |
+-----+
| Nkosi      |
+-----+
1 row in set (0.17 sec)

```

Esta búsqueda devuelve un resultado aunque se especifique *nkosi* en lugar de *Nkosi*. Si modifica la tabla, especificando la columna `first_name` como `BINARY`, no recuperara ningun resultado, como se muestra a continuacion:

```

mysql> ALTER TABLE test5 CHANGE first_name first_name CHAR(10)
BINARY;
Query OK, 1 row affected (0.16 sec)
Records: 1 Duplicates: 0 Warnings: 0

mysql> SELECT first_name FROM test5 WHERE first_name='nkosi';
Empty set (0.17 sec)

```

NOTA: La realización de búsquedas sobre campos `CHAR` y `VARCHAR` sin que se discrimine entre mayúsculas y minúsculas no suele ser habitual en la mayor parte de los DBMS, por lo que debe tener cuidado si está realizando el tránsito a MySQL desde otro DBMS.

La palabra clave `NATIONAL` solo se incluye por razones de compatibilidad con SQL ANSI. (ANSI equivale a Instituto americano de normalización y han desarrollado un estandar para SQL.

La mayor parte de los sistemas de administración de bases de datos, DBMS, se adhieren a este estandar en algún grado; son pocos los que lo hacen de forma completa y gran parte de ellos incluyen de elementos propios.) Indica al DBMS que utilice el conjunto de caracteres predeterminado de MySQL (que por otra parte es el estandar de MySQL).

NOTA: Si se utiliza `CHAR` en lugar de `VARCHAR`, el resultado serán tablas de mayor tamaño, pero por regla general más rápidas en cuanto a su procesamiento ya que MySQL sabe dónde comienza cada registro de manera exacta. En una sección posterior se ampliara este tema.

Las columnas `ENUM` incluyen algunas funciones especiales. Si agrega un valor no válido, se insertará una cadena vacia (""), como se puede ver a continuacion:

```

mysql> CREATE TABLE test6(bool ENUM("true","false"));
Query OK, 0 rows affected (0.17 sec)

mysql> INSERT INTO test6(bool) VALUES ('true');
Query OK, 1 row affected (0.17 sec)

```

```
mysql> INSERT INTO test6(bool) VALUES ('troo');
Query OK, 1 row affected (0.06 sec)
```

```
mysql> SELECT bool from test6;
+----+
| bool |
+----+
| true  |
|      |
+----+
2 rows in set (0.11 sec)
```

Tambien puede realizar consultas sobre campos enumerados en funcion de sus indices (el primer valor comicnza en 1). En el ejemplo anterior, `true` se reflejará como un índice 1, `false` como un índice 2, `NULL` como un índice `NULL`, y cualquier otro valor ("") como índice 0. Por ejemplo:

```
mysql> SELECT * FROM test6 WHERE bool=0;
+----+
| bool |
+----+
|      |
+----+
1 row in set (0.17 sec)
```

```
mysql> SELECT • FROM test6 WHERE bool=1;
+----+
| bool |
+----+
| true  |
+----+
1 row in set (0.16 sec)
```

ADVERTENCIA: LOAD DATA no permite agregar registros a un campo enumerado utilizando el índice porque trata todas las entradas como cadenas.

Los campos enumerados se ordenan por los valores de los indices, no de forma alfabética. En otras palabras, se ordenan en el orden en el que se definen los valores.

```
mysql> SELECT • FROM test6 ORDER BY bool ASC;
+----+
| bool |
+----+
| true  |
| false |
+----+
3 rows in set (0.22 sec)
```


Los conjuntos funcionan de forma similar a los campos enumerados:

```
mysql> CREATE TABLE test7 (fruit
SET('apple','mango','litchi','banana'));
Query OK, 0 rows affected (0.11 sec)
```

```
mysql> INSERT INTO test7 VALUES('banana');
Query OK, 1 row affected (0.17 sec)
```

```
mysql> INSERT INTO test7 VALUES('litchi');
Query OK, 1 row affected (0.05 sec)
```

```
mysql> INSERT INTO test7 VALUES('paw-paw');
Query OK, 1 row affected (0.00 sec)
```

La diferencia de un tipo SET es que permite agregar varias instancias:

```
mysql> INSERT INTO test7 VALUES('apple,mango');
Query OK, 1 row affected (0.06 sec)
```

```
mysql> SELECT * FROM test7;
+-----+
| fruit          |
+-----+
| banana         |
| litchi         |
|                |
| apple,mango   |
+-----+
4 rows in set (0.17 sec)
```

Como en el caso de las enumeraciones, la ordenación se realiza por el índice:

```
mysql> INSERT INTO test7 VALUES('mango,apple');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT * FROM test7 ORDER BY fruit;
+-----+
| fruit          |
+-----+
| apple,mango   |
| apple,mango   |
| litchi        |
| banana        |
+-----+
5 rows in set (0.11 sec)
```

Fíjese en que el orden de los elementos es siempre igual al especificado por la instrucción CREATE TABLE.

Por ello, *mango,apple* se almacena como *apple,mango*, y aparece de esta forma en los resultados ordenados.

NOTA: Puede crear una columna de tipo CHAR(1). Resulta de utilidad al trabajar con aplicaciones antiguas que dependan de la existencia de un campo pero no almacenen nada en él. También puede utilizarlas si necesita un campo que contenga sob dos valores, NULL y ""

Tipos de columna de fecha y hora

Los tipos de columna de fecha y hora estan diseiados para trabajar con las necesidades especiales que exigen los datos de tipo temporal y se puede utilizar para almacenar datos tales como la hora del dia o fechas de nacimiento. La tabla 2.3 describe los tipos de columnas de fecha disponibles para MySQL.

Tabla 2.3. Tipos de fecha

Tipo	Descripción
DATETIME	AAAA-MM-DD HH:MM:ss desde 1000-01-01 00:00:00 a 9999-12-31 23:59:59.
DATE	AAAA-MM-DD desde 1000-01-01 a 9999-12-31.
TIMESTAMP	AAAAMDDHHMMSS.
TIME	HH:MM:SS.
YEAR	AAAA.

El tipo de columna TIMESTAMP se puede visualizar de diferentes formas como muestra la tabla 2.4.

Tabla 2.4. Tipos TIMESTAMP

Tipo	Descripción
TIMESTAMP(14)	AAAAMDDHHMMSS
TIMESTAMP(12)	AAMDDHHMMSS
TIMESTAMP(10)	AAMDDHHMM
TIMESTAMP(8)	AAAAMDD
TIMESTAMP(6)	AAMDD
TIMESTAMP(4)	AAM
TIMESTAMP(2)	AA

Esto no implica la perdida de datos. El numero solo afecta a la visualización de los datos; incluso en las columnas definidas como `TIMESTAMP (2)`, se alma-

cenan los 14 dígitos, por lo que si en un momento posterior modificáramos la definición de la tabla, la marca de tiempo se mostraría correctamente.

ADVERTENCIA: Las funciones, a excepción de `UNIX_TIMESTAMP()`, operan sobre el valor de representación. Por lo tanto, la función `DAYOFWEEK()` no funcionara con un tipo `TIMESTAMP(2)` o `TIMESTAMP(4)`.

MySQL acepta diferentes formatos de fecha. Puede sustituir el guion (-) y los dos puntos (:) por cualquier otro carácter de puntuación sin efectos sobre la validez. Por ejemplo:

```
mysql> CREATE TABLE tt(ts DATETIME);

mysql> INSERT INTO tt(ts) VALUES('1999+11+11 23-24');
Query OK, 1 row affected (0.06 sec)
```

Puede incluso sustituir el espacio por otro carácter. El siguiente ejemplo lo sustituye por signos de igual:

```
mysql> INSERT INTO tt(ts) VALUES('1999+12=12-12'12');
Query OK, 1 row affected (0.05 sec)
```

Si el valor introducido no es válido, no se generará un mensaje de error. En su lugar, se asignará 0 como resultado (0000 para un tipo YEAR, 00:00:00 para un tipo TIME, etc.)

Opciones de MySQL

Al ejecutar el comando `mysql` para establecer una conexión a MySQL, puede utilizar cualquiera de las opciones que se muestran en la tabla 2.5.

Tabla 2.5. Opciones de MySQL

Opciones	Descripción
-?, -help	Muestra la ayuda y sale.
-A, -no-auto-rehash	Permite un inicio más rápido. La función de asignación automática permite pulsar la tecla Tab para que MySQL intente completar la tabla o campo. MySQL asigna los nombres del campo o de la tabla al inicio, pero en ocasiones, si el número de tablas y campos es grande, la operación de inicio puede ralentizarse. Esta opción permite desactivar la fun-

Opciones	Descripción
-A, -no-auto-rehash	cion de reasignacion. Para utilizar la funcion de asignacion cuando se ha especificado esta opción en el proceso de inicio, escriba <code>rehash</code> en la línea de comandos.
-b, -no-beep	Desactiva el pitido que se emite cada vez que tiene lugar un error.
-B, -batch	Acepta instrucciones SQL en modo de procesamiento por lotes. Muestra los resultados separados mediante tabuladores . No utiliza el historial.
-character-sets-dir=...	Indica a MySQL en que directorio se encuentran ubicados los conjuntos de caracteres.
-C, -compress	Utiliza la compresion en el protocolo servidor cliente.
-#, -debug[=...]	Crea un registro de depuracion. El valor predeterminado es <code>d:t:o:;/tmp/mysql.trace</code> , que permite labores de depuracion, activa la entrada de llamada de funcion y el rastreo de salida, y dirige el resultado a <code>/tmp/mysql.trace</code> . Puede reemplazar este parametro especificando otro.
-D, -database=..	Indica que base de datos utilizar. Por regla general, puede seleccionar una base de datos sin especificar esta opción, pero resulta útil emplearla en el archivo de configuración.
-default-character-set=...	Establece el conjunto de caracteres predeterminado.
-e, -execute=...	Ejecuta el comando y sale. Produce el mismo resultado que la opción <code>-B</code> .
-E, -vertical	Imprime el resultado de una consulta verticalmente, incluyendo cada campo en una línea diferente. Sin esta opción, puede lograr el mismo resultado colocando <code>\G</code> al final de cada instrucción .
-f, -force	Obliga a MySQL a continuar con el procesamiento aunque reciba un error SQL.

Opciones

Descripción

<code>-f, -force</code>	Esta opción resulta útil en modo de procesamiento por lotes cuando este se realiza desde archivos.
<code>-g, -no-named-commands</code>	Deshabilita comandos con nombres. Utiliza <code>*</code> únicamente o comandos con nombres sólo al principio de la línea que termine en un punto y coma (;). Desde la versión 10.9, el cliente se inicia con esta opción activada de forma predefinida. Sin embargo, con la opción <code>-g</code> , los comandos con formato largo seguirán funcionando desde la primera línea.
<code>-G, -enable-named-commands</code>	Permite el uso de comandos con nombre. Se admiten comandos con formato largo así como comandos <code>*</code> abreviados.
<code>-h, -host=...</code>	Establece la conexión a un equipo dado.
<code>-H, -html</code>	Aplica formato a los resultados de una consulta en HTML. Por regla general, se utilizara un lenguaje de programación para aplicar este formato , pero esta opción puede resultar útil para generar HTML de forma más rudimentaria.
<code>-i, -ignore-space</code>	Ignora los espacios incluidos tras nombres de funciones.
<code>-L, -skip-line-numbers</code>	Impide que MySQL escriba el número de línea asociado a los errores. Puede resultar de utilidad para representar archivos de resultados en los que se necesiten buscar errores o comparar.
<code>-no-pager</code>	Deshabilita el paginador y los resultados dirigidos a un dispositivo estándar de salida. Véase la opción <code>-pager</code> .
<code>-no-tee</code>	Deshabilita la salida. Consulte también la ayuda interactiva (<code>\h</code>).
<code>-n, -unbuffered</code>	Vacia el búfer tras cada consulta.
<code>-N, -skip-column-names</code>	Impide que MySQL escriba los nombres de columna en los resultados.
<code>-O, -set-variable var=option</code>	Asigna un valor a una variable. <code>-help</code> lista las variables.

Opciones	Descripción
<code>-o, --one-database</code>	Solo actualiza la base de datos predeterminada . Esta opcion puede resultar util para evitar actualizaciones de otras bases de datos en el registro de actualizaciones.
<code>-p[er]=[...]</code>	Los resultados con muchos datos suelen salirse de la pantalla. Puede dirigirlos a un paginador. Los paginadores validos son <code>less</code> , <code>more</code> , <code>cat [> nombre de archivo]</code> , etc. Esta opcion no funciona en modo de procesamiento por lotes. <code>Pager</code> solo funciona en Unix.
<code>-p[contraseña], --password[=...]</code>	Contraseña que se utiliza al establecer la conexión al servidor. Si no se indica en la línea de comandos, se le pedira. Si introduce la contraseña en la línea de comandos, no podra incluir espacios entre la opcion y la contraseña .
<code>-P, --port=...</code>	De manera predeterminada, se utiliza el puerto 3306 para establecer la conexión a MySQL . Puede cambiar esta opcion especificando otro numero de puerto TCP/IP para la conexion.
<code>-q, --quick</code>	Obliga a mostrar los resultados fila a fila. Este método aumenta la velocidad de visualizacion de los resultados si hay muchos , pero puede ralentizar el servidor si se suspende la salida. No utiliza el archive de historial.
<code>-r, --raw</code>	Escribe valores de columna sin conversion de escape. Se utiliza con <code>--batch</code> .
<code>-s, --silent</code>	No visualiza una gran cantidad de resultados .
<code>-S, --socket=...</code>	Archivo de socket utilizado para establecer la conexion.
<code>-t, --table</code>	Devuelve los resultados en formato de tabla. Éste es el formato predeterminado en modo no por lotes.
<code>-T, --debug-info</code>	Imprime determinada información de depuracion al salir.
<code>-tee=...</code>	Anexa todo en el archivo de salida. Véase tambien la ayuda interactiva (<code>\h</code>). No funciona en modo por lotes.

Opciones	Descripción
<code>-u, -user=#</code>	Especifica un usuario para el inicio de sesion. Si no se especifica un usuario, MySQL asumira el actual (si lo hubiera).
<code>-U, -safe-updates[=#], -i-am-a-dummy[=#]</code>	Sólo permite UPDATE y DELETE que utilicen claves. Si esta opción lleva asignado el valor predeterminado, puede reiniciarla utilizando <code>-safe-updates=0</code> .
<code>-v, -verbose</code>	Obliga a MySQL a generar salida detallada (<code>-v -v -v</code> aplica formato de tabla a los resultados, <code>-t</code>).
<code>-V, -version</code>	Devuelve información sobre la version y sale.
<code>-w, -wait</code>	Si la conexión no esta establecida, esta opcion espera e intenta establecerla mas tarde, en lugar de abortarla.

La funcion de reasignacion automática permite pulsar la tecla **Tab** y completar la tabla o el campo. MySQL realiza esta operacion al establecer la conesion, pero en ocasiones, cuando el numero de tablas y campos es grande, la operacion de inicio puede resultar muy lenta. Las opciones `-A` o `-no-auto-rehash` desactiva esta funcion.

La opcion `-E` imprime los resultados verticalmente. Puede obtener este tipo de resultados, aunque no tenga establecida la conexión a MySQL con esta opcion activada si utiliza `\G` al final de la consulta:

```
mysql> SELECT * FROM customer\G;
***** 1. row *****
      id: 1
first-name: Yvonne
      surname: Clegg
***** 2. row *****
      id: 2
first-name: Johnny
      surname: Chaka-Chaka
***** 3. row *****
      id: 3
first-name: Winston
      surname: Powers
***** 4. row *****
      id: 4
first-name: Patricia
      surname: Mankunku
***** 5. row *****
      id: 5
first-name: Francois
```

```

    surname: Papo
***** 6. row *****
    id: 7
first-name: Winnie
    surname: Dlamini
***** 7. row *****
    id: 6
first_name: Neil
    surname: Beneke
7 rows in set (0.00 sec)

```

La opción para la **omisión** de espacios (`-i`) brinda una mayor flexibilidad a la hora de reutilizar funciones en las consultas. Por ejemplo, la siguiente secuencia genera un error (**fíjese** en el espacio utilizado tras `MAX`):

```

mysql> SELECT MAX (value) FROM sales;
ERROR 1064: You have an error in your SQL syntax near '(value)
from
sales' at line 1

```

Si utiliza la opción `-i` al conectar, no surgirá ningún problema:

```

mysql> SELECT MAX(value) FROM sales;
+-----+
| MAX (value) |
+-----+
|          3800 |
+-----+

```

La opción `-H` (o `-html`) coloca los resultados de la consulta dentro de una tabla HTML. Si establece la **conexión** con esta opción, se generará el siguiente resultado:

```

mysql> SELECT * FROM customer;
<TABLE BORDER=1><TR><TH>id</TH><TH>first_name</
TH><TH>surname</TH></TR>
<TR><TD>1</TD><TD>Yvonne</TD><TD>Clegg</TD></TR>
<TR><TD>2</TD><TD>Johnny</TD><TD>Chaka-Chaka</TD></TR>
<TR><TD>3</TD><TD>Winston</TD><TD>Powers</TD></TR>
<TR><TD>4</TD><TD>Patricia</TD><TD>Mankunku</TD></TR>
<TR><TD>5</TD><TD>Francois</TD><TD>Papo</TD></TR>
<TR><TD>7</TD><TD>Winnie</TD><TD>Dlamini</TD></TR>
<TR><TD>6</TD><TD>Neil</TD><TD>Beneke</TD></TR></TABLE>
7 rows in set (0.00 sec)

```

La opción `-o` sólo permite realizar actualizaciones sobre la base de datos predeterminada. Si establece la **conexión** utilizando esta opción, no podrá realizar actualizaciones sobre ninguna de las tablas de la base de datos `firstdb`:

```

mysql> UPDATE customer SET first_name='John' WHERE
first_name='Johnny';
Ignoring query to other database

```


La opción `-U` (también conocida como la opción "soy un poco torpe") ayuda a evitar sorpresas desagradables ya que no **permite** realizar operaciones de **actualización** o **eliminación** sin una clave (tema que se analizara en un capítulo posterior). Si establece la **conexión** utilizando esta opción, el siguiente comando no funcionara:

```
mysql> DELETE FROM customer;  
ERROR 1175: You are using safe update mode and you tried to  
update a  
table without a WHERE that uses a KEY column
```

Análisis de los distintos tipos de tablas

Existen dos tipos de tablas de transacción segura (**InnoDB** y **BDB**). El resto (**ISAM**, **MyISAM**, **MERGE** y **HEAP**) no son de transacción segura. La **elección** del tipo de tabla adecuado puede afectar enormemente al rendimiento.

Tablas ISAM

Las tablas del tipo **Método de acceso secuencial indexado (ISAM)** era el **estándar** antiguo de MySQL. **Éstas fueron** sustituidas por las tablas **MyISAM** en la versión 3.23.0 (aunque los tipos **ISAM** seguirán estando disponibles hasta MySQL 4.1). Por lo **tanto**, es probable que **sólo** se tope con este tipo de tablas si está trabajando con bases de datos antiguas. La principal diferencia entre las dos es que el **índice** de las tablas **MyISAM** es mucho más pequeño que el de las tablas **ISAM**, de **manera** que una **instrucción SELECT** con un **índice sobre** una tabla **MyISAM** utilizara muchos **menos** recursos del sistema. En contrapartida, las tablas de tipo **MyISAM** necesitan mucha más **potencia** de procesador para **insertar** un registro dentro de un **índice** más comprimido.

Las tablas **ISAM** presentan las siguientes características:

- **ISAM** almacena los archivos de datos con una extensión **.ISD** y el **archivo de índice** con una extensión **.ISM**.
- Las tablas no son archivos binarios portables entre diferentes equipos o sistemas operativos. En otras palabras, no basta con copiar los archivos **ISD** e **ISM**. Necesitará utilizar un **método** de volcado, como **mysqldump** (analizado en un capítulo posterior).

Si se **topa** con una tabla de **tipo ISAM**, debería convertirla a tipo **MyISAM** ya que **resultan** más eficaces. Las tablas **MyISAM** **permiten además** utilizar un mayor número de las funciones de MySQL. Utilice la siguiente secuencia para convertir una tabla **ISAM** a una tabla **MyISAM**:

```
ALTER TABLE nombre_de_tabla TYPE = MYISAM;
```

Tablas MyISAM

Las tablas de tipo **ISAM** sustituyeron a las tablas **ISAM** en la version 3.23.0. Los índices MyISAM son mucho mas pequeeios que los índices **ISAM**. Debido a **ello**, el sistema utiliza **menos** recursos **al** realizar una **operación** de **selección** mediante un **índice** de una tabla MyISAM. Sin embargo, MyISAM requiere mas **potencia** de procesador para insertar un registro dentro de un **índice** mucho mas comprimido.

Los archivos de datos MyISAM llevan asignada la extension **.MYD** y la extension de los índices es **.MYI**. Las bases de datos MyISAM se almacenan en un directorio. Por lo **tanto**, si ha realizado los ejercicios del capitulo anterior y **dispone** de **permiso** para examinar el directorio `firstdb`, vera los siguientes **archivos**:

- `sales_rep.MYI`
- `sales_rep.MYD`
- `sales.MYD`
- `sales.MYI`
- `customer.MYD`
- `customer.MYI`

Los archivos de datos deberian ser siempre mas grandes que los archivos de **índice**. En un capitulo posterior, se explicara como utilizar correctamente los índices y se analizara su contenido.

Existen tres subtipos de tablas MyISAM: estaticas, dinamicas y comprimidas.

Al crear las tablas, **MySQL** escoge entre el tipo **dinámico** o el tipo estatico. El **tipo** predeterminado son las tablas estaticas y se crean si no incluyen **columnas** `VARCHAR`, `BLOB` o `TEXT`. De lo contrario, la tabla se convierte en tabla **dinámica**.

Tablas estaticas

Las tablas estaticas (tambien denominadas de **forma** mas descriptiva **tablas de longitud fija**) tienen longitud **fija**. En la figura 2.1, se muestran los caracteres almacenados en una mini tabla. El **campo** es un nombre definido como `CHAR(10)`.

I	A	N							
V	I	N	C	E	N	T			
M	I	R	I	A	M				

Figura 2.1. Datos almacenados en formato estatico

Cada registro lleva asignados exactamente 10 bytes. Si el nombre ocupara **menos** espacio, el resto **de** la **columna** se rellenaria con espacios para ajustarse a los 10 caracteres.

Las tablas estaticas se caracterizan por:

- Ser muy **rápidas** (ya que MySQL sabe que el segundo nombre comienza siempre en el caracter numero once).
- **Resultan** sencillas de almacenar en cache.
- **Resultan** sencillas de reconstruir tras un fallo (ya que como las posiciones de los registros son fijas, MySQL sabe donde se encuentra; de esta **forma** sólo se perdera el registro escrito durante el fallo).
- Requieren mas espacio de disco (se necesitan 30 caracteres para tres **registros**, aunque los nombres ocupen sólo 16).
- No resulta necesario reorganizarlas con `myisamchk` (en un capitulo posterior examinaremos este aspecto).

Tablas dinamicas

Las **columnas** de las tablas dinamicas **tienen** diferentes tamaños. Si los mismos datos utilizados en la tabla estatica se colocan en una tabla dinamica, se almacenaran como se muestra en la **figura 2.2**:

I	A	N				
V	I	N	C	E	N	T
M	I	R	I	A	M	

Figura 2.2. Datos almacenados en formato dinamico

Aunque este **formato** de datos ahorra espacio, resulta sin embargo mas **complejo**. Cada registro consta de un encabezado que indica su longitud.

Las tablas de **tipo** dinamico presentan las siguientes características:

- Todas las **columnas** de cadena son dinamicas, a **menos** que su tamaño sea inferior a 4 bytes. (En este **caso**, el espacio ahorrado resultaria insignificante y la complejidad adicional provocaria una perdida de rendimiento.)
- Por **regla** general, ocupan mucho **menos** espacio de disco que las tablas fijas.
- Las tablas requieren un mantenimiento regular para evitar su fragmentacion. (Por ejemplo, si actualizamos *Ian* a *Iane*, la *e* no puede aparecer en

el espacio inmediatamente posterior a la porque este espacio esta ocupado por el inicio de la siguiente columna o registro.) En un capitulo posterior se amplia el tema del mantenimiento.

- En caso de **columnas** fragmentadas, cada nuevo **vínculo** supondra 6 bytes adicionales y tendra **al menos** 20 bytes de tamaño (además de poder tener vinculos propios si se aplican otras actualizaciones que aumenten dicho tamaño).
- No **resultan** tan sencillas de reconstruir tras un **fallo** del sistema, especialmente si las tablas estan muy fragmentadas.
- Si se excluyen **los vinculos**, el tamaño de un registro dinamico se puede calcular con la siguiente formula:

```
3
+ (numero de columnas + 7) / 8
+ (numero de columnas de caracter)
+ tamaño empaquetado de las columnas numericas
+ longitud de cadenas
+ (numeros de columnas NULL + 7) / 8
```

- Cada registro consta de un encabezado, lo que indica que **columnas** de cadena estan vacias y que **columnas** numericas contienen un cero (no registros NULL), en cuyo caso no se almacenan en el disco. Las **cadena**s no vacias contienen un byte de longitud **más** los contenidos de la cadena.

Tablas comprimidas

Las tablas comprimidas son tablas de **sólo** lectura que utilizan mucho **menos** espacio de disco.

Son ideales para su uso con datos comprimidos que no cambien (que sólo se pueden leer y no escribir) y donde no exista mucho espacio disponible, como en un CD-ROM.

Las tablas comprimidas presentan las siguientes características:

- Se crean utilizando la utilidad myisampack (fijese en que la **opción** **ROW_FORMAT="compressed"** del comando **CREATE TABLE** sólo **funcionará** si el codigo de myisampack se ha agregado al servidor).
- Las tablas son mucho mas pequeñas.
- Como cada registro se comprime de **forma** separada, la carga de acceso es reducida.
- Cada columna se **podría** comprimir de **forma** diferente, utilizando distintos algoritmos de compresion.
- Se pueden comprimir formatos de tabla **fija** y dinamica.

- Para crear una tabla comprimida con `myisampack`, basta con ejecutar la siguiente secuencia:

```
myisampack [opciones] nombre del archivo
```

En la tabla 2.6 se recogen las opciones correspondientes a las tablas comprimidas.

Tabla 2.6. Opciones de tabla comprimida

Opción	Descripción
<code>-b, -backup</code>	Crea un volcado de la tabla llamado <code>nombre_de_tabla.OLD</code> .
<code>-#, -debug=opciones_depuracion</code>	Genera el registro de depuración. La cadena <code>opciones_depuracion</code> suele ser <code>d:t:o</code> , nombre de archivo.
<code>-f, -force</code>	Durante el proceso de compresión, MySQL crea un archivo temporal llamado <code>nombre_de_tabla.TMD</code> . Si este proceso finaliza de forma inesperada por alguna razón, puede que el archivo temporal no se elimine. Esta opción obliga a MySQL a comprimir la tabla aunque exista el archivo temporal, si el proceso de compresión aumenta el tamaño de la tabla o si la tabla es demasiado pequeña para comprimirse.
<code>-?, -help</code>	Muestra un mensaje de ayuda y sale.
<code>-j big_nombre_de_tabla, -join=nombre_de_tabla</code>	Combina todas las tablas incluidas en la línea de comandos en una tabla mayor. Las tablas deben ser idénticas (en todos los aspectos, como columnas e índices).
<code>-p #, --packlength=#s</code>	Por regla general , esta opción solo se utiliza al ejecutar <code>myisampack</code> por segunda vez. <code>myisampack</code> almacena todas las filas con un puntero de longitud de 1-3. Ocasionalmente, puede que detecte la necesidad de utilizar un puntero de longitud inferior durante el proceso de compresión (por lo general suele deducirlos correctamente). Cuando volvamos a comprimir la tabla, podemos indicar a <code>myisampack</code> que utilice el tamaño de almacenamiento de longitud óptima .
<code>-s, -silent</code>	Modo silencioso. Solo muestra errores.
<code>-t, -test</code>	Esta opción no comprime la tabla, solo prueba el proceso de compresión.

Opción	Descripción
-T nombre_directorio, -tmp_dir= nombre_directorio	Escribe la tabla temporal dentro del directorio especificado.
-v, - verbose	Modo detallado. Escribe información sobre el progreso y el resultado del proceso de compresión.
-V, - version	Muestra información sobre la versión y sale.
-w, - wait	Si se está utilizando la tabla, esta opción espera y vuelve a intentarlo. No es aconsejable utilizar esta opción en combinación con <code>- skip-external-locking</code> si existe la posibilidad de que la tabla se vaya a actualizar durante el proceso de compresión.

Vamos comprimir una de las tablas que hemos estado utilizando hasta el momento. Tenemos que utilizar la opción `-f` porque la tabla es demasiado pequeña para comprimirla normalmente:

```
C:\Archivos de programa\MySQL\bin>myisampack -v -f
..\data\firstdb\sales_~1
Compressing ..\data\firstdb\sales_~1.MYD: (5 records)
- Calculating statistics

normal:      3  empty-space:      0  empty-zero:      2
empty-fill:  1
pre-space:   0  end-space:      2  intervall-fields:  0
zero:        0
Original trees: 7  After join: 1
- Compressing file
Min record length:  10  Max length:  17  Mean total
length:  40
-35.81%
```

Para descomprimir una tabla, ejecute el comando `myisamchk -unpack nombre de archivo`:

```
C:\Archivos de programa\MySQL\bin>myisamchk -unpack
..\data\firstdb\sales_~1
- recovering (with keycache) MyISAM-table
'..\data\firstdb\sales_~1'
Data records: 5
```

Tablas MERGE

Las tablas MERGE son la fusión de tablas MyISAM iguales. Este tipo de tablas se introdujeron en la versión 3.23.25.

Por regla general sólo se utilizan cuando las tablas **MyISAM** empiezan a resultar demasiado grandes.

Entre las ventajas de estas tablas se pueden mencionar las siguientes:

- **Resultan** mas rápidas en determinadas situaciones (se pueden dividir varias tablas en discos diferentes y utilizar una tabla **MERGE** para acceder a ellas como si se tratara de una sola tabla).
- El tamaño de tabla es mas pequeño. Algunos sistemas operativos tienen un límite en cuanto a los tamaños de archivo y la división de las tablas y la creación de una tabla **MERGE** permite solucionar este problema. Así mismo, los archivos **resultan** mas fáciles de transferir, como por ejemplo para copiarlos a un CD.
- Puede convertir la mayor parte de las tablas originales en tablas de sólo lectura y permitir la inserción de elementos en la tabla mas reciente. De esta forma sólo se corre el riesgo de dañar una pequeña tabla durante el proceso de actualización o inserción y el proceso de reparación resultara mucho mas rápido.

Entre las desventajas de las tablas **MERGE** se incluyen las siguientes:

- **Resultan** mucho mas lentas en búsquedas `eq_ref`
- Es necesario tener cuidado al cambiar una de las tablas subyacentes, ya que puede dañarse la tabla **MERGE** (en realidad no sufren daños, sólo puede ocurrir que no este disponible).
- El comando **REPLACE** no funciona sobre ellas.
- Las tablas utilizan algunos descriptores mas de archivos.

A continuación crearemos una tabla **MERGE**. En primer lugar, necesitamos crear dos tablas idénticas:

```
CREATE TABLE sales-rep1 (  
id INT AUTO_INCREMENT PRIMARY KEY,  
employee-number INT(11),  
surname VARCHAR(40),  
first-name VARCHAR(30),  
commission TINYINT(4),  
date-joined DATE,  
birthday DATE  
) TYPE=MyISAM;
```

```
CREATE TABLE sales-rep2 (  
id INT AUTO-INCREMENT PRIMARY KEY,  
employee-number INT(11),  
surname VARCHAR(40),  
first_name VARCHAR(30),  
commission TINYINT(4),
```

```

date-joined DATE,
birthday DATE
) TYPE=MyISAM;

CREATE TABLE sales_rep1_2 (
id INT AUTO-INCREMENT PRIMARY KEY,
employee_number INT(11),
surname VARCHAR(40),
first-name VARCHAR(30),
commission TINYINT(4),
date-joined DATE,
birthday DATE
) TYPE=MERGE
UNION=(sales_rep1,sales_rep2);

```

A continuacion, insertaremos algunos datos dentro de las tablas para poder probarlas:

```

INSERT INTO sales-rep1 ('employee-number', 'surname',
'first-name', 'commission', 'date-joined', 'birthday')
VALUES (1,'Tshwete','Paul',15,'1999-01-03','1970-03-04');

INSERT INTO sales-rep2 ('employee-number', 'surname',
'first-name', 'commission', 'date-joined', 'birthday')
VALUES (2,'Grobler','Peggy-Sue',12,'2001-11-19','1956-08-25');

```

Ahora, si realizamos una consulta sobre la tabla combinada, todos los registros de sales_rep1 y sales_rep2 estaran disponibles:

```

mysql> SELECT first_name,surname FROM sales_rep1_2;
+-----+-----+
| first-name | surname |
+-----+-----+
| Paul      | Tshwete |
| Peggy-Sue | Grobler |
+-----+-----+
2 rows in set (0.00 sec)

```

En funcion de los resultados anteriores, no es posible saber de que tabla subyacente proceden. Afortunadamente, no necesitaremos saberlo si estamos actualizando un registro. La siguiente instrucción

```

mysql> UPDATE sales_rep1_2 set first_name = "Peggy"
WHERE first_name="Peggy-Sue";
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

```

actualizara el registro correctamente. Como el registro sólo existe físicamente en el nivel subyacente, las consultas realizadas sobre la tabla MERGE y sobre la tabla MyISAM subyacente reflejaran los datos correctamente, como se demuestra a continuación:

```

mysql> SELECT first_name,surname FROM sales-rep1-2;

```



```

+-----+-----+
| first-name | surname |
+-----+-----+
| Paul       | Tshwete |
| Peggy      | Grobler |
+-----+-----+
2 rows in set (0.00 sec)

```

```

mysql> SELECT first_name,surname FROM sales-rep2;
+-----+-----+
| first-name | surname |
+-----+-----+
| Peggy      | Grobler |
+-----+-----+
1 row in set (0.00 sec)

```

Los mismo se aplica a las instrucciones DELETE:

```

mysql> DELETE FROM sales_rep1_2 WHERE first_name='Peggy';
Query OK, 1 row affected (0.00 sec)

```

El registro se elimina en el nivel subyacente, por lo que desaparecera de las consultas en la tabla MERGE y en la tabla subyacente:

```

mysql> SELECT first_name,surname FROM sales_rep1_2;
+-----+
| first-name | surname |
+-----+-----+
| Paul       | Tshwete |
+-----+-----+
1 row in set (0.00 sec)

```

```

mysql> SELECT first_name,surname FROM sales-rep2;
Empty set (0.00 sec)

```

Sin embargo, si intenta realizar una operación de inserción, MySQL no sabra en que tabla subyacente insertar el registro y devolvera un error:

```

mysql> INSERT INTO sales_rep1_2 ('surname', 'first_name',
  'commission', 'date-joined', 'birthday')
  VALUES ('Shephard', 'Earl',11,'2002-12-15','1961-05-31');
ERROR 1031: Table handler for 'sales_rep1_2' doesn't have this
option

```

Por suerte existe una solución, que fue introducida en la version 4 (antes no se podian insertar registros en las tablas MERGE). Al crear una tabla MERGE, podemos especificar en que tabla realizar las inserciones. Fijese en la ultima cláusula de la siguiente instrucción CREATE:

```

CREATE TABLE sales_rep1_2 (
  id INT AUTO-INCREMENT PRIMARY KEY,
  employee-number INT(11),
  surname VARCHAR(40),

```

```

first-name VARCHAR(30),
commission TINYINT(4),
date-joined DATE,
birthday DATE
) TYPE=MERGE
UNION=(sales_rep1,sales_rep2)
INSERT-METHOD = LAST

```

INSERT_METHOD puede ser NO, FIRST o LAST. A continuación, los registros insertados se colocan dentro de la primera tabla en la lista de unión, en la última tabla o en ninguna. El valor predeterminado es NO.

ADVERTENCIA: Si realiza algún cambio estructural en las tablas subyacentes, como modificar el nombre o reconstruir los índices, necesitará volver a construir la tabla MERGE. En primer lugar, elimine la tabla MERGE, realice los cambios deseados y vuelva a construir la tabla MERGE. Si realiza los cambios y olvida eliminar la tabla MERGE, puede que descubra que no puede acceder a la tabla correctamente. Para solucionar este problema, elimine la tabla MERGE y vuelva a reconstruirla.

Tablas HEAP

Las tablas HEAP son el tipo de tabla más rápido porque se almacenan en memoria y utilizan un índice asignado. La contrapartida es que, como se almacenan en memoria, todos los datos se pierden en caso de un fallo en el sistema. Además, no pueden contener una gran cantidad de datos (a menos que disponga de un gran presupuesto para RAM).

Como en el caso de cualquier otra tabla, puede crear una en función de los contenidos de otra. Las tablas HEAR se suelen utilizar para acceder rápidamente a una tabla ya existente (se deja la tabla original para labores de inserción y de actualización, y la nueva tabla se utiliza para realizar lecturas rápidas.) A continuación, crearemos una tabla a partir de la tabla sales_rep. Si no creo la tabla sales_rep en el capítulo anterior, hágalo ahora y rellénela utilizando las siguientes instrucciones:

```

CREATE TABLE sales_rep (
  employee-number int(11) default NULL,
  surname varchar(40) default NULL,
  first_name varchar(30) default NULL,
  commission tinyint(4) default NULL,
  date-joined date default NULL,
  birthday date default NULL
) TYPE=MyISAM;

INSERT INTO sales_rep VALUES (1, 'Rive', 'Sol', 10,
  '2000-02-15', '1976-03-18');

```

```

INSERT INTO sales-rep VALUES (2, 'Gordimer', 'Charlene', 15,
'1998-07-09', '1958-11-30');
INSERT INTO sales-rep VALUES (3, 'Serote', 'Mike', 10,
'2001-05-14', '1971-06-18');
INSERT INTO sales-rep VALUES (4, 'Rive', 'Mongane', 10,
'2002-11-23', '1982-01-04');

```

A **continuación**, crearemos una tabla HEAP que tome un subconjunto de **elementos** de `sales-rep` y los coloque en **memoria** para brindar un acceso mas rapido:

```

mysql> CREATE TABLE heaptest TYPE=HEAP SELECT
first_name,surname
FROM sales-rep;
Query OK, 4 rows affected (0.02 sec)
Records: 4 Duplicates: 0 Warnings: 0

```

```

mysql> SELECT * FROM heaptest;
+-----+-----+
| first-name | surname |
+-----+-----+
| Sol       | Rive    |
| Charlene  | Gordimer |
| Mike      | Serote  |
| Mongane   | Rive    |
+-----+-----+
4 rows in set (0.00 sec)

```

Entre las características de las tablas HEAP se pueden **destacar** las siguientes:

- Como las tablas HEAP utilizan **memoria**, no es deseable que su tamaño sea demasiado grande. El tamaño de las tablas se limita mediante el uso de la variable `max_heap_table_size` de `mysql`.
- Las claves no se utilizan de la misma **forma** que en las tablas **MyISAM**. No se pueden **utilizar** con una **instrucción** `ORDER BY`.
- Sólo utilizan la clave completa para **buscar una fila**, no parte de una clave.
- Sólo utilizan `=` y `<=>` **al buscar** índices.
- El optimizador de rangos de **MySQL** no puede descubrir cuantas **filas existen** entre dos valores.
- Sin embargo, si las claves se **usan de manera correcta** sobre tablas HEAP el resultado es mas rapida.
- Las tablas HEAP, a diferencia de otras tablas asignadas, **permiten** el uso de claves no unicas.
- No **admiten** el uso de índices en una **columna** `NULL`.
- No **admiten** **columnas** `AUTO_INCREMENT`.
- No **admiten** **columnas** `BLOB` o `TEXT`.

Como puede ver, existen bastantes diferencias entre los índices MyISAM y los índices HEAP. Una tabla HEAP puede resultar **más lenta** si nos basamos en un índice que no utilice. En un capítulo posterior se analiza el uso de las claves con mas detenimiento.

NOTA: Además del límite `max_heap_table_size` y del límite de memoria de su equipo, se podría alcanzar un límite de 4GB por tabla en algunas configuraciones dado que esa es la limitación impuesta por el espacio de dirección en los equipos de 32 bits.

Tablas InnoDB

Las tablas InnoDB son tablas de transacción segura (lo que significa que disponen de las funciones COMMIT y ROLLBACK). En una tabla MyISAM, la tabla entera se bloquea al realizar funciones de inserción. Durante esa fracción de segundo, no se puede ejecutar ninguna otra instrucción sobre la tabla. InnoDB utiliza funciones de bloqueo en el nivel de fila de manera que solo se bloquee dicha fila y no toda la tabla, y se puedan seguir aplicando instrucciones sobre otras filas.

Por razones de rendimiento, es aconsejable utilizar tablas InnoDB si necesita realizar una gran cantidad de operaciones de inserción y actualización sobre los datos de sus tablas en comparación con operaciones de selección. Por el contrario, si las operaciones de selección superan a las de actualización o inserción, es preferible inclinarse por las tablas MyISAM.

Para utilizar tablas InnoDB, es necesario compilar MySQL con compatibilidad InnoDB (en un capítulo posterior se explicarán los detalles), como la distribución `mysqld-max`. También existe una serie de parámetros de configuración que deberían configurarse antes de confiar en este tipo de tablas para obtener un buen rendimiento.

Al iniciar MySQL con las opciones InnoDB compiladas y utilizar solo los valores predeterminados, verá aparecer una secuencia parecida a la siguiente:

```
C:\MySQL\bin>mysqld-max
InnoDB: The first specified data file .\ibdata1 did not exist:
InnoDB: a new database to be created!
InnoDB: Setting file .\ibdata1 size to 64 MB
InnoDB: Database physically writes the file full: wait...
InnoDB: Log file .\ib_logfile0 did not exist: new to be created
InnoDB: Setting log file .\ib_logfile0 size to 5 MB
InnoDB: Log file .\ib_logfile1 did not exist: new to be created
InnoDB: Setting log file .\ib_logfile1 size to 5 MB
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: Creating foreign key constraint system tables
InnoDB: Foreign key constraint system tables created
```

```
020504 12:42:52 InnoDB: Started
C:\MYSQL\BIN\MYSQLD~2.EXE: ready for connections
```

ADVERTENCIA: Antes de la versión 4, no bastaba con iniciar MySQL. Era necesario configurar al menos el archivo `innodb_data_file_path`. Este archivo se comenta en un capítulo posterior.

De manera predeterminada, MySQL crea un archivo `ibdata1` en el directorio de datos predeterminado (por lo general, `C:\MSQL\data` en Windows, o `/usr/local/mysql/data` o `/usr/local/var/` en Unix).

Las tablas InnoDB se diferencian de las tablas MyISAM en que las bases de datos no se almacenan en un directorio, con las tablas como archivos. Todas las tablas y los índices se almacenan en un *espacio de tabla* InnoDB (que puede componerse de una o varias tablas; en el ejemplo anterior era `ibdata1`).

Por lo tanto, la restricción en cuanto a los datos no viene dada por el límite del tamaño de archivos del sistema operativo.

NOTA: En una tabla MyISAM, un sistema operativo con un límite de 2GB permitirá un tamaño de tabla **máximo** de 2GB. En el caso de las tablas InnoDB no existe dicho límite, pero la responsabilidad de optimizar el espacio de las tablas recae en el administrador.

El tamaño inicial de las tablas se establece en 16MB. En versiones anteriores de MySQL anteriores a la 4, el tamaño asignado era de 64 pero no se podía extender (lo que significa que una vez agotado el espacio, no se podía ampliar). En las versiones posteriores este parámetro pasó a extenderse automáticamente de manera predeterminada, lo que significa que el espacio de las tablas crece a medida que lo hacen los datos. Sin embargo, MySQL brinda control manual sobre este parámetro para poder optimizar el rendimiento. En breve se explicará como (en un capítulo posterior se desarrolla este tema de manera más detallada).

Utilice la siguiente secuencia para crear una tabla InnoDB:

```
mysql> CREATE TABLE innotest (f1 INT,f2 CHAR(10),INDEX (f1))
TYPE=InnoDB;
Query OK, 0 rows affected (0.10 sec)
```

Tablas DBD

DBD equivale a Base de datos de Berkeley (creada originalmente en la University of California, Berkeley). Se trata también de un tipo de tabla habilitado para transacciones. Como en el caso de las tablas InnoDB, es necesario compilar la compatibilidad BDB en MySQL para que funcione (la distribución `mysql-max` incorpora dicha función).

Para crear una tabla BDB, basta con utilizar `TYPE=BDB` tras la **instrucción** `CREATE TABLE`:

```
mysql> CREATE TABLE bdbtest (f1 INT, f2 CHAR(10)) TYPE=BDB;  
Query OK, 0 rows affected (0.28 sec)
```

En la actualidad, la interfaz entre MySQL y DBD (que existe de forma independiente a MySQL) sigue todavía en versión beta. MySQL y BDB llevan **mucho años** entre nosotros, y se pueden considerar como estables, no así la interfaz entre ambos. Examine la última **documentación al respecto** para verificar si se ha producido algún **avance**.

Resumen

La selección de los tipos de campos correctos es una **cuestión** importante si desea obtener el **mejor** rendimiento de MySQL. Los tipos numéricos **permiten** realizar cálculos y suelen resultar más pequeños que los tipos de cadena. Los tipos de fecha **permiten** almacenar fechas y horas de **forma** sencilla.

De la misma **forma**, para utilizar MySQL de **manera** correcta, es necesario comprender los operadores que utiliza. Los operadores lógicos como `AND` y `OR`, y los operadores de comparación como `=` y `LIKE`, **ayudan** a restringir los resultados de las consultas a los registros deseados. Los operadores bit a bit **resultan** útiles para trabajar con datos matemáticos binarios.

MySQL incluye varios tipos de tablas para su uso en diferentes situaciones. Las tablas de tipo **MyISAM** (el tipo predeterminado) es ideal para sistemas en los que se realizan una **gran cantidad** de consultas de actualización (como en los sitios Web). Las tablas `MERGE` son combinaciones de tablas **MyISAM idénticas**; estas tablas facilitan las **labores** de actualización y brindan una mayor velocidad de procesamiento en determinadas situaciones.

Las tablas `HEAP` son las más **rápidas** y se almacenan en **memoria**. Las tablas `InnoDB` y `BDB` garantizan la seguridad de las transacciones, lo que **permite** la agrupación de instrucciones para asegurar la integridad de los datos. Las tablas `InnoDB` realizan lecturas uniformes, lo que significa que los resultados de las tablas se muestran tal y como aparecen tras una **transacción** realizada. Esta **opción** no resulta siempre adecuada y puede reemplazar este comportamiento con bloqueos de lectura para operaciones de actualización y uso compartido.

Ya hemos tenido la oportunidad de conocer un poco de **SQL**, pero todavía queda mucho por aprender. Para comenzar a apreciar el potencial que ofrece **MySQL** y avanzar en su dominio, es necesario analizar varios operadores aritméticos, de comparación y de bit a bit. Estos operadores **permiten** construir consultas mucho más **complejas** que las vistas en el primer capítulo. De **manera** similar, examinaremos consultas que **extraen** datos de varias tablas, llamadas *combinaciones*. Las características **propias** de las combinaciones por la izquierda, por la derecha, externas, **internas** y naturales pueden resultar **confusas**. En este capítulo se abordarán todos estos aspectos y le indicaremos cuando usarlas y como **hacerlo**.

En este capítulo se abordan los siguientes temas:

- Operadores lógicos, aritméticos y bit a bit
- Combinaciones avanzadas (internas, externas, por la izquierda, por la derecha y naturales)
- **Combinación** de resultados con el comando UNION
- Reescritura de subselectores como combinaciones
- **Eliminación** de registros con DELETE y TRUNCATE

- Variables de usuario
- Ejecucion de MySQL en modo de procesamiento por lotes
- **Realización** de transacciones con BEGIN y COMMIT
- Lecturas coherentes
- Bloqueos de tabla
- Bloqueos de lectura para operaciones de actualización y de uso compartido

Operadores

Los operadores son los bloques con los que se construyen las consultas complejas. Los *operadores lógicos* (como AND y OR) permiten asociar varias condiciones de distintas formas. Los *operadores aritméticos* (como + o *) permiten realizar operaciones matemáticas básicas en sus consultas. Los *operadores de comparación* (como > o <) permiten comparar valores y restringir los conjuntos de resultados. Por último, los *operadores bit a bit*, aunque no se utilicen habitualmente, permiten trabajar con bits en las consultas.

Operadores lógicos

Los operadores lógicos reducen las opciones a true (1) o false (0). Por ejemplo, si le pregunto si es hombre OR mujer (estoy asumiendo que quiero una respuesta afirmativa o negativa), la respuesta será sí o true. Si la pregunta fuera si es hombre AND mujer, la respuesta sería no, o false. Los operadores AND y OR utilizados en las preguntas son operadores lógicos. En la tabla 3.1 se describen los operadores de forma más detallada.

Tabla 3.1. Operadores lógicos

Operadores	Sintaxis	Descripción
AND, &&	c1 AND c2, c1 && c2	Sólo es verdad si ambas condiciones, c1 y c2, son verdaderas.
OR,	c1 OR c2, c1 c2	Verdad si c1 o c2 es verdad.
!, NOT	! c1, NOT c1	Verdad si c1 es falsa y falsa si c1 es verdadera.

En lugar de rellenar la tabla y ejecutar consultas sobre ella, los siguientes ejemplos devolverán 1 o 0. Cada fila de las tablas que se consulte se reducirá a un

1 o un 0. Los unos se devolverán y los ceros no. Si entiende la **lógica**, puede aplicar **los principios** a cualquiera de sus tablas. Si es la **primera vez** que trabaja con estos operadores, compruebe si puede predecir los resultados en función de la tabla 3.1.

```
mysql> SELECT 1 AND 0;
+-----+
| 1 AND 0 |
+-----+
|         0 |
+-----+

mysql> SELECT NOT(1 AND 0);
+-----+
| NOT(1 AND 0) |
+-----+
|              1 |
+-----+

mysql> SELECT !((1 OR 0) AND (0 OR 1));
+-----+
| !((1 OR 0) AND (0 OR 1)) |
+-----+
|                            0 |
+-----+
```

Recuerde que las condiciones **incluidas** en los **parentesis** mas **internos** se **resuelven** en primer lugar. Por lo tanto, MySQL **simplifica** la compleja instrucción del ejemplo anterior de la siguiente forma:

```
!((1 OR 0) AND (0 OR 1))
!((1) AND (1))
!(1)
0
```

Operadores aritmeticos

Los operadores aritmeticos se usan para realizar operaciones aritmeticas **elementales**. Por ejemplo, en la expresion $2 + 3 = 5$, el signo mas (+) es un operador **aritmético**. En la tabla 3.2 se describen los operadores aritmeticos disponibles en MySQL.

Tabla 3.2. Operadores aritméticos

Operadores	Sintaxis	Descripción
+	$a + b$	Agrega a y b, y devuelve la suma de ambos.
	$a - b$	Resta b de a, y devuelve la diferencia
	$a * b$	Multiplica a y b, y devuelve el producto de ambos.

Operadores	Sintaxis	Descripción
/	a / b	Divide a entre b, y devuelve el cociente.
%	a % b	A modulo b, y devuelve el resto de a / b.

Por ejemplo, la suma de dos columnas de tipo INT generara otro entero:

```
mysql> SELECT 2+1;
+----+
| 2+1 |
+----+
|    3 |
+----+

mysql> SELECT 4-2/4;
+----+
| 4-2/4 |
+----+
|  3.50 |
+----+
```

El valor que se devuelve en este ejemplo es 3,5, no 0,5, porque la division se realiza en primer lugar (¿recuerda la reglas de prioridad de las operaciones aprendidas en la escuela?). Sin embargo, es aconsejable utilizar siempre parentesis para dejar claro las operaciones que se realizaran en primer lugar a aquellas personas que no conozcan las reglas. Para simplificar la consulta anterior, podemos escribirla de esta forma:

```
mysql> SELECT 4-(2/4);
+----+
| 4-(2/4) |
+----+
|    3.50 |
+----+
```

NOTA: Aunque todos los valores de esta consulta son enteros, como el resultado es un elemento no entero, se devuelve como numero decimal. En el siguiente ejemplo se muestra el funcionamiento del operador de modulo:

```
mysql> SELECT 5 % 3;
+----+
| 5 % 3 |
+----+
|    2 |
+----+
```

El operador de modulo devuelve el resto de una division. En el ejemplo anterior, 5 dividido entre 3 es 1, y el resto es 2.

Operadores de comparacion

Los operadores de comparacion se utilizan para **realizar** comparaciones entre valores. Por ejemplo, podemos afirmar que 34 es mayor que 2. La **expresión es mayor que** es un operador de comparacion. La tabla 3.3 lista y describe los operadores de comparacion utilizados en MySQL.

Tabla 3.2. Operadores de comparacion

Operador	Sintaxis	Descripción
=	a = b	Verdad si a y b son iguales (excluyendo NULL).
!=, <>	a != b, a <> b	Verdad si a no es igual a b.
>	a > b	Verdad si a es mayor que b.
<	a < b	Verdad si a es menor que b.
>=	a >= b	Verdad si a es mayor o igual que b.
<=	a <= b	Verdad si a es menor o igual que b.
<=>	a <=> b	Verdad si a y b son iguales (incluyendo NULL).
IS NULL	a IS NULL	Verdad si a contiene un valor NULL.
IS NOT NULL	a IS NOT NULL	Verdad si a no contiene un valor NULL.
BETWEEN	a BETWEEN b and c	Verdad si a esta entre los valores de b y c, ambos inclusive.
NOT BETWEEN	a NOT BETWEEN b and c	Verdad si a no está entre los valores de b y c, ambos inclusive.
LIKE	a LIKE b	Verdad si a equivale a b en una correspondencia de patron SQL.
NOT LIKE	a NOT LIKE b	Verdad si a no equivale con b en una correspondencia de patron SQL. Los dos comodines aceptados son % (que equivale a cualquier número de caracteres) y _ (que equivale a un carácter).

Operador	Sintaxis	Descripción
IN	a IN (b1, b2, b3)	Verdad si a es igual a algún elemento de la lista.
NOT IN	a NOT IN (b1,b2,b3)	Verdad si a no es igual a algún elemento de la lista.
REGEXP, RLIKE	a REGEXP b, a RLIKE b	Verdad si a equivale a b con una expresion regular.
NOT REGEXP, NOT RLIKE	a NOT REGEXP b, a NOT RLIKE B	Verdad si a no equivale a b con una expresión regular.

Con los operadores de comparacion, vamos a utilizar tambien 1 y 0 para representar los valores verdadero y falso. Recuerde que vamos a sustituir estos con nuestras **propias** constantes y campos procedentes de las tablas de la base de datos. Por ejemplo, tome una tabla con dos **filas**, como se muestra en la figura 3.1.

FIELD1
15
13

Figura 3.1. Tabla1

El siguiente codigo representa el oprador de comparacion =:

```
SELECT * FROM TABLE1 WHERE FIELD1 = 13.
```

Seguidamente se compara **cada** fila de la tabla para comprobar si la **condición** es verdadera o falsa. En la **primera** fila, la cpsresion se reduce a este resultado:

15 = 13

Esta expresion es falsa, por lo que no se devuelve la fila. En la segunda, la cpsresion se reduce a la siguiente:

13 = 13

Esta secuencia es verdadera, por lo que se devuelve la fila.

Una vez entendido este concepto, puede aplicarlo a sus **propias** tablas.

Otro ejemplo:

```
mysql> SELECT 13=11;
+-----+
| 13=11 |
+-----+
|    0  |
+-----+
```

Si viene del mundo de la programación, es probable que conozca las complejidades de comparar tipos diferentes (como cadenas y números). Por ejemplo, ¿qué tipo de respuesta esperaría recibir si pregunta si la cadena "treinta" es inferior al número 29? MySQL intenta ser lo más útil posible cuando se desean comparar valores de diferente tipo (para lo cual convierte los tipos lo mejor que puede; esta operación se conoce como conversiones de tipos). Si está comparando cadenas y números o números decimales y enteros, MySQL los comparará como si se trata del mismo tipo. Por ejemplo:

```
mysql> SELECT '4200' = 4200.0;
+-----+
| '4200' = 4200.0 |
+-----+
|                  1 |
+-----+
```

La cadena "4200" convertida a un número es igual a 4200,0. Sin embargo, las cadenas "4200" y "4200,0" no son iguales:

```
mysql> SELECT '4200' = '4200.0';
+-----+
| '4200' = '4200,0' |
+-----+
|                   0 |
+-----+
```

El siguiente ejemplo demuestra la naturaleza ajena a la discriminación entre mayúsculas y minúsculas de la comparación de cadenas:

```
mysql> SELECT 'abc' = 'ABC';
+-----+
| 'abc' = 'ABC' |
+-----+
|                1 |
+-----+
```

En el siguiente ejemplo, se ignora un espacio a la derecha dentro de una búsqueda de igualdad en la que no se hace distinción entre mayúsculas y minúsculas:

```
mysql> SELECT 'abc' = 'ABC ';
+-----+
| 'abc' = 'ABC' |
+-----+
|                1 |
+-----+
1 row in set (0.00 sec)
```

A continuación se incluye un ejemplo en el que fijarse; el resultado no es 0 (falso) sino NULL:

```
mysql> SELECT NULL=0;
```

```

+-----+
| NULL=0 |
+-----+
| NULL |
+-----+

```

Para evaluar filas NULL, necesitaremos utilizar la siguiente secuencia:

```

mysql> SELECT NULL<=>0;
+-----+
| NULL<=>0 |
+-----+
| 0 |
+-----+

```

NULL es básicamente el tercer resultado posible de una evaluación; los tres resultados son verdadero, falso y NULL. Ninguna de las consultas que se incluyen a continuación suministra resultados útiles al compararse con un valor NULL:

```

mysql> SELECT 200 = NULL, 200 <> NULL, 200 < NULL, 200 > NULL;
+-----+ +-----+ +-----+ +-----+
| 200 = NULL | 200 <> NULL | 200 < NULL | 200 > NULL |
+-----+ +-----+ +-----+ +-----+
| NULL | NULL | NULL | NULL |
+-----+ +-----+ +-----+ +-----+

```

Necesitara utilizar la comparacion IS NULL (o IS NOT NULL) en su lugar:

```

mysql> SELECT NULL IS NULL;
+-----+
| NULL IS NULL |
+-----+
| 1 |
+-----+

```

```

mysql> SELECT 4.5 BETWEEN 4 and 5;
+-----+
| 4.5 BETWEEN 4 and 5 |
+-----+
| 1 |
+-----+

```

El siguiente ejemplo muestra un error comun al utilizar BETWEEN

```

mysql> SELECT 5 BETWEEN 6 and 4;
+-----+
| 5 BETWEEN 6 and 4 |
+-----+
| 0 |
+-----+

```

Como la letra **a** va antes en el alfabeto que la letra b, el resultado del siguiente ejemplo es verdadero.

Las comparaciones de cadenas se realizan de izquierda a derecha y de carácter en carácter:

```
mysql> SELECT 'abc' < 'b';
+-----+
| 'abc' < 'b' |
+-----+
|           1 |
+-----+
```

ADVERTENCIA: MySQL no ordena los dos valores situados tras un comando BETWEEN. Por lo tanto, si los incluye en el orden incorrecto, los resultados serán falsos para todas las filas. Asegúrese de que el primer número es el menor.

En el siguiente ejemplo, la letra *b* es menor o igual que la letra *b*; sin embargo, la letra *b* siguiente no es menor o igual a nada (el segundo carácter situado a la derecha de la cadena):

```
mysql> SELECT 'bbc' <= 'b';
+-----+
| 'bbc' <= 'b' |
+-----+
|           0 |
+-----+
```

La función `IN()` se puede utilizar para probar un valor con respecto a una serie de valores posibles. El campo puede coincidir con cualquiera de los valores separados por coma incluidos dentro del paréntesis, como se muestra en el siguiente ejemplo:

```
mysql> SELECT 'a' IN ('b', 'c', 'a');
+-----+
| 'a' in ('b', 'c', 'a') |
+-----+
|           1 |
+-----+
```

Como usar LIKE en equivalencias de patrón de SQL

Los caracteres comodín incluidos en la tabla 3.4 se suelen utilizar junto a operadores de comparación. Este hecho nos permite realizar comparaciones con respecto a un carácter (o a un número de caracteres) sobre el que no estamos seguros, en lugar de caracteres específicos.

Tabla 3.3. Caracteres comodines

Carácter	Descripción
<code>*</code>	Cualquier número de caracteres
<code>_</code>	Un carácter

El siguiente ejemplo muestra el uso del comodín `%`

```
mysql> SELECT 'abcd' LIKE '%bc%';
+-----+
| 'abcd' LIKE '%bc%' |
+-----+
|                      1 |
+-----+
```

El comodín `%` devuelve cualquier número de caracteres. Por ello, la siguiente secuencia también coincidiría:

```
mysql> SELECT 'abcd' LIKE '%b%';
+-----+
| 'abcd' LIKE '%b%' |
+-----+
|                      1 |
+-----+
mysql> SELECT 'abcd' LIKE 'a_ _ _',
+-----+
| 'abcd' LIKE 'a_ _ _' |
+-----+
|                      1 |
+-----+
```

Los guiones bajos (`_`) equivalen a un único carácter, por lo que si sólo se utilizan dos guiones bajos, en lugar de tres como se muestra a continuación, no se produciría la correspondencia:

```
mysql> SELECT 'abcd' LIKE 'a_ _',
+-----+
| 'abcd' LIKE 'a_ _' |
+-----+
|                      0 |
+-----+
```

Expresiones regulares

Las expresiones regulares permiten realizar comparaciones complejas en MySQL y suelen generar comportamientos de rechazo. Muchas personas fruncen inmediatamente el ceño al oír esta expresión, ponen excusas ya preparadas para evitarlas y no confían en absoluto en su uso. Es verdad que el tema puede resultar complicado (se han escrito libros enteros al respecto), pero su uso en MySQL no resulta difícil y pueden contribuir a incrementar la flexibilidad de las comparaciones. La tabla 3.5 describe los operadores de expresiones regulares en MySQL.

Tabla 3.5. Expresiones regulares (REGEXP, RLIKE)

Carácter	Descripción
*	Equivale a una o varias instancias de la cadena situadas por delante.

Carácter	Descripción
+	Equivale a cero o una instancia situadas por delante .
?	Equivale a un solo caracter.
[xyz]	Equivale a cualquier x, y o z (los caracteres incluidos dentro de los corchetes).
[A-Z]	Equivale a cualquier letra mayúscula .
[a-z]	Equivale a cualquier letra minúscula.
[0-9]	Equivale a cualquier dígito.
^	Fija la equivalencia desde el comienzo.
\$	Fija la correspondencia hasta el final.
	Separa cadena s de una expresión regular.
{n,m}	La cadena debe tener lugar al menos n veces, pero no más.
{n}	La cadena debe tener lugar n veces exactamente.
{n,	La cadena debe tener lugar n veces al menos .

Las coincidencias de **expresiones** regulares (REGEXP) pueden generar **resultados** similares a las coincidencias SQL (LIKE). Sin embargo, también **existen** diferencias importantes entre **ambas**. Una expresión regular, a **menos** que se **especifique** otra cosa, establece **equivalencias** en **cualquier** parte de la cadena. No es **necesario utilizar** comodines en **ninguno** de sus lados, **como** ocurre con LIKE. Fíjese en la diferencia entre **los** siguientes dos resultados:

```
mysql> SELECT 'abcdef' REGEXP 'abc';
```

```
+-----+
| 'abcdef' REGEXP 'abc' |
+-----+
|                          1 |
+-----+
```

```
mysql> SELECT 'abcdef' LIKE 'abc';
```

```
+-----+
| 'abcdef' LIKE 'abc' |
+-----+
|                          0 |
+-----+
```

Para obtener el equivalente con LIKE, tendríamos que **utilizar** el comodín % **al** final:

```
mysql> SELECT 'abcdef' LIKE 'abc%';
```

```

+-----+
| 'abcdef' LIKE 'abc%' |
+-----+
|                               1 |
+-----+

```

El siguiente ejemplo coincide cuando la letra *a* es el primer caracter:
mysql> SELECT 'abc' REGEXP "'a'";

```

+-----+
| 'abc' REGEXP "'a' |
+-----+
|                               1 |
+-----+

```

Sin embargo, en este otro ejemplo no se produce la correspondencia, ya que el signo mas (+) indica que la letra g debe aparecer una o dos veces:

```

mysql> SELECT 'abcdef' REGEXP 'g+' ;
+-----+
| 'abcdef' REGEXP 'g+' |
+-----+
|                               0 |
+-----+

```

La siguiente **consulta** si coincide porque el asterisco (*) indica cero o mas instancias. En efecto, la equivalencia se produciria utilizando cualquier elemento:

```

mysql> SELECT 'abcdef' REGEXP 'g*';
+-----+
| 'abcdef' REGEXP 'g*' |
+-----+
|                               1 |
+-----+

```

Tambien podriamos utilizar el asterisco para **buscar** correspondencias con el nombre *ian* o escrito como *iaiiin*. El uso de cualquier otra letra tras la *a* haría que fallara la equivalencia. Por ejemplo:

```

mysql> SELECT 'ian' REGEXP 'iai*n';
+-----+
| 'ian' REGEXP 'iai*n' |
+-----+
|                               1 |
+-----+

```

Sin embargo, el problema es que se obtendria el mismo resultado si **estableciéramos** la equivalencia con "iaiiin", ya que el asterisco equivale a cualquier número de caracteres, como se puede ver a **continuación**:

```

mysql> SELECT 'iaiiiiiin' REGEXP 'iai*n';
+-----+
| 'iaiiiiiin' REGEXP 'iai*n' |

```

```

+-----+
|                                     1 |
+-----+

```

Para solucionar este problema, debemos limitar la equivalencia sobre la "i" a una instancia o cero instancias. Para ello, debemos sustituir el asterisco por un signo de **interrogación** invertido.

Como resultado, seguiria equivaliendo a "ian" y a "iain", pero no a "iaiiin", como se puede ver a **continuación**:

```

mysql> SELECT 'iaiiiiiin' REGEXP 'iai?n';
+-----+
| 'iaiiiiiin' REGEXP 'iai?n' |
+-----+
|                               0 |
+-----+

```

El siguiente ejemplo coincide porque {3, } implica que a debe tener lugar **al menos** tres veces:

```

mysql> SELECT 'aaaa' REGEXP 'a{3,}';
+-----+
| 'aaaa' REGEXP 'a{3,}' |
+-----+
|                               1 |
+-----+

```

A **primera** vista, puede que piense que el siguiente ejemplo no coincidiria porque la letra a coincide tres cuatro veces y {3} significa que debe hacerlo tres veces exactamente. Sin embargo, coincide tres veces, asi como dos, una y cuatro veces.

```

mysql> SELECT 'aaaa' REGEXP 'a{3}';
+-----+
| 'aaaa' REGEXP 'a{3}' |
+-----+
|                               1 |
+-----+

```

Si queremos que coincida con la secuencia aaa unicamente, necesitaríamos **utilizar** la siguiente **consulta**:

```

mysql> SELECT 'aaaa' REGEXP '^aaa$';
+-----+
| 'aaaa' REGEXP '^aaa$' |
+-----+
|                               0 |
+-----+

```

La marca de **inserción** (^) fija el **punto inicial** y el simbolo del dolar (\$) fija el **punto final**; si se omite cualquiera de los dos la correspondencia tendra lugar.

En el siguiente ejemplo, no se produce la correspondencia porque {3} sólo equivale a c, no a abc:

```
mysql> SELECT 'abcabcabc' REGEXP 'abc{3}';
+-----+
| 'abcabcabc' REGEXP 'abc{3}' |
+-----+
|                               0 |
+-----+
```

Por lo tanto, la siguiente consulta coincide:

```
mysql> SELECT 'abccc' REGEXP 'abc{3}';
+-----+
| 'abccc' REGEXP 'abc{3}' |
+-----+
|                           1 |
+-----+
```

Para hacer coincidir abcabcabc, necesitara utilizar parentesis, de la siguiente forma:

```
mysql> SELECT 'abcabcabc' REGEXP '(abc){3}';
+-----+
| 'abcabcabc' REGEXP '(abc){3}' |
+-----+
|                               1 |
+-----+
```

Fijese en la diferencia entre el uso de parentesis y corchetes en el siguiente ejemplo.

Los parentesis agrupan la secuencia abc en un conjunto y los corchetes permiten la correspondencia de la letra a, la b o la c, lo que brinda toda una serie de posibilidades, como las siguientes:

```
mysql> SELECT 'abcbbcccc' REGEXP '[abc]{3}';
+-----+
| 'abcbbcccc' REGEXP '[abc]{3}' |
+-----+
|                               1 |
+-----+
```

El siguiente ejemplo utiliza parentesis para obtener el mismo resultado, agrupando las subcadenas alternativas con el caracter barra (|):

```
mysql> SELECT 'abcbbcccc' REGEXP '(a|b|c){3}';
+-----+
| 'abcbbcccc' REGEXP '(a|b|c){3}' |
+-----+
|                               1 |
+-----+
```

Operadores bit a bit

Para **entender** como funcionan las operaciones bit a bit, es necesario conocer un poco **los** numeros booleanos y la aritmetica booleana. Este **tipo** de **consulta** no se suele utilizar, **pero** cualquier **experto** en ciernes que se precie necesitara **in-**cluirlas en su repertorio. En la tabla 3.6 se describen **los** operadores bit a bit.

Tabla 3.6. Operadores de bit a bit

Operador	Sintaxis	Descripción
&	a & b	Operador de bit AND.
	a b	Operador de bit OR.
<<	a << b	Desplaza los bits de a b posiciones hacia la izquierda.
>>	a >> b	Desplaza los bits de a b posiciones hacia la derecha.

El sistema de numeros utilizado habitualmente, denominado *sistema de numeros decimal*, funciona **sobre** la base del numero 10. Tiene **sentido**, ya que despues de **todo** tenemos 10 dedos. Contamos de cero a **nueve** y **al** llegar **al** diez, pasamos a la columna de las decenas y empezamos de nuevo.

00 01 02 03 04 05 06 07 08 09 10

El sistema de numeros decimales consta de diez digitos, que van desde el cero **al** **nueve**. Sin embargo, **los** informaticos han descubierto que a **menudo** resulta util trabajar con un sistema de numeros **basado** en dos digitos, cero y uno. Estos valores representan **los** dos estados de una **conexión** electrica, con carga y si carga.

00 01 10 11

En lugar de pasar a la **columnas** de las decenas cuando se acaban **los** digitos (en el sistema decimal, despues del **nueve** viene el diez), se pasa a la columna de **los** "doses" (en el sistema binario, tras el uno viene el uno cero (**10**), que se indican como "uno cero" para evitar la confusion con el numero decimal).

En el sistema decimal las **columnas** aumentan de tamaio en potencias de diez, como muestra la **figura 3.2**.

10^6	10^5	10^4	10^3	10^2	10^1	10^0
Millones	Cientos de miles	Decenas de miles	Miles	Centenas	Diez	Unos
4	3	9	2	4	2	1

Figura 3.2. Potencias de 10

Por lo **tanto**, el numero, cuatro millones, trescientos noventa y dos mil, cuatrocientos veinte uno, **podría** representarse de la siguiente **forma**:

$$\begin{aligned}
 &4 * 100\ 000\ 000 + \\
 &3 * 100\ 000 + \\
 &9 * 10\ 000 + \\
 &2 * 1000 + \\
 &4 * 100 + \\
 &2 * 10 + \\
 &1 * 1
 \end{aligned}$$

Si puede seguir este ejemplo (para ayudarle, imagine que esta aprendiendo a **contar** con el sistema decimal), le resultara sencillo aplicar **los** mismos conceptos a **los** numeros binarios.

En el sistema binario, las columnnas aumentan de tamaaiio en potencias de dos, **como** muestra la figura 3.3.

2^6	2^5	2^4	2^3	2^2	2^1	2^0
64s	32s	16s	8s	4s	2s	1s
1	1	1	1	1	1	1

Figura 3.3. Potencias de 2

El numero binario anterior (1111111) se lee de la siguiente **forma** **al** convertirlo **al** sistema decimal:

$$\begin{aligned}
 &1 * 64 + \\
 &1 * 32 + \\
 &1 * 16 + \\
 &1 * 8 + \\
 &1 * 4 + \\
 &1 * 2 + \\
 &1 * 1
 \end{aligned}$$

Lo que equivale a $64 + 32 + 16 + 8 + 4 + 2 + 1$, que a su vez es 127.

De **manera** similar, el numero 101001 equivaldria a $1 * 1 + 1 * 8 + 1 * 32 = 41$.

Por lo **tanto**, la conversion de numeros binarios a decimales resulta sencilla y otro **tanto** ocurre **al** revés. Para convertir el numero 18 **al** sistema binario, comience con la figura 3.4.

2^6	2^5	2^4	2^3	2^2	2^1	2^0
64s	32s	16s	8s	4s	2s	1s

Figura 3.4. Paso 1, dibujar las columnnas

Empezando a la izquierda, no hay ningun 64 en 18, ni ningun 32.

Sin embargo, si existe un 16 en 18. Por lo tanto escriba 1 en la columna del 16, como se muestra en la figura 3.5.

2^6	2^5	2^4	2^3	2^2	2^1	2^0
64s	32s	16s	8s	4s	2s	1s
0	0	1				

Figura 3.5. Paso 2, rellenar los valores

Ya hemos dado cuenta de un 16 para el 18, por lo que procedemos a **restar** 16 de 18, lo que nos da como resultado 2. Siguiendo **hacia** la derecha, en 2 no hay ochos, ni cuatros, **sólo** un 2. Y como 2 **menos** 2 es igual a 0, nos detenemos tras escribir un uno en la columna del dos, como se muestra en la figura 3.6.

2^6	2^5	2^4	2^3	2^2	2^1	2^0
64s	32s	16s	8s	4s	2s	1s
0	0	1	0	0	1	0

Figura 3.6. Paso 3, convertir el valor decimal a binario

En el sistema binario, 18 equivale a 10010. Para realizar la conversión de números más grandes, basta con utilizar más **columnas** a la izquierda (para **representar** 128,256, etc.). Los números binarios pueden crecer en **tamaño** rápidamente. Por esta razón, no se suele utilizar este sistema para almacenar números. El sistema octal (con base en 8) y el sistema hexadecimal (con base en 16) son otros dos sistemas **prácticos** de usar.

Volvamos a los operadores bit a bit y **tomemos** dos números, el 9 y el 7. En el sistema binario, equivalen a 1001 y 111, respectivamente. Los operadores bit a bit **operan sobre** los bits individuales del número binario que compone los **números** 9 y 7.

En una operación de bit AND, ambos bits **deben** ser 1 para que el resultado sea 1 (como en una operación AND ordinaria). La figura 3.7 muestra dos números binarios.

1	0	0	1
0	1	1	1
0	0	0	1

Figura 3.7. Operación de bit AND: 9&7

Comenzando por la izquierda, 1 AND 0 es 0, de **manera** que la columna más a la izquierda (la de los ochos) es 0. Moviendonos **hacia** la derecha, 0 AND 1 es 0 y,

de nuevo, 0 AND 1 es 0. Sólo en la **columna** situada en el extremo derecho tenemos que 1 AND 1 es 1.

Por lo **tanto**, el resultado de una operación de bit AND entre 7 y 9 es 1. Por ejemplo:

```
mysql> SELECT 9&7;
+----+
| 9&7 |
+----+
|    1 |
+----+
```

NOTA: Un operador de bit AND da el mismo resultado independientemente de la forma en la que se distribuyan sus elementos; en otras palabras, 9&7 es lo mismo que 7&9.

En el **caso** de un operador de bit OR, basta con que un dígito sea 1 para que el resultado sea 1. Por lo **tanto**, la figura 3.8 muestra una operación de bit OR realizada sobre los mismos números anteriores.

1	0	0	1
0	1	1	1
1	1	1	1

Figura 3.8. Operación de bit OR: 9|7

Todas las **columnas** tienen al menos un 1 presente de manera que el resultado para **cada una de ellas es un 1**, y 1111 es equivalente a 15 en el sistema binario.

```
mysql> SELECT 9|7;
+----+
| 9|7 |
+----+
|   15 |
+----+
```

<< es el operador de desplazamiento **hacia** la izquierda. a << b significa que los bits de a se desplazan a la izquierda en función de las **columnas** b. Por ejemplo, 2 << 1; en el sistema binario 2 es 10. Si desplazamos esta **cifra hacia** la izquierda 1 bit, obtendremos 100, que equivale a 4. Por ejemplo:

```
mysql> SELECT 2 << 1;
+-----+
| 2 << 1 |
+-----+
|      4 |
+-----+

mysql> SELECT 15 << 4;
```



```

+-----+
| 15 << 4 |
+-----+
|      240 |
+-----+

```

Ahora tenemos 15, que equivale a 1111; cuando se desplaza 4 bits a la izquierda, se obtiene 11110000. En la figura 3.9 se convierte este valor al sistema decimal.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128s	64s	32s	16s	8s	4s	2s	1s
1	1	1	1	0	0	0	0

Figura 3.9. Conversion del número binario 11110000 al sistema decimal

A continuación, halle el siguiente total:

$$128 + 64 + 32 + 16 = 240$$

Las operaciones de bits se realizan como BIGINT, lo que significa que existe un límite de 64 bits. Si se realiza un desplazamiento mas alla de 64 bits o se utiliza un numero negativo, se obtendra 0. Por ejemplo:

```

mysql> SELECT 3 << 64;
+-----+
| 3 << 64 |
+-----+
|      0 |
+-----+

```

>> es el operador de desplazamiento a la derecha. a >> b desplaza los bits de a en función de las columnas de b. Los bits desplazados mas alla de las columnas de los unos se pierden. Y, de nuevo, los desplazamientos con numeros negativos devuelven 0. Por ejemplo:

```

mysql> SELECT 3 >> 1;
+-----+
| 3 >> 1 |
+-----+
|      1 |
+-----+

```

En el sistema binario, 3 es 11, desplazado hacia la derecha por 1 con 1 decimal mas alla de la columna de los unos (o 1,1 si lo prefiere, aunque no existen comas decimales en la notación binaria). Como estamos trabajando con enteros, los numeros situados a la derecha de la "coma decimal" se eliminan (quizás deberíamos llamarla coma binaria) para quedarnos con 1 (tanto en el sistema decimal como en el binario).

Por ejemplo:

```
mysql> SELECT 19 >> 3;
+-----+
| 19 >> 3 |
+-----+
|         2 |
+-----+
```

En el ejemplo anterior, 19 equivale a 10011, que desplazado por 3 es 10, eliminando la secuencia 011. Y 10 equivale a 2 en el sistema decimal.

```
mysql> SELECT 4 >> 3;
+-----+
| 4 >> 3 |
+-----+
|         0 |
+-----+
```

Éste se desplaza demasiado a la derecha y pierde todos los bits.

Combinaciones avanzadas

En un capítulo anterior examinamos un tipo de combinación básico de dos tablas. Pero las combinaciones pueden complicarse mucho más y su incorrecta creación es la culpable de la gran mayoría de los problemas de rendimiento graves.

Volvamos a las tablas creadas en el capítulo anterior. Si se saltó dicho capítulo, puede volver a crearlas ejecutando las siguientes instrucciones:

```
CREATE TABLE customer (
  id int(11) default NULL,
  first-name varchar(30) default NULL,
  surname varchar(40) default NULL
) TYPE=MyISAM;

INSERT INTO customer VALUES (1, 'Yvonne', 'Clegg');
INSERT INTO customer VALUES (2, 'Johnny', 'Chaka-Chaka');
INSERT INTO customer VALUES (3, 'Winston', 'Powers');
INSERT INTO customer VALUES (4, 'Patricia', 'Mankunku');

CREATE TABLE sales (
  code int(11) default NULL,
  sales-rep int(11) default NULL,
  customer int(11) default NULL,
  value int(11) default NULL
) TYPE=MyISAM;

INSERT INTO sales VALUES (1, 1, 1, 2000);
INSERT INTO sales VALUES (2, 4, 3, 250);
```

```

INSERT INTO sales VALUES (3, 2, 3, 500);
INSERT INTO sales VALUES (4, 1, 4, 450);
INSERT INTO sales VALUES (5, 3, 1, 3800);
INSERT INTO sales VALUES (6, 1, 2, 500);

```

```

CREATE TABLE sales-rep (
    employee-number int(11) default NULL,
    surname varchar(40) default NULL,
    first_name varchar(30) default NULL,
    commission tinyint(4) default NULL,
    date-joined date default NULL,
    birthday date default NULL
) TYPE=MyISAM;

```

```

INSERT INTO sales-rep VALUES (1, 'Rive', 'Sol', 10,
    '2000-02-15', '1976-03-18');
INSERT INTO sales-rep VALUES (2, 'Gordimer', 'Charlene', 15,
    '1998-07-09', '1958-11-30');
INSERT INTO sales-rep VALUES (3, 'Serote', 'Mike', 10,
    '2001-05-14', '1971-06-18');
INSERT INTO sales-rep VALUES (4, 'Rive', 'Mongane', 10,
    '2002-11-23', '1982-01-04');

```

Comencemos por una combinacion básica:

```

mysql> SELECT sales-rep, customer,value, first_name,surname
    FROM sales, sales-rep WHERE code=1 AND
    sales_rep.employee_number=sales.sales_rep;

```

sales_rep	customer	value	first-name	surname
1	1	2000	Sol	Rive

Como la **relación** entre las tablas `sales_rep` y `sales` se establece a **partir** de `employee_number` o `sales_rep`, estos dos campos **forman** la **condición** de **combinación** de la **cláusula WHERE**.

La **implementación** de una combinacion mas compleja **sobre** las tres tablas no resulta mucho mas compleja. Si desea devolver **los** nombres y apellidos del **comercial** y del **cliente**, asi **como** el **valor** de la **venta**, utilice esta **consulta**:

```

mysql> SELECT sales_rep.first_name,sales_rep.surname,
    value,customer.first_name, customer.surname FROM
    sales,sales_rep,customer WHERE sales_rep.employee_number =
    sales.sales_rep AND customer.id = sales.customer;

```

first-name	surname	value	first-name	surname
Sol	Rive	2000	Yvonne	Clegg
Mike	Serote	3800	Yvonne	Clegg
Sol	Rive	500	Johnny	Chaka-Chaka
Charlene	Gordimer	500	Winston	Powers

Mongane	Rive	250	Winston	Powers
Sol	Rive	450	Patricia	Mankunku

El campo `employee_number` de la tabla `sales_rep` está relacionado con el campo `sales_rep` de la tabla `sales`. Y el campo `id` de la tabla `customer` está relacionado con el campo `customer` de la tabla `sales`. No existen otras condiciones, por lo que esta consulta devuelve todas las ventas para las que existen filas correspondientes en la tabla `sales_rep` y en la tabla `customer`.

Combinaciones internas

Las combinaciones internas son otra forma de describir el primer tipo de combinación aprendido. Las siguientes dos consultas son idénticas:

```
mysql> SELECT first_name,surname,value FROM customer,sales WHERE
id=customer;
```

first-name	surname	value
Yvonne	Clegg	2000
Winston	Powers	250
Winston	Powers	500
Patricia	Mankunku	450
Yvonne	Clegg	3800
Johnny	Chaka-Chaka	500

6 rows in set (0.00 sec)

```
mysql> SELECT first_name,surname,value FROM customer INNER JOIN sales
ON id=customer;
```

first-name	surname	value
Yvonne	Clegg	2000
Winston	Powers	250
Winston	Powers	500
Patricia	Mankunku	450
Yvonne	Clegg	3800
Johnny	Chaka-Chaka	500

Combinaciones por la izquierda (o combinaciones externas por la izquierda)

Imagine que hemos hecho otra venta, con la diferencia de que esta vez el pago se ha realizado al contado y el cliente se ha marchado con los artículos sin que le

hayamos **tomado los datos**. No hay problema porque todavía podemos agregarlos a la tabla `sales` utilizando un valor `NULL` para el cliente.

```
mysql> INSERT INTO sales(code,sales_rep,customer,value) VALUES
(7, 2,NULL,670);
```

Vamos a ejecutar de nuevo la **consulta** que devuelve el valor y los nombres de los comerciales y clientes para cada venta:

```
mysql> SELECT sales_rep.first_name, sales_rep.surname, value,
customer.first_name, customer.surname FROM sales,sales_rep,
customer WHERE sales_rep.employee_number = sales.sales_rep
AND customer.id = sales.customer;
```

first-name	surname	value	first-name	surname
Sol	Rive	2000	Yvonne	Clegg
Mike	Serote	3800	Yvonne	Clegg
Sol	Rive	500	Johnny	Chaka-Chaka
Charlene	Gordimer	500	Winston	Powers
Mongane	Rive	250	Winston	Powers
Sol	Rive	450	Patricia	Mankunku

¿Qué ocurre? ¿Dónde está la nueva venta? El problema está en que como el cliente es `NULL` en la tabla `sales`, la **condición** de combinación no se cumple. Como recordara por una **sección** anterior, el operador `=` excluye a los valores `NULL`. El operador `<=>` no nos servirá de ayuda porque la tabla `customer` no incluye registros `NULL`, de **manera** que no serviría una igualdad que admita valores nulos.

La **solución** en este **caso** consiste en realizar una combinación externa. Esta combinación devolverá un resultado para cada registro coincidente de una tabla, independientemente de que exista un registro asociado en la otra tabla. Por lo **tanto**, aunque el campo `customer` sea `NULL` en la tabla `sales` y no exista **relación** con la tabla `customer`, se devolverá un registro. Una combinación externa por la izquierda devuelve todas las filas coincidentes de la tabla **izquierda**, independientemente de si existe una fila correspondiente en la tabla de la derecha. La **sintaxis** de las combinaciones **externas** por la izquierda es la **siguiente**:

```
SELECT campo1, campo2 FROM tabla1 LEFT JOIN tabla2 ON
campo1=campo2
```

En primer lugar vamos a **probar** con un ejemplo sencillo que realiza una combinación por la izquierda sobre las tablas `customer` y `sales`.

```
mysql> SELECT first_name,surname,value FROM sales LEFT JOIN customer
ON id=customer;
```

first-name	surname	value
Yvonne	Clegg	2000
Winston	Powers	250
Winston	Powers	500
Patricia	Mankunku	450
Yvonne	Clegg	3800
Johnny	Chaka-Chaka	500
NULL	NULL	670

Se devuelven los siete registros, como se esperaba.

El orden de la tabla es importante en una combinacion por la izquierda. La tabla desde la que se devuelven todas las **filas** coincidentes debe ser la tabla de la izquierda (antes de las palabras clave `LEFT JOIN`). Si invertimos el orden e intentamos lo siguiente:

```
mysql> SELECT first_name,surname,value FROM customer LEFT JOIN sales
ON id=customer;
```

first-name	surname	value
Yvonne	Clegg	2000
Yvonne	Clegg	3800
Johnny	Chaka-Chaka	500
Winston	Powers	250
Winston	Powers	500
Patricia	Mankunku	450

solo obtendriamos 6 registros. Como la tabla izquierda es la tabla de clientes en esta consulta y la operacion de combinacion solo busca coincidencias en los registros de la tabla izquierda, no se devuelve el registro de **ventas** con el cliente NULL (lo que significa que no hay relación con la tabla `customer`).

NOTA: Las combinaciones por la izquierda se solian llamar **combinaciones exteriores** por la izquierda en el pasado. Por familiaridad, MySQL sigue aceptando este término.

Obviamente, esta operacion se puede extender a una tercera tabla para dar respuesta a la consulta original (nombres de clientes y comerciales asi como valores de **ventas**, para **cada** venta). Pruebe a crearla. A **continuación**, se incluye una **opción**:

```
mysql> SELECT sales_rep.first_name, sales_rep.surname, value,
customer.first_name, customer.surname FROM sales LEFT JOIN
sales_rep ON sales_rep.employee-number = sales.sales_rep
LEFT JOIN customer ON customer.id = sales.customer;
```

first-name	surname	value	first-name	surname
Sol	Rive	2000	Yvonne	Clegg
Mongane	Rive	250	Winston	Powers
Charlene	Gordimer	500	Winston	Powers
Sol	Rive	450	Patricia	Mankunku
Mike	Serote	3800	Yvonne	Clegg
Sol	Rive	500	Johnny	Chaka-Chaka
Charlene	Gordimer	670	NULL	NULL

Combinaciones por la derecha (o combinaciones externas por la derecha)

Las combinaciones por la derecha son exactamente iguales a las combinaciones por la izquierda, con la salvedad de que el orden de la combinación se invierte. Para recuperar el nombre de todos los clientes para cada venta, incluyendo aquellas de las que no se dispongan de datos de los clientes, debemos colocar la tabla `sales` en la parte derecha de la combinación.

```
mysql> SELECT first_name,surname,value FROM customer RIGHT JOIN
sales ON id=customer;
```

first-name	surname	value
Yvonne	Clegg	2000
Winston	Powers	250
Winston	Powers	500
Patricia	Mankunku	450
Yvonne	Clegg	3800
Johnny	Chaka-Chaka	500
NULL	NULL	670

TRUCO: Si duda en cuanto al lado en el que colocar la tabla, recuerde que una combinación por la derecha lee todos los registros de la tabla derecha, incluyendo los nulos, y una combinación por la izquierda lee todos los registros por la izquierda desde la tabla izquierda, incluyendo los nulos.

Combinaciones externas completas

En el momento de escribir estas líneas, MySQL no admite las combinaciones externas completas. En estas combinaciones, cada registro de la primera tabla, incluyendo aquellos que no tengan una correspondencia en la segunda, se devuelve junto a cada registro de la segunda tabla, incluyendo aquellos sin correspondencias en la primera. Equivalen a una combinación por la izquierda

da y a una combinacion por la derecha. MySQL no tardara en incorporar este tipo de combinaciones, por lo que es aconsejable que consulte la documentacion mas reciente. La sintaxis es la misma que para las otras combinaciones:

```
SELECT campo1,campo2 FROM tabla1 FULL OUTER JOIN tabla2
```

Combinaciones naturales y la palabra clave USING

El campo `id` de la tabla `customer` y el campo `customer` de la tabla `sales` estan relacionados.

Si les asignaramos el mismo nombre, podriamos utilizar varios metodos de SQL que permiten que las instrucciones JOIN resulten mas sencillas de manejar.

Para demostrarlo, vamos a convertir `sales.customer` en `sales.id`:

```
mysql> ALTER TABLE sales CHANGE customer id INT;
```

Ahora, como las dos tablas constan de campos con nombres identicos, podemos realizar una combinacion natural, que busca campos con nombres iguales sobre los que realizar una union:

```
mysql> SELECT first_name,surname,value FROM customer NATURAL JOIN
sales;
```

first_name	surname	value
Yvonne	Clegg	2000
Winston	Powers	250
Winston	Powers	500
Patricia	Mankunku	450
Yvonne	Clegg	3800
Johnny	Chaka-Chaka	500

Esta secuencia es identica a la siguiente:

```
mysql> SELECT first_name,surname,value FROM customer INNER JOIN
sales ON customer.id=sales.id;
```

first_name	surname	value
Yvonne	Clegg	2000
Winston	Powers	250
Winston	Powers	500
Patricia	Mankunku	450
Yvonne	Clegg	3800
Johnny	Chaka-Chaka	500

Solo existe un campo identico en **ambas** tablas, pero si hubiera otros, **cada uno** de ellos se convertiria en parte de la **condición** de combinacion.

Las combinaciones naturales **también** pueden ser por la izquierda o por la derecha. Las siguientes dos instrucciones son identicas:

```
mysql> SELECT first_name,surname,value FROM customer LEFT JOIN sales
ON customer.id=sales.id;
```

first_name	surname	value
Yvonne	Clegg	2000
Yvonne	Clegg	3800
Johnny	Chaka-Chaka	500
Winston	Powers	250
Winston	Powers	500
Patricia	Mankunku	450

```
mysql> SELECT first_name,surname,value FROM customer NATURAL
LEFT JOIN sales;
```

first_name	surname	value
Yvonne	Clegg	2000
Yvonne	Clegg	3800
Johnny	Chaka-Chaka	500
Winston	Powers	250
Winston	Powers	500
Patricia	Mankunku	450

La palabra clave **USING** brinda un mayor control sobre una combinacion natural. Si dos tablas **constan** de varios campos identicos, esta **palabra clave permite especificar** aquellos que se **utilizarán como condiciones** de combinacion. Por ejemplo, si tomamos dos tablas A y B, con los mismos campos a, b, c, d, las siguientes instrucciones resultaran identicas:

```
SELECT * FROM A LEFT JOIN B USING (a,b,c,d)
SELECT * FROM A NATURAL LEFT JOIN B
```

La palabra clave **USING** brinda una mayor flexibilidad porque **permite utilizar** los campos deseados en la combinacion. Por ejemplo:

```
SELECT * FROM A LEFT JOIN B USING (a,d)
```

NOTA: En las combinaciones naturales, cuando se habla de campos idénticos, se hace referencia al nombre de los campos, no a su tipo. Los campos pueden ser de tipo INT y DECIMAL o incluso INT y VARCHAR, siempre y cuando tengan el mismo nombre.

Recuperación de los datos encontrados en una tabla pero no en la otra

Hasta el momento hemos recuperado las filas que aparecen en **ambas** tablas en las que se establecía una combinación interna. En las combinaciones externas, también devolvíamos los registros de una tabla en la que no se encontraban correspondencias en la segunda.

A menudo resulta útil realizar la operación **inversa** y devolver únicamente los resultados encontrados en una tabla pero no en la otra. Para demostrarlo, en primer lugar vamos a agregar un nuevo comercial:

```
mysql> INSERT INTO sales_rep VALUES(5, 'Jomo', 'Iglesund', 10, '2002-11-29', '1968-12-01');
```

A continuación, si realiza una combinación interna, puede recuperar todos los comerciales que **hayan** realizado una venta:

```
mysql> SELECT DISTINCT first_name,surname FROM sales_rep
INNER JOIN sales ON sales_rep=employee_number;
+-----+-----+
| first_name | surname |
+-----+-----+
| Sol        | Rive    |
| Mongane    | Rive    |
| Charlene   | Gordimer|
| Mike       | Serote  |
+-----+-----+
```

Se utiliza la **palabra** clave DISTINCT para evitar duplicados porque hay comerciales que han realizado mas de una venta.

Pero la operación **inversa** también resulta útil. Suponga que su jefe está negro con las **ventas** y ha decidido que van a rodar cabezas.

Nos pide que busquemos los comerciales que no han realizado ninguna venta.

Puede **buscar** esta **información** examinando los comerciales que aparecen en la tabla `sales_rep` sin una entrada correspondiente en la tabla `sales`.

```
mysql> SELECT first_name,surname FROM sales_rep LEFT JOIN sales
ON sales_rep=employee_number WHERE sales_rep IS NULL;
+-----+-----+
| first_name | surname |
+-----+-----+
| Igesund    | Jomo    |
+-----+-----+
```

Necesitará realizar una combinación por la izquierda (externa, no interna) porque **sólo** las combinaciones **externas** devuelven todos los registros sin correspondencias (o valores nulos).

Combinación de resultados con UNION

MySQL 4 introdujo la instrucción de SQL ANSI UNION, que combina los resultados de diferentes instrucciones SELECT. Cada instrucción debe constar del mismo número de columnas.

Para demostrar el uso de esta instrucción, vamos a crear otra tabla que contenga una lista de clientes recibida del antiguo propietario de su establecimiento:

```
mysql> CREATE TABLE old_customer(id int, first_name varchar(30),
mysql> INSERT INTO old_customer VALUES (5432, 'Thulani', 'Salie'),
(2342, 'Shahiem', 'Papo');
```

A continuación, para obtener una lista con todos los clientes, tanto los antiguos como los nuevos, puede utilizar la siguiente instrucción:

```
mysql> SELECT id, first_name, surname FROM old_customer UNION SELECT
id, first_name, surname FROM customer;
+----+-----+-----+
| id  | first-name | surname  |
+----+-----+-----+
| 5432 | Thulani    | Salie    |
| 2342 | Shahiem    | Papo     |
| 1    | Yvonne     | Clegg    |
| 2    | Johnny     | Chaka-Chaka |
| 3    | Winston    | Powers   |
| 4    | Patricia   | Mankunku |
+----+-----+-----+
```

También puede ordenar el resultado de la forma habitual. Sólo debe tener cuidado al decidir si aplicar la cláusula ORDER BY a toda la unión o sólo a una selección.

```
mysql> SELECT id, first_name, surname FROM old_customer UNION SELECT
id, first_name, surname FROM customer ORDER BY surname, first_name;
+----+-----+-----+
| id  | first-name | surname  |
+----+-----+-----+
| 2    | Johnny     | Chaka-Chaka |
| 1    | Yvonne     | Clegg    |
| 4    | Patricia   | Mankunku |
| 2342 | Shahiem    | Papo     |
| 3    | Winston    | Powers   |
| 5432 | Thulani    | Salie    |
+----+-----+-----+
```

La ordenación se realiza sobre toda la unión. Si sólo quisiéramos ordenar la segunda selección, necesitaríamos utilizar paréntesis.

```
mysql> SELECT id, first_name, surname FROM old-customer UNION
(SELECT id, first-name, surname FROM customer ORDER BY surname,
 first-name);
+----+-----+-----+
| id  | first-name | surname  |
+----+-----+-----+
| 5432 | Thulani    | Salie    |
| 2342 | Shahiem    | Papo     |
| 2    | Johnny     | Chaka-Chaka |
| 1    | Yvonne     | Clegg    |
| 4    | Patricia   | Mankunku |
| 3    | Winston    | Powers   |
+----+-----+-----+
```

TRUCO: Cuando exista una posible ambigüedad, como dónde aplicar la ordenación, utilice paréntesis. De esta forma quedará clara la parte que se ordena y ayudaremos a otras personas a interpretar nuestras instrucciones. No asuma que todo el mundo sabe tanto como usted.

De manera predeterminada, la instrucción UNION no devuelve resultados duplicados (de manera similar a la palabra clave DISTINCT) . Puede modificar este comportamiento especificando que todos los resultados se devuelvan con la palabra clave ALL:

```
mysql> SELECT id FROM customer UNION ALL SELECT id FROM sales;
+----+
| id  |
+----+
| 1   |
| 2   |
| 3   |
| 4   |
| 1   |
| 3   |
| 3   |
| 4   |
| 1   |
| 2   |
| NULL |
+----+
```

El uso de UNION requiere cierta reflexión. Puede unir fácilmente campos no relacionados siempre y cuando los campos devueltos en cada operación de selección y los tipos de datos sean iguales. MySQL devolverá estos datos sin problemas aunque no tengan mucho sentido.

```
mysql> SELECT id, surname FROM customer UNION ALL SELECT value,
 sales-rep FROM sales;
```

```

+-----+-----+
| id    | surname      |
+-----+-----+
| 1    | Clegg        |
| 2    | Chaka-Chaka |
| 3    | Powers       |
| 4    | Mankunku     |
| 2000 | 1            |
| 250  | 4            |
| 500  | 2            |
| 450  | 1            |
| 3800 | 3            |
| 500  | 1            |
| 670  | 2            |
+-----+-----+

```

Subselecciones

Muchas consultas realizan una **operación** de seleccion dentro de una seleccion. La **implementación** de las subselecciones esta programada para la version 4.1. Hasta ahora, MySQL no permitia las subselecciones, en parte por razones de diseiio (son **menos** eficientes que las alternativas, como veremos mas adelante) y en parte porque se encontraban en la parte baja de la lista de **los 1001** elementos "importantes" que implementar. **Ahora** que MySQL esta a **punto** de integrarlas, necesitaremos ver como **funcionan**.

Como escribir subselecciones como combinaciones

Spongamos una consulta en la que deseemos recuperar todos los comerciales que han realizado una venta por un valor superior a 1.000 dolares. Si puede ejecutar una subseleccion, pruebe a utilizar la siguiente secuencia:

```

mysql> SELECT first_name,surname FROM sales_rep WHERE
       sales_rep.employee_number IN (SELECT code FROM sales WHERE
       value>1000);
+-----+-----+
| first-name | surname |
+-----+-----+
| Sol       | Rive    |
+-----+-----+

```

La hazaiia sólo la ha logrado Sol Rive.

La consulta se realiza resolviendo en primer lugar la seleccion interna, es decir, llevando **cabo** el siguiente **paso** en primer lugar:

```

mysql> SELECT id FROM sales WHERE value>1000;

```

```

+----+
| id  |
+----+
|    1 |
|    1 |
+----+

```

y, a continuación, el resto:

```

mysql> SELECT first-name, surname FROM sales-rep WHERE
sales-rep.employee-number IN (1);
+-----+-----+
| first-name | surname |
+-----+-----+
| Sol       | Rive    |
+-----+-----+

```

Pero ya conocemos otra forma mejor de realizar esta consulta mediante una combinación:

```

mysql> SELECT DISTINCT first-name,surname FROM sales-rep INNER
JOIN sales ON employee_number=id WHERE value>1000;
+-----+-----+
| first-name | surname |
+-----+-----+
| Sol       | Rive    |
| Sol       | Rive    |
+-----+-----+

```

o, alternativamente:

```

mysql> SELECT DISTINCT first_name,surname FROM sales-rep,sales WHERE
sales.id=sales_rep.employee_number AND value>1000;
+-----+-----+
| first-name | surname |
+-----+-----+
| Sol       | Rive    |
| Sol       | Rive    |
+-----+-----+

```

Esta opción es mejor porque las combinaciones suelen ser resultar mas eficientes para realizar consultas y los resultados se recuperan con mayor rapidez. Puede que en una base de datos pequeña no se note mucho la diferencia, pero en tablas grandes con mucho tráfico en las que el rendimiento resulta fundamental, nos interesa aprovechar cada micro segundo que podamos obtener de MySQL.

Para recuperar todos los comerciales que todavía no hayan realizado una venta, puede utilizar una subselección, si su DBMS lo permite, de la siguiente forma:

```

mysql> SELECT first_name,surname FROM sales-rep WHERE employee_number
NOT IN (SELECT DISTINCT code from sales);

```

```

+-----+-----+
| first-name | surname |
+-----+-----+
| Igesund   | Jomo    |
+-----+-----+

```

Pero ya conocemos otra forma mejor:

```

mysql> SELECT DISTINCT first-name,surname FROM sales-rep LEFT
JOIN sales ON sales-rep=employee_number WHERE
sales-rep IS NULL;
+-----+-----+
| first-name | surname |
+-----+-----+
| Igesund   | Jomo    |
+-----+-----+

```

Como agregar registros a una tabla desde otras tablas con INSERT SELECT

La instrucción `INSERT` tambien permite agregar registros, o partes de registros, de otras tablas.

Por ejemplo, supongamos que desea crear una nueva tabla que contenga los nombres de los clientes y los valores de todas las compras realizadas. La consulta para devolver los resultados deseados sera la siguiente:

```

mysql> SELECT first-name,surname,SUM(value) FROM sales NATURAL JOIN
customer GROUP BY first-name, surname;
+-----+-----+-----+
| first-name | surname      | SUM(value) |
+-----+-----+-----+
| Johnny    | Chaka-Chaka |          500 |
| Patricia  | Mankunku    |          450 |
| Winston   | Powers      |          750 |
| Yvonne    | Clegg       |         5800 |
+-----+-----+-----+

```

En primer lugar, necesitaremos crear la tabla para que reciba los siguientes resultados:

```

mysql> CREATE TABLE customer_sales_values(first_name
VARCHAR(30), surname VARCHAR(40), value INT);

```

A continuación, se insertan los resultados en la tabla:

```

mysql> INSERT INTO customer_sales_values(first_name,surname,value)
SELECT first_name,surname, SUM(value) FROM sales NATURAL JOIN
customer GROUP BY first-name, surname;

```

La tabla `customer_sales_values` contiene ahora los siguientes elementos:

```
mysql> SELECT * FROM customer_sales_values;
+-----+-----+
| first-name | surname | value |
+-----+-----+
| Johnny    | Chaka-Chaka | 500 |
| Patricia  | Mankunku   | 450 |
| Winston   | Powers     | 750 |
| Yvonne    | Clegg      | 5800 |
+-----+-----+
```

Mas sobre la agregacion de registros

INSERT tambien permite una sintaxis similar a la utilizada por una instruccion UPDATE.

En lugar de utilizar la siguiente secuencia:

```
mysql> INSERT INTO customer-sales-values(first_name, surname, value)
VALUES('Charles', 'Dube', 0);
```

podriamos utilizar esta otra:

```
mysql> INSERT INTO customer_sales_values SET first-name =
'Charles', surname='Dube', value=0;
```

Tambien se puede llevar a cabo una forma limitada de calculo al agregar registros. Para demostrarlos, agregue otro campo sobre la tabla `customer-sales-value`:

```
mysql> ALTER TABLE customer-sales-values ADD value2 INT;
```

A **continuación**, puede insertar elementos dentro de esta tabla y completar `value2` con el doble del valor:

```
mysql> INSERT INTO customer-sales-values(first_name, surname,
value, value2) VALUES('Gladys', 'Malherbe', 5, value*2);
```

Este registro contiene los siguientes elementos:

```
mysql> SELECT * FROM customer_sales_values WHERE
first_name='Gladys';
+-----+-----+-----+-----+
| first-name | surname | value | value2 |
+-----+-----+-----+-----+
| Gladys    | Malherbe | 5     | 10    |
+-----+-----+-----+-----+
```


Mas sobre como eliminar registros (DELETE y TRUNCATE)

Ya sabe como eliminar un registro con la instruccion DELETE. Y ya sabra que si no utiliza una cláusula WHERE se eliminarán todos los registros. Un problema asociado a la eliminación de registros utilizando este método es que puede resultar muy lento si la tabla es de gran tamaño. Por suerte, existe otra forma de realizar dicha operación.

En primer lugar vamos a eliminar todos los registros de la tabla `customer_sales_value` con la instrucción DELETE.

```
mysql> DELETE FROM customer_sales_values;
Query OK, 7 rows affected (0.00 sec)
```

La forma mas rapida de eliminar estos valores consiste en utilizar la instruccion TRUNCATE. A continuación, volveremos a agregar los registros y utilizaremos dicha instruccion:

```
mysql> INSERT INTO customer_sales_values(first_name, surname, value,
value2) VALUES('Johnny', 'Chaka-Chaka', 500, NULL), ('Patricia',
'Mankunku', 450, NULL), ('Winston', 'Powers', 750, NULL), ('Yvonne',
'Clegg', 5800, NULL), ('Charles', 'Dube', 0, NULL), ('Charles',
'Dube', 0, NULL), ('Gladys', 'Malherbe', 5, 10);
```

```
mysql> TRUNCATE customer_sales_values;
Query OK, 0 rows affected (0.00 sec)
```

Observe la diferencia entre el resultado de las dos instrucciones. DELETE nos informa del numero de filas que se han eliminado, cosa que no hace TRUNCATE. Por tanto, TRUNCATE elimina el conjunto completo sin contar los elementos eliminados. Para realizar esta tarea, elimina la tabla y vuelve a crearla.

Variable de usuario

MySQL consta de una funcion que permite almacenar valores como variables temporales para poder utilizarlas en una instruccion posterior. En la gran mayoría de los casos se utiliza un lenguaje de programacion para realizar este tipo de acciones (como se vera en un capitulo posterior), pero las variables de MySQL resultan utiles cuando se trabaja en la linea de comandos de MySQL.

El valor de la variable se establece con la instruccion SET o en una instruccion SELECT con `:=`.

Para recuperar todos los comerciales con una comision mas alta que la comision media, podemos utilizar las siguientes secuencias:

```
mysql> SELECT @avg := AVG(commission) FROM sales-rep;
+-----+
| @avg := AVG(commission) |
+-----+
|                11.0000 |
+-----+
```

```
mysql> SELECT surname,first_name FROM sales-rep WHERE
commission>@avg;
+-----+-----+
| surname | first-name |
+-----+-----+
| Gordimer | Charlene |
+-----+-----+
```

El simbolo @ indica que se trata de una variable de MySQL. La comision media se almacena en la variable @avg, a la que se puede acceder en un momento posterior.

Tambien puede establecer una variable de manera específica. Por ejemplo, en lugar de repetir un calculo complejo cada vez que resulte necesario, puede establecer la variable con el valor deseado como paso previo:

```
mysql> SET @result = 22/7*33.23;

mysql> SELECT @result;
+-----+
| @result |
+-----+
| 104.43714285714 |
+-----+
```

Las variables de usuario pueden ser cadenas, enteros y numeros decimales. Se les puede asignar una expresion (excluyendo aquellos lugares en los que se necesitan determinados valores literales, como en la clausula LIMIT). Sin embargo, no se pueden utilizar para sustituir parte de la consulta, como para reemplazar el nombre de una tabla.

Por ejemplo:

```
mysql> SET @t = 'sales';
mysql> SELECT * FROM @t;
ERROR 1064: You have an error in your SQL syntax near '@t' at
line 1

mysql> SET @v=2;
mysql> SELECT * FROM sales LIMIT 0,@v;
ERROR 1064: You have an error in your SQL syntax near '@v' at
line 1
```

Las variables de usuario se establecen en un subproceso dado (o **conexión** a un servidor) y ningun otro proceso puede acceder a **ellas**. Al cerrar el proceso o al **peder la conexión las variables dejan de estar asignadas**.

Ejecute la siguiente secuencia desde el primer subproceso, ventana 1:

```
mysql> SET @a = 1;
```

```
mysql> SELECT @a;
+----+
| @a  |
+----+
|    1 |
+----+
```

A esta variable no se puede acceder desde otro subproceso. Ejecute la **siguien- te** secuencia desde la ventana 2:

```
mysql> SELECT @a;
+----+
| @a  |
+----+
| NULL |
+----+
```

Si cierra la **conexión** y vuelve a conectarse a la ventana 1, MySQL **habrá** vaciado la variable de ventana 1, de la siguiente **forma**:

```
mysql> exit
```

```
% mysql firstdb
```

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14 to server version: 4.0.1-alpha-
max
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the buffer
```

```
mysql> SELECT @a;
+----+
| @a  |
+----+
| NULL |
+----+
```

Fijese en que en una instrucción SELECT, la **cláusula WHERE** se **calcula** en primer lugar y, a **continuación**, se lista el campo. Si no se devuelve ningun **regis- tro**, la variable de usuario no se establecera para dicha instrucción. Por ejemplo, **como** esta instrucción no ha devuelto ningun registro, la variable de usuario no se establecera:

```
mysql> SELECT @a:=2 FROM sales WHERE value>10000;
Empty set (0.00 sec)
```

```
mysql> SELECT @a;
+----+
| @a  |
+----+
| NULL |
+----+
```

Sin embargo, si recupera al menos un registro, la variable de usuario se establecera correctamente:

```
mysql> SELECT @a:=2 FROM sales WHERE value>2000;
+----+
| @a:=2 |
+----+
|      2 |
+----+
```

```
mysql> SELECT @a;
+----+
| @a  |
+----+
| 2   |
+----+
```

De manera similar, una variable de usuario establecida en la lista de campos no se puede utilizar como condicion. La siguiente instrucción no funcionara porque la variable de usuario no ha sido establecida para dicha condicion:

```
mysql> SELECT @d:=2000,value FROM sales WHERE value>@d;
Empty set (0.00 sec)
```

A continuación tendremos que establecer la variable especificamente antes que la consulta de la siguiente forma:

```
mysql> SET @d=2000;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @d,value FROM sales WHERE value>@d;
+----+----+
| @d  | value |
+----+----+
| 2000 | 3800 |
+----+----+
```

Tambien puede establecer una variable en la propia clausula WHERE. Tenga en cuenta que no se reflejara correctamente en la lista de campos a menos que vuelva a establecer las variables. Por ejemplo:

```
mysql> SELECT @e,value FROM sales WHERE value>(@e:=2000);
+----+----+
| @e  | value |
+----+----+
```

```
| NULL | 3800 |
+-----+-----+
```

Para reflejarlo correctamente, deberá establecer la variable de nuevo en la lista de campos:

```
mysql> SELECT @f:=2000,value FROM sales WHERE value>(@f:=2000);
+-----+-----+
| @f:=2000 | value |
+-----+-----+
|      2000 | 3800 |
+-----+-----+
```

Ésta no es una forma elegante de implementar variables de usuario; en su lugar, establézcalas de manera separada de antemano.

ADVERTENCIA: Recuerde que las variables de usuario se mantienen durante el periodo de vida del subproceso. Puede que no obtenga los resultados esperados si olvida inicializar una variable de usuario.

Ejecucion de instrucciones SQL almacenadas en archivos

A menudo se suelen guardar grupos de instrucciones SQL en un archivo para volver a utilizarlas. Puede ejecutar estos comandos desde la línea de comandos de su sistema operativo de forma sencilla. Esta operación se conoce como ejecutar MySQL en *modo de procesamiento por lotes* (en contraposición a hacerlo en modo interactivo al establecer una conexión al servidor y escribir los comandos deseados). Cree un archivo de texto `test.sql` que contenga las siguientes dos líneas:

```
INSERT INTO customer(id,first_name,surname)
VALUES(5,'Francois','Papo');
INSERT INTO customer(id,first_name,surname)
VALUES(6,'Neil','Beneke');
```

Puede ejecutar estas dos instrucciones desde la línea de comandos de su sistema operativo de la siguiente forma:

```
% mysql firstdb < test.sql
```

Recuerde agregar un nombre de anfitrión, un nombre de usuario y una contraseña si resultara necesario. (Este ejemplo muestra la versión abreviada para facilitar la lectura.)

Si establece una conexión al servidor MySQL ahora, verá que se han añadido estos dos registros:

```
mysql> SELECT * FROM customer;
```

id	first-name	surname
1	Yvonne	Clegg
2	Johnny	Chaka-Chaka
3	Winston	Powers
4	Patricia	Mankunku
5	Francois	Papo
6	Neil	Beneke

Si alguna de las líneas del archivo contiene un error SQL, MySQL interrumpirá el procesamiento del archivo. Modifique `test.sql` como se indica a continuación. Agregamos la instrucción `DELETE` a la parte superior para que si volvemos a ejecutar el conjunto de instrucciones varias veces, no nos quedemos atascados con registros duplicados:

```
DELETE FROM customer WHERE id>=6;
INSERT INTO customer(id,first_name,surname)
VALUES(6,'Neil','Beneke');
INSERT INTO customer(id,first_name,surname)
VALUES(,'Sandile','Cohen');
INSERT INTO customer(id,first_name,surname)
VALUES(7,'Winnie','Dlamini');
```

Al ejecutar esta secuencia desde la línea de comandos, MySQL devolverá un error:

```
<L% mysql firstdb < test.sql
ERROR 1064 at line 2: You have an error in your SQL syntax near
  'Sandile','Cohen')' at line 1
```

Si examina los contenidos de la tabla `customer`, verá que el primer registro se ha insertado correctamente, pero como la segunda línea contiene un error (el campo `id` no se ha especificado), MySQL detuvo el procesamiento en dicho punto:

```
mysql> SELECT * FROM customer;
+----+-----+-----+
| id | first-name | surname |
+----+-----+-----+
| 1 | Yvonne | Clegg |
| 2 | Johnny | Chaka-Chaka |
| 3 | Winston | Powers |
| 4 | Patricia | Mankunku |
| 5 | Francois | Papo |
| 6 | Neil | Beneke |
+----+-----+-----+
```

Puede obligar a MySQL a continuar procesando la acción aunque existan errores con la opción `force` (en un capítulo anterior encontrará una lista completa de las opciones de MySQL):

```
% mysql -f firstdb < test.sql
```

```
ERROR 1064 at line 2: You have an error in your SQL syntax near
" Sandile','Cohen')' at line 1
```

Aunque el error sigue apareciendo, todos los registros validos se han insertado como puede observar si visualiza la tabla de nuevo:

```
mysql> SELECT * FROM customer;
+----+-----+-----+
| id  | first-name | surname  |
+----+-----+-----+
| 1  | Yvonne     | Clegg    |
| 2  | Johnny     | Chaka-Chaka |
| 3  | Winston    | Powers    |
| 4  | Patricia   | Mankunku  |
| 5  | Francois   | Papo      |
| 7  | Winnie     | Dlamini   |
| 6  | Neil       | Beneke    |
+----+-----+-----+
```

Redireccionamiento de la salida hacia un archivo

Puede capturar el resultado en otro archivo.

Por ejemplo, en lugar de ejecutar la instrucción `SELECT` desde la línea de comandos, puede agregarla al archivo original y dirigir los resultados de la consulta a un tercer archivo. Si modifica el archivo `test.sql` de la siguiente forma:

```
DELETE FROM customer WHERE id>=6;
INSERT INTO customer(id,first_name,surname)
VALUES (6,'Neil','Beneke');
INSERT INTO customer(id,first_name,surname)
VALUES (7,'Winnie','Dlamini');
SELECT * FROM customer;
```

puede dirigir los resultados a un archivo, `test_output.txt`, como se indica en la siguiente secuencia:

```
% mysql firstdb < test.sql > test-output.txt
The file test_output.txt now contains the following:
id      first-name      surname
1       Yvonne          Clegg
2       Johnny          Chaka-Chaka
3       Winston         Powers
4       Patricia        Mankunku
5       Francois        Papo
7       Winnie          Dlamini
6       Neil            Beneke
```

Fijese en que el resultado no es exactamente el mismo que se obtendría en caso de ejecutar la consulta en modo interactivo. Los datos están separados por tabuladores y no se generan líneas de formato.

Para activar el formato interactivo en el archivo de salida, puede utilizar la opción `-t`, por ejemplo:

```
% mysql -t firstdb < test.sql > test_output.txt
```

Como usar los archivos desde la línea de comandos MySQL

También puede ejecutar instrucciones SQL almacenadas dentro de un archivo desde la línea de comandos de MySQL con el comando `SOURCE`:

```
mysql> SOURCE test.sql
Query OK, 2 rows affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

+----+-----+-----+
| id  | first-name | surname |
+----+-----+-----+
| 1  | Yvonne     | Clegg  |
| 2  | Johnny     | Chaka-Chaka |
| 3  | Winston    | Powers |
| 4  | Patricia   | Mankunku |
| 5  | Francois   | Papo   |
| 7  | Winnie     | Dlamini |
| 6  | Neil      | Beneke |
+----+-----+-----+
7 rows in set (0.00 sec)
```

Puede eliminar los registros agregados a través de los archivos de texto ya que no los necesitaremos posteriormente:

```
mysql> DELETE FROM customer WHERE id > 4;
```

Entre las razones más destacadas para utilizar el modo de procesamiento por lotes se pueden citar las siguientes:

- Puede utilizar las instrucciones SQL si las necesita de nuevo
- Puede copiar y enviar archivos a otras personas.
- Resulta sencillo realizar cambios en un archivo si surgieran errores.
- En ocasiones resulta necesario utilizar el modo de procesamiento por lotes, por ejemplo si queremos ejecutar determinados comandos de SQL de manera repetida en un momento dado del día (por ejemplo, con la instrucción `crontab` de Unix).

Transacciones y bloqueos

Las consultas sobre bases de datos se ejecutan una después de otra. En el caso de un sitio Web que sirva páginas, da lo mismo el orden en el que la base de datos

realice las consultas, siempre y cuando lo haga rápidamente. Sin embargo, ciertos tipos de consultas necesitan realizarse en un **orden** dado, como las que **dependen** de los resultados de una **consulta** anterior, o grupos de actualizaciones que **necesitan** realizarse en conjunto. Todos los tipos de tabla pueden utilizar la función de bloqueo, pero sólo los tipos **InnoDB** y **BDB** disponen de funciones transaccionales integradas. En esta **sección** se analizan los distintos mecanismos de transacción y bloqueo.

Las transacciones en las tablas InnoDB

La **potencia** de las tablas InnoDB **procede** del uso de *transacciones* o **instrucciones SQL** agrupadas en una. Un ejemplo típico son las transacciones bancarias. Por ejemplo, si se transfiere una **cantidad** de dinero desde la cuenta de una persona a otra, se realizarán **al menos** dos consultas:

```
UPDATE person1 SET balance = balance-transfer_amount;
UPDATE person2 SET balance = balance+transfer_amount;
```

El proceso parece claro, pero ¿qué ocurriría si algo sale **mal** y el sistema falla entre las dos consultas sin que llegue a completarse la segunda? Se **habrán retirado** los fondos de la cuenta de la **primera** persona, que creará que el **pago** se ha realizado. Sin embargo, la segunda persona no estará muy **contenta** porque el **pago** no se ha realizado. En este tipo de situaciones, resulta fundamental **asegurarse** de que **ambas** consultas se llevan a **cabo** o que no lo hacen ninguna de las dos. Para **ello**, se empaquetan en lo que se conoce como una *transacción*, con una instrucción **BEGIN** para indicar el **inicio** de la transacción y una instrucción **COMMIT** para indicar el **final**. Sólo tras procesar la instrucción **COMMIT**, las consultas se **habrán** convertido en permanentes. Si algo sale **mal** entre medias, podemos utilizar el comando **ROLLBACK** para **invertir** la parte incompleta de la transacción.

Vamos a ejecutar algunas consultas para comprobar su funcionamiento. **Tendra** que crear la tabla si no lo hizo en el capítulo anterior:

```
mysql> CREATE TABLE innotest (f1 INT,f2 CHAR(10),INDEX
(f1))TYPE=InnoDB;
Query OK, 0 rows affected (0.10 sec)
```

```
mysql> INSERT INTO innotest(f1) VALUES (1);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT f1 FROM innotest;
+----+
| f1  |
+----+
|    1 |
+----+
1 row in set (0.21 sec)
```

Nada especial hasta el momento. A continuacion, procederemos a empaquetar una consulta en las instrucciones BEGIN/COMMIT:

```
mysql> BEGIN;
Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO innotest(f1) VALUES (2);
Query OK, 1 row affected (0.05 sec)

mysql> SELECT f1 FROM innotest;
+----+
| f1  |
+----+
|    1 |
|    2 |
+----+
2 rows in set (0.16 sec)
```

Si ahora invertimos la acción con un comando ROLLBACK, desharemos la transaccion que todavia no se ha confirmado:

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT f1 FROM innotest;
+----+
| f1  |
+----+
|    1 |
+----+
1 row in set (0.17 sec)
```

A continuacion, vamos a examinar que ocurriria si se interrumpe la conexión antes de que se complete la transaccion:

```
mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO innotest(f1) VALUES (2);
Query OK, 1 row affected (0.00 sec)

mysql> EXIT
Bye
```

```
C:\MySQL\bin> mysql firstdb
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8 to server version: 4.0.1-alpha-
max

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

```
mysql> SELECT f1 FROM innotest;
+----+
```

```

| f1    |
+----+
|     1 |
+----+
1 row in set (0.11 sec)

```

Puede repetir la **instrucción** anterior utilizando esta vez una **instrucción COMMIT** antes de salir. Tras **ello**, la transaccion quedara completada, de **forma** que **al** volver a establecer la conexion, se presentara el nuevo registro:

```

mysql> BEGIN;
Query OK, 0 rows affected (0.05 sec)

```

```

mysql> INSERT INTO innotest(f1) VALUES(2);
Query OK, 1 row affected (0.06 sec)

```

```

mysql> COMMIT;
Query OK, 0 rows affected (0.05 sec)

```

```

mysql> EXIT
Bye

```

```

C:\Program Files\MySQL\bin> mysql firstdb
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9 to server version: 4.0.1-alpha-
max

```

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

```

mysql> SELECT f1 FROM innotest;
+----+
| f1    |
+----+
|     1 |
|     2 |
+----+
2 rows in set (0.11 sec)

```

Lecturas coherentes

De **manera** predeterminada, las tablas **InnoDB** realizan una *lectura coherente*. Esto significa que **al** realizar una consulta de **selección**, MySQL devuelve los **valores** presentes de la base de datos hasta la ultima transaccion completada. Si en el **momento** de realizar la consulta existe alguna transaccion en progreso, los **resultados** de las **instrucciones** UPDATE o INSERT no se **reflejarán**, con una **excepción**: la transaccion abierta puede modificarse (puede que haya observado que **al** realizar la consulta BEGIN-INSERT-SELECT, se **visualizó** el resultado insertado). Para **po-**der verlo, necesita dos tener dos **ventanas** abiertas y estar conectado a la base de datos. En primer lugar agregue un registro desde una transaccion en la ventana 1:

```

mysql> BEGIN;

```

```
Query OK, 0 rows affected (0.11 sec)
```

```
mysql> INSERT INTO innotest(f1) VALUES(3);  
Query OK, 1 row affected (0.05 sec)
```

A continuacion, pase a la ventana 2:

```
mysql> SELECT f1 FROM innotest;  
+----+  
| f1  |  
+----+  
|    1 |  
|    2 |  
+----+  
2 rows in set (0.16 sec)
```

El 3 que hemos insertado no se devuelve porque **forma** parte de una transaccion incompleta. Si se devolvieran los resultados de una transaccion incompleta, la lectura resultaria incoherente.

A continuacion volvamos a la ventana 1:

```
mysql> SELECT f1 FROM innotest;  
+----+  
| f1  |  
+----+  
|    1 |  
|    2 |  
|    3 |  
+----+
```

Se muestra el 3 porque estamos dentro de una transaccion.

A continuacion, y todavia dentro de la ventana 1. **confirme** la transaccion:

```
mysql> COMMIT;
```

En la ventana 2, la **consulta reflejará** la transaccion completada:

```
mysql> SELECT f1 FROM innotest;  
+----+  
| f1  |  
+----+  
|    1 |  
|    2 |  
|    3 |  
+----+
```

Lectura de bloqueos para actualizaciones

Las lecturas coherentes no siempre **resultan** adecuadas. Por ejemplo, ¿qué ocurría si varios usuarios estan intentando agregar un nuevo registro en una tabla `innotest`? Cada nuevo registro **inserta** un numero ascendente **exclusivo**. Como

en este ejemplo: el campo f1 no es la clave principal o un campo de **incremento** automatico, por lo que nada impide que se **crea** un registro duplicado. Sin embargo, no queremos que eso ocurra. Lo que **desearíamos** es leer el valor actual de f1 e insertar un nuevo valor, incrementado en una unidad. Pero esta **acción no garantiza** un valor unico. Examine el siguiente ejemplo, comenzando en la ventana 1:

```
mysql> BEGIN;

mysql> SELECT MAX(f1) FROM innotest;
+-----+
| MAX(f1) |
+-----+
|        3 |
+-----+
```

Simultáneamente, otro usuario realiza la misma **operación** en la ventana 2:

```
mysql> BEGIN;

mysql> SELECT MAX(f1) FROM innotest;
+-----+
| MAX(f1) |
+-----+
|        3 |
+-----+
1 row in set (0.11 sec)
```

Ahora, los dos usuarios (ventana 1 y ventana 2) agregan un nuevo registro y confirman sus transacciones:

```
mysql> INSERT INTO innotest(f1) VALUES(4);
Query OK, 1 row affected (0.11 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
```

Si uno de los usuarios realiza una **consulta de selección**, recibirán los siguientes resultados:

```
mysql> SELECT f1 FROM innotest;
+-----+
| f1   |
+-----+
|     1 |
|     2 |
|     3 |
|     4 |
|     4 |
+-----+
```

La lectura coherente no ha **devuelto** lo esperado: los registros con los valores 4 y 5. La **forma** de evitar este resultado es realizando un bloqueo de **actualización**

sobre la operacion de **selección**. Si indicamos a MySQL que estamos realizando una lectura de actualizacion, no permitira que nadie mas lea el valor hasta que nuestra transaccion se haya completado. En primer lugar, elimine el 4 **incorrecto** de la tabla, para realizar la operacion correctamente esta vez:

```
mysql> DELETE FROM innotest WHERE f1=4;
Query OK, 2 rows affected (0.00 sec)
```

A **continuación**, establezca el bloqueo de actualizacion **como** se indica en la ventana 1:

```
mysql> BEGIN;
```

```
mysql> SELECT MAX(f1) FROM innotest FOR UPDATE;
+-----+
| MAX(f1) |
+-----+
|         3 |
+-----+
```

```
mysql> INSERT INTO innotest(f1) VALUES(4);
Query OK, 1 row affected (0.05 sec)
```

Entretanto, la ventana 2 tambien intenta establecer un bloqueo de **actualiza-**
cion:

```
mysql> BEGIN;
```

```
mysql> SELECT MAX(f1) FROM innotest FOR UPDATE;
```

Fíjese en que no se devuelve ningun resultado. MySQL espera a que se complete la transaccion de la ventana 1. Complete la transaccion en la ventana 1:

```
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
```

La ventana 2 devolvera los resultados de su **consulta**, tras esperar a que se complete la operacion de insercion.

```
mysql> SELECT MAX(f1) FROM innotest FOR UPDATE;
+-----+
| MAX(f1) |
+-----+
|         4 |
+-----+
1 row in set (4 min 32.65 sec)
```

Ahora, una vez seguros de que el 4 es el ultimo valor de la tabla, podemos agregar el 5 la ventana 2:

```
mysql> INSERT INTO innotest(f1) VALUES(5);
Query OK, 1 row affected (0.06 sec)
```

```
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
```

Bloqueos de lectura en modo compartido

Existe otro tipo de bloqueo de lectura que no devuelve un valor si el valor que esta leyendo ha sido modificado por otra transaccion incompleta. Devuelve el ultimo valor, pero no forma parte de una transaccion cuya intención es modificar el valor. Por ejemplo, vamos a utilizar el campo `f2` creado en la tabla `innotest`. Asumamos que el campo `f1` consta ya de elementos, pero hasta un momento posterior de la transaccion no se introdujera un valor para el campo `f2`. Al realizar una consulta de seleccion, no queremos recuperar un registro que disponga de un valor para `f1` pero no para `f2`, sino que queremos obtener siempre el ultimo registro. En este caso, necesitaremos esperar a que se complete la transaccion antes de que se recuperen los resultados. Por ejemplo, una transaccion comienza en la ventana 1:

```
mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO innotest(f1) VALUES(6);
Query OK, 1 row affected (0.00 sec)

mysql> UPDATE innotest set f2='Sebastian' WHERE f1=6;
Query OK, 1 row affected (0.05 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

Si realiza una consulta normal de seleccion en la ventana 2, no recuperaremos el ultimo valor (porque la transaccion anterior no se ha completado y la tabla **InnoDB** realiza una lectura coherente como parte de su comportamiento predeterminado). Sin embargo, si realiza una consulta con un bloqueo en modo compartido, no obtendra un resultado hasta que se complete la transaccion en la ventana 1.

Si ejecutamos una consulta normal en la ventana 2, se recuperaran los siguientes resultados:

```
mysql> SELECT MAX(f1) FROM innotest;
+-----+
| MAX(f1) |
+-----+
|         5 |
+-----+
1 row in set (0.17 sec)
```

Todavía en la ventana 2, si realiza la misma consulta en modo de bloqueo de uso compartido no se generara ningun resultado:

```
mysql> SELECT MAX(f1) FROM INNOTEST LOCK IN SHARE MODE;
Complete la transaccion en la ventana 1

mysql> COMMIT;
```

```
Query OK, 0 rows affected (0.00 sec)
```

A continuación la ventana 2 devolverá el resultado correcto:

```
mysql> SELECT MAX(f1) FROM innotest LOCK IN SHARE MODE;
+-----+
| MAX(f1) |
+-----+
|         6 |
+-----+
1 row in set (4 min 32.98 sec)
```

```
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
```

Confirmaciones automáticas

De manera predeterminada, y a menos que se especifique una **transacción** con `BEGIN`, MySQL **confirma** automáticamente las instrucciones. Por ejemplo, una **consulta** en la ventana 1 devolvería los siguientes resultados:

```
mysql> SELECT f1 FROM innotest;
+-----+
| f1   |
+-----+
|     1 |
|     2 |
|     3 |
|     4 |
|     5 |
|     6 |
+-----+
6 rows in set (0.11 sec)
```

A continuación, el usuario de la ventana 2 **inserta** un registro:

```
mysql> INSERT INTO innotest(f1) VALUES (7);
Query OK, 1 row affected (0.00 sec)
```

Esta inmediatamente disponible en la ventana 1 (recuerde completar todos los ejemplos anteriores con la **instrucción** `COMMIT`):

```
mysql> SELECT f1 FROM innotest;
+-----+
| f1   |
+-----+
|     1 |
|     2 |
|     3 |
|     4 |
|     5 |
|     6 |
|     7 |
+-----+
```



```

|    7 |
+----+
7 rows in set (0.11 sec)

```

El registro **insertado** en la ventana 2 está **inmediatamente** disponible para el resto de **las ventanas** porque la **acción** predeterminada es AUTOCOMMIT. Sin embargo, en **las tablas** de transaccion **segura (InnoDB, DBD)**, puede cambiar este **comportamiento** asignando 0 a AUTOCOMMIT. **En primer lugar, realice dicha operacion** en la ventana 1:

```

mysql> SET AUTOCOMMIT=0;
Query OK, 0 rows affected (0.00 sec)

```

A **continuación**, ejecute la **consulta** en la ventana 2:

```

mysql> SELECT f1 FROM innotest;
+----+
| f1  |
+----+
|    1 |
|    2 |
|    3 |
|    4 |
|    5 |
|    6 |
|    7 |
+----+
7 rows in set (0.22 sec)

```

Seguidamente, **inserte** un registro en la ventana 1:

```

mysql> INSERT INTO innotest(f1) VALUES(8);
Query OK, 1 row affected (0.00 sec)

```

En esta ocasion, no esta inmediatamente disponible en la ventana 2:

```

mysql> SELECT f1 FROM innotest;
+----+
| f1  |
+----+
|    1 |
|    2 |
|    3 |
|    4 |
|    5 |
|    6 |
|    7 |
+----+
7 rows in set (0.16 sec)

```

Como no se ha desactivado la **función de confirmación automática**, la operacion de **inserción** de la ventana 1 no se confirmara hasta que se ejecute un comando COMMIT de **manera** predeterminada. **Confirme** la transaccion desde la ventana 1:

```

mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

```

Ahora el nuevo registro esta disponible en la ventana 2.

```
mysql> SELECT f1 FROM innotest;
+----+
| f1  |
+----+
|    1 |
|    2 |
|    3 |
|    4 |
|    5 |
|    6 |
|    7 |
|    8 |
+----+
8 rows in set (0.11 sec)
```

Sin embargo, la secuencia `AUTOCOMMIT=0` no se aplica a todo el servidor, sino sólo a la sesion especifica. Si se asigna el valor 0 al comando `AUTOCOMMIT` en la ventana 2, el comportamiento sera diferente.

En primer lugar, establezca `AUTOCOMMIT` en la ventana 1 y en la ventana 2:

```
mysql> SET AUTOCOMMIT=0;
Query OK, 0 rows affected (0.00 sec)
```

A continuacion, ejecute la siguiente secuencia en la ventana 1 para comprobar sus elementos presentes:

```
mysql> SELECT f1 FROM innotest;
+----+
| f1  |
+----+
|    1 |
|    2 |
|    3 |
|    4 |
|    5 |
|    6 |
|    7 |
|    8 |
+----+
8 rows in set (0.17 sec)
```

Agregue un registro en la ventana 2 y confirme la transacción:

```
mysql> INSERT INTO innotest(f1) VALUES (9);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
```

A continuacion compruebe si aparece en la ventana 1

```
mysql> SELECT f1 FROM innotest;
```

```

+----+
| f1  |
+----+
|    1 |
|    2 |
|    3 |
|    4 |
|    5 |
|    6 |
|    7 |
|    8 |
+----+
8 rows in set (0.11 sec)

```

El 9 del nuevo registro no aparece, aunque hayamos confirmado los resultados. La razón es que la **instrucción de selección** de la ventana 1 **forma** también parte de una transacción. A la lectura coherente se le ha asignado un **punto temporal** y este **punto temporal** avanza si la transacción en la que se estableció se ha completado. **Confirme** la transacción en la ventana 1:

```

mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

```

```

mysql> SELECT f1 FROM innotest;

```

```

+----+
| f1  |
+----+
|    1 |
|    2 |
|    3 |
|    4 |
|    5 |
|    6 |
|    7 |
|    8 |
|    9 |
+----+
9 rows in set (0.22 sec)

```

Como vimos anteriormente, la única **forma** de examinar los últimos resultados consiste en seleccionarlos en **modo** bloqueo de uso compartido. En este **caso**, se hubiera esperado hasta que la transacción que realiza la operación de **inserción** haya realizado una operación de confirmación.

Transacciones en tablas DBD

Las tablas DBD procesan las transacciones de **forma** ligeramente diferente a las tablas **InnoDB**. En primer lugar, **Cree** la tabla (si no lo ha hecho en el capítulo anterior) e inserte un registro desde la ventana 1:

```

mysql> CREATE TABLE bdbtest(f1 INT,f2 CHAR(10))TYPE=BDB;

```

```
Query OK, 0 rows affected (0.28 sec)
```

```
mysql> BEGIN;
```

```
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> INSERT INTO bdbtest(f1) VALUES(1);
```

```
Query OK, 1 row affected (0.00 sec)
```

A continuación, **realice** la siguiente consulta desde la ventana 2:

```
mysql> SELECT f1 FROM bdbtest;
```

La ventana 2 espera a que la transacción de la ventana 1 este completada. (No devuelve un **conjunto** de resultados en **función** de la **situación** antes de que de comienzo la transacción de la ventana 1, **como** ocurre en las tablas **InnoDB**.)

Sólo cuando la ventana 1 **confirma** la **acción**, la ventana 2 recibe **los resultados**. Complete la transacción de la ventana 1:

```
mysql> COMMIT;
```

```
Query OK, 0 rows affected (0.00 sec)
```

Y la consulta de la ventana 2 se completa (no necesita escribirla de nuevo):

```
mysql> SELECT f1 FROM bdbtest;
```

```
+----+
```

```
| f1  |
```

```
+----+
```

```
|    1 |
```

```
+----+
```

```
1 row in set (3 min 13.99 sec)
```

Fijese en el largo **periodo** de tiempo que llevo la consulta. El hecho de que no se trate de una consulta de **selección** "rápida" en las tablas DBD significa que todas **las** transacciones que se pospongan pueden dar lugar a graves problemas de rendimiento.

Como en el **caso** de las tablas **InnoDB**, el **modo** predeterminado es AUTOCOMMIT=1. Esto significa que a **menos** que coloque sus cambios dentro de una transacción (comenzando con BEGIN), se **completarán** inmediatamente.

Ejecute la siguiente consulta desde la ventana 1:

```
mysql> SELECT f1 FROM bdbtest;
```

```
+----+
```

```
| f1  |
```

```
+----+
```

```
|    1 |
```

```
+----+
```

```
1 row in set (0.17 sec)
```

A continuación ejecute una **inserción** desde la ventana 2:

```
mysql> INSERT INTO bdbtest(f1) VALUES(2);
```

```
Query OK, 1 row affected (0.06 sec)
```

Resulta inmediatamente recuperable desde la ventana 1:

```
mysql> SELECT f1 FROM bdbtest;
+----+
| f1  |
+----+
|    1 |
|    2 |
+----+
2 rows in set (0.16 sec)
```

Si AUTOCOMMIT se define como 0, el efecto es el **mismo al** de encerrar todas las instrucciones en un comando BEGIN.

Asigne 0 a AUTOCOMMIT e inserte un registro en la ventana 1:

```
mysql> SET OPTION AUTOCOMMIT=0;
Query OK, 0 rows affected (0.11 sec)
mysql> INSERT INTO bdbtest(f1) VALUES(3);
Query OK, 1 row affected (0.11 sec)
```

Una consulta **ejecutada** desde la ventana 2 esperara a que la transaccion este activa:

```
mysql> SELECT f1 FROM bdbtest;
```

El resultado aparecera **sólo** cuando la transaccion se haya confirmado:
Confirme la transaccion en la ventana 1:

```
mysql> COMMIT;
Query OK, 0 rows affected (0.05 sec)
```

Ahora la consulta **recupera** los resultados en la ventana 2 (no necesita volver a escribir la consulta):

```
mysql> SELECT f1 FROM bdbtest;
+----+
| f1  |
+----+
|    1 |
|    2 |
|    3 |
+----+
```

Otros comportamientos transaccionales

Existe una serie de comandos adicionales que finalizan automaticamente una transaccion (en otras palabras, que se comportan como **si** hubieramos realizado una **operación de confirmación**).

- BEGIN

- ALTER TABLE
- CREATE INDEX
- RENAME TABLE (ese un sinonimo de ALTER TABLE x RENAME)
- TRUNCATE
- DROP TABLE
- DROP DATABASE

Incluso si el comando no produce un resultado satisfactorio, el mero hecho de aplicarlo genera una operacion de **confirmación**. Por ejemplo, comencemos por la siguiente transaccion en la ventana 1:

```
mysql> BEGIN;

mysql> SELECT MAX(f1) FROM innotest FOR UPDATE;
+-----+
| MAX(f1) |
+-----+
|         9 |
+-----+
```

Y comience otra transaccion en la ventana 2:

```
mysql> BEGIN;

mysql> SELECT MAX(f1) FROM innotest FOR UPDATE;
```

Los resultados no se muestran, ya que la ventana 1 ha bloqueado la fila para su **actualización**.

Sin embargo, el usuario de la ventana 1 cambia de opinion y decide modificar **primero** la estructura de la tabla.

- Ejecutamos el comando ALTER TABLE en la ventana 1

```
mysql> ALTER TABLE innotest add f1 INT;
ERROR 1060: Duplicate column name 'f1'
```

Aunque la operacion ALTER falló, se levantó el bloqueo, se **confirmó la transaccion** y la **consulta** de la ventana 2 se **completó** (no es necesario volver a introducirla).

```
mysql> SELECT MAX(f1) FROM innotest FOR UPDATE;
+-----+
| MAX(f1) |
+-----+
|         9 |
+-----+
1 row in set (2 min 23.52 sec)
```

Bloqueo de tablas

En la sección dedicada al análisis de las tablas InnoDB y BDB, se menciona el concepto de bloqueo en el nivel de fila, en el que se bloqueaban filas individuales durante un periodo de tiempo. Los bloqueos en el nivel de fila son mucho más eficaces cuando se necesita realizar una gran cantidad de inserciones o actualizaciones en la tabla. El bloqueo en el nivel de fila, sin embargo, solo está disponible para los tipos de tabla de transacción segura (BDB e InnoDB). MySQL también incorpora la función de bloqueo en el nivel de tablas, que está disponible para todos los tipos de tabla.

Existen dos tipos de bloqueos de tabla: los *bloqueos de lectura* y los *bloqueos de escritura*. Los bloqueos de lectura sólo permiten realizar lecturas sobre la tabla, quedando bloqueadas las operaciones de escritura. Los bloqueos de escritura impiden la realización de operaciones de lectura o escritura sobre la tabla durante el bloqueo. La sintaxis para bloquear una tabla es la siguiente:

```
LOCK TABLE nombre_de_tabla (READ|WRITE)
```

Para desbloquear una tabla, basta con utilizar la instrucción UNLOCK TABLE de la siguiente forma:

```
UNLOCK TABLES
```

La siguiente secuencia ilustra un bloqueo en el nivel de tabla, que funcionará con todo tipo de tablas.

En primer lugar, bloquee la tabla desde la ventana 1:

```
mysql> LOCK TABLE customer READ;
Query OK, 0 rows affected (0.01 sec)
```

Se pueden leer otros subprocesos, pero no se pueden escribir, como puede observar si prueba a utilizar el siguiente comando en la ventana 2:

```
mysql> SELECT * FROM customer;
+----+-----+-----+
| id  | first-name | surname |
+----+-----+-----+
| 1   | Yvonne     | Clegg   |
| 2   | Johnny     | Chaka-Chaka |
| 3   | Winston    | Powers  |
| 4   | Patricia   | Mankunku |
+----+-----+-----+
```

```
mysql> INSERT INTO customer(id,first_name,surname)
VALUES(5,'Francois','Papo');
```

La instrucción INSERT no se procesa hasta que el bloqueo se libera en la ventana 1:

```
mysql> UNLOCK TABLES;
```

Seguidamente se completa la operacion de insercion en la ventana 2 (no es necesario volver a escribirla):

```
mysql> INSERT INTO customer(id,first_name,surname)
VALUES (5,'Francois','Papo');
Query OK, 1 row affected (7 min 0.74 sec)
```

Tambien puede bloquear mas de una tabla a la vez. Aplique los siguientes bloqueos desde la ventana 1:

```
mysql> LOCK TABLE customer READ,sales WRITE;
```

Otros subprocesos pueden leer la tabla customer ,pero no la tabla sales . Intente ejecutar una instruccion SELECT desde a ventana 2:

```
mysql> SELECT * FROM sales;
```

Si el subproceso que creo el bloqueo intenta agregar un registro a la tabla customer ,fallara. No esperara a que el bloqueo se libere (como se creo el bloqueo, si se suspende, no volvera a poder liberarlo nunca); en su lugar la operacion de insercion simplemente fallara. Pruebe la siguiente instruccion en la ventana 1:

```
mysql> INSERT INTO customer VALUES (1,'a','b');
ERROR 1099: Table 'customer' was locked with a READ lock and
can't be updated
```

Sin embargo, puede realizar lecturas sobre la tabla cuya escritura bloqueo, de la siguiente forma, desde la ventana 1:

```
mysql> SELECT * FROM sales;
+----+-----+-----+
| code | sales-rep | id  | value |
+----+-----+-----+
| 1 | 1 | 1 | 2000 |
| 2 | 4 | 3 | 250 |
| 3 | 2 | 3 | 500 |
| 4 | 1 | 4 | 450 |
| 5 | 3 | 1 | 3800 |
| 6 | 1 | 2 | 500 |
| 7 | 2 | NULL | 670 |
+----+-----+-----+
```

```
mysql> UNLOCK TABLES;
```

Y con el bloqueo liberado, la ventana 2 realiza el proceso de selección (no es necesario volver a escribir la instruccion):

```
mysql> SELECT * FROM sales;
+----+-----+-----+
| code | sales-rep | id  | value |
+----+-----+-----+
| 1 | 1 | 1 | 2000 |
```



```

| 2 |          4 | 3 1 2501
| 3 |          2 | 3 | 500 |
| 4 |          1 | 4 | 450 |
| 5 |          3 | 1 | 3800 |
| 6 |          1 | 2 | 500 |
| 7 |          2 | NULL | 6701
+-----+
7 rows in set (5 min 59.35 sec)

```

NOTA: Puede utilizar esta instrucción en su forma singular o plural: [UN] LOCK TABLE y [UN]LOCK TABLES. Ambas son válidas, independientemente de la cantidad de tablas que estemos bloqueando.

Los bloqueos de escritura tienen prioridad sobre los bloqueos de lectura, de manera que si un subproceso espera un bloqueo de lectura y recibe un bloqueo de escritura; el bloqueo de lectura deberá esperar hasta obtener el bloqueo de escritura y a su liberación, de la forma que se indica a continuación.

Aplicar un bloqueo de escritura desde la ventana 1:

```
mysql> LOCK TABLE customer WRITE;
Query OK, 0 rows affected (0.00 sec)
```

Ahora, intentar aplicar un bloqueo de lectura desde la ventana 2:

```
mysql> LOCK TABLE customer READ;
```

El bloqueo de lectura no se puede obtener hasta que se libere el bloqueo de escritura. Entretanto, se recibe otra petición por un bloqueo de escritura, que también debe esperar hasta que se libere el primero.

Intentar aplicar otro bloqueo de escritura desde una tercera ventana:

```
mysql> LOCK TABLE customer WRITE;
```

A continuación, libere el bloqueo desde la ventana 1:

```
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)
```

Ahora se obtiene el bloqueo de escritura de la ventana 2, aunque fue solicitado tras el bloqueo de lectura, de la siguiente forma (no es necesario volver a escribir la instrucción LOCK):

```
mysql> LOCK TABLE customer WRITE;
Query OK, 0 rows affected (33.93 sec)
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)
```

Sólo cuando se libera el bloqueo de escritura de la ventana 3 se puede obtener el bloqueo de escritura de la ventana 2 (no es necesario volver a escribirlo):

```
mysql> LOCK TABLE customer READ;
```

```
Query OK, 0 rows affected (4 min 2.46 sec)
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)
```

Puede variar este comportamiento especificando una prioridad inferior para el bloqueo de escritura, mediante la **palabra clave** `LOW_PRIORITY`.

Si **vuelve** a ejecutar el ejemplo anterior con una solicitud de prioridad baja para un bloqueo de escritura, se obtendra primero el bloqueo de lectura anterior.

En primer lugar, solicite el bloqueo de escritura en la ventana 1:

```
mysql> LOCK TABLE customer WRITE;
Query OK, 0 rows affected (0.00 sec)
```

A continuacion pruebe a **realizar** un bloqueo de lectura desde la ventana 2:

```
mysql> LOCK TABLE customer READ;
```

Y un bloqueo de escritura de prioridad baja desde la ventana 3:

```
mysql> LOCK TABLE customer LOW-PRIORITY WRITE;
```

Seguidamente, libere el bloqueo de la ventana 1:

```
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)
```

En esta ocasion la ventana 2 obtendra su bloqueo en primer lugar (no es necesario volver a escribir la **instrucción** `LOCK`):

```
mysql> LOCK TABLE customer READ;
Query OK, 0 rows affected (20.88 sec)
```

```
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)
```

Por ultimo, se obtiene el bloqueo de escritura desde la ventana 3 (no es necesario volver a escribirlo):

```
mysql> LOCK TABLE customer LOW-PRIORITY WRITE;
Query OK, 0 rows affected (1 min 25.94 sec)
```

A continuacion, libere de nuevo el bloqueo para poder **utilizar** la tabla en un momento posterior:

```
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)
```

ADVERTENCIA: La **instrucción** `LOCK TABLES` no es de **transacción** segura. **Confirmará** todas las **transacciones activas** antes de intentar **bloquear las tablas**.

Los bloques de tabla se suelen utilizar de esta forma sobre tablas que no admiten transacciones. Si esta utilizando una tabla InnoDB o BDB, utilice los comandos BEGIN y COMMIT para evitar anomalías en los datos. A continuación se incluye un ejemplo en el que podría utilizarse. Si su tabla customer_sales_values esta vacía, rellénela con algunos registros:

```
mysql> INSERT INTO customer_sales_values(first_name, surname, value,
value2) VALUES('Johnny', , 500, NULL), ('Patricia',
'Mankunku', 450, NULL), ('Winston', 'Powers', 750, NULL),
('Yvonne', 'Clegg', 5800, NULL), ('Charles', 'Dube', 0, NULL),
('Charles', 'Dube', 0, NULL), ('Gladys', 'Malherbe', 5, 10);
```

Imaginemos que Johnny Chaka-Chaka ha realizado dos ventas, cada una procesada por un administrativo diferente. La primera es de 100 dolares y la segunda de 300 dolares. Los administrativos leen el valor existente, agregan 100 o 300 a dicho valor y actualizan el registro. El problema surge si ambos realizan la operación de selección antes de que cualquiera de los dos se actualice.

En este caso, una de las actualizaciones sobrescribira a la otra y se perderá dicho valor, de la forma en la que indica a continuación.

En primer lugar, realice la consulta desde la ventana 1:

```
mysql> SELECT value from customer_sales_values WHERE
first_name='Johnny' and surname='Chaka-Chaka';
+----+
| value |
+----+
| 500 |
+----+
```

A continuación realice la consulta desde la ventana 2:

```
mysql> SELECT value from customer_sales_values WHERE
first_name='Johnny' and surname='Chaka-Chaka';
+----+
| value |
+----+
| 500 |
+----+
```

Ésta es la ventana 1:

```
mysql> UPDATE customer_sales_values SET value=500+100 WHERE
first_name='Johnny' and surname='Chaka-Chaka';
Query OK, 1 row affected (0.01 sec)
```

Ésta es la ventana 2:

```
mysql> UPDATE customer_sales_values SET value=500+300 WHERE
first_name='Johnny' and surname='Chaka-Chaka';
```

Query OK, 1 row affected (0.01 sec)

Una vez capturadas las dos ventanas, el valor de las **ventas** de Johnny sera de 800 dolares, lo que **supone** 100 dolares **menos** de lo realmente vendido. Si **hubiéramos** bloqueado la tabla, habriamos evitado el problema.

Tras restablecer **los** datos y comenzar de nuevo, ejecute la siguiente **operación de actualización**.

```
mysql> UPDATE customer-sales-values SET value=500 WHERE
  first_name='Johnny' and surname='Chaka-Chaka';
Query OK, 1 row affected (0.00 sec)
```

A continuacion, coloque un bloqueo de escritura en la ventana 1:

```
mysql> LOCK TABLE customer_sales_values WRITE;
mysql> SELECT value from customer_sales_values WHERE
  first_name='Johnny' and surname='Chaka-Chaka';
+----+
| value |
+----+
|  500 |
+----+
```

La ventana 2 intenta obtener un bloqueo de escritura tambien:

```
mysql> LOCK TABLE customer_sales_values WRITE;
```

No lo logra porque ya ha sido asignado uno a la ventana 1. A continuacion, la ventana 1 puede actualizar el registro, antes de liberar el bloqueo y permitir que la ventana 2 continúe.

Ejecute la siguiente **instrucción UPDATE** en la ventana 1 y **libere** el bloqueo:

```
mysql> UPDATE customer_sales_values SET value=500+100 WHERE
  first_name='Johnny' and surname='Chaka-Chaka';
Query OK, 1 row affected (0.00 sec)
mysql> UNLOCK TABLES;
```

La ventana 2 obtiene el bloqueo (no es necesario volver a escribirlo) y puede completar el resto de la **transacción** de la siguiente forma:

```
mysql> LOCK TABLE customer-sales-values WRITE;
Query OK, 0 rows affected (1 min 35.87 sec)
mysql> SELECT value from customer-sales-values WHERE
  first_name='Johnny' and surname='Chaka-Chaka';
+----+
| value |
+----+
|  600 |
+----+
1 row in set (0.00 sec)
```

```
mysql> UPDATE customer_sales_values SET value=600+300 WHERE
  first_name='Johnny' and surname='Chaka-Chaka';
Query OK, 1 row affected (0.01 sec)
```

```
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)
```

Johnny recibe el crédito que se merece, y la tabla refleja correctamente los 900 dolares de las ventas realizadas.

Como evitar los bloqueos de tabla

Deberia evitar los bloqueos sobre tablas que necesiten realizar un gran volumen de actualizaciones, ya que, en el caso de los bloqueos de escritura, no se puede leer o escribir ningun registro de la tabla durante el bloqueo.

Además, como los bloqueos de escritura tienen prioridad sobre los de lectura de manera predeterminada, no se puede leer ningun registro hasta que todas las operaciones de actualizacion e inserción se completen, lo que puede provocar que MySQL se atasque de forma terrible. Existen varias formas de evitar los bloqueos de tabla. Una de ellas consiste en realizar la operación de lectura y actualizacion dentro de la misma instrucción (es lo que se conoce como *actualizacion incremental*).

Ejecute la siguiente actualizacion incremental desde la ventana 1:

```
mysql> UPDATE customer_sales_values SET value=value+300 WHERE
  first_name='Johnny' and surname='Chaka-Chaka';
```

La ventana 2 tambien puede realizar su actualizacion:

```
mysql> UPDATE customer_sales_values SET value=value+100 WHERE
  first_name='Johnny' and surname='Chaka-Chaka';
```

A continuación, independientemente del orden en el que se distribuyan las instrucciones, la actualizacion siempre se realizara sobre el valor mas reciente:

Niveles de transaccion

Puede modificar el comportamiento predeterminado al trabajar con transacciones mediante el establecimiento del nivel de transaccion. Existen varios niveles de transaccion en MySQL. En concreto admite los siguientes niveles de aislamiento de transaccion:

```
READ UNCOMMITTED
```

Este nivel permite transacciones para leer datos sin confirmar desde otras transacciones (es lo que se conoce como *lectura sucia*).

Este nivel no permite lecturas sucias.

REPEATABLE READ

Este nivel no **permite** lecturas no susceptibles de **repetición** (que son las que se dan cuando otra transacción ha modificado **los** datos, incluso si se han **confirmado**).

SERIALIZABLE

Este nivel no **permite** lecturas fantasma, que **tienen** lugar cuando otra transacción ha confirmado una nueva fila que coincide con **los** resultados de nuestra consulta. Los datos serán **los** mismos en cada ocasión.

Para cambiar el nivel de transacción, utilice la siguiente sintaxis:

```
SET [ámbito] TRANSACTION ISOLATION LEVEL
{ nivel de aislamiento }
```

La opción *ambito* puede ser **GLOBAL** o **SESSION**. Esta opción reemplaza el ámbito habitual de la instrucción, que se encarga de establecer el nivel de **aislamiento** en el que debe comenzar la siguiente transacción. **GLOBAL** establece el nivel para todas las nuevas transacciones y **SESSION** para las nuevas transacciones de dicho subproceso. La opción *nivel de aislamiento* es una de los cuatro niveles expuestos más arriba.

Resumen

Las combinaciones pueden adoptar **formas** mucho más **complejas** que la unión de dos tablas vista en un capítulo anterior. Las combinaciones **internas omiten los** valores **NULL** en las tablas que se estén combinando (o en las **filas** que no tengan registros asociados) y las combinaciones **externas** incluyen datos **NULL**. Las **combinaciones externas** por la derecha devuelven todos los datos de una tabla **especificada** en primer lugar (a la izquierda), incluyendo aquellos que no tengan un registro asociado en la tabla derecha, mientras que las combinaciones **externas** por la **derecha** devuelven todos los datos de la tabla especificados a la derecha de la combinación. Las combinaciones **externas** completas **combinan** las características de las uniones por la izquierda y por la derecha, **pero MySQL** no las **admite** todavía.

Las combinaciones naturales aprovechan el hecho de que **los** campos comunes pueden recibir **los** mismos nombres y simplificar la **sintaxis** si este **fuera** el caso.

El comando **UNION** combina **los** resultados de varias consultas en una.

Las subselecciones son consultas dentro de consultas. Por **regla** general, **resultan** más eficaces si se describen en **forma** de combinación.

La **eliminación** de registros uno a uno, como ocurre con **la** instrucción **DELETE**, no es un **método eficaz** si necesitamos eliminar todos los registros de una tabla. La instrucción **TRUNCATE** es **la forma** más rápida de **realizar** esta tarea, aunque no devuelva el número de registros eliminados, como **hace DELETE**.

Las variables de usuario **permiten** almacenar valores para su uso en una consulta posterior. **Ahora bien, al utilizarlas** es necesario tener cuidado de establecer la variable de usuario antes de que resulte necesaria. En **las** instrucciones **SELECT**, la **condición** (la cláusula **WHERE**) se realiza en primer lugar, antes de la lista de campos (inmediatamente después de la **instrucción SELECT** y en el lugar en el que se establezcan las variables de usuario de **manera** general).

MySQL también se puede ejecutar en **modo** de procesamiento por **lotes**, con **las** instrucciones **SQL** almacenadas en **archivos** para facilitar **las** operaciones de **edición** y reutilización. También puede dirigir la salida a un **archivo**, por ejemplo para facilitar el **examen** de los resultados de las consultas en un **momento** posterior.

Todos **los** tipos de tablas **permiten** el bloqueo de tablas. Esta **operación** **permite** bloquear una tabla entera en contraposición a **los** bloqueos de filas que se utilizan en **las** tablas de **transacción** segura.

En el siguiente capítulo, seguiremos analizando nuevos conceptos y examinaremos varios métodos para optimizar el rendimiento de nuestras bases de datos. Analizaremos el tema de la **creación** de índices, la escritura de consultas más eficaces y la mejora del rendimiento del **servidor**.

4

Índices y optimización de consultas

Una cosa es lograr que una **consulta** funcione y otra que lo haga rápidamente cuando los clientes se amontonan.

Puede agilizar la velocidad de sus consultas mediante el uso de métodos **básicos**.

El uso inteligente de índices puede dar resultados sorprendentes.

Así mismo, el **ajuste** cuidadoso de su sistema puede contribuir a lograr **mejoras** notables.

En este capítulo se abordan los siguientes temas:

- Creation y uso de índices
- Claves principales, índices únicos, índices de texto completo e índices ordinarios
- Búsquedas de texto completo
- Eliminación y modificación de un índice
- Campos de incremento automático
- Análisis de consultas con EXPLAIN
- Optimización de instrucciones SELECT

Comprension de los indices

Hasta ahora, ninguna de las tablas creadas en los capitulos anteriores **consta** de un **índice**. Al agregar un nuevo registro, este se suele colocar **al final** de la tabla, **pero** tambien puede hacerse en la **mitad** de la tabla si se ha eliminado otro o si existe dicho espacio. En otras palabras, **los registros no se almacenan en ningun orden**. Considere, por ejemplo, la tabla de clientes creada en el capitulo anterior, la cual, suponiendo que ha seguido **los ejemplos** de dicho capitulo, contiene **registros distribuidos** en el siguiente **orden**:

id	first-name	surname
1	Yvonne	Clegg
2	Johnny	Chaka-Chaka
3	Winston	Powers
4	Patricia	Mankunku
5	Francois	Papo
7	Winnie	Dlamini
6	Neil	Beneke

A **continuación**, imagine que estuvieramos **haciendo** el trabajo de MySQL. Si quisieramos recuperar todos **los registros** que tengan como apellido Beneke, es probable que empezaramos por la **parte superior** de la tabla y examinaramos **cada uno** de ellos. Sin mas datos, no hay **forma** de saber (ni nosotros ni MySQL) donde **buscar los registros** que **cumplan** dichos criterios. La operacion de recorrer la tabla de esta **forma** (de principio a fin, examinando todos **los registros**) se conoce como **examen completo de la tabla**. Cuando las tablas son de gran **tamaño**, esta operacion resulta poco eficiente ya que la labor de examinar tablas compuestas de varios cientos de miles de registros puede resultar muy lenta. Para evitar esta circunstancia, es aconsejable ordenar **los registros**. Vamos a **buscar** el mismo registro de antes **pero sobre** una tabla ordenada por el campo del apellido:

id	first-name	surname
6	Neil	Beneke
2	Johnny	Chaka-Chaka
1	Yvonne	Clegg
7	Winnie	Dlamini
4	Patricia	Mankunku
5	Francois	Papo
3	Winston	Powers

Las **busquedas sobre** esta tabla resultaran mucho mas **rápidas**. Como sabemos que **los registros** estan almacenados alfabeticamente por el campo **surname** al llegar a la entrada Chaka-Chaka, sabemos que no hay mas registros **Beneke**.

Por lo tanto, basta con examinar un registro, en contraposición a los siete que deberíamos examinar en la tabla sin ordenar. El resultado es un gran ahorro, que resultara incluso mas importante en tablas de mayor tamaño.

Por lo tanto, la solución parece estar en la ordenación de la tabla. Sin embargo, puede ocurrir que deseemos buscar registros de la tabla utilizando otros criterios. Por ejemplo, imagine que queremos recuperar un registro con un id de 3. Si la tabla sigue ordenada por el campo del apellido, necesitaremos examinar todos los registros de nuevo para hallar el deseado, con lo que la consulta resultaria lenta e ineficiente.

La solución consiste en crear listas separadas para cada campo que necesite ordenar. No contendran todos los campos, sólo aquellos que necesite ordenar y un puntero a un registro completo de la tabla. Estas tablas se denominan *índices* y son uno de los elementos menos y peor usados de las bases de datos relacionales (vease figura 4.1). Los índices se almacenan en *archivos* separados en algunos casos (tablas MyISAM) o como parte de algun espacio de tabla en otros (tablas InnoDB).

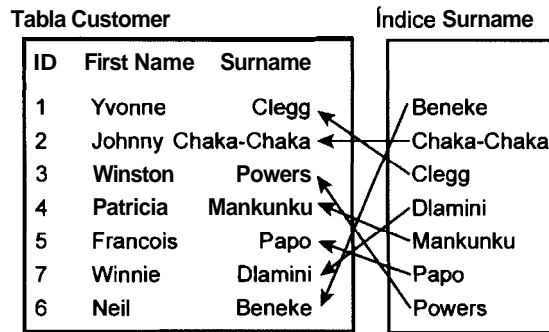


Figura 4.1. Los registros de índice apuntan a registros de la tabla customer

Creacion de un índice

Existen cuatro tipos de índice en MySQL: una clave primaria, un índice exclusivo, un índice de texto completo y un índice ordinario.

Creacion de una clave primaria

Una *clave primaria* es un índice establecido sobre un campo en el que cada valor es exclusivo y ninguno de los valores es NULL.

Para establecer una clave primaria al crear una tabla, utilice la instrucción **PRIMARY KEY** al final de las definiciones de campo, junto a una lista de los campos que se incluirán:

```
CREATE TABLE nombre_de_tabla (nombre_de_campo tipo_de_columna
NOT NULL,
```

```
{nombre_de_campo2...} PRIMARY KEY(nombre_de_campo1
[,nombre_de_campo2...]);
```

NOTA: El término **clave primaria** es, estrictamente hablando, un término lógico, pero MySQL lo utiliza para denotar un índice físico. Cuando MySQL indica que existe una clave primaria, siempre existe un índice asociado. A lo largo de este texto, el término clave indica la presencia de un índice físico.

Fijese en que la palabra clave NOT NULL resulta obligatoria al crear un campo primario; las claves primarias no pueden contener un valor nulo. MySQL le avisara si se olvida de especificarlo:

```
mysql> CREATE TABLE pk_test(f1 INT, PRIMARY KEY(f1));
ERROR 1171: All parts of a PRIMARY KEY must be NOT NULL;
If you need NULL in a key, use UNIQUE instead
```

Para crear una clave **primaria** en una tabla existente, puede utilizar la palabra clave ALTER:

```
ALTER TABLE nombre_de_tabla ADD PRIMARY KEY(nombre_de_campo1
[,nombre_de_campo2...]);
```

La elección de una clave **primaria** para la tabla de clientes es bastante sencilla ya que consta del campo `id` que resulta perfecto para dicha **función** al asignar un identificador diferente a cada cliente y no incluir campos nulos. Los otros dos campos de nombre no son adecuados ya que pueden incluir duplicados en el futuro. Para agregar una clave **primaria** al campo `id` de la tabla de clientes, es necesario modificarlo para no permitir registros nulos y, a continuación, agregar la clave **primaria**. Estas dos operaciones se pueden encerrar en una sola instrucción:

```
mysql> ALTER TABLE customer MODIFY id INT NOT NULL, ADD PRIMARY
KEY(id);
Query OK, 7 rows affected (0.00 sec)
Records: 7 Duplicates: 0 Warnings: 0
```

Puede ver los cambios realizados en la tabla con esta instrucción examinando las columnas:

```
mysql> DESCRIBE customer;
+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id             | int(11)       |      | PRI | 0        |      |
| first-name    | varchar(30)   | YES  |     | NULL     |      |
| surname       | varchar(40)   | YES  |     | NULL     |      |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

El campo `id` no contiene el valor `YES` en la **columna** `Null`, lo que indica que no puede aceptar valores **nu**los. Tambien lleva asignado el valor `PRI` en la **columna** `Key`, lo que indica que se trata de una clave primaria.

Las claves primarias tambien pueden componerse de mas de un campo. En ocasiones no existe un campo que identifique un registro de **manera** exclusiva.

Para agregar una clave **primaria** en este **caso**, separe los campos con una coma:

```
mysql> CREATE TABLE pk2(id INT NOT NULL, id2 INT NOT NULL,
PRIMARY KEY(id,id2));
Query OK, 0 rows affected (0.00 sec)
```

o de la siguiente **forma** si ya existe la tabla:

```
mysql> ALTER TABLE pk2 ADD PRIMARY KEY(id,id2);
Query OK, 0 rows affected (0.01 sec)
```

La tabla `sales` utilizada en los capitulos anteriores no consta todavia de una clave:

```
mysql> SHOW COLUMNS FROM sales;
+-----+-----+-----+-----+
| Field      | Type      | Null  | Key  | Default | Extra |
+-----+-----+-----+-----+
| code       | int(11)   | YES   |      | NULL    |      |
| sales_rep  | int(11)   | YES   |      | NULL    |      |
| id         | int(11)   | YES   |      | NULL    |      |
| value      | int(11)   | YES   |      | NULL    |      |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Imagine que ha agregado un nuevo registro con el mismo codigo que uno existente:

```
mysql> SELECT * FROM sales;
+-----+-----+-----+-----+
| code | sales_rep | id  | value |
+-----+-----+-----+-----+
| 1    |          | 1  | 2000  |
| 2    |          | 4  | 250   |
| 3    |          | 2  | 500   |
| 4    |          | 1  | 450   |
| 5    |          | 3  | 3800  |
| 6    |          | 1  | 500   |
| 7    |          | 2  | 670   |
+-----+-----+-----+-----+
7 rows in set (0.00 sec)
mysql> INSERT INTO sales VALUES(7,3,3,1000);
Query OK, 1 row affected (0.00 sec)
```

Por ahora no surgen problemas. Aunque tenemos dos registros con el codigo 7, no hay nada en la tabla que lo impida. Sin embargo, suponga que queremos

aplicar el nuevo conocimiento adquirido y decidimos convertir el campo de código en una clave primaria.

```
mysql> ALTER TABLE sales MODIFY code INT NOT NULL,ADD PRIMARY KEY(code);  
ERROR 1062: Duplicate entry '7' for key 1
```

Tenemos un valor duplicado en el campo de código y por **definición** las claves primarias **deben** ser siempre únicas.

Para solucionar este problema necesitaremos eliminar o actualizar los duplicados o **utilizar** un índice ordinario que los admita. La mayor **parte** de las tablas funcionan correctamente con una clave primaria. Por **ello**, resulta **sencillo** actualizar el registro responsable:

```
mysql> UPDATE sales SET code=8 WHERE code=7 AND sales_rep=3;  
Query OK, 1 row affected (0.00 sec)  
Rows matched: 1 Changed: 1 Warnings: 0  
mysql> ALTER TABLE sales MODIFY code INT NOT NULL,ADD PRIMARY KEY(code);  
Query OK, 8 rows affected (0.01 sec)  
Records: 8 Duplicates: 0 Warnings: 0
```

TRUCO: Trabajé en un sistema en el que un campo "único" resultó tener miles de duplicados debido a una combinación de dos circunstancias: no constar de clave primaria y no llevar asignado ningún bloqueo. Es aconsejable agregar siempre claves, especialmente una clave primaria, al crear una tabla.

Creación de un índice primario

Los índices que no son primarios permiten el uso de valores duplicados (a menos que los campos se especifiquen como únicos). Como siempre, es mejor crear el índice a la vez que se crea la tabla:

```
CREATE TABLE nombre_de_tabla(nombre_de_campo tipo_de_columna,  
nombre_de_campo2 tipo_de_columna, INDEX [nombre_de_indice]  
(nombre_de_campo1 [,nombre_de_campo2...]));
```

También puede crear más de un índice al crear la tabla, separándolas sencillamente mediante comas:

```
CREATE TABLE nombre_de_tabla (nombre_de_campo tipo_de_columna,  
nombre_de_campo2 tipo_de_columna, INDEX [nombre_de_indice1]  
(nombre_de_campo1,nombre_de_campo2),INDEX [nombre_de_indice2]  
(nombre_de_campo1 [,nombre_de_campo2...]));
```

También puede crear un índice en un momento posterior mediante el siguiente código:

```
ALTER TABLE nombre_de_tabla ADD INDEX [nombre_de_índice]
(nombre_de_campo1 [,nombre_de_campo2...]);
```

o con el siguiente código:

```
mysql> CREATE INDEX nombre_de_índice ON nombre_de_tabla
nombre_de_campo1 [,nombre_de_campo2...];
```

En estas dos instrucciones se solicita un nombre de índice, aunque con la instrucción `CREATE INDEX` el nombre del índice es obligatorio. Si en la instrucción `ALTER TABLE ADD INDEX` no se indica un nombre al índice, MySQL lo hará en función del nombre de campo. MySQL toma el primer campo como nombre del índice si el índice va a constar de varios campos. Si existe un segundo índice con el mismo campo, MySQL adjuntará las secuencias `_2`, `_3` y etc. al nombre del índice. La siguiente tabla de ventas consta de una tabla primaria, pero también podrá utilizar un índice sobre el campo de valor. Las búsquedas por registros con valor mayor o menor que una determinada cantidad o las consultas que ordenen los registros por el valor suelen ser comunes:

```
mysql> SHOW COLUMNS FROM sales;
+-----+-----+-----+-----+
| Field      | Type      | Null  | Key  | Default | Extra |
+-----+-----+-----+-----+
| code       | int(11)   |       | PRI  | 0        |       |
| sales-rep  | int(11)   | YES   |      | NULL     |       |
| id         | int(11)   | YES   |      | NULL     |       |
| value      | int(11)   | YES   |      | NULL     |       |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
mysql> ALTER TABLE sales ADD INDEX(value);
Query OK, 8 rows affected (0.02 sec)
Records: 8 Duplicates: 0 Warnings: 0
```

TRUCO: Puede utilizar la palabra clave `KEY` en lugar de `INDEX` en las instrucciones de MySQL, si lo desea. Personalmente, prefiero utilizar `INDEX` ya que `KEY` suele hacer referencia a la estructura lógica e `INDEX` al índice físico del disco.

Creación de un índice de texto completo

Puede crear índices de texto completo en tablas `MyISAM` sobre cualquier campo `CHAR`, `VARCHAR` o `TEXT`. Los índices de texto completo están diseñados para facilitar la búsqueda sobre palabras clave en campos de texto de tablas grandes.

Para crear un **índice** de texto completo al crear la tabla, utilice la siguiente sintaxis:

```
CREATE TABLE nombre_de_tabla (nombre_de_campo tipo_de_columna,  
nombre_de_campo2  
    tipo_de_columna, FULLTEXT(nombre_de_campo1  
[,nombre_de_campo2...]));
```

Se puede agregar la **palabra clave** opcional INDEX, como muestra la siguiente sintaxis:

```
CREATE TABLE nombre_de_tabla (nombre_de_campo tipo_de_columna,  
nombre_de_campo2  
    tipo_de_columna, FULLTEXT INDEX(nombre_de_campo1  
[,nombre_de_campo2...]));
```

Para crear un **índice** de texto completo una vez creada la tabla, utilice la siguiente sintaxis:

```
ALTER TABLE nombre_de_tabla ADD FULLTEXT [nombre_de_índice]  
(nombre_de_campo1 [,nombre_de_campo2...]);
```

o el siguiente código:

```
mysql> CREATE FULLTEXT INDEX nombre_de_índice on  
nombre_de_tabla nombre_de_campo1 [,nombre_de_campo2...]);
```

Vamos a crear una tabla e **intentar** crear índices de texto completo sobre alguno de los campos, como se indica a continuación:

```
mysql> CREATE TABLE ft(f1 VARCHAR(255),f2 TEXT,f3 BLOB,f4 INT);  
Query OK, 0 rows affected (0.01 sec)  
mysql> ALTER TABLE ft ADD FULLTEXT (f1,f2);  
Query OK, 0 rows affected (0.01 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

El campo f1 y f2 son de tipo VARCHAR y TEXT, respectivamente, por lo que se puede crear sobre ellos un **índice** de texto completo:

```
mysql> ALTER TABLE ft ADD FULLTEXT (f1,f4);  
ERROR 1005: Can't create table './firstdb/#sql-52eb_4f.frm'  
(errno: 140)  
mysql> ALTER TABLE ft ADD FULLTEXT (f2,f3);  
ERROR 1005: Can't create table './firstdb/#sql-52eb_4f.frm'  
(errno: 140)
```

En estos **ejemplo**, el campo f4 es del tipo INT y f3 es de tipo BLOB, por lo que no se **permite** un **índice** de texto completo.

Uso de los índices de texto completo

Vamos a crear una tabla con un **índice** de texto completo y a insertar algunos títulos de libros para probarlo:

```
mysql> CREATE TABLE ft2(f1 VARCHAR(255),FULLTEXT(f1));
```

```

Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO ft2 VALUES('Waiting for the
  Barbarians'),
  ('In the Heart of the Country'),
  ('The Master of Petersburg'),
  ('Writing and Being'),
  ('Heart of the Beast'),
  ('Heart of the Beest'),
  ('The Beginning and the End'),
  ('Master Master'),
  ('A Barbarian at my Door');
Query OK, 9 rows affected (0.00 sec)
Records: 9  Duplicates: 0  Warnings: 0

```

Para devolver los resultados de una búsqueda de texto completo, se utiliza la función `MATCH()`, y se busca la correspondencia de un campo con un valor, como en el siguiente ejemplo, que busca ocurrencias de la palabra `Master`:

```

mysql> SELECT * FROM ft2 WHERE MATCH(f1) AGAINST ('Master');
+-----+
| f1                |
+-----+
| Master Master     |
| The Master of Petersburg |
+-----+
2 rows in set (0.01 sec)

```

No es una coincidencia que `Master Master` aparezca en primer lugar, aunque fuera añadido en segundo lugar. `MySQL` calcula la relevancia del resultado para cada correspondencia y devuelve el resultado en dicho orden.

NOTA: Recuerde que las búsquedas sobre campos de tipo `TEXT` no distinguen entre mayúsculas y minúsculas, ni los campos `VARCHAR` o `CHAR` declarados sin la palabra clave `BINARY`.

Palabras ruido

Ahora vamos a ejecutar otra búsqueda:

```

mysql> SELECT * FROM ft2 WHERE MATCH(f1) AGAINST ('The
Master');
+-----+
| f1                |
+-----+
| Master Master     |
| The Master of Petersburg |
+-----+
2 rows in set (0.00 sec)

```

Los resultados no son los esperados.

La mayor parte de los títulos contienen artículos (*the* en inglés) y *The Beginning and the End* contiene dos, aunque no aparezca reflejado. Existen varias razones para ello:

- MySQL tiene lo que se conoce como umbral del 50 por ciento. Todas las palabras que aparecen en más de un 50 por ciento de los campos se consideran como ruido y se ignoran.
- Todas las palabras que tengan un número igual o inferior a tres letras se excluyen del índice.
- Existe una lista predefinida de palabras ruido, entre las que se incluye *the*.

Por lo tanto, el título *The Beginning and the End* no tiene ninguna oportunidad.

ADVERTENCIA: Si tiene una tabla con un solo registro, todas las palabras se consideran como palabras ruido. Por lo tanto una búsqueda de texto completo no devolverá nada. Las tablas con muy pocos registros también aumentan la probabilidad de que las palabras se consideren como palabras ruido.

La siguiente consulta no devuelve nada, aunque la palabra *for* aparezca en los datos ya que solo tiene tres caracteres y queda excluida de manera predeterminada del índice.

```
mysql> SELECT * FROM ft2 WHERE MATCH(f1) AGAINST ('for');
Empty set (0.00 sec)
```

Relevancia

No estamos limitados al uso de la función `MATCH()` en la condición `WHERE`. También puede recuperar los resultados de la siguiente forma:

```
mysql> SELECT f1, (MATCH(f1) AGAINST ('Master')) FROM ft2;
+-----+
| f1                | (MATCH(f1) AGAINST ('Master')) |
+-----+
| Waiting for the Barbarians | 0 |
| In the Heart of the Country | 0 |
| The Master of Petersburg   | 1.2245972156525 |
| Writing and Being          | 0 |
| Heart of the Beast         | 0 |
```

```

| Heart of the Beest | 0
|
| A Barbarian at my Door | 0
|
| Master Master | 1.238520026207
|
| The Beginning and the End | 0
|
+-----+
9 rows in set (0.00 sec)

```

Puede que la relevancia que obtenga en su caso no coincida con la de los ejemplos, ya que MySQL puede realizar cambios en funcion de un esquema de ponderacion.

El calculo de relevancia es bastante inteligente. Se basa en el numero de palabras del campo de indice de la fila, el numero de palabras unicas de dicha fila, el numero total de palabras de los resultados, el numero de registros que contiene dicha palabra en particular y la importancia de la palabra. Las palabras poco comunes reciben una mayor ponderacion y cuanto mayor es el numero de registros que contiene la palabra, menor sera su ponderacion.

MySQL puede devolver la relevancia asi como los campos requeridos sin que ello suponga un mayor coste en terminos de tiempo ya que las dos llamadas a la funcion MATCH () son identicas.

```

mysql> SELECT f1,(MATCH(f1) AGAINST ('Master')) FROM ft2
WHERE MATCH(f1) AGAINST ('Master');
+-----+-----+
| f1 | (MATCH(f1) AGAINST ('Master')) |
+-----+-----+
| Master Master | 1.238520026207 |
| The Master of Petersburg | 1.2245972156525 |
+-----+-----+

```

Busquedas booleanas de texto completo

Una de las optimizaciones mas utiles de MySQL 4 es su capacidad para realizar busquedas completas booleanas. Esta funcion utiliza un conjunto completo de elementos para buscar por palabras, combinaciones de palabras, porciones de palabras y otras variantes (vease la tabla 4.1).

Tabla 4.1. Operadores de busqueda booleanas

Operadores	Descripción
+	La palabra siguiente es obligatoria y debe aparecer en todas las filas devueltas.
-	La palabra siguiente esta prohibida y no debe aparecer en ninguna de las filas devueltas.

Operadores

Descripción

La **palabra** siguiente tiene una relevancia inferior al resto de palabras.

La **palabra** siguiente tiene una relevancia superior al resto de palabras.

()

Se utiliza para agrupar **palabras** en subexpresiones.

~

La **palabra** siguiente realiza una contribución negativa a la relevancia de la fila (no es igual al operador -, que excluye la fila completamente si se encuentra la **palabra**, o al operador <, que sigue asignando una relevancia positiva, aunque mas baja, a la palabra).

•

El comodín que indica cero o varios caracteres. Solo puede aparecer al final de la **palabra**.

"

Cualquier expresion encerrada entre comillas dobles se toma como un conjunto.

Las búsquedas booleanas de texto completo no tienen en cuenta el umbral del 50 por ciento. Para realizar una búsqueda booleana de texto completo se utiliza la cláusula `IN BOOLEAN MODE`:

```
mysql> SELECT • FROM ft2 WHERE MATCH(f1) AGAINST
  ('+Master -Petersburg' IN BOOLEAN MODE);
+-----+
| f1                |
+-----+
| Master Master |
+-----+
1 row in set (0.00 sec)
```

En este ejemplo, se excluye la **palabra** Petersburg, por lo que no se recupera el título The Master of Petersburg aunque incluya la **palabra** Master. Fíjese en la diferencia entre estos dos conjuntos de resultados:

```
mysql> SELECT • FROM ft2 WHERE MATCH(f1) AGAINST
  ('Country Master' IN BOOLEAN MODE);
+-----+
| f1                |
+-----+
| In the Heart of the Country |
| The Master of Petersburg   |
| Master Master              |
+-----+
3 rows in set (0.00 sec)
mysql> SELECT • FROM ft2 WHERE MATCH(f1) AGAINST
  ('+Country Master' IN BOOLEAN MODE);
```

```

+-----+
| f1                |
+-----+
| In the Heart of the Country |
+-----+
1 row in set (0.00 sec)

```

La palabra **Country** es obligatoria en la segunda búsqueda (de manera predeterminada una **palabra** es **opcional**), por lo que no se devuelve **The Master of Petersburg** ni **Master Master**.

El siguiente ejemplo muestra un caso habitual de confusión.

```

mysql> SELECT * FROM ft2 WHERE MATCH(f1) AGAINST
(' +Dog Master' IN BOOLEAN MODE);
+-----+
| f1                |
+-----+
| The Master of Petersburg |
| Master Master      |
+-----+
2 rows in set (0.00 sec)

```

Este resultado puede parecer sorprendente si lo comparamos con el ejemplo anterior, **pero como la palabra Dog** consta de tres letras queda excluida de la búsqueda. Los dos ejemplos que se incluyen a continuación muestran la diferencia entre realizar la búsqueda en función de una **palabra** completa y en función de **parte** de una **palabra** (utilizando el operador *****):

```

mysql> SELECT * FROM ft2 WHERE MATCH(f1) AGAINST
('Barbarian' IN BOOLEAN MODE);
+-----+
| f1                |
+-----+
| A Barbarian at my Door |
+-----+
1 row in set (0.00 sec)
mysql> SELECT * FROM ft2 WHERE MATCH(f1) AGAINST
('Barbarian*' IN BOOLEAN MODE);
+-----+
| f1                |
+-----+
| A Barbarian at my Door |
| Waiting for the Barbarians |
+-----+
2 rows in set (0.01 sec)

```

De manera predeterminada sólo se realizan búsquedas por la **palabra** completa, a **menos** que se utilice el operador *****.

En los tres ejemplos que se incluyen a continuación se demuestra el uso de los operadores **>** y **<** para incrementar y reducir las ponderaciones, respectivamente:

```

mysql> SELECT f1,MATCH(f1) AGAINST ('Heart Beast Beast')

```

```

IN BOOLEAN MODE) AS m FROM ft2 WHERE MATCH(f1)
AGAINST ('Heart Beest Beast' IN BOOLEAN MODE);
+-----+-----+
| f1                                     | m      |
+-----+-----+
| In the Heart of the Country          | 1      |
| Heart of the Beast                   | 2      |
| Heart of the Beest                  | 2      |
+-----+-----+
3 rows in set (0.00 sec)
mysql> SELECT f1,MATCH(f1) AGAINST ('Heart Beest >Beast'
IN BOOLEAN MODE) AS m FROM ft2 WHERE MATCH(f1)
AGAINST ('Heart Beest >Beast' IN BOOLEAN MODE);
+-----+-----+
| f1                                     | m      |
+-----+-----+
| In the Heart of the Country          | 1      |
| Heart of the Beast                   | 2.5    |
| Heart of the Beest                  | 2      |
+-----+-----+
3 rows in set (0.00 sec)

```

El operador > **incrementa** la relevancia de Heart of the Beast

```

mysql> SELECT f1,MATCH(f1) AGAINST ('Heart <Beest Beast'
IN BOOLEAN MODE) AS m FROM ft2 WHERE MATCH(f1)
AGAINST ('Heart <Beest Beast' IN BOOLEAN MODE);
+-----+-----+
| f1                                     | m      |
+-----+-----+
| In the Heart of the Country          | 1      |
| Heart of the Beast                   | 2      |
| Heart of the Beest                  | 1.6666667461395 |
+-----+-----+
3 rows in set (0.00 sec)

```

Los cinco ejemplos que se incluyen a **continuación** muestran la diferencia entre el operador <, que aplica una ponderación ligeramente positiva a la **correspondencia**; el operador = que aplica una ponderación negativa a la correspondencia; y el operador -, que **prohíbe** la correspondencia.

El primer ejemplo es una búsqueda booleana **básica**, con una ponderación de 1 para una correspondencia.

```

mysql> SELECT *,MATCH(f1) AGAINST ('Door' IN BOOLEAN MODE)
AS m FROM ft2 WHERE MATCH(f1) AGAINST ('Door' IN BOOLEAN
MODE);
+-----+-----+
| f1                                     | m      |
+-----+-----+
| A Barbarian at my Door               | 1      |
+-----+-----+
1 row in set (0.00 sec)

```

A continuación, el operador < reduce su ponderación a 2/3, que todavía sigue siendo positiva:

```
mysql> SELECT •,MATCH(f1) AGAINST ('<Door' IN BOOLEAN MODE)
      AS m FROM ft2 WHERE MATCH(f1) AGAINST ('<Door' IN BOOLEAN
MODE);
+-----+-----+
| f1                                | m          |
+-----+-----+
| A Barbarian at my Door           | 0.66666668653488 |
+-----+-----+
1 row in set (0.00 sec)
```

El operador ~ reduce su ponderación a un valor negativo, y por ello, no se devuelve la fila cuando se iguala a A Barbarian at my Door:

```
mysql> SELECT •,MATCH(f1) AGAINST ('- Door' IN BOOLEAN MODE)
      AS m FROM ft2 WHERE MATCH(f1) AGAINST ('~Door' IN BOOLEAN
MODE);
Empty set (0.00 sec)
```

Utilizando el operador ~ en combinación con una correspondencia común nos permitiría ver lo que se ha reducido la ponderación, que en este caso es 0,5:

```
mysql> SELECT •,MATCH(f1) AGAINST ('- Door Barbarian*' IN
BOOLEAN MODE)
      AS m FROM ft2 WHERE MATCH(f1) AGAINST ('- Door Barbarian*' IN
BOOLEAN MODE);
+-----+-----+
| f1                                | m          |
+-----+-----+
| A Barbarian at my Door           | 0.5        |
| Waiting for the Barbarians       | 1          |
+-----+-----+
2 rows in set (0.01 sec)
```

En este caso, por ejemplo, se muestra la diferencia entre los operadores ~ y -, donde el operador - impide la equivalencia cuando aparece la palabra Door.

```
mysql> SELECT •,MATCH(f1) AGAINST ('- Door Barbarian*' IN
BOOLEAN MODE)
      AS m FROM ft2 WHERE MATCH(f1) AGAINST ('- Door Barbarian*' IN
BOOLEAN MODE);
+-----+-----+
| f1                                | m          |
+-----+-----+
| Waiting for the Barbarians       | 1          |
+-----+-----+
1 row in set (0.00 sec)
```

A **continuación** se muestra un ejemplo de agrupación de **palabras** en una subexpresión:

```
mysql> SELECT f1,MATCH(f1) AGAINST ('+Heart +(<Beest >Beast)'
    IN BOOLEAN MODE) AS m FROM ft2 WHERE MATCH(f1)
    AGAINST ('+Heart +(<Beest >Beast)' IN BOOLEAN MODE);
+-----+-----+
| f1                | m                |
+-----+-----+
| Heart of the Beast | 1.25             |
| Heart of the Beest | 0.83333337306976 |
+-----+-----+
2 rows in set (0.00 sec)
```

El operador **+** se aplica a la subcadena completa encerrada entre parentesis, lo que significa que **al menos** uno de los terminos **Beest** y **Beast** debe estar presente en la cadena. In the Heart of the Country no aparece porque no incluye ninguno de los dos. Compare esto con el siguiente código.

El siguiente ejemplo muestra una **forma** habitualmente utilizada de búsqueda, en la que todas las **palabras** son obligatorias:

```
mysql> SELECT f1,MATCH(f1) AGAINST ('+Heart +<Beest +>Beast)'
    IN BOOLEAN MODE)
    AS m FROM ft2 WHERE MATCH(f1) AGAINST ('+Heart +<Beest
    +>Beast)' IN BOOLEAN MODE);
Empty set (0.00 sec)
```

No se **recupera** nada porque ninguna de las filas contiene las tres **palabras** (Heart, Beest y Beast) **a la vez**.

Los siguientes dos ejemplos muestran las diferencias entre una búsqueda **utilizando** los operadores **"** y sin ellos. Estos operadores **permiten** realizar **búsquedas** con una correspondencia **exacta** en una frase:

```
mysql> SELECT • FROM ft2 WHERE MATCH(f1)
    AGAINST ('the Heart of the' IN BOOLEAN MODE);
+-----+-----+
| f1                |
+-----+-----+
| In the Heart of the Country |
| Heart of the Beast          |
| Heart of the Beest         |
+-----+-----+
3 rows in set (0.01 sec)
```

```
mysql> SELECT • FROM ft2 WHERE MATCH(f1)
    AGAINST ('"the Heart of the"' IN BOOLEAN MODE);
+-----+-----+
| f1                |
+-----+-----+
| In the Heart of the Country |
+-----+-----+
1 row in set (0.00 sec)
```

No olvide colocar la comillas iniciales al utilizar el operador de comillas dobles. Si lo hace, es como si no utilizara ningun operador.

Por ejemplo:

```
mysql> SELECT * FROM ft2 WHERE MATCH(f1)
  AGAINST ("the Heart of the" IN BOOLEAN MODE);
+-----+
| f1                |
+-----+
| In the Heart of the Country |
| Heart of the Beast          |
| Heart of the Beest         |
+-----+
3 rows in set (0.00 sec)
```

ADVERTENCIA: Los índices de texto completo pueden tardar bastante tiempo en generarse y obligar a que les ocurra lo mismo a las instrucciones **OPTIMIZE**.

Creación de un índice único

Un índice unico es lo mismo que un índice ordinario con la salvedad de que no se permiten duplicaciones.

Para establecer un índice unico al crear una tabla, utilice la siguiente sintaxis:

```
CREATE TABLE nombre-de-tabla (nombre-de-campo tipo_de_columna,
  nombre_de_campo2
  tipo_de_columna, UNIQUE(nombre_de_campo
  [,nombre_de_campo2...]));
```

O si la tabla ya existe, puede utilizar esta otra sintaxis:

```
ALTER TABLE nombre-de-tabla ADD UNIQUE [nombre_de_indice ]
(nombre_de_campo
  [,nombre_de_campo2...]);
```

O esta otra:

```
CREATE UNIQUE INDEX nombre_de_indice ON nombre-de-tabla
(nombre_de_campo [,nombre_de_campo2...]);
```

Si el índice contiene un solo campo, dicho campo no puede contener valores duplicados:

```
mysql> CREATE TABLE ui_test(f1 INT,f2 INT,UNIQUE(f1));
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO ui_test VALUES(1,2);
Query OK, 1 row affected (0.01 sec)
mysql> INSERT INTO ui_test VALUES(1,3);
ERROR 1062: Duplicate entry '1' for key 1
```


Aunque el campo `f1` no se especificó como `UNIQUE` en el momento de su creación, la existencia de un índice exclusivo impide cualquier duplicación. Si el índice contiene más de un campo, los valores de los campos individuales se pueden duplicar, pero la combinación de los valores de campo que componen el índice entero no se pueden duplicar:

```
mysql> CREATE TABLE ui_test2(f1 INT,f2 INT,UNIQUE(f1,f2));
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO ui_test2 VALUES(1,2);
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO ui_test2 VALUES(1,3);
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO ui_test2 VALUES(1,3);
ERROR 1062: Duplicate entry '1-3' for key 1
```

Creación de índices sobre una parte de un campo

MySQL permite crear un índice que no utilice el campo completo en las columnas de tipo `VARCHAR`, `CHAR`, `BLOB` y `TEXT`. Por ejemplo, aunque el nombre del cliente alcance los 40 caracteres, es probable que el apellido varíe en los primeros 10 caracteres. El uso de los primeros 10 caracteres para crear el índice permite reducir enormemente su tamaño. De esta forma, las actualizaciones y operaciones de inserción resultarán más rápidas (ya que sólo se necesitará escribir un cuarto de lo que resultaría necesario si se utilizara la columna completa) y la velocidad de selección no se verá afectada siempre que no se recorte excesivamente el índice. Si asignáramos un carácter de tamaño al índice de una columna de apellidos anularíamos su función. Para crear un índice sobre una parte de un campo, indique su tamaño en el paréntesis situado tras el nombre de la columna. Por ejemplo, para crear un índice de 10 caracteres sobre el campo `surname` en la tabla de clientes, utilice el siguiente código:

```
mysql> ALTER TABLE customer ADD INDEX (surname(10));
Query OK, 8 rows affected (0.00 sec)
Records: 8 Duplicates: 0 Warnings: 0
```

NOTA: No se pueden crear índices (a excepción de los índices de texto completo) sobre un campo `BLOB` o `TEXT` completo, por lo que en estos casos necesitará especificar el tamaño del índice.

Como utilizar un campo de incremento automático

Los campos de incremento automático resultan útiles porque permiten incrementar el valor de un campo automáticamente cada vez que se inserta un nuevo

registro. Sólo se puede incrementar automáticamente un campo de un registro y dicho campo debe ser una clave **primaria numérica** o un **índice exclusivo numérico**.

Creación de un campo de incremento automatico

La **sintaxis** para crear un campo de incremento automatico es la siguiente:

```
CREATE TABLE nombre_de_tabla(nombre_de_campo INT
AUTO-INCREMENT, [nombre_de_campo2...,] PRIMARY
KEY(nombre_de_campo));
Query OK, 0 rows affected (0.00 sec)
```

Para crear un campo de incremento automatico en una tabla ya existente, utilice la siguiente **sintaxis**:

```
ALTER TABLE nombre_de_tabla MODIFY nombre_de_campo
tipo_de_columna AUTO-INCREMENT;
Query OK, 0 rows affected (0.00 sec)
```

La tabla de clientes consta de un candidato ideal para un campo de incremento automatico, el campo **id**:

```
mysql> SHOW COLUMNS FROM customer;
+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| id         | int(11)       |      | PRI | 0        |      |
| first-name | varchar(30)   | YES  |     | NULL     |      |
| surname    | varchar(40)   | YES  | MUL | NULL     |      |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

El campo **id** es un campo **numérico** que consta de una clave **primaria** y como hemos asignado el **id** en secuencia según la fecha, su conversión en un campo de incremento automatico permitira a **MySQL** automatizar el proceso por nosotros. El siguiente código convierte el campo **id** en un campo de incremento **automático**:

```
mysql> ALTER TABLE customer MODIFY id INT AUTO-INCREMENT;
Query OK, 7 rows affected (0.01 sec)
Records: 7 Duplicates: 0 Warnings: 0
mysql> SHOW COLUMNS FROM customer;
+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| id         | int(11)       |      | PRI | NULL     |      |
| auto-increment |              |      |     |          |      |
| first-name | varchar(30)   | YES  |     | NULL     |      |
+-----+-----+-----+-----+
```

```
| surname      | varchar(40) | YES | MUL | NULL      |
|
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Insercion de registros que contienen un campo de incremento automatico

A continuación, si agrega un registro, no necesitara especificar el valor del campo `id` ya que MySQL se encargara de agregar automaticamente el siguiente valor:

```
mysql> SELECT * FROM customer;
+----+-----+-----+
| id | first-name | surname |
+----+-----+-----+
| 1  | Yvonne     | Clegg   |
| 2  | Johnny     | Chaka-Chaka |
| 3  | Winston    | Powers  |
| 4  | Patricia   | Mankunku |
| 5  | Francois   | Papo    |
| 7  | Winnie     | Dlamini |
| 6  | Neil       | Beneke  |
+----+-----+-----+
7 rows in set (0.00 sec)
```

```
mysql> INSERT INTO customer(first_name,surname)
VALUES ('Breyton','Tshbalala');
Query OK, 1 row affected (0.00 sec)
mysql> SELECT * FROM customer;
+----+-----+-----+
| id | first-name | surname |
+----+-----+-----+
| 1  | Yvonne     | Clegg   |
| 2  | Johnny     | Chaka-Chaka |
| 3  | Winston    | Powers  |
| 4  | Patricia   | Mankunku |
| 5  | Francois   | Papo    |
| 7  | Winnie     | Dlamini |
| 6  | Neil       | Beneke  |
| 8  | Breyton    | Tshbalala |
+----+-----+-----+
8 rows in set (0.00 sec)
```

El contador automatico de MySQL recuerda el ultimo numero agregado, aunque se elimine el registro. De esta forma queda garantizada la asignacion de un nuevo `id` al registro agregado y evitara que se produzcan colisiones con los registros de la entrada antigua:

```
mysql> DELETE FROM customer WHERE id=8;
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO customer(first_name,surname)
VALUES ('Breyton','Tshbalala');
Query OK, 1 row affected (0.00 sec)
mysql> SELECT * FROM customer;
+----+-----+-----+
| id | first-name | surname |
+----+-----+-----+
| 1 | Yvonne     | Clegg   |
| 2 | Johnny     | Chaka-Chaka |
| 3 | Winston    | Powers  |
| 4 | Patricia   | Mankunku |
| 5 | Francois   | Papo    |
| 7 | Winnie     | Dlamini |
| 6 | Neil       | Beneke  |
| 9 | Breyton    | Tshbalala |
+----+-----+-----+
8 rows in set (0.01 sec)
```

El id es ahora 9. Aunque el registro con el siguiente id mas alto que queda es 7, el valor insertado mas recientemente fue el 8.

Como recuperar y reiniciar el valor de incremento automatico

Puede devolver el valor de incremento automatico insertado mas recientemente mediante la funcion `LAST_INSERT_ID()`:

```
mysql> SELECT LAST_INSERT_ID() FROM customer LIMIT 1;
+-----+
| last-insert-id() |
+-----+
| 9 |
+-----+
1 row in set (0.00 sec)
```

Esta funcion puede resultar util para actualizaciones en las que se necesite crear un nuevo valor de incremento. Por ejemplo, el siguiente codigo busca el valor de incremento automatico insertado mas recientemente y le agrega uno para establecer un nuevo id para Breyton Tshbalala:

```
mysql> UPDATE customer set id=LAST_INSERT_ID()+1 WHERE
first_name='Breyton' AND surname='Tshbalala';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0
mysql> SELECT * FROM customer;
+----+-----+-----+
| id | first-name | surname |
+----+-----+-----+
| 1 | Yvonne     | Clegg   |
| 2 | Johnny     | Chaka-Chaka |
| 3 | Winston    | Powers  |
```

```

| 4 | Patricia | Mankunku |
| 5 | Francois | Papo |
| 7 | Winnie | Dlamini |
| 6 | Neil | Beneke |
| 10 | Breyton | Tshbalala |
+---+-----+
8 rows in set (0.00 sec)

```

ADVERTENCIA: La función `LAST_INSERT_ID()` presenta problemas al reiniciar el contador de incremento automático, los cuales serán analizados en una sección posterior.

Si desea restablecer el **contador** de incremento automático para que se inicie en un valor **concreto** (como en 1 tras eliminar todos los registros) puede utilizar el siguiente código:

```
ALTER TABLE nombre_de_tabla AUTO_INCREMENT=valor_inc_auto;
```

Vamos a crear una prueba para **examinar** su comportamiento:

```

mysql> CREATE TABLE ai-test(id INT NOT NULL AUTO-INCREMENT,
  f1 VARCHAR(10),PRIMARY KEY (id));
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO ai-test(f1) VALUES ('one'),('two');
Query OK, 2 rows affected (0.00 sec)
Records: 2 Duplicates: 0 Warnings: 0
mysql> SELECT * FROM ai-test;
+---+
| id | f1 |
+---+
| 1 | one |
| 2 | two |
+---+
2 rows in set (0.00 sec)
mysql> DELETE FROM ai-test;
Query OK, 2 rows affected (0.00 sec)
mysql> INSERT INTO ai-test(f1) VALUES ('three');
Query OK, 1 row affected (0.01 sec)
mysql> SELECT * FROM ai-test;
+---+
| id | f1 |
+---+
| 3 | three |
+---+
1 row in set (0.00 sec)

```

El **contador** de incremento automático mantiene su valor, aunque la tabla este vacía. Puede utilizar `TRUNCATE` para vaciar la tabla y que se **reinicie** el **contador** de incremento automático:

```
mysql> DELETE FROM ai-test;
```

```

Query OK, 1 row affected (0.00 sec)
mysql> ALTER TABLE ai-test AUTO_INCREMENT=1;
Query OK, 0 rows affected (0.01 sec)
Records: 0 Duplicates: 0 Warnings: 0

```

Esta **instrucción** reinicia el **contador** de incremento automatico con el valor 1 de **manera** especifica:

```

mysql> INSERT INTO ai_test(f1) VALUES('four');
Query OK, 1 row affected (0.00 sec)
mysql> SELECT * FROM ai-test;
+----+----+
| id | f1  |
+----+----+
|  1 | four |
+----+----+
1 row in set (0.00 sec)
mysql> TRUNCATE ai-test;
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO ai_test(f1) VALUES('five');
Query OK, 1 row affected (0.01 sec)
mysql> SELECT * FROM ai-test;
+----+----+
| id | f1  |
+----+----+
|  1 | five |
+----+----+
1 row in set (0.00 sec)

```

TRUNCATE, en contraposicion a **DELETE**, reinicia el **contador** de incremento automatico.

ADVERTENCIA: En la actualidad sólo funciona con tablas MyISAM. En otros tipos de tablas, resulta necesario establecer manualmente el contador de incremento automático.

La asignacion de un valor de incremento automatico diferente a 1 resulta sencilla. Puede hacerlo al crear la tabla, por ejemplo:

```

mysql> CREATE TABLE ai_test2(id INT NOT NULL AUTO_INCREMENT,
  f1 VARCHAR(5),PRIMARY KEY(id) AUTO_INCREMENT=50;
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO ai_test2(f1) VALUES('one');
Query OK, 1 row affected (0.00 sec)
mysql> SELECT * FROM ai_test2;
+----+----+
| id | f1  |
+----+----+
| 50 | one |
+----+----+
1 row in set (0.00 sec)

```

O puede establecer el **contador** cuando la tabla ya existe:

```
mysql> DELETE FROM ai-test;
Query OK, 3 rows affected (0.00 sec)
mysql> ALTER TABLE ai_test AUTO_INCREMENT=1000;
Query OK, 0 rows affected (0.00 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> INSERT INTO ai-test(f1) VALUES('one') ;
Query OK, 1 row affected (0.00 sec)
mysql> SELECT * FROM ai-test;
+----+----+
| id  | f1  |
+----+----+
| 1000 | one |
+----+----+
1 row in set (0.01 sec)
```

En la mayor **parte** de los casos esta función se utiliza cuando la tabla esta vacia, pero no es una **condición** necesaria; puede restablecer el **contador** incluso con registros en la tabla:

```
mysql> ALTER TABLE ai_test2 AUTO_INCREMENT=500;
Query OK, 1 row affected (0.01 sec)
Records: 1 Duplicates: 0 Warnings: 0
mysql> INSERT INTO ai_test2(f1) VALUES('two');
Query OK, 1 row affected (0.01 sec)
mysql> SELECT * FROM ai_test2;
+----+----+
| id  | f1  |
+----+----+
| 50  | one |
| 500 | two |
+----+----+
2 rows in set (0.00 sec)
```

NOTA: En la actualidad, esta operación sólo funciona con las tablas **MyISAM**. Por ejemplo, el contador de incremento automático de las tablas **InnoDB** no se puede establecer en un valor diferente a 1.

En los ejemplos anteriores se insertaban registros sin especificar el campo **id**. Si prefiere utilizar una **sintaxis** alternativa en la que se especifique el valor de un campo de incremento automatico, asigne **NULL** o **0** como valor del campo incrementado automaticamente:

```
mysql> INSERT INTO ai-test VALUES(NULL, 'two');
Query OK, 1 row affected (0.01 sec)
mysql> INSERT INTO ai_test VALUES(0, 'three');
Query OK, 1 row affected (0.00 sec)
mysql> SELECT * FROM ai-test;
+----+----+
| id  | f1  |
+----+----+
```

```

| id   | f1   |
+----+-----+
| 1000 | one   |
| 1001 | two   |
| 1002 | three |
+----+-----+
3 rows in set (0.00 sec)

```

Mas allá de los límites

Fijese en que el contador automatico solo puede contener un numero positivo aunque el campo adjunto sea un campo con firma. Si intenta asignarle un numero negativo, surgiran problemas extraiios:

```

mysql> ALTER TABLE ai_test2 AUTO-INCREMENT=-500;
Query OK, 2 rows affected (0.01 sec)
Records: 2 Duplicates: 0 Warnings: 0
mysql> INSERT INTO ai_test2(f1) VALUES('three');
Query OK, 1 row affected (0.00 sec)
mysql> SELECT * FROM ai_test2;
+----+-----+
| id   | f1   |
+----+-----+
|      50 | one   |
|      500 | two   |
| 2147483647 | three |
+----+-----+
3 rows in set (0.00 sec)

```

Como -500 queda fuera del rango positivo de valores permitidos para un incremento automatico, MySQL le asigna el maximo valor permitido a un entero: 2147483647. Si intenta agregar otro registro, obtendra un error de clave duplicada porque MySQL no puede aumentar un entero por encima de dicho valor:

```

mysql> INSERT INTO ai_test2(f1) VALUES('four');
ERROR 1062: Duplicate entry '2147483647' for key 1

```

ADVERTENCIA: Asegúrese siempre de disponer de espacio suficiente para sus registros. Si crea un incremento automático para un campo SIGNED TINYINT, al alcanzar el número 127 empezará a obtener errores de clave duplicada.

Problemas con LAST-INSERT-ID()

La función LAST_INSERT_ID() consta de un numero de funciones que podrian causar problemas al utilizarlas:

- El valor devuelto por LAST_INSERT_ID() no es igual al que establecimos al restaurar el contador de incremento automatico. En su lugar vuelve a 1.

- El numero se mantiene de **conexión** en conexión, de **manera** que si se agregan otros registros en una **conexión** distinta, el numero devuelto por esta funcion no se actualizara.

A continuación se incluyen algunos ejemplos:

```
mysql> SELECT * FROM ai_test2;
+-----+-----+
| id      | f1      |
+-----+-----+
|         | 50 lone |
|         | 500 two  |
| 2147483647 | three |
+-----+-----+
3 rows in set (0.00 sec)

mysql> ALTER TABLE ai_test2 AUTO_INCREMENT=501;
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0
mysql> UPDATE ai_test2 SET id=LAST_INSERT_ID()+1 WHERE
f1='three';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

En este ejemplo esperaríamos que el id tuviera el valor 501. Sin embargo, recibimos una sorpresa poco agradable.

Examine la siguiente secuencia:

```
mysql> SELECT * FROM ai_test2;
+----+-----+
| id | f1      |
+----+-----+
| 50 | one     |
| 500 | two    |
| 1 | three  |
+----+-----+
3 rows in set (0.00 sec)
```

LAST_INSERT_ID se restablece en 1 al reiniciar el contador de incremento automático. Pero el resultado puede ser peor todavía si prueba el siguiente código:

```
mysql> ALTER TABLE ai_test2 AUTO_INCREMENT=501;
Query OK, 3 rows affected (0.01 sec)
Records: 3 Duplicates: 0 Warnings: 0
mysql> UPDATE ai_test2 SET id=LAST_INSERT_ID()+1 WHERE
f1='two';
ERROR 1062: Duplicate entry '1' for key 1
```

Ahora tenemos una clave duplicada y la instrucción UPDATE falla. El segundo error tiene lugar **cuando** se establecen varias conexiones. Abra dos ventanas para establecer dos conexiones a la base de datos.

Desde la ventana 1:

```
mysql> TRUNCATE ai_test2;
Query OK, 0 rows affected (0.01 sec)
mysql> INSERT INTO ai_test2(f1) VALUES('one');
Query OK, 1 row affected (0.00 sec)
mysql> SELECT * FROM ai_test2;
+----+-----+
| id | f1 |
+----+-----+
| 1 | one |
+----+-----+
1 row in set (0.01 sec)
mysql> SELECT LAST-INSERT-ID() FROM ai_test2;
+-----+
| last-insert-id() |
+-----+
| 1 |
+-----+
1 row in set (0.00 sec)
```

Por ahora todo correcto. A continuación, pase a la segunda ventana e inserte otro registro. Desde la ventana 2:

```
mysql> INSERT INTO ai_test2(f1) VALUES('two');
Query OK, 1 row affected (0.00 sec)
Window1:
mysql> SELECT LAST-INSERT-ID() FROM ai_test2;
+-----+
| last-insert-id() |
+-----+
| 1 |
| 1 |
+-----+
2 rows in set (0.00 sec)
```

El valor devuelto sigue siendo 1, cuando debería ser 2. Por lo tanto, si intentamos utilizar el valor para realizar una actualización, obtendremos el familiar error de clave duplicada:

```
mysql> UPDATE ai_test2 SET id=LAST_INSERT_ID()+1 WHERE
f1='one';
ERROR 1062: Duplicate entry '2' for key 1
```

Índices de varias columnas y campos de incremento automatico

Las tablas de tipo MyISAM y BDB permiten además convertir el segundo campo de índice de un índice de varias columnas en un campo de incremento automatico. Esta opción resulta de utilidad al crear agrupaciones de datos. En

este ejemplo, vamos a crear una tabla para la plantilla de trabajadores, en la que agruparemos a sus miembros en gerentes, empleados y servicios subcontratados, y se les asignara una posición en cada categoria:

```
mysql> CREATE TABLE staff(rank ENUM('Employee','Manager',
'Contractor') NOT NULL,position VARCHAR(100),
id INT NOT NULL AUTO-INCREMENT,PRIMARY KEY(rank,id));
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO staff(rank,position) VALUES
('Employee','Cleaner'),
('Contractor','Network maintenance'),
('Manager','Sales manager');
Query OK, 3 rows affected (0.01 sec)
mysql> SELECT * FROM staff;
+-----+-----+-----+
| rank | position | id |
+-----+-----+-----+
| Employee | Cleaner | 1 |
| Contractor | Network maintenance | 1 |
| Manager | Sales manager | 1 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

Los tres registros constan del mismo id, ya que la clave principal se compone de dos campos: rank e id.

Al agregar otros registros a cada rango, observara el comportamiento habitual de incremento:

```
mysql> INSERT INTO staff(rank,position) VALUES
('Employee','Security guard'),
('Employee','Receptionist'),
('Manager','Head of security');
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0
mysql> SELECT * FROM staff;
+-----+-----+-----+
| rank | position | id |
+-----+-----+-----+
| Employee | Cleaner | 1 |
| Contractor | Network maintenance | 1 |
| Manager | Sales manager | 1 |
| Employee | Security guard | 2 |
| Employee | Receptionist | 3 |
| Manager | Head of security | 2 |
+-----+-----+-----+
6 rows in set (0.01 sec)
```

En este ejemplo, tenemos un empleado 1,2 y 3; un gerente 1 y 2; y un servicio subcontratado 1. El contador de incremento automatico genera los valores correctamente para cada grupo.

Sin embargo, en este caso no se puede restablecer el contador de incremento automatico.

```
mysql> ALTER TABLE staff AUTO_INCREMENT=500;
Query OK, 6 rows affected (0.01 sec)
Records: 6 Duplicates: 0 Warnings: 0

mysql> INSERT INTO staff(rank,position) VALUES
  ('Employee','Stationary administrator'),
  ('Manager','Personnel manager'),
  ('Contractor','Programmer');
Query OK, 3 rows affected (0.01 sec)
Records: 3 Duplicates: 0 Warnings: 0
mysql> SELECT * FROM staff;
+-----+-----+-----+
| rank | position | id |
+-----+-----+-----+
| Employee | Cleaner | 1 |
| Contractor | Network maintenance | 1 |
| Manager | Sales manager | 1 |
| Employee | Security guard | 2 |
| Employee | Receptionist | 3 |
| Manager | Head of security | 2 |
| Employee | Stationary administrator | 4 |
| Manager | Personnel manager | 3 |
| Contractor | Programmer | 2 |
+-----+-----+-----+
9 rows in set (0.00 sec)
```

Los valores van aumentando a partir de donde se quedaron, independientemente de la instrucción ALTER.

Eliminación o modificación de un índice

En ocasiones, los índices dejan de ser útiles y necesitan modificarse o eliminarse. Al realizar un cambio sobre un índice, el primer paso consiste en eliminar el índice y volver a generarlo con la nueva definición.

Para eliminar una clave primaria utilice esta sintaxis

```
ALTER TABLE nombre_de_tabla DROP PRIMARY KEY;
```

Para eliminar un índice ordinario, exclusivo o de texto completo, debemos especificar el nombre del índice, de la siguiente forma:

```
ALTER TABLE nombre_de_tabla DROP INDEX nombre_de_indice;
```

o de esta otra forma:

```
DROP INDEX nombre_de_indice ON nombre_de_tabla;
```

Si no está seguro del nombre del índice, la instrucción SHOW KEYS se lo desvelará:

```
SHOW KEYS FROM nombre_de_tabla;
```

Tipos de tabla e índices

Cada tipo de tabla tiene su propio comportamiento en **materia** de índices y cada una de **ellas los procesa de manera** diferente. No todos los tipos índices **están** disponibles para los distintos tipos de tabla. Es importante tener claro **cómo** se va a utilizar una tabla y los índices que se van a necesitar antes de seleccionar el tipo de tabla. En ocasiones, lo que parece ser el tipo de tabla **perfecta** se convierte en una **elección** errónea porque no se puede utilizar un determinado **tipo de índice en ella**. En la siguiente lista se destacan las **funciones** y las diferencias de índices para cada **tipo de tabla**. Las tablas **MyISAM** presentan las siguientes características:

- Los índices se almacenan en archivos con la extensión **.MY I**.
- Los índices de número se almacenan con el byte alto **primero** para permitir una mejor compresión del **índice**.
- Se pueden utilizar índices **BLOB** y **TEXT**.
- Se **permiten** valores nulos en los índices (no claves primarias).
- Los datos y el **índice** se pueden incluir en **directorios** diferentes (lo que **permite** una mayor velocidad).

Las tablas **MERGE** presentan las siguientes características:

- Las tablas **MERGE** no contienen índices propios.
- El **archivo .MRG** contiene una lista de los archivos **.MY I** de **índice** procedentes de las tablas **MyISAM** integrantes.
- Sigue siendo necesario especificar los índices **al** crear la tabla **MERGE**.

Las tablas **HEAP** presentan las siguientes características:

- Utilizan un **índice** de asignación almacenado en **memoria**, que resulta muy rápido.
- Solo pueden utilizar índices con los operadores **=** y **<=>**
- No pueden usar un **índice** en una **columna** que permita valores **NULL**.
- Los índices no se pueden utilizar con la **cláusula** **ORDER BY**.
- **MySQL** no puede determinar el número aproximado de filas que **existen** entre los dos valores (este resultado es utilizado por el optimizador de consultas para seleccionar el **índice** más eficaz que utilizar). En una **sección** posterior se ampliará este tema.

Las tablas **ISAM** utilizan un **índice** B-Tree almacenado en archivos con la extensión **.i s m**.

Las tablas **InnoDB** no pueden utilizar índices de texto completo.

Uso eficaz de los índices

Las tablas con pocos índices devolverán los resultados muy despacio. Pero la inclusión de demasiados índices, aunque no suele ser normal, también ocasiona problemas. Los índices ocupan espacio de disco y, como están ordenados, cada vez que se realice una operación de inserción o de actualización, es necesario volver a organizar el índice para incluir los cambios, lo que da como resultado una carga de trabajo adicional significativa. En las siguientes secciones se explica cuando utilizar los índices. La eficiencia en el uso de índices depende de la configuración de MySQL (en un capítulo posterior se abordará este tema).

Donde utilizar los índices

El uso más común de un índice consiste en recuperar filas que cumplan una condición incluida en una cláusula WHERE:

```
mysql> SELECT first_name FROM customer WHERE surname>'C';
+-----+
| first_name |
+-----+
| Yvonne     |
| Johnny     |
| Winston    |
| Patricia   |
| Francois   |
| Winnie     |
| Breyton    |
+-----+
7 rows in set (0.00 sec)
```

En este caso, resultaría útil crear un índice sobre el campo surname. No se utilizaría un índice sobre el campo first_name en esta consulta porque no forma parte de la condición.

Los campos que aparecen únicamente en la lista de campos (inmediatamente después de SELECT) no utilizan un índice.

Al buscar valores máximos o mínimos, MySQL sólo necesita tomar el primer valor o el último de una tabla ordenada con un índice, lo que resulta extremadamente rápido.

Si se solicitan valores máximos o mínimos con frecuencia, resultaría extremadamente útil crear un índice sobre el campo pertinente.

```
mysql> SELECT MAX(id) FROM customer;
+-----+
| MAX(id) |
+-----+
|         10 |
+-----+
```

Existe otro caso en el que MySQL nunca necesita examinar la tabla completa, sólo el **índice**: cuando todos los campos que se desean recuperar forman parte de un **índice**. Observe el siguiente ejemplo:

```
mysql> SELECT id FROM customer;
+----+
| id |
+----+
|  1 |
|  2 |
|  3 |
|  4 |
|  5 |
|  6 |
|  7 |
| 10 |
+----+
8 rows in set (0.01 sec)
```

Si creamos un **índice sobre** el campo `id`, MySQL ni siquiera necesitaría examinar los datos. Esto no sería de aplicación si el **índice** se compusiera únicamente de una parte de los datos de la columna (por ejemplo, si el **índice** se hubiera creado **sobre los** primeros diez caracteres de un campo de tipo VARCHAR de 20 caracteres de longitud).

Otro caso en el que los índices resultan útiles es cuando se utiliza la instrucción `ORDER BY` para ordenar un campo, como en el siguiente ejemplo:

```
mysql> SELECT * FROM customer ORDER BY surname;
+----+-----+-----+
| id | first-name | surname |
+----+-----+-----+
|  6 | Neil       | Beneke  |
|  2 | Johnny    | Chaka-Chaka |
|  1 | Yvonne    | Clegg   |
|  7 | Winnie    | Dlamini |
|  4 | Patricia  | Mankunku |
|  5 | Francois  | Papo    |
|  3 | Winston   | Powers  |
| 10 | Breyton   | Tshbalala |
+----+-----+-----+
8 rows in set (0.01 sec)
```

Como los registros se devuelven ordenados por un campo, que es la tarea que realizan los índices, la **creación** de un **índice sobre** el campo `surname` resultaría útil en esta consulta. Si el tipo de orden solicitado es descendente, bastaría con leer el **índice en orden inverso**.

ADVERTENCIA: En la actualidad, los índices no se pueden utilizar en cláusulas `ORDER BY` con tablas **HEAP**.

Los índices también se utilizan para agilizar combinaciones, como en el siguiente ejemplo:

```
mysql> SELECT first_name,surname,commission FROM
sales,sales_rep
WHERE code=8 AND sales.sales_rep=sales_rep.employee_number;
+-----+-----+-----+
| first-name | surname | commission |
+-----+-----+-----+
| Mike      | Serote  |          10 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Se utiliza un índice para llevar a cabo la condición de combinación (en otras palabras, para buscar en los campos sales.sales_rep y sales_rep.employee_number). Esto resulta de aplicación aunque se utilice la sintaxis alternativa:

```
mysql> SELECT first_name,surname,commission FROM sales INNER
JOIN sales_rep ON sales.sales_rep=sales_rep.employee_number
WHERE code=8;
+-----+-----+-----+
| first-name | surname | commission |
+-----+-----+-----+
| Mike      | Serote  |          10 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Los índices se pueden utilizar cuando la consulta contenga un comodín, como:

```
mysql> SELECT * FROM sales_rep WHERE surname LIKE 'Ser%';
+-----+-----+-----+
| first-name | surname | commission |
+-----+-----+-----+
| Mike      | Serote  |          10 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Sin embargo, los índices no se pueden utilizar en el siguiente caso:

```
mysql> SELECT * FROM sales_rep WHERE surname LIKE '%Ser%';
+-----+-----+-----+
| first-name | surname | commission |
+-----+-----+-----+
| Mike      | Serote  |          10 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

La diferencia está en que en el último ejemplo se utiliza un comodín como primer carácter y, como el índice se ordena alfabéticamente a partir del primer carácter, la presencia del comodín inutiliza la función del índice.

Selección de índices

Ahora que ya sabemos donde utiliza MySQL los índices, tenga en cuenta los siguientes consejos a la hora de seleccionarlos:

- Los índices **sólo** deberían crearse en aquellas consultas que los utilicen (por ejemplo, **sobre** campos de la condicion WHERE) y no **sobre** campos que no vayan a utilizarlos (como en caso de que el primer caracter de la condicion sea un comodin).
- Cree índices que devuelvan el **menor** numero de **filas** posible. El mejor lugar es **sobre** una clave **primaria** ya que estas se asocian de **manera exclusiva** a un registro. De **manera** similar, los índices **sobre** campos **enumerados** no **resultan** particularmente utiles (por ejemplo, un **índice** sobre un campo que contenga valores **sí** o **no**, **sólo** serviria para reducir la **seleccion** a la **mitad**, con toda la carga que **supone** el mantenimiento de un **índice**).
- Utilice índices cortos (**Cree un índice sobre los diez primeros caracteres** de un nombre, por ejemplo, en lugar de hacerlo **sobre** el campo completo).
- No **Cree** demasiados índices. Los índices aumentan el tiempo necesario para actualizar o agregar un registro. Por **ello**, si el **índice** se crea para una **consulta** que **se** utilice en **contadas** ocasiones y pueda aceptarse un **rendimiento** ligeramente mas lento, considere la **opción** de no crear el **índice**.
- Utilice el sistema de prefijacion mas a la izquierda (vease la siguiente seccion).

Uso del sistema de prefijacion mas a la izquierda

Ya sabemos que podemos crear un **índice** sobre varios campos. Los apellidos y los nombres son buenos ejemplos en los que hacerlo, de **manera** que, aunque existan muchos apellidos duplicados, el nombre hara que el **índice** resulte **prácticamente** unico. En efecto, MySQL dispondra en este caso de dos índices. El **primero** es el apellido y el nombre, y el segundo es simplemente el apellido. Comenzando por la parte izquierda de la lista de campos del **índice**, MySQL puede utilizar **cada** uno de ellos, uno tras otro, siempre y cuando sigan la secuencia empezando por la izquierda. Utilizaremos algunos ejemplos para aclarar este concepto. En el siguiente **fragmento** agregaremos un campo **inicial** a la tabla de clientes, algunos valores, asi como un **índice**:

```
mysql> ALTER TABLE customer ADD initial VARCHAR(5);
Query OK, 8 rows affected (0.01 sec)
Records: 8 Duplicates: 0 Warnings: 0
mysql> ALTER TABLE customer ADD INDEX (surname,initial,
first-name);
```

```

Query OK, 8 rows affected (0.01 sec)
Records: 8 Duplicates: 0 Warnings: 0
mysql> UPDATE customer SET initial='X' WHERE id=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
mysql> UPDATE customer SET initial='B' WHERE id=2;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
mysql> UPDATE customer SET initial='M' WHERE id=3;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
mysql> UPDATE customer SET initial='C' WHERE id=4;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
mysql> UPDATE customer SET initial='P' WHERE id=5;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
mysql> UPDATE customer SET initial='B' WHERE id=10;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

```

Si realiza una **consulta** incluyendo los tres campos en la **condición**, lograremos aprovechar **al** máximo el **índice**:

```

mysql> SELECT * FROM customer WHERE surname='Clegg' AND
initial='X' AND first_name='Yvonne';

```

También sacariamos el máximo **partido** del **índice** si buscáramos por el **apellido** y la **inicial**:

```

mysql> SELECT * FROM customer WHERE surname='Clegg' AND
initial='X':

```

o simplemente por el **apellido**:

```

mysql> SELECT * FROM customer WHERE surname='Clegg';

```

Sin embargo, si rompe la **secuencia de disposición** más a la izquierda y realiza la **busqueda** por el **nombre** o la **inicial** o por **ambos** campos, **MySQL** no utilizará el **índice**. Por ejemplo, ninguna de las siguientes **busquedas** utiliza un **índice**:

```

mysql> SELECT * FROM customer WHERE initial='X' AND
first_name='Yvonne';
mysql> SELECT * FROM customer WHERE first_name='Yvonne';
mysql> SELECT * FROM customer WHERE initial='X';

```

Si realiza la **busqueda** por el **primer** y **tercer** campo del **índice** (**surname** y **first_name**), romperíamos la **secuencia** y el **índice** no se utilizaría **completamente**. Sin embargo, como el **apellido** constituye la **primera** parte del **índice**, se seguirá utilizando esta **porción**:

```

mysql> SELECT * FROM customer WHERE surname='Clegg' AND
first_name='Yvonne';

```

```

+-----+-----+-----+
| id | first-name | surname | initial |
+-----+-----+-----+
| 1 | Yvonne | Clegg | X |
+-----+-----+-----+
1 row in set (0.00 sec)

```

Puede utilizar el sistema de **prefijación** a la izquierda siempre y cuando se utilice un índice, como en el caso de utilizar una cláusula ORDER BY.

Como utiliza MySQL los índices con EXPLAIN

Uno de los **secretos** mejor guardados de MySQL es la instrucción EXPLAIN. Incluso la gente que lleva años utilizando MySQL parece haber pasado por alto esta instrucción y la verdad es que facilita enormemente el trabajo.

La instrucción EXPLAIN muestra (explica) la **forma** en que MySQL procesa las instrucciones SELECT, utiliza índices y combina tablas. Esta instrucción puede ayudarle a seleccionar índices mejores y a escribir sus consultas de **manera** opcional.

Para utilizarla, colóquela delante de una instrucción SELECT:

```

mysql> EXPLAIN SELECT surname,first_name FROM customer WHERE
id=1;
+-----+-----+-----+-----+-----+-----+
| table | type | possible-keys | key | key-len | ref |
| rows | Extra |
+-----+-----+-----+-----+-----+-----+
| customer | const | PRIMARY | PRIMARY | 4 | const |
| 1 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

¿Qué significan todas estas columnas?
En la tabla 4.2 encontrará la respuesta.

Tabla 4.2. Significado de las columnas EXPLAIN

Columna	Descripción
table	Muestra a que tabla se refiere el resto de la columna (en este ejemplo es obvio, pero resulta útil cuando se combina mas de una tabla en una columna).
type	Esta columna es importante porque indica el tipo de combinación que se esta utilizando. Los tipos de combinación son, ordenados de mejor al peor, const, eq_ref, ref, range, index y ALL.

Columna	Descripción
<code>possible_keys</code>	Muestra los índices que se podrían aplicar a la tabla. Si esta vacía, no existirán índices disponibles. Podemos disponer de uno si realizamos una búsqueda sobre un campo relevante desde la cláusula <code>WHERE</code> .
<code>key</code>	El índice que se está utilizando. Si el valor de esta columna fuera <code>NULL</code> , no estamos utilizando ningún índice. En ocasiones, MySQL selecciona un índice que resulta menos óptimo que otro. En este caso, puede obligar a MySQL a seleccionar otro índice utilizando <code>USE INDEX (nombre de índice)</code> dentro de la instrucción <code>SELECT</code> o a ignorar un índice con <code>IGNORE INDEX (nombre de índice)</code> .
<code>key_len</code>	La longitud del índice utilizado. Cuanto menor sea su tamaño (sin detrimento de la exactitud), mejor.
<code>ref</code>	Indica la columna del índice que se está utilizando o una constante, si fuera posible.
<code>rows</code>	Número de filas que MySQL considera que debe examinar para poder devolver los datos requeridos.
<code>extra</code>	Información extra sobre cómo resulte MySQL la consulta. Estas opciones se examinan en la tabla 4.3. En este caso nos interesa estar atentos a <code>using temporary</code> y <code>Using filesort</code> , que indican que MySQL no puede utilizar el índice en absoluto y que los resultados se recuperarán lentamente.

La tabla 4.3 recoge las descripciones de la columna `extra`.

Tabla 4.3. Significado de las descripciones de la columna `extra` `EXPLAIN`

Descripción de la columna <code>extra</code>	Descripción
<code>Distinct</code>	Cuando MySQL encuentra una fila que coincide con la combinación de fila, deja de buscar otras.
<code>Not exists</code>	MySQL optimizó la combinación por la izquierda y, tras encontrar una fila que cumpla los criterios de esta combinación, deja de buscar otras.
<code>range checked for each record (index map: #)</code>	No se encontró un índice ideal, de manera que para cada combinación de filas de las tablas anteriores, MySQL comprueba que índice utilizar y lo utiliza para recuperar filas de la tabla. Se trata de una de las combinaciones más lentas con un índice.

Descripción de la columna extra	Descripción
Using filesort	Cuando vea esta descripción, sepa que la consulta necesita optimizarse. MySQL necesitará realizar un paso extra para determinar cómo ordenar las filas que devuelve. Para ordenarlas, recorre todas las filas en función del tipo de combinación y almacena la clave de ordenación y un puntero a la fila de todas las filas que cumplen la condición . A continuación, ordena las claves y, finalmente, devuelve las filas en el orden determinado.
Using index	Los datos de la columna se devuelven desde la tabla utilizando únicamente la información del índice , sin necesidad de leer la fila. Esto ocurre cuando todas las columnas requeridas de la tabla forman parte del mismo índice .
Using temporary	Cuando vea esta descripción, sepa que la consulta necesita optimizarse. En este caso MySQL necesita crear una tabla temporal para recoger los datos, lo que suele ocurrir al realizar una operación de ordenación sobre un conjunto de columnas diferentes a las agrupadas.
Where used	Una cláusula <code>WHERE</code> se utiliza para restringir las filas que se utilizarán en la selección de la tabla siguiente o que se devolverán al cliente. Si no de sea recuperar todas las filas de la tabla y el tipo de combinación es <code>ALL</code> o <code>index</code> , debería aparecer esta descripción; de lo contrario, es posible que su consulta presente algún problema.

El tipo de columna devuelta por `EXPLAIN` indica el tipo de combinación que se está utilizando. La tabla 4.4 explica el tipo de combinación, ordenadas de mayor a menor por su eficacia.

Tabla 4.4. Los diferentes tipos de combinación

Combinación	Descripción
<code>system</code>	La tabla solo tiene una fila: una tabla de sistema. Éste es un caso especial del tipo de combinación <code>const</code> .
<code>const</code>	El número máximo de coincidencias que puede extraer esta consulta de la tabla es de un registro (el

Combinación	Descripción
	índice sera una clave primaria o un índice exclusivo). Si solo hay una fila, el valor es una constante, ya que MySQL lee el valor y, a continuación, lo utiliza de forma identica a una constante.
eq_ref	En una combinacion, MySQL lee un registro de la tabla para cada combinacion de registros de las tablas anteriores de la consulta. Se utiliza cuando la consulta utiliza partes de un índice que sea una clave primaria o una clave unica.
ref	Este tipo de combinacion ocurre si la consulta utiliza una clave que no sea unica o primaria o que solo forme parte de uno de estos tipos (por ejemplo, si utiliza el sistema de prefijacion mas a la izquierda). Todos los registros que coincidan seran leidos de la tabla para cada combinacion de filas de las tablas anteriores. Este tipo de combinacion dependen enormemente de la cantidad de registros que coincidan en el índice (cuantos menos mejor).
index	Este tipo de combinacion examina el índice completo para cada combinacion de registros de las tablas anteriores (lo que resulta mejor que la opcion ALL , ya que el tamaño de los indices suele ser menor que los datos de la tabla).
ALL	Esta combinacion examina toda la tabla para cada combinacion de registros de las tablas anteriores. Por regla general se trata de una combinacion muy mala y debe evitarse siempre que resulte posible.

Volvamos al ejemplo:

```
mysql> EXPLAIN SELECT surname,first_name FROM customer WHERE
id=1;
+----+-----+-----+-----+-----+
| table      | type  | possible-keys | key      | key-len | ref  |
| rows | Extra |               |          |         |     |
+----+-----+-----+-----+-----+
| customer | const | PRIMARY      | PRIMARY | 4       | const
| 1       |      |               |          |         |     |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Ya tenemos una clave primaria en la tabla cliente sobre el campo **id** y como nuestra consulta consta unicamente de una condición, este campo es equivalente a una constante por lo que la consulta no puede mejorarse mas. La columna **rows**

indica que **MySQL** sólo necesita examinar una fila para devolver los resultados. No se puede obtener un proceso mejor. Así mismo, el tipo de la combinación (que en este caso no es tal) es `const`, que equivale a constante, que es el mejor tipo. Examinemos que ocurriría si realiza una consulta similar sobre una tabla sin índices:

```
mysql> EXPLAIN SELECT * FROM sales-rep WHERE employee_number=2;
+-----+-----+-----+-----+-----+-----+
| table      | type | possible-keys | key  | key-len | ref  |
+-----+-----+-----+-----+-----+
| sales-rep | ALL  | NULL          | NULL | NULL    | NULL |
5 | where used |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Se trata de la peor combinación. La combinación es de tipo `ALL`, la peor de todas: no existen claves posibles y se examinan las cinco filas para devolver los resultados (la tabla `sales_rep` consta únicamente de cinco registros). Veamos cómo podemos mejorar esta situación:

```
mysql> SHOW COLUMNS FROM sales-rep;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| employee-number | int(11)   | YES  |     | NULL    |      |
| surname       | varchar(40) | YES  |     | NULL    |      |
| first-name    | varchar(30) | YES  |     | NULL    |      |
| commission    | tinyint(4) | YES  |     | NULL    |      |
| date-joined   | date      | YES  |     | NULL    |      |
| birthday      | date      | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
mysql> SELECT * FROM sales-rep;
+-----+-----+-----+-----+-----+-----+
| employee-number | surname   | first-name | commission | date-joined | birthday |
+-----+-----+-----+-----+-----+-----+
| 1 | Rive | Sol | 10 | 2000-02-15 | 1976-03-18 |
| 2 | Gordimer | Charlene | 15 | 1998-07-09 | 1958-11-30 |
| 3 | Serote | Mike | 10 | 2001-05-14 | 1971-06-18 |
| 4 | Rive | Mongane | 10 | 2002-11-23 | 1982-01-04 |
```

```

|          5 | Jomo      | Igenesund |          10 | 2002-
11-29 | 1968-12-01 |
+-----+-----+-----+-----+
5 rows in set (0.01 sec)

```

Un candidato obvio para una clave **primaria** es el campo `employee_number`. No existen valores duplicados y no deseara tener ninguno de **manera que puede convertirla en una clave primaria** sin demasiadas complicaciones:

```

mysql> ALTER TABLE sales-rep MODIFY employee-number INT NOT NULL
PRIMARY KEY;
Query OK, 5 rows affected (0.00 sec)
Records: 5 Duplicates: 0 Warnings: 0

```

El resultado se aprecia inmediatamente si vuelve a ejecutar el comando **EXPLAIN** sobre esta consulta:

```

mysql> EXPLAIN SELECT * FROM sales-rep WHERE employee_number=2;
+-----+-----+-----+-----+-----+
| table | type | possible-keys | key | key-len | ref |
| rows | Extra |
+-----+-----+-----+-----+
| sales-rep | const | PRIMARY | PRIMARY | 4 | const |
| 1 | |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

La mejora es sobresaliente.

La **instrucción EXPLAIN** es también **sinónimo** de las secuencias `DESCRIBE nombre_de_tabla` o `SHOW COLUMNS FROM nombre_de_tabla` si se utilizan **delante** de un nombre de tabla.

Realización de calculos en una consulta

Vamos a examinar algunos **casos** mas complejos. Por ejemplo, suponga que deseamos recuperar todos los comerciales con una comision inferior **al 20** por ciento **si** le sumaramos un 5 por ciento a la que **tienen** asignada. **A continuación se incluye** una posible consulta con la **instrucción EXPLAIN**:

```

mysql> EXPLAIN SELECT * FROM sales-rep WHERE (commission+5)<20;
+-----+-----+-----+-----+-----+
| table | type | possible-keys | key | key-len | ref |
| rows | Extra |
+-----+-----+-----+-----+
| sales-rep | ALL | NULL | NULL | NULL | NULL |
5 | where used |
+-----+-----+-----+-----+

```

El resultado no es muy bueno ya que la consulta no utiliza ninguno de los **indices**. No deberiamos sorprendernos porque **sólo** existe un **índice**, una clave **primaria**

sobre el campo `employee_number`. A primera vista, si le añadieramos un índice al campo `commission` los resultados mejorarian. Como este campo consta de valores duplicados no se le puede asignar una clave primaria (cuyo uso se reduce a una por tabla) o una clave unica. Y como tampoco se trata de un campo de caracter, la unica opción disponible es recurrir a un índice ordinario:

```
mysql> ALTER TABLE sales-rep ADD INDEX(commission);
Query OK, 5 rows affected (0.01 sec)
Records: 5 Duplicates: 0 Warnings: 0
```

A continuacion, si vuelve a ejecutar la consulta obtendra el siguiente resultado:

```
mysql> EXPLAIN SELECT * FROM sales-rep WHERE (commission+5)<20;
+----+-----+-----+-----+-----+-----+
| table | type | possible-keys | key | key-len | ref |
rows | Extra |
+----+-----+-----+-----+-----+-----+
| sales-rep | ALL | NULL | NULL | NULL | NULL |
5 | where used |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

No hemos conseguido ninguna mejora. MySQL sigue examinando cada uno de los registros. La razon es que debe calcular `commission+5` para cada registro. Necesita leer el campo `commission` de cada registro para agregarle 5 unidades y, continuacion, compararlo con 20. No deberiamos realizar ningun calculo en el campo de índice. La solución consiste en llevar a cabo el calculo sobre la constante, no sobre el campo indexado. En terminos algebraicos, $(x + 5 < y)$ es lo mismo que $(x < y - 5)$, por lo que se puede escribir de la siguiente forma:

```
mysql> EXPLAIN SELECT * FROM sales-rep WHERE commission<20-5;
+----+-----+-----+-----+-----+-----+
| table | type | possible-keys | key | key_len |
ref | rows | Extra |
+----+-----+-----+-----+-----+
| sales-rep | range | commission | commission | 2 |
NULL | 3 | where used |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

El resultado mejora bastante. MySQL realiza los calculos una vez, obtiene la constante 15 y recorre el índice buscando valores inferiores. Tambien podriamos haber utilizado la consulta de la siguiente forma:

```
mysql> EXPLAIN SELECT * FROM sales-rep WHERE commission<15;
+----+-----+-----+-----+-----+-----+
| table | type | possible-keys | key | key_len |
ref | rows | Extra |
+----+-----+-----+-----+-----+
| sales-rep | range | commission | commission | 2 |
NULL | 3 | where used |
```

```

+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

donde nosotros mismos obtenemos la constante, **pero** la diferencia de velocidad apenas resulta perceptible. La operacion de restar 5 unidades a 20, no le supone ningun trabajo a MySQL (intente medirlo si puede). Fijese en que ocurriria si quisieramos recuperar todos los comerciales con un 20 por ciento exacto de comision tras el aumento del 5 por ciento:

```

mysql> EXPLAIN SELECT * FROM sales_rep WHERE commission=15;
+-----+-----+-----+-----+-----+-----+
| table      | type | possible-keys | key          | key-len | ref |
| rows      | Extra |               |             |         |     |
+-----+-----+-----+-----+-----+-----+
| sales_rep | ref  | commission    | commission  |         | 2 |
const      | 1 | where used    |             |         |   |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

El tipo de consulta varia de range a ref, cuyo uso es preferible ya que la operacion de devolver un valor exacto supone menos trabajo que devolver un rango de valores (o una gran cantidad de valores exactos).

Uso de EXPLAIN con el sistema de prefijos a la izquierda

Vamos a volver sobre el tema del sistema de prefijacion mas a la izquierda y a ver cómo EXPLAIN puede ayudarnos a entenderlo mejor. Considere la siguiente consulta:

```

mysql> EXPLAIN SELECT * FROM customer WHERE
first_name='Yvonne';
+-----+-----+-----+-----+-----+-----+
| table      | type | possible-keys | key          | key-len | ref |
| rows      | Extra |               |             |         |     |
+-----+-----+-----+-----+-----+-----+
| customer  | ALL  | NULL          | NULL        |         | NULL |
8 | where used |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)

```

Como el campo `first_name` no es la parte del índice situada mas a la izquierda, no nos sirve para funciones de indexación y el tipo de combinación ALL nos lo indica claramente. El siguiente ejemplo muestra el uso que hace la instrucción EXPLAIN del sistema de prefijacion mas a la izquierda:

```

mysql> EXPLAIN SELECT * FROM customer WHERE surname='Clegg'
AND initial='X' AND first_name='Yvonne';
+-----+-----+-----+-----+-----+-----+
| table      | type | possible-keys | key          | key-len | ref |
| rows      | Extra |               |             |         |     |
+-----+-----+-----+-----+-----+-----+

```

```

+-----+-----+-----+-----+-----+
| customer | ref | surname | surname | 78 |
const,const,const | 1 | where used |
+-----+-----+-----+-----+
1 row in set (0.01 sec)

```

En este ejemplo, se utilizan los tres campos completos del índice y el tipo de combinación es `ref` porque el índice en cuestión permite duplicados. Si la estructura de la tabla excluye la posibilidad de una combinación duplicada de apellidos, iniciales y nombres, el tipo de combinación debería ser `eq_ref`. Fijese en la columna `ref`, `const,const,const`, que indica que las tres partes del índice se comparan con un valor constante. El siguiente ejemplo ilustra una situación similar.

```

mysql> EXPLAIN SELECT * FROM customer WHERE
      surname='Clegg' AND initial='X';
+-----+-----+-----+-----+-----+
| table | type | possible-keys | key | key-len | ref |
rows | Extra | | | | |
+-----+-----+-----+-----+-----+
| customer | ref | surname | surname | 47 |
const,const | 1 | where used |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

Nuevamente, el índice se utiliza correctamente, pero en este caso sólo se usan los dos primeros campos. La longitud de la clave en este caso es mas corta (lo que significa que MySQL tiene menos que examinar y, por lo tanto, lo hace de forma mas rapida). El siguiente ejemplo no utiliza el sistema de prefijación más a la izquierda:

```

mysql> EXPLAIN SELECT * FROM customer WHERE initial='X';
+-----+-----+-----+-----+-----+
| table | type | possible-keys | key | key-len | ref |
rows | Extra | | | | |
+-----+-----+-----+-----+-----+
| customer | ALL | NULL | NULL | NULL | NULL |
8 | where used |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

Esta consulta no se adhiere a los principios del sistema de prefijacion mas a la izquierda y no utiliza un índice. El siguiente ejemplo tampoco utiliza el sistema de prefijacion mas a la izquierda pero si utiliza un índice:

```

mysql> EXPLAIN SELECT * FROM customer WHERE surname='Clegg'
AND first_name='Yvonne';
+-----+-----+-----+-----+-----+
| table | type | possible-keys | key | key-len | ref |
rows | Extra | | | | |
+-----+-----+-----+-----+-----+

```

```

| customer | ref | surname | surname | 41 | const |
1 | where used |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

Aunque en este ejemplo no se utiliza el sistema de prefijacion mas a la izquierda, porque el campo `first_name` queda fuera de secuencia, basta el campo `surname` para aprovechar dicho índice. En este caso, limita el número de filas que MySQL necesita examinar a una, ya que el apellido Clegg es unico, independientemente de los nombres y las iniciales.

Optimización de las selecciones

En una combinacion, puede calcular el numero de filas que MySQL necesita buscar multiplicando todas las filas juntas. En el siguiente ejemplo, MySQL necesitara examinar $5*8*1$ filas, lo que da un total de 40:

```

mysql> EXPLAIN SELECT * FROM customer,sales-rep,sales
WHERE sales.sales_rep=sales_rep.employee_number
AND customer.id=sales.id;
+-----+-----+-----+-----+
| table | type | possible-keys | key | key-len | ref |
| rows | Extra | | | | |
+-----+-----+-----+-----+
| sales-rep | ALL | PRIMARY | NULL | NULL | NULL |
| 5 | | | | | |
| sales | ALL | NULL | NULL | NULL | NULL |
| 8 | where used | | | | |
| customer | eq_ref | PRIMARY | PRIMARY | 4 | |
sales.id | 1 | | | | |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

```

Como puede observar, cuanto mayor sea el numero de tablas que se combinan, mayor sera la cantidad de filas examinadas. Parte del buen diseño de las bases de datos consiste en hallar un equilibrio entre las tablas pequeñas de las bases de datos que necesitan mas combinaciones y las tablas de mayor tamaño que resultan mas difíciles de mantener. En un capítulo posterior se presentaran algunas tecnicas utiles para conseguir este objetivo.

Para tareas de optimización, vamos a concentrarnos en la combinacion de `sales_rep` y `sales` de la consulta anterior. A continuación, se ha recreado dicha combinacion (utilizando la sintaxis alternativa de la instrucción `LEFT JOIN`):

```

mysql> EXPLAIN SELECT * FROM sales-rep LEFT JOIN sales
ON sales.sales_rep = sales_rep.employee_number;
+-----+-----+-----+-----+
| table | type | possible-keys | key | key-len | ref |
| rows | Extra | | | | |
+-----+-----+-----+-----+

```

```

+-----+-----+-----+-----+-----+
| sales-rep | ALL | NULL | NULL | NULL | NULL |
5 |
| sales | ALL | NULL | NULL | NULL | NULL |
8 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

El número de **filas** que se van a examinar en esta consulta es cinco veces **ocho** (de la **columna** rows), cuyo resultado es 40. No se utiliza ningún **índice**. Si examina la estructura de las dos **primeras** tablas, verá por qué:

```

mysql> DESCRIBE sales;
+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| code       | int(11)   |      | PRI | 0        |       |
| sales-rep  | int(11)   | YES  |     | NULL     |       |
| id         | int(11)   | YES  |     | NULL     |       |
| value      | int(11)   | YES  |     | NULL     |       |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
mysql> DESCRIBE sales-rep;
+-----+-----+-----+-----+-----+
| Field              | Type              | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| employee-number    | int(11)           |      | PRI | 0        |       |
| surname            | varchar(40)       | YES  |     | NULL     |       |
| first-name         | varchar(30)       | YES  |     | NULL     |       |
| commission         | tinyint(4)        | YES  | MUL | NULL     |       |
| date-joined        | date              | YES  |     | NULL     |       |
| birthday           | date              | YES  |     | NULL     |       |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

```

Como no tenemos ninguna **condición** WHERE, la **consulta** devolverá todos los registros de la **primera** tabla (sales_rep). A continuación, se llevará a cabo la **combinación** entre la tabla sales_rep (para la que no se puede utilizar un **índice** porque vamos a recuperar todos los valores) y el campo sales_rep de la tabla sales (sin indexar). El problema está en que la **condición** de combinación no utiliza ningún **índice**. Si agrega un **índice** al campo sales_rep, mejorará el rendimiento:

```
mysql> CREATE INDEX sales-rep ON sales(sales_rep);
```

```

Query OK, 8 rows affected (0.01 sec)
Records: 8 Duplicates: 0 Warnings: 0
mysql> EXPLAIN SELECT * FROM sales_rep LEFT JOIN sales
ON sales.sales_rep = sales_rep.employee_number;
+-----+-----+-----+-----+-----+-----+
| table      | type | possible-keys | key          | key-len | ref |
| rows | Extra |
+-----+-----+-----+-----+-----+-----+
| sales_rep | ALL  | NULL          | NULL        | NULL    | NULL |
| 5 | |
| sales      | ref  | sales_rep    | sales_rep   | 5 |
sales_rep.employee_number | 2 | |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

Se ha reducido el numero de filas que MySQL necesita leer de 40 a 10 (5*2), lo que supone una gran mejora.

Si lleva a cabo la combinacion por la izquierda al revés (de forma que la tabla sales contenga los posibles valores nulos en lugar de la tabla sales_rep), obtendra un resultado diferente con EXPLAIN:

```

mysql> EXPLAIN SELECT * FROM sales LEFT JOIN sales_rep ON
sales.sales_rep = sales_rep.employee_number;
+-----+-----+-----+-----+-----+-----+
| table      | type | possible-keys | key          | key-len | ref |
| rows | Extra |
+-----+-----+-----+-----+-----+-----+
| sales      | ALL  | NULL          | NULL        | NULL    | NULL |
| 8 | |
| sales_rep | eq_ref | PRIMARY      | PRIMARY     | 4 |
sales.sales_rep | 1 | |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

Sólo se examinan ocho filas porque aunque se recuperan todas las filas de la tabla sales se esta utilizando la clave primaria de la tabla sales_rep (employee_number) para realizar la combinaci3n.

Examine otro ejemplo:

```

mysql> EXPLAIN SELECT * FROM sales_rep LEFT JOIN sales
ON sales.sales_rep = sales_rep.employee_number
WHERE sales.sales_rep IS NULL;
+-----+-----+-----+-----+-----+-----+
| table      | type | possible-keys | key          | key-len | ref |
| rows | Extra |
+-----+-----+-----+-----+-----+-----+
| sales_rep | ALL  | NULL          | NULL        | NULL    | NULL |
| 5 | |
| sales      | ref  | sales_rep    | sales_rep   | 5 |
sales_rep.employee_number | 2 | where used |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

A continuación, si cambia el campo `sales_rep` para evitar valores nulos, verá algunos cambios:

```
mysql> ALTER TABLE sales CHANGE sales_rep sales_rep INT NOT
NULL;
Query OK, 8 rows affected (0.00 sec)
Records: 8 Duplicates: 0 Warnings: 0
mysql> EXPLAIN SELECT * FROM sales_rep LEFT JOIN sales ON
  sales.sales_rep = sales_rep.employee_number WHERE
  sales.sales_rep IS NULL;
```

table	type	possible-keys	key	key-len	ref
rows	Extra				
sales_rep	ALL	NULL	NULL	NULL	NULL
5					
sales_rep	ref	sales_rep	sales_rep	4	
sales_rep-employee-number		2	where used; Not exists		

```
2 rows in set (0.00 sec)
```

Fijese en que la longitud del índice (indicada en campo `key_len`) es 4 y no 5. Como ya no se permiten valores nulos, los registros no necesitan almacenar información al respecto, por lo que el tamaño se reduce en un byte. Fijese también en el comentario `Not exists` de la columna `Extra`. Como el campo `sales_rep` ya no contiene valores nulos, cuando MySQL encuentra un registro que cumple los criterios de la combinación por la izquierda, no necesita seguir buscando.

El orden en el que las tablas se presentan a MySQL puede, en algunos casos, marcar la diferencia en cuanto a la velocidad de la consulta. MySQL intenta seleccionar las mejores opciones, pero no siempre sabe por adelantado cuál será el camino más rápido. En la siguiente sección se explica cómo ayudar a MySQL a almacenar con anticipación la mayor cantidad de información posible sobre la composición del índice, pero, como muestra el siguiente ejemplo, a veces ni siquiera resulta suficiente. En primer lugar, cree cuatro tablas idénticas:

```
mysql> CREATE TABLE t1 (f1 int unique not null, primary
key(f1));
Query OK, 0 rows affected (0.15 sec)
mysql> CREATE TABLE t2 (f2 int unique not null, primary
key(f2));
Query OK, 0 rows affected (0.15 sec)
mysql> CREATE TABLE t3 (f3 int unique not null, primary
key(f3));
Query OK, 0 rows affected (0.15 sec)
mysql> CREATE TABLE t4 (f4 int unique not null, primary
key(f4));
Query OK, 0 rows affected (0.15 sec)
```

A continuación, agregue dos registros a cada una:

```
mysql> INSERT INTO t1 VALUES(1),(2);
```

```

Query OK, 2 rows affected (0.12 sec)
Records: 2 Duplicates: 0 Warnings: 0
mysql> INSERT INTO t2 VALUES(1),(2);
Query OK, 2 rows affected (0.12 sec)
Records: 2 Duplicates: 0 Warnings: 0
mysql> INSERT INTO t3 VALUES(1),(2);
Query OK, 2 rows affected (0.12 sec)
Records: 2 Duplicates: 0 Warnings: 0
mysql> INSERT INTO t4 VALUES(1),(2);
Query OK, 2 rows affected (0.12 sec)
Records: 2 Duplicates: 0 Warnings: 0

```

A continuación, imagine que necesita combinar estas tablas. La siguiente consulta devolverá los resultados requeridos:

```

mysql> SELECT * FROM t1,t2 LEFT JOIN t3 ON (t3.f3=t1.f1)
LEFT JOIN t4 ON (t4.f4=t1.f1) WHERE t2.f2=t4.f4;
+----+----+----+----+
| f1 | f2 | f3 | f4 |
+----+----+----+----+
| 1  | 1  | 1  | 1  |
| 2  | 2  | 2  | 2  |
+----+----+----+----+
2 rows in set (0.02 sec)

```

Si utiliza la instrucción EXPLAIN para examinar el registro, observará lo siguiente:

```

mysql> EXPLAIN SELECT * FROM t1,t2 LEFT JOIN t3 ON
(t3.f3=t1.f1) LEFT JOIN t4 ON (t4.f4=t1.f1)
WHERE t2.f2=t4.f4;
+----+----+----+----+----+----+
| table | type | possible-keys | key | key-len | ref |
| rows | Extra | | | | |
+----+----+----+----+----+----+
| t1 | index | NULL | PRIMARY | 4 | NULL |
2 | Using index | | | | |
| t2 | index | PRIMARY,b,f2 | PRIMARY | 4 | NULL |
2 | Using index | | | | |
| t3 | eq_ref | PRIMARY,c,f3 | PRIMARY | 4 | t1.f1 |
1 | Using index | | | | |
| t4 | eq_ref | PRIMARY,d,f4 | PRIMARY | 4 | t1.f1 |
1 | where used; Using index | | | | |
+----+----+----+----+----+----+
4 rows in set (0.00 sec)

```

El índice se examina dos veces, una sobre t1 y otra sobre t2, lo que significa que MySQL necesita examinar todo el índice. Si analiza atentamente la consulta, observará que la combinación por la izquierda es lo que necesita t2 para leerse antes que t4. Puede evitar esta operación modificando el orden de las tablas y separando t2 de la combinación por la izquierda:

```

mysql> EXPLAIN SELECT * FROM t2,t1 LEFT JOIN t3

```



```

ON (t3.f3=t1.f1) LEFT JOIN t4 ON (t4.f4=t1.f1)
WHERE t2.f2=t4.f4;
+----+-----+-----+-----+-----+
| table | type   | possible-keys | key      | key-len | ref  |
rows | Extra |               |          |          |     |
+----+-----+-----+-----+-----+
| t1    | index  | NULL          | PRIMARY |         |     |
2 | Using index |
| t3    | eq_ref | PRIMARY,c,f3  | PRIMARY |         | t1.f1 |
1 | Using index |
| t4    | eq_ref | PRIMARY,d,f4  | PRIMARY |         | t1.f1 |
1 | Using index |
| t2    | eq_ref | PRIMARY,b,f2  | PRIMARY |         | t4.f4 |
1 | Using index |
+----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

```

¡Observe la diferencia! Según la **columna** rows, sólo es necesario leer $2 \times 1 \times 1^*$ filas (2 en total), en lugar de $2 \times 2 \times 1^* \times 1$ filas (4 en total) de la **consulta** anterior. Por supuesto, los resultados son idénticos:

```

mysql> SELECT * FROM t2,t1 LEFT JOIN t3 ON (t3.f3=t1.f1)
LEFT JOIN t4 ON (t4.f4=t1.f1) WHERE t2.f2=t4.f4;
+----+-----+-----+-----+
| f2 | f1 | f3 | f4 |
+----+-----+-----+-----+
| 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 |
+----+-----+-----+-----+
2 rows in set (0.00 sec)

```

Este ejemplo demuestra, de nuevo, la importancia de probar las consultas con EXPLAIN.

Sin una buena comprensión del funcionamiento interno de MySQL, puede que nunca se hubiera dado cuenta de cual de las dos consultas anteriores es la mas rápida, aunque tuviera la sospecha de que existía una diferencia. La instrucción EXPLAIN cuantifica nuestras suposiciones, que se desarrollan con la experiencia.

Como ayudar al optimizador de MySQL con ANALYZE

El mecanismo de MySQL que decide que clave utilizar (si es que selecciona alguna) se conoce como el optimizador de consultas. Este mecanismo examina rápidamente los índices para determinar el que conviene utilizar. Los humanos hacemos algo parecido al buscar un libro. Por ejemplo, suponga que estamos buscando un libro escrito por Zakes Mda titulado *Ways of Dying* y sabemos que sólo existen dos índices. Si en uno de ellos se ordenan los autores por orden alfabético y consta de 4.000 entradas y el otro recoge los títulos de libros y consta

de 12.000 entradas, es probable que nos decantáramos por el primero. Pero si sabemos que Zakes Mda ha escrito 200 libros pero que sólo uno de ellos se titula *Ways of Dying*, es probable que seleccionemos el segundo índice. MySQL también funciona mejor con una idea del contenido de cada índice. Podemos suministrarle este tipo de información (conocida como cardinalidad o número de valores únicos) ejecutando el siguiente comando: **ANALYZE TABLE** nombre_de_tabla.

En los ejemplos que hemos estado utilizando hasta ahora no tiene mucho sentido utilizar esta función, pero en tablas de mayor tamaño con una gran cantidad de inserciones, actualizaciones y eliminaciones, el análisis regular de la tabla puede contribuir a mejorar su rendimiento.

ANALYZE TABLE actualiza la distribución de la clave de la tabla si no está actualizada. (La ejecución de la instrucción **ANALYZE** equivale a ejecutar `myisamchk -a` o `myismachk -analyze`. En un capítulo posterior, se ampliará este tema).

ADVERTENCIA: Esta función sólo se puede utilizar con las tablas MyISAM y BDB. Además, la tabla queda bloqueada con un bloqueo de lectura durante el proceso. Por lo tanto, no es aconsejable realizar esta operación de análisis cuando la base de datos gestiona mucho tráfico.

Si desea ver la información a disposición de MySQL ejecutando el comando **SHOW INDEX:**

```
mysql> SHOW INDEX FROM customer;
+-----+-----+-----+-----+-----+-----+
+
| Table      | Non_unique | Key_name | Seq_in_index | Column_name |
| Collation | Cardinality | Sub_part | Packed      | Comment     |
+-----+-----+-----+-----+-----+-----+
+
| customer  |          0 | PRIMARY  |             1 | id          |
| A         |          8 |          | NULL        | NULL        |
| customer  |          1 | surname  |             1 | surname     |
| A         |          8 |          | NULL        | NULL        |
| customer  |          1 | surname  |             2 | initial     |
| A         |          8 |          | NULL        | NULL        |
| customer  |          1 | surname  |             3 | first-name  |
| A         |          8 |          | NULL        | NULL        |
+-----+-----+-----+-----+-----+-----+
+
4 rows in set (0.01 sec)
```

La tabla 4.5 explica el significado de las columnas devueltas por la instrucción **SHOW INDEX:**

Tabla 4.5. Significado de las columnas devueltas por SHOW INDEX.

Columna	Descripción
Table	Nombre de la tabla que se esta examinando.
Non_unique	0 o 1. 0 indica que el índice no contiene duplicados (una clave primaria o índice exclusivo) y 1 significa que puede contenerlos.
Key_name	Nombre del índice.
Seq_in_index	Orden de las columnas del índice, comenzando en 1.
Column_name	Nombre de la columna.
Collation	A O NULL. A indica que el índice se ha ordenado en orden ascendente y NULL indica que no esta ordenado.
Cardinality	Numero de valores exclusivo del índice. Esta opción se actualiza de manera especifica ejecutando <code>ANALYZE TABLE 0 myisamchk -a</code> .
Sub_part	NULL si se indexa toda la columna. De lo contrario indica el tamaño del índice, en caracteres.
Packed	Indica si el índice esta comprimido o no.
Null	YES si la columna puede contener valores NULL.
Comment	Varios comentarios.

Las operaciones de eliminación y **actualización** pueden dejar huecos en la tabla (especialmente si las tablas contienen campos TEXT, BLOB o VARCHAR). En estos casos, aumenta el trabajo del disco porque las cabezas necesitan saltar los huecos al leer.

La instrucción `OPTIMIZE TABLE` **soluciona** este problema, eliminando los huecos en los datos; para ello, une los registros fragmentados, lo que equivale a una **operación** de defragmentación aplicada a los datos de una tabla.

TRUCO: En un capítulo posterior se analizarán más detalles de la instrucción `OPTIMIZE`. Tenga en cuenta que la tabla queda **bloqueada durante el proceso**, por lo que no es aconsejable ejecutar esta instrucción en las horas de mas tráfico.

Optimización de las instrucciones SELECT y seguridad

Cuando mas complejos sean sus permisos, mayor sera la carga de trabajo que experimentaran sus consultas. Con **ello** no queremos decir que deba **escatimar**

medidas de seguridad, pero si tiene un **conjunto** de consultas de gran volumen y otro **conjunto** de bajo volumen con permisos complejos, puede que resulte útil mantener lo mas separado posible el conjunto de bajo volumen.

Evaluación del rendimiento de las funciones

La funcion `BENCHMARK()` indica cuanto tiempo necesita MySQL para realizar una tarea un numero dado de veces.

Esta funcion brinda una idea general sobre la diferencia de potencia entre dos equipos. Su sintaxis es la siguiente:

```
SELECT BENCHMARK(número_de_repeticiones,expresión)
```

Compare los resultados que obtuvo MySQL al calcular diez millones de veces la raiz cuadrada de 999 en los siguientes equipos: un ordenador con procesador Duron de 1GB con Windows 98 sin mucha carga de trabajo y un Pentium III a 850Mhz con Linux Red Hat 7 con bastante carga de trabajo. Para establecer mejor la comparacion, se ejecuto una tercera vez, en un antiguo Cyrix 200MMX con FreeBSD 4.6, sin otro proceso en marcha:

```
mysql> SELECT BENCHMARK(10000000, SQRT(999));
+-----+
| BENCHMARK(10000000, SQRT(999)) |
+-----+
|                                0 |
+-----+
1 row in set (0.66 sec)
mysql> SELECT BENCHMARK(10000000, SQRT(999));
+-----+
| BENCHMARK(10000000, SQRT(999)) |
+-----+
|                                0 |
+-----+
1 row in set (2.73 sec)
mysql> SELECT BENCHMARK(10000000, SQRT(999));
+-----+
| BENCHMARK(10000000, SQRT(999)) |
+-----+
|                                0 |
+-----+
1 row in set (13.24 sec)
```

ADVERTENCIA: La instrucción `BENCHMARK()` debe utilizarse con precaución para comparar equipos. En el caso de una base de datos activa, es necesario tener en cuenta muchos otros factores como la velocidad del disco duro, que no se ha tenido en cuenta en la prueba anterior. Su principal objetivo es ayudar a optimizar las funciones.

Optimización de actualizaciones y eliminaciones e inserciones

Una operación de **actualización** es prácticamente igual a una operación de **selección** con la diferencia de que se realiza una operación de escritura al final. Por ejemplo, para fines de optimización, el siguiente código:

```
UPDATE nombre_de_campo FROM nombre-de-tabla WHERE condicion
```

es igual al este otro:

```
SELECT nombre_de_campo FROM nombre-de-tabla WHERE condicion
```

Puede optimizar una instrucción UPDATE de la misma forma que haríamos con la instrucción SELECT equivalente. Así mismo, tenga en cuenta que cuanto menor sea el número de índices y el número de datos, más rápida resultará la operación. Procure no utilizar índices superfluos o que el tamaño de los campos o de los índices resulte mayor de lo necesario.

La velocidad de la instrucción DELETE depende del número de índices. Al eliminar registros, resulta necesario suprimir cada uno de ellos de todos los índices asociados así como del archivo de datos principal.

Por esta razón, la instrucción TRUNCATE nombre_de_tabla resulta más rápida que DELETE nombre_de_tabla, ya que la tabla entera se elimina de una vez, sin necesidad de tener que suprimir cada índice y registro de datos de manera individual.

El mejor método para insertar datos consiste en utilizar LOAD DATA en lugar de INSERT, ya que puede resultar 20 veces más rápido.

Puede acelerar este proceso deshabilitando las claves durante el intervalo dedicado a agregar datos. MySQL sólo tendrá que concentrarse en agregar los datos despreocupándose de agregar los archivos de índice a la vez. La operación de agregar los datos resultará mucho más rápida y si los índices se generan de manera separada, el proceso resultará además mucho más óptimo. Puede utilizar el siguiente procedimiento:

```
ALTER TABLE nombre_tbl DISABLE KEYS;  
LOAD DATA INFILE nombre_de_archivo INTO TABLE nombre-de-tabla  
ALTER TABLE nombre_tbl ENABLE KEYS;
```

Sin embargo, no siempre se puede realizar la inserción desde un archivo de texto.

Pero si puede agrupar sus inserciones, las listas de varios valores se agregan mucho más rápidamente que las instrucciones separadas. Por ejemplo, la siguiente consulta:

```
INSERT INTO nombre-de-tabla VALUES(registro1),(registro2)  
(registro3);
```

es mucho mas rapida que la siguiente alternativa:

```
INSERT INTO nombre-de-tabla VALUES (registro1);
INSERT INTO nombre-de-tabla VALUES (registro2);

INSERT INTO nombre-de-tabla VALUES (registron);
```

La razon es que los indices sólo se vacian una vez por *cada* instruccion INSERT. Si necesita realizar varias instrucciones INSERT, puede utilizar bloqueos para lograr el mismo resultado. Utilice instrucciones como las siguientes para tablas no transaccionales:

```
LOCK TABLES nombre-de-tabla WRITE;
INSERT INTO nombre-de-tabla VALUES (registro1), (registro2)
(registro3);
INSERT INTO nombre-de-tabla VALUES (registro4), (registro5)
(registro6);
UNLOCK TABLES;
```

Tenga en cuenta que nadie podra leer las tablas mientras estas instrucciones esten en progreso.

Para realizar la misma operacion con tablas transaccionales, utilice las siguientes instrucciones:

```
BEGIN;
INSERT INTO nombre-de-tabla VALUES (registro1), (registro2)
(registro3);
INSERT INTO nombre-de-tabla VALUES (registro4), (registro5)
(registro6);
COMMIT;
```

Las cosas se complican un poco al agregar registros desde diferentes subprocesos. Imagine un caso en el que el primer subproceso agrega 10.000 registros y el segundo un solo registro. Si se utiliza la funcion de bloqueo, se mejorara la velocidad general de la operacion pero el segundo subproceso sólo se completara cuando termine el primero. Si no se utiliza el bloqueo, el segundo subproceso se completara de forma mucho mas rapida, pero la velocidad de la operacion completa resultara mas lenta. La importancia del segundo subproceso con respecto al primero determinara el método que debemos seleccionar.

Sin embargo, puede ocurrir que su aplicacion necesite realizar una gran cantidad de inserciones no relacionadas de manera continua. Si esta utilizando un bloqueo de nivel de fila (como el que se puede realizar en tablas InnoDB), puede que descubra que las hordas de usuarios que consultan sus tablas necesitan esperar una cantidad de tiempo inusualmente larga debido a unas cuantas operaciones de insercion. No desespere, ya que existen formas de minimizar este efecto.

La primera consiste en utilizar la instruccion INSERT LOW PRIORITY. Esta instruccion reduce la alta prioridad habitual asignada a los comportamientos de insercion y obliga a hacerlos esperar hasta que no existan mas consultas de lectura en la cola. El problema, sin embargo, es que si su base de datos tiene

mucho trafico, puede que el cliente que **realice** la operacion de insercion con prioridad baja necesite esperar mucho tiempo para realizar la operacion (si es que surge un hueco).

Otra alternativa es la instrucción `INSERT DELAYED`. El cliente queda liberado inmediatamente y la insercion se coloca en cola (con el resto de instrucciones `INSERT DELAYED` esperando a que **la cola finalice**). La desventaja de este método es que no se pasa informacion significativa al cliente (como el valor `auto_increment`) dado que la insercion no se ha procesado cuando el cliente se libera. ¡Pero hay cosas por las que no merece la pena esperar! Asi mismo, tenga en cuenta que existe la posibilidad de que todas las inserciones de la cola se pierdan si tiene lugar una catastrofe como un fallo en la corriente electrica.

ADVERTENCIA: El uso de las instrucciones `INSERT LOW PRIORITY` e `INSERT DELAYED` no permite saber el momento en el que se realizarán las inserciones, si es que se realizan. Por ello, aconsejable utilizarlas con precaución.

Resumen

El reducido uso de los indices es probablemente la causa mas importante que **explique** los problemas de rendimiento. Un **índice** es un pequeño archivo ordenado que apunta al archivo de datos principal. La busqueda de un registro **concreto** resultara mas rapida porque solo es necesario **realizarla sobre un pequeño archivo de índice**.

Los indices pueden ser de clave **primaria** (un **índice** unico que no puede **contener** valores nulos), un **índice exclusivo**, un **índice ordinario** (que puede contener duplicados) o un **índice** de texto completo. Los indices de texto completo **permiten** un alto nivel de sofisticacion en la busqueda de campos de texto para **determinadas** combinaciones de **palabras clave**.

Los campos de incremento **automático** se asocian con la clave **primaria** y **permiten** que MySQL se encargue de la secuenciacion del campo. Si se inserta un registro, MySQL agregara una unidad **al** valor anterior incrementado automaticamente y lo utilizara como valor para el campo de incremento **automático** insertado.

La **instrucción EXPLAIN** devuelve informacion **útil** sobre la forma en que MySQL utiliza **los** indices en una consulta concreta. Puede utilizarla para **determinar** si MySQL esta utilizando **los** indices creados y, si la consulta no resulta optima, obtener informacion relativa a **los** campos **sobre los** que crear indices o **sobre** como cambiar la consulta para mejorarla.

5

Programacion con MySQL

En este capitulo no vamos a enseiarse a programar. Existen muchos libros que **intentan** combinar la enseianza de MySQL y de un lenguaje de programacion, y el resultado final es un trabajo **incompleto** en ambos aspectos. En este libro asumimos que el lector es un programador competente, que su interes es aprender o centrarse en el **papel** de administrador de bases de datos (DBA) y que no esta interesado en la programacion.

El verdadero potencial de una base de datos se aprecia cuando se integra en un sistema de **información**, con aplicaciones completamente funcionales que incorporen su propio valor **al** sistema. Un sitio Web de noticias, por ejemplo, necesita herramientas para agregar y ordenar los articulos de noticias, para visualizarlos en el sitio Web y para realizar el seguimiento de las historias de mayor eco. A la mayor parte de los periodistas no les atrae mucho aprender SQL (lenguaje de **consulta** estructurado).

Sin embargo, necesitan una interfaz bien diseiada para comunicarse con la base de datos. Esta interfaz **podría** ser una pagina Web con un formulario HTML (lenguaje de **marcado** de hipertexto), con un boton de envio que invoque una secuencia de comandos para ejecutar una **instrucción** INSERT. La interfaz tambien **podría** adoptar la **forma** de un sistema de suministro de noticias que tome los articulos de un sistema **QuarkXPress** y que los agregue automaticamente a la base de datos.

Otro ejemplo de sistema de informacion **podría** consistir en el uso de una aplicacion para asesores financieros, donde el **servidor** se alimenta de las ultimas cotizaciones de valores y monedas, para que **los asesores** puedan acceder y analizar la informacion con el fin de controlar la tendencia de **los mercados**.

Las posibilidades para **los sistemas** de informacion son infinitas. Estos escenarios se caracterizan por incluir una aplicacion desarrollada para agregar niveles adicionales de lógica que MySQL no puede suministrar. En teoria, puede utilizar cualquier lenguaje de programacion para desarrollar aplicaciones. Los lenguajes de uso mas habitual son Java, C, PHP, Perl, C++, Visual Basic, Python y Tcl, **los cuales** disponen, en la **mayoría los casos**, de interfaces de programacion de aplicaciones (API) para interactuar con MySQL. En **los apendices** de este libro se **recogen los API** de la **mayoría** de estos lenguajes de programacion.

Todos **los ejemplos** de este capitulo estan escritos en PHP, no porque deba conocer este lenguaje sino simplemente porque es probable lo haya utilizado, porque su **sintaxis** resulta familiar a todas aquellas personas que tengan experiencia con lenguajes del tipo C (como C, Perl o C++) y porque su sencillez facilita que otros programadores puedan seguir **los ejemplos**. Lo importante son **los principios** de programacion, no la sintaxis. En este capitulo, el codigo viene acompaiiado de amplios comentarios lo que le permitira seguirlo independientemente del lenguaje que utilice o su nivel de conocimientos.

En este capitulo se abordan **los siguientes temas**:

- Uso de conexiones permanentes
- Como lograr que nuestro codigo resulte portable y sencillo de mantener
- Valoracion de la carga de trabajo de la base de datos frente a la de la aplicacion
- Exploración del proceso de desarrollo de la aplicacion

Uso de buenas tecnicas de programacion de bases de datos

En las siguientes secciones se presentan algunas de las tecnicas habituales utilizadas por **los programadores** de bases de datos para lograr que sus aplicaciones **resulten sólidas** (que no **fallan** facilmente), portables (faciles de trasladar a otros **entornos** y plataformas) y faciles de mantener. Las conexiones permanentes son utiles si la aplicacion realiza un numero alto de peticiones de **conexión** que procedan de la misma fuente en un corto **periodo** de tiempo. Los programadores sin experiencia, con prisa o simplemente vagos **tienden** a crear codigo que **dificulta** la tarea de **los siguientes programadores** (y a **menudo** la de ellos mismos) al pasar por alto aspectos relacionados con la portabilidad y el mantenimiento, car-

gando demasiada el trabajo **sobre** la aplicacion en lugar de **sobre** la base de datos. Puede evitar muchos problemas en fases posteriores si planea un poco el trabajo **al principio**.

Uso de conexiones permanentes

MySQL se ha caracterizado por su rapidez de conexión, si la comparamos con otras bases de datos. Sin embargo, la **conexión** a una base de datos sigue siendo una tarea **bastante** pesada y, si necesita realizar un gran numero de conexiones en un corto **periodo** de tiempo (como **al** establecer una **conexión** desde un servidor Web), es aconsejable **simplificar al** maximo la **operación**. Las conexiones **permanentes** mantienen la comunicacion abierta una vez completada la secuencia de comandos. Las siguientes peticiones utilizan la **conexión** existente con el ahorro de carga resultante. En PHP, puede utilizar la funcion `mysql_pconnect()` para crear una **conexión** permanente:

```
mysql_pconnect($host,$user,$pass);  
    // la funcion mysql_pconnect crea una conexión permanente a una  
    // base de datos mysql, para lo cual toma los parametros de  
    //anfitrión, usuario y contraseiia
```

No siempre es necesario mantener las conexiones abiertas durante mucho tiempo. En la mayor parte de los **casos**, el servidor Web se encarga de limpiar las conexiones. Sin embargo, puedo hablarles de un caso en el que un servidor Web presentaba problemas porque no limpiaba las conexiones tras iniciarse. El servidor Web estaba configurado para permitir 400 instancias y MySQL podia admitir 750 conexiones. Debido a este comportamiento erroneo, el servidor Web **duplicaba** el numero de conexiones que realizaba **al** servidor de la base de datos, es decir 800. De repente el servidor de la base de datos se quedaba sin conexiones disponibles. Puede minimizar el riesgo de mantener abiertas las conexiones permanentes durante demasiado tiempo reduciendo el valor de la variable `wait_timeout` de MySQL (o `interactive_timeout` en **función** del tipo de **conexión**), que determina la **cantidad de tiempo** que MySQL **permite** que una **conexión** se mantenga inactiva antes de cerrarse. (En un capitulo posterior se explica como configurar estas variables.) Su valor predeterminado es de 28.800 segundos (8 horas). En el caso anterior, se redujo a 600 segundos para impedir que el problema volviera a surgir. ¡A partir de ahí la preocupacion se reducía **al** servidor Web!

Como lograr codigo portable y sencillo de mantener

Basta con aplicar unos sencillos pasos para mejorar enormemente la flexibilidad del codigo. Entre ellos, se incluye el mantenimiento de los detalles de co-

nexión aparte y dentro de una única ubicación así como la **construcción** de consultas de base de datos de **manera** flexible para que los cambios futuros que se apliquen a la estructura de la base de datos no afecten a la aplicación.

La conexión

La mayor **parte** de los lenguajes de programación facilita la tarea de establecer la **conexión** a una base de datos a través de funciones nativas. Por ejemplo, PHP cuenta con un **conjunto** de funciones para su uso con **MySQL**, como `mysql_connect()`, `mysql_query()`, etc.

Al **programar** una pequeña **aplicación** con una **conexión** a la base de datos, con clases nativas, resulta fácil utilizar algo sencillo para establecer la **conexión** a MySQL (vease el listado 5.1).

Listado 5.1. totally_importable.php

```
$db = mysql_pconnect("dbhostname.co.za", "db_app", "g00r002b");
// la función mysql_pconnect crea una conexión permanente a una
// base de datos mysql para lo cual toma los parámetros de
//anfitrión, usuario y contraseña
// donde 'dbhostname' es el anfitrión, 'db_app' el usuario y
// 'g00r002b' la contraseña
if (!$db) {
    echo "There was a problem connecting to the database.";
    exit;
}
// verificación básica de errores - si la conexión no resulta
//satisfactoria, muestre
// un mensaje de error y salga de la secuencia de comandos
```

Muchos de los ejemplos con los que se encontrara utilizan este **método** porque resulta sencillo de **entender** y funciona correctamente en aplicaciones pequeñas. Sin embargo, cuando se trata de una base de datos de una aplicación más grande, es aconsejable que resulte lo más portable, sencilla de mantener y segura posible. Imagine que tiene 10 secuencias de comando que se conectan a la base de datos. Si las 10 secuencias de comando se conectan de la misma **forma** y un día necesita trasladar la base de datos a un nuevo servidor o desea cambiar su contraseña, necesitará hacerlo en todas las secuencias de comandos.

Ahora imagínese que en lugar de 10 fueran 100.

En una ocasión heredé una **situación** como esta y, ante la posibilidad de que la contraseña pudiera verse comprometida (que **además** estaba situada en cientos de ubicaciones por lo que resultaba muy sencilla de encontrar), me **tocó** la agradable tarea de realizar todos los cambios. La mejor **solución** consiste en crear la aplicación de **manera correcta** desde el principio.

Coloque los detalles de la **conexión** de la base de datos en una ubicación aparte. Estos se incluirán en las secuencias de comandos que establecen la conexión a la base de datos. Posteriormente, cuando necesite modificar los detalles, sólo tendrá que hacerlo en un lugar (y estará seguro de no olvidar nada). La

operación de cambiar la contraseña en cientos de lugares implica el riesgo de olvidar uno y descubrirlo cuando falle la funcionalidad.

Las soluciones que se recogieron en el listado 5.2 y 5.3 son mejores.

Listado 5.2. db.inc

```
$host = "dbhostname.co.za";
$user = "db_app";
$pass = "g00r002b";
```

Listado 5.3. not_too_portable.php

```
require_once "$include_path/db.inc";
// incluye el archivo que contiene los detalles de la conexión,
//db.inc
// ubicado en la ruta: $include_path, que debería ser una
//ubicación segura
$db = mysql_pconnect($host, $user, $pass);
    // la función mysql_pconnect crea una conexión permanente a una
    // base de datos mysql, para lo cual toma los parámetros de
//anfitrión, usuario y contraseña
if (!$db) {
    echo "There was a problem connecting to the database.";
    exit;
}
// verificación básica de errores - si la conexión no resulta
//satisfactoria, muestre
// un mensaje de error y salga de la secuencia de comandos
```

En este ejemplo, la contraseña, el nombre del anfitrión y el nombre de usuario se almacenan en un archivo, por lo que solo se necesita modificarlos en un único lugar (db.inc).

ADVERTENCIA: Si está creando una aplicación Web, asegúrese de que el archivo `db.inc` no se incluye dentro del **árbol** de la Web. (Su servidor Web no debería **servir archivos .inc**, pero en este **caso**, es conveniente mantener la **información** sensible en un lugar lo más alejado posible.)

Se incorpora otra mejora que se concreta en un nivel ligeramente superior de **abstracción**. En los listados 5.2 y 5.3, imagine que la dirección decide realizar la migración a otro DBMS. Puede ocurrir que MySQL no sea un sistema ideal para una situación dada o, como ocurre a menudo, que se tomen decisiones extrañas, como una de la que fui testigo en la que la dirección quería gastarse una enorme cantidad de dinero en una segunda base de datos cuando bastaba con configurar MySQL correctamente. Afortunadamente, logré convencerles de lo contrario porque, nuevamente, el código no resultaba muy portable.

Para lograr un código lo más portable posible, los detalles del DBMS deberían poder modificarse en un único lugar. Esta operación implica la creación de una segunda función que se encargue de procesar los detalles de la conexión, como se muestra en el listado 5.4 y 5.5.

La función `db_pconnect()` se coloca en el archivo `db.inc` y se utiliza en el archivo `portable.php` para realizar la conexión a la base de datos en lugar de `mysql_connect()`.

Listado 5.4. `db.inc`

```
// esta función establece la conexión a la base de datos y
//devuelve la función de conexión
db_pconnect() {
    $host = "dbhostname.co.za";
    $user = "db_app";
    $pass = "g00r002b";
    return mysql_pconnect($host, $user, $pass);
}
```

Listado 5.5. `portable.php`

```
require_once "$include_path/db.inc";
// incluye el archivo que contiene los detalles de la conexión,
//db.inc
// ubicado en la ruta: $include_path, que debería ser una //
ubicación segura
$db = db_pconnect($host, $user, $pass);
if (!$db) {
    echo "There was a problem connecting to the database.";
    exit;
}
// verificación básica de errores - si la conexión no resulta
//satisfactoria, muestre
// un mensaje de error y salga de la secuencia de comandos
```

A partir de ahora si sustituye MySQL por otra base de datos, bastará con reemplazar `mysql_pconnect()` por la función oportuna, como `odbc_pconnect()`.

NOTA: Nuestra intención en este libro no es imponerle un estilo de programación. Cada lenguaje tiene sus puntos fuertes y sus puntos débiles. Java es un lenguaje mucho más orientado a objetos que PHP, por ejemplo, de manera que los ejemplos anteriores no funcionarían bien si se traducen &-rectamente a Java. Lo importante es el principio de lograr que las aplicaciones resulten lo más sencillas de mantener (almacenando la información de conexión en una ubicación) y lo más portables (evitando el uso de código específico de la base de datos) posible.

Consultas de base de datos

Puede utilizar **atajos** como `SELECT *` al consultar MySQL de manera directa. Sin embargo, debería evitar este tipo de métodos en sus aplicaciones ya que merman la portabilidad. Imagine una **situación** en la que tengamos una tabla de miembros con tres campos: `id`, `first name` y `surname`. El código de programación podría parecerse al del listado—5.6.

Listado 5.6. `totally_inflexible_select.php`

```
// supongamos que la conexión a $db ya se ha establecido
$result = mysql_query("SELECT * FROM entrants",$db);
// ejecute la consulta en la conexión activa
while ($row = mysql_fetch_array($result,MYSQL_NUM)) {
    // cuando se invoca mysql_fetch_array con MYSQL_NUM
    // como parametro, se devuelve una matriz numericamente
    // indexada, en la que cada elemento se corresponde con un
//campo
    // recuperado de una fila devuelta
    $id = $row[0];
    // Como el primer campo de la base de datos es id,
    // se devuelve como el primer elemento de la matriz,
    // cuyo primer valor es, obviamente, 0
    $first-name = $row[1];
    $surname = $row[2];
    // .. realice alguna operación con los detalles
}
```

Esta consulta funcionaba **al principio**. Pero suponga que alguien (siempre un tercero) realiza un **cambio** en la estructura de la base de datos e introduce un nuevo campo entre `first_name` y `surname`, **llamado** `initial`. Su código no necesita la inicial y, **de repente, deja** de funcionar, ya que la inicial es el tercer elemento de la matriz (o `$row[2]`) y esta almacenado como `$surname`. Como resultado, nunca se accede **al** campo `surname`.

La secuencia de comandos `totally_inflexible_select.php` **presenta** una serie de problemas que **necesitan resolverse**. Además de no funcionar si se modifica la estructura de base de datos, la función utilizada para recuperar campos devuelve una matriz **numérica** en lugar de una matriz asociativa. Como consecuencia, su código resultará **menos** legible, ya que cualquier persona sin conocimiento **sobre** la estructura de la base de datos no sabrá que se **recupera** de la base de datos. En PHP, puede corregir este problema utilizando una función que devuelva una matriz asociativa, como se ilustra en el **listado 5.7**

Listado 5.7. `inflexible_select.php`

```
// supongamos que la conexión a $db ya se ha establecido
$result = mysql_query("SELECT * FROM entrants",$db);
while ($row = mysql_fetch_array($result,MYSQL_ASSOC)) {
    // cuando se invoca mysql_fetch_array con MYSQL_NUM
```

```

        // como parametro, se devuelve una matriz asociativa,
        // en la que cada clave de la matriz es el nombre de un
//campo
        // devuelto de la fila
        $id = $row["id"];
        $first-name = $row["first_name"];
        $surname = $row["surname"];
        // .. realice alguna operacion con los detalles

```

Este código resulta mejor ya que seguirá funcionando, incluso tras agregar el campo `initial` a la tabla de la base de datos. Es capaz de procesar varios cambios en la estructura de la base de datos y resulta más legible. Un programador sin conocimientos sobre la estructura de la base de datos, sabrá que campos se están devolviendo. Pero todavía podemos incorporar otra optimización. Al ejecutar una consulta `SELECT *`, le estamos pidiendo a MySQL que devuelva todos los campos de la tabla. Como nuestro código sólo necesita utilizar tres campos, ¿por qué malgastar recursos para devolverlos todos, con una carga adicional de operaciones de entrada y salida del disco que implica y el mayor tráfico sobre la red?

Basta con especificar los campos que queramos devolver. De esta forma, no sólo lograremos reducir el uso de recursos, sino que además mejoraremos la legibilidad del código. De hecho, en algunos casos la devolución de la matriz asociativa absorbe más recursos que la operación de recuperar una matriz numérica. En este caso, podemos mantener la legibilidad de código, incluso al recuperar una matriz numérica, si especificamos los campos como se ilustra en el listado 5.8.

Listado 5.8. flexible_select.php

```

// supongamos que la conexión a $db ya se ha establecido
$result = mysql_query("SELECT id,first_name,surname FROM
entrants",$db);
while ($row = mysql_fetch_array($result,MYSQL_NUM)) {
    // cuando se invoca mysql_fetch_array con MYSQL_NUM
    // como parametro, se devuelve una matriz numericamente
    // indexada, en la que cada elemento se corresponde con un
//campo
    // recuperado de una fila devuelta

    $id = $row[0];
    $first-name = $row[1];
    $surname = $row[2];
    // .. realice alguna operacion con los detalles
}

```

Este mismo principio se aplica a las consultas `INSERT`. No utilice nunca una consulta `INSERT` sin una lista de campos dentro de una aplicación. Por ejemplo, tomando la tabla original con tres campos (`id`, `first_name` y `surname`), podríamos utilizar código como el que se muestra en el listado 5.9.

Listado 5.9. inflexible_insert.php

```
// supongamos que la conexión a $db ya se ha establecido
$result = mysql_query("INSERT INTO entrants?
VALUES ('$id', '$first_name', '$surname')", $db);
```

Si la estructura de la tabla cambia, el código dejara de funcionar de nuevo. Si se añade otro campo, `initial`, el número de campos insertados no coincidirá con los campos de la tabla y la consulta fallará.

La forma de resolver este problema consiste en especificar los campos de la base de datos que se están insertando, como se muestra en el listado 5.10.

Listado 5.10. flexible_insert.php

```
// supongamos que la conexión a $db ya se ha establecido
$result = mysql_query("INSERT INTO entrants(id, first_name,?
surname) VALUES ('$id', '$first_name', '$surname')", $db);
```

Esta solución presenta también la ventaja de resultar más legible, especialmente si consideramos que los valores de campo no siempre coinciden con los nombres de campo como en este ejemplo.

¿Cuánto trabajo debería realizar el servidor de la base de datos?

Uno de los debates constantes entre los desarrolladores es como debería repartirse la carga de trabajo entre el servidor de la base de datos y la aplicación.

En un primer momento, los desarrolladores de MySQL se mostraron muy a favor de delegar todo el peso en la aplicación, en parte porque no incorporaba algunas funciones, como procedimientos almacenados y desencadenadores, y en parte por una cuestión de principios. Esta actitud les convirtió en objeto de críticas y la ausencia de estas funciones llevó a la gente a considerar a MySQL como una base de datos poco seria (una etiqueta de la que sólo ahora, con la versión 4, está comenzando a superar).

En general, la base de datos debería hacer todo el trabajo posible. Los ejemplos siguientes producen el mismo resultado de formas diferentes. El listado 5.11 devuelve todos los datos, sin ordenar, y utiliza la función `sort()` de PHP para ordenarlos. En su lugar el listado 5.12 utiliza la cláusula `ORDER BY` para ordenar los datos.

Listado 5.11. work_the_script.php

```
// supongamos que la conexión a $db ya se ha establecido
$result = mysql_query("SELECT surname FROM entrants", $db);
while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
    // cuando se invoca mysql_fetch_array con MYSQL_NUM
    // como parametro, se devuelve una matriz asociativa,
```



```

        // en la que cada clave de la matriz es el nombre de un
//campo
        // devuelto de la fila
        $surname[] = $row["surname"];
        // agregue el apellido como el siguiente elemento de la
//matriz
        // de apellido (y cree la matriz si no se ha creado todavia)
    }
    sort($surname)
    // la funcion sort() ordena la matriz
    // continue con el procesamiento de los datos ordenados

```

Listado 5.12. work_the_db.php

```

// supongamos que la conexión a $db ya se ha establecido
$result = mysql_query("SELECT surname FROM entrants ORDER BY
surname", $db);
while ($row = mysql_fetch_array($result,MYSQL_ASSOC)) {
    // cuando se invoca mysql_fetch_array con MYSQL_NUM
    // como parametro, se devuelve una matriz asociativa,
    // en la que cada clave de la matriz es el nombre de un campo
    // devuelto de la fila
    $surname[] = $row["surname"];
    // agregue el apellido como el siguiente elemento de la
//matriz
    // de apellido (y cree la matriz si no se ha creado todavia)
}
// continue con el procesamiento de los datos ordenados

```

El listado 5.12 resulta mucho mas lógico. MySQL podría (o debería) llevar asignado un **índice** sobre el campo surname si se tratara de una operacion comun y la operacion de leer los datos ordenados, a partir de un **índice**, resultaria mucho mas rapida que hacerlo en **formato** desordenado y, a **continuación**, usar la aplicacion para ordenarlos.

De hecho, es posible que la lectura de los datos ordenados desde la base de datos resulte mas rapida que lectura de los datos desordenados (incluso antes de tener en cuenta la funcion sort()) ya que es probable que los datos ordenados solo necesiten leerse desde el **índice** y no desde el **archivo** de datos.

Existen excepciones (como aquellas situaciones en las que no se pueda utilizar un **índice** y el servidor de la base de datos cree el principal **cuello** de botella), pero en la inmensa **mayoría** de los casos, la tecnica que se muestra en el listado 5.12 resultara muy superior.

Un ejemplo parecido, aunque mas extremo (pero comun) es aquel en el que la aplicacion realiza el trabajo de la **cláusula** WHERE como ilustra el listado 5.13.

Listado 5.13. work_the_script2.php

```

// supongamos que la conexión a $db ya se ha establecido
$result = mysql_query("SELECT surname FROM entrants", $db);
while ($row = mysql_fetch_array($result,MYSQL_ASSOC)) {

```

```

// cuando se invoca mysql-fetch-array con MYSQL_NUM
// como parametro, se devuelve una matriz asociativa,
// en la que cada clave de la matriz es el nombre de un campo
// devuelto de la fila
if ($row["surname"] == 'Johnson') {
    $johnson[] = $row["surname"];
    // agregue el apellido como siguiente elemento a la matriz
    // johnson (y cree la matriz si no se ha creado todavia)
}
elseif ($row["surname"] == 'Makeba') {
    $makeba[] = $row["surname"];
    // agregue el apellido como siguiente elemento a la matriz
    // makeba (y cree la matriz si no se ha creado todavia)
}
}
// continue con el procesamiento de las matrices makeba y
//johnson

```

Es mejor utilizar la cláusula WHERE, como se ilustra en el listado 5.14 y no perder el tiempo recuperando todos los registros extra no deseados.

Listado 5.14. work_the_db2.php

```

// supongamos que la conexión a $db ya se ha establecido
$result = mysql_query("SELECT surname FROM?
entrants WHERE surname = 'Makeba' OR surname='Johnson'", $db);
while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
    // cuando se invoca mysql-fetch-array con MYSQL_NUM
    // como parametro, se devuelve una matriz asociativa,
    // en la que cada clave de la matriz es el nombre de un campo
    // devuelto de la fila
    if ($row["surname"] == 'Johnson') {
        $johnson[] = $row["surname"];
        // agregue el apellido como siguiente elemento a la matriz
        // johnson (y cree la matriz si no se ha creado todavia)
    }
    elseif ($row["surname"] == 'Makeba') {
        $makeba[] = $row["surname"];
        // agregue el apellido como siguiente elemento a la matriz
        // makeba (y cree la matriz si no se ha creado todavia)
    }
}
// continue con el procesamiento de los datos ordenados

```

Puede escribir estos fragmentos de código de manera más elegante si está procesando muchos nombres, pero la cuestión es que el listado 5.14 resulta mucho más eficiente porque MySQL realiza el trabajo, con lo que se limita el número de resultados recibidos y se reducen los recursos utilizados.

El listado 5.15 muestra una solución que suele implementar la gente que trabaja con otras bases de datos. Como la versión 4.0 de MySQL no implementa completamente las subselecciones (aunque la situación variara en la versión 4.1), se

asume que no existe mas remedio que utilizar la aplicacion. Por ejemplo, suponemos una **situación** con dos tablas de clientes en la que deseamos determinar que clientes de una tabla no aparecen en la otra.

En este **caso** la siguiente consulta **ANSI** estandar no funcionara en **MySQL**:

```
SELECT first_name,surname FROM entrants WHERE code NOT IN
(SELECT code FROM referred-entrants);
```

Por esta razon, el **tipo** de codigo que se muestra en el **listado 5.15** se suele implementar con demasiada frecuencia.

Listado 5.15. work_the_script3.php

```
// supongamos que la conexión a $db ya se ha establecido
$result = mysql_query("SELECT code FROM entrants",$db);
$scodelist = "";
    // inicialice al lista de codigos
while ($row = mysql_fetch_array($result,MYSQL_ASSOC)) {
    // cuando se invoca mysql_fetch_array con MYSQL_NUM
    // como parametro, se devuelve una matriz asociativa,
    // en la que cada clave de la matriz es el nombre de un campo
    // devuelto de la fila
    Scodelist .= $row["code"].",";
    // agregue el codigo, seguido de una coma hasta la variable
// Scodelist
}
Scodelist = substr(Scodelist, 0, -1);
    // elimina la ultima coma, lo que da como resultado una lista
    // como "1,3,4,8,12";
$result = mysql_query("SELECT first_name,surname FROM?
referred-entrants WHERE code NOT IN($scodelist)", $db);
while ($row = mysql_fetch_array($result,MYSQL_ASSOC)) {
    // procese los detalles de las entradas
}
}
```

El **listado 5.15** funcionará, pero, de nuevo, vuelve a delegar demasiada carga en la aplicacion. En su lugar, con un poco de reflexion, **MySQL** podría realizar una consulta para devolver los resultados, como demuestra el **listado 5.16**.

Listado 5.16. work_the_db3.php

```
// supongamos que la conexión a $db ya se ha establecido
$result = mysql_query("SELECT entrants.first_name,?
entrants.surname FROM entrants LEFT JOIN referred-entrants?
ON entrants.code = referred-entrants-code WHERE?
referred_entrants.code IS NULL",$db);
while ($row = mysql_fetch_array($result,MYSQL_ASSOC)) {
    // procese los detalles de las entradas
}
}
```

Cuando se implementen las subselecciones (en la version 4.1, segun el programa actual) el codigo resultante, ilustrado en el **listado 5.17**, resultara incluso mas sencillo:

Listado 5.17. work_the_db3_2.php

```
// supongamos que la conexión a $db ya se ha establecido
$result = mysql_query("SELECT first-name,surname FROM?
entrants WHERE code NOT IN (SELECT code FROM?
referred-entrants", $db);
while ($row = mysql_fetch_array($result,MYSQL_ASSOC)) {
    // procese los detalles de las entradas
}
```

Las fases del desarrollo de aplicaciones

En este capítulo se asume que el lector ya sabe programar o que puede acudir a otro manual para aprender. Muchos programadores novatos, especialmente aquellos sin una **formación académica**, suelen olvidarse de dar un **paso** atrás y examinar el proyecto de desarrollo dentro de su **contexto** y **planear** situaciones futuras.

Son muchos los programadores dedicados a extinguir fuegos, a reinventar la rueda y, en general, a aprovechar muy poco su tiempo, culpando de ello a los coordinadores de proyectos, a los usuarios y a **todo** el mundo de su **entorno**. Obviamente, puede que estos no esten libres de culpa, **pero** como este libro va dirigido a los desarrolladores de MySQL, se incluyen una serie de sugerencias para ayudarles a mejorar la **forma** de dirigir sus proyectos. Los desarrolladores Web en **concreto**, que suelen llegar a MySQL y al aprendizaje de un lenguaje de programación a través de HTML y JavaScript, no suelen ser conscientes de la complejidad que entrañan los proyectos de mayor tamaño y a **menudo** quedan atrapados cuando los proyectos crecen.

En las siguientes secciones se **expone** brevemente los pasos implicados en el desarrollo de aplicaciones. No se trata de un **conjunto** estricto de pasos, **sino** de varios marcos de trabajo posibles. Cualquier estructura de desarrollo de aplicaciones debe permitir cierto grado de flexibilidad que viene dado por los recursos disponibles y las condiciones específicas del proyecto. **Existen** muchas metodologías buenas para el desarrollo de aplicaciones, y debería utilizar aquella que se adapte a sus necesidades. Ahora bien, los principios generales son los mismos.

Fase 1: análisis de las necesidades

El **análisis** de los necesidades de un proyecto es un **paso** obvio en el desarrollo de una aplicación. Los expertos **repiten** esta regla una y otra vez **al** deplorar el pésimo estado del desarrollo de software.

Sin embargo, se suelen escuchar excusas como "no sabíamos que quería eso" o "nunca se nos dijo **al** principio" para justificar el pobre resultado final de los proyectos. La **primera** fase de un proyecto, y **quizás** la más importante, consiste en determinar las necesidades.

Determinación de las necesidades del usuario

La mayor parte de las peticiones que realizan los usuarios **resultan** triviales, para su propia sorpresa. He conocido a usuarios, que tras trabajar con desarrolladores incompetentes o vagos, temían pedir algo más complicado que un campo adicional de una tabla. En la mayor parte de **los casos**, prácticamente **todo** es posible siempre y cuando se solicite por adelantado. La principal dificultad no es satisfacer las necesidades del usuario, si no hacerlo una vez desarrollado el marco de trabajo **inicial**. Los usuarios no siempre saben lo que quieren. **Necesitan** que les ayudemos a formalizar sus necesidades. Pero, como este libro va dirigido a los desarrolladores, no a los responsables de **tomar** decisiones de **negocio**, depositaremos toda la carga **sobre** aquellos. Asegúrese, antes de nada, de que las necesidades de usuarios han quedado claras, **tanto** para el equipo de desarrollo como para **los propios** usuarios. El equipo del proyecto necesita **realizar** los siguientes pasos para determinar las necesidades de **los usuarios**:

- El equipo debe ayudar a **los usuarios** a que determinen sus necesidades. El equipo debe guiar a **los usuarios**, **explicándoles** por que determinadas **sugerencias** no **resultan** prácticas o proponiéndoles alternativas mejores. El **equipo** debe utilizar su experiencia. Debe exponer necesidades no mencionadas para poder documentarlas. Lo que resulta obvio para el usuario puede que no lo sea para **los desarrolladores** y puede que las necesidades importantes se **pasen** por alto. Las necesidades **deben** ser lo más completas posibles. Por ejemplo, un usuario puede solicitar un sistema para "realizar **reservas**". Sin embargo, esta solicitud no es adecuada ya que no se aclaran **los campos** necesarios para llevar a **cabo** la reserva ni **los procesos** subyacentes.
- Tras comprender las necesidades del usuario, el equipo debe ponerlas por escrito y presentárselas de nuevo a **los usuarios** y a **los propietarios** del proyecto (los que pagan por su desarrollo) para su **confirmación**. **Los usuarios** **deben** estar seguros de lo que van a obtener. Hay que evitar las **sorpresas** posteriores.
- Obligue a **los propietarios** a confirmar formalmente las necesidades. De esta **forma**, se limita la posibilidad de que alguna de las partes quede **descontenta**. El **incremento** continuo de las necesidades durante la fase de desarrollo de un proyecto es un problema insidioso que suele producirse con frecuencia cuando no se ha llegado a un acuerdo formal **sobre** las necesidades en un **primer momento**. **O** bien los propietarios del proyecto no dejan de pedir nuevos elementos o una de las partes descubre nuevos **aspectos** no previstos hasta entonces.

Determinación de tecnología necesaria

La determinación de la tecnología necesaria es tan importante como la fase de determinación de las necesidades de los usuarios. Puede que no resulte imposible

ejecutar un sitio Web con mas de 20 millones de solicitudes al mes en un solo servidor **pero sí** se necesitara, **al menos**, un buen ordenador que cumpla una serie de condiciones. El equipo del proyecto no debe imponer ningun requisito previo al proyecto, como que ejecute Linux y MySQL. Las necesidades en materia de tecnologia se abordan en un momento posterior para poder adaptarlas al proyecto, dentro de los limites definidos. Se **deben** responder cuestiones como las siguientes:

- ¿Número de equipos y tipo de arquitectura necesaria para unirlos?
- ¿Qué tipos de equipos se necesita utilizar?
- ¿Qué sistemas operativos, sistemas de bases de datos y otras aplicaciones de software se necesitan (como servidores Web, clientes de correo y demas)?
- ¿Qué lenguajes se utilizan para desarrollar la aplicacion? ¿Seran orientados a objetos?

Fase 2: Diseño de la aplicacion

Una vez definidas las necesidades, llega el momento de diseiir la aplicacion

Modelado

Un modelo simplifica la estructura del programa y traduce las necesidades de los usuarios a un formato que el programador comprende de manera sencilla. Puede tratarse de modelos formales, como los del lenguaje unificado de modelado de sistemas (UML), un diagrama de flujo de datos o sencillamente un dibujo sobre un pedazo de papel.

Los elementos fundamentales que debe incluir son los datos que necesita cada proceso y la información que genera cada uno de ellos.

Uso de pseudocodigo

El pseudocodigo es otro paso que puede ayudar a los programadores a desarrollar una aplicacion de forma mas rapida y sencilla. En lugar de preocuparse por los requisitos exactos de sintaxis del lenguaje de programacion, el pseudocodigo responde a las necesidades lógicas, creando los algoritmos necesarios para resolver los problemas.

Codificación

Éste es el paso que se suele considerar como unico. Sin embargo, la labor de codificacion resulta a menudo mucho mas sencilla cuando se ha creado documentacion en los pasos anteriores.

Utilice los siguientes consejos durante la fase de **codificación**:

- Documente siempre su código. Incluya comentarios dentro del código y cree documentos aparte en los que se describan como se organiza la aplicación y su **funcionamiento**. Si no quiere hacerlo para su uso personal, **hágalo pensando** en los programadores que lo vayan a utilizar posteriormente. (No sabe lo sencillo que resulta olvidar algún detalle "trivial" después de unos meses.) Se puede llamar **egoístas** a los programadores que no dejan una extensa **documentación** para su uso posterior. Asegurese de asignar tiempo a esta tarea y no utilice la excusa de los plazos para pasarla por alto.
- Utilice nombres de **archivo**, de función y de variable claros. Por ejemplo, la función `f3()` no es muy **intuitiva**, mientras que `clear_interes` resulta más clara.
- No intente reinventar la rueda. **Existen** muchas clases y código de ejemplo disponibles. Son raros los casos en los que los programadores necesitan crear algo único. La mayor **parte** del trabajo suele ser repetitivo, convirtiendo la labor de los programadores en bibliotecarios a la búsqueda del código **correcto** para realizar su trabajo.
- Inicialice todas las variables y documentelas en un lugar, incluso si está codificando en lenguajes que **permiten** el uso de variables antes de su inicialización. Con **ello**, no sólo logrará que **resulten** más legibles sino también más seguras.
- Divida su código en partes pequeñas. De esta **forma** le resultará más **sencillo** de entender, de depurar y de mantener. En el mundo del desarrollo Web, por ejemplo, en algunos códigos se adjudica **todo** el proceso a una sola secuencia de comandos, lo que suele dar como resultado una mezcla **incomprensible**.
- Utilice los **directorios** de **forma** inteligente. No debería incluir las distintas partes de una aplicación dentro del mismo directorio. Agrúpelas de **forma lógica** y en función de criterios de seguridad.
- Vuelva a utilizar su propio código. Tenga presente las funciones y las clases creadas y utilícelas constantemente. Escríbalas para poder utilizarlas en diferentes **tareas**. De esta **forma**, le resultará mucho más sencillo cambiar el código en un **momento** posterior, especialmente **al** realizar conexiones a bases de datos.
- Separe la **lógica** de la **presentación** (en los entornos Web, se suele mezclar HTML y lenguajes de secuencia de comandos).
- Al depurar una consulta, puede que le resulte de ayuda ejecutarla **directamente** en MySQL y no a través de una aplicación (en mi caso, suelo visualizar la consulta y pegarla dentro del cliente de MySQL). Esta opera-

cion contribuye a limitar **los errores** ya que **permite** determinar si se alojan en la **consulta** que se esta ejecutando o en otro **elemento** de la aplicacion.

- Acostumbrese a cerrar las conexiones (**ejecutando** `mysql_close` en PHP) y a vaciar **los recursos**, incluso en el **caso** de que **esté trabajando** en un **entorno** en el que no resulte necesario **realizar** estas **tareas**. Los desarrolladores Web suelen tener problemas en este **sentido** cuando **pasan** a trabajar con otros tipos de aplicaciones, ya que estan acostumbrados a un **entorno** tolerante como el de la Web, en el que todos **los recursos** se liberan cuando el proceso del servidor Web se completa. De **manera** similar, los lenguajes como PHP son mucho **menos** estrictos que otros como C++.
- El codigo sencillo es buen codigo. El hecho de escribir un **fragmento** de codigo en una **sola** linea ilegible, simplemente porque esta permitido, puede hacerle parecer mas listo, **pero** con **ello sólo** lograra fastidiar a aquellos que necesiten leer su codigo posteriormente. Los programadores que **complican** su codigo demasiado **fracasan** en lo que se **supone** que **tienen** que **hacer**: simplificar lo las **tareas** complejas. **Además**, el codigo sencillo suele resultar igual o incluso mas rapido que el complicado, por ejemplo, **al** utilizar funciones en lugar de expresiones regulares.
- En **los proyectos** en los que participen varios programadores, utilice estandares de codificacion. De esta **forma**, cuando **los miembros** del equipo tengan que trabajar **sobre** el codigo mutuo, el trabajo resultara mas productivo porque les llevara mucho **menos** tiempo ajustarlo **al** nuevo estandar (o a su ausencia). Dentro de **los estandares** de codificacion se incluyen aspectos como el numero de espacios (o tabuladores) que utilizar **al** sangrar el codigo o las convenciones que se emplean para nombrar variables (**uso** de letras **mayúsculas**, por ejemplo, `$totalConnects`; guiones bajos, por ejemplo, `$total_connects`; o la ausencia de espacios, por ejemplo, `$totalconnects`). Los **estándares** pueden incluso establecer **qué** editor utilizar ya que **cada** uno de ellos alinea el codigo de **manera** diferente, lo que merma su legibilidad.
- Los proyectos de mayor tamaño tambien necesitan algun tipo de control de version. De esta **forma** se evitan **los conflictos** cuando mas de una persona trabaja **sobre** ellos. Resulta sencillo perder el control incluso en proyectos realizados por una **sola** persona, ya que es habitual trabajar en varios equipos y guardar versiones en todos ellos. En **los proyectos** grandes se pueden utilizar aplicaciones como CVS o Visual SourceSafe (se pueden encontrar en <http://www.cvshome.org/> y <http://msdn.microsoft.com/ssafe/>, respectivamente) y para proyectos pequeños puede utilizar un sistema de **numeración**.
- Utilice prototipos, si el proyecto lo autoriza (en especial si **existen dudas** **sobre las** necesidades). En **los prototipos** se desarrolla un **modelo** del sis-

tema final, sin toda la funcionalidad, y se trabaja sobre él hasta que quede listo. El desarrollo de prototipos exige una mayor **participación** de los usuarios y puede utilizarse para hacerles sentir mas implicados.

Fase 4: fase de pruebas e implementación

No se salte nunca la fase de pruebas. La presión de los plazos puede tentarle a reducir las tres semanas de pruebas prevista a una, **pero** las consecuencias pueden resultar muy negativas. No considere la fase de prueba como un **fardo** innecesario, sino como un **elemento** vital para ajustar la aplicación y garantizar su correcta ejecución.

Existen varios tipos de pruebas:

- **Pruebas de unidades:** Estas pruebas garantizan que todas las clases, métodos y funciones funcionan correctamente en si mismas. Se comprueba que los resultados devueltos son correctos independientemente de las entradas; en otras palabras, se prueban todos los posibles escenarios.
- **Pruebas de integración:** Estas pruebas garantizan que **cada** unidad funciona como debería **al** integrarlas con otras unidades.
- **Pruebas de sistema:** Se prueba el sistema entero. Aquí se incluye la **comprobación** del rendimiento del sistema con carga (pruebas de estrés), varios usuarios, etc.
- **Pruebas de regresión:** Se trata de las pruebas que se realizan para **comprobar** si las modificaciones introducidas afectan a otras funcionalidades. Los "arreglos rápidos" **originan** consecuencias imprevisibles habitualmente.

Tras analizar las necesidades, diseñar la aplicación, codificarla y probarla, estamos en **disposición** de poder implementarla. Puede hacerlo en un **entorno** formado por un pequeño grupo de usuarios o en una pequeña oficina inicialmente, para poder resolver los imprevistos que surjan, o desplegar el nuevo sistema de **manera** completa. En este **caso**, prepárese para repetir **todo** el proceso si algún usuario solicita nuevas funciones.

Resumen

Existen varias técnicas para aumentar la portabilidad y simplificar el mantenimiento de sus aplicaciones de base de datos, especialmente en lo que se **refiere** a las consultas. Separe los detalles de **conexión** de las secuencias de comandos que establecen las conexiones y asegúrese de que se almacenan en un solo lugar para poder modificar los detalles de **manera** sencilla. Asegúrese de que los cam-

bios **futuros** en la estructura de la base de datos no afectan a las secuencias de comandos si **los** cambios no estan relacionados. Especifique **los** nombres de **los** campos en sus **consultas** SELECT e INSERT.

Determinados **entornos** (en especial en aplicaciones Web) pueden **beneficiarse** del uso de conexiones permanentes. Utilizelas para reducir la carga de la **conexión al** realizar conexiones frecuentes desde un mismo lugar.

MySQL suele ser un **entorno** mas eficiente para realizar una tarea que un lenguaje de programacion. MySQL esta optimizado para resultar rapido y **hacer** uso de indices, caches y **memoria** para acceder a la **información** rapidamente. Por ejemplo, el uso de un lenguaje de aplicacion en lugar de MySQL para ordenar datos, aunque resulta posible, no es **eficaz**. En general, utilice MySQL siempre que le resulte posible.

Al desarrollar aplicaciones, resulta necesario llevar a **cabo** una planificacion cuidadosa. Recoja y formalice todas las necesidades de **los** usuarios antes de determinar las necesidades de tecnologia, de **diseñar** el **modelo** y de codificar la aplicacion.

Al codificar, asegurese de evitar **los** fallos comunes para que codigo resulte flexible, portable y facil de mantener. Seguidamente, antes de implementar la aplicacion, disponga tiempo para **probarla** de **manera** extensiva y asigne a esta fase la misma importancia que a la **codificación** de la aplicacion.

6

Como ampliar las funciones de MySQL

Una de las grandes ventajas de MySQL es que sus funciones resultan relativamente sencillas de **ampliar**. Cuando en las primeras versiones de MySQL se le achacaba una **carencia** de funciones, la respuesta era "escribalas". MySQL facilita esta tarea para cualquier persona competente **C** o **C++**.

Puede agregar funciones a MySQL de dos **formas**: creando una funcion definida por el usuario (UDF) o agregando una funcion nativa (integrada). Las funciones definidas por el usuario se pueden incorporar a distribuciones fuente o distribuciones binarias. Las funciones integradas sólo se pueden **añadir** en distribuciones fuente (se modifica el codigo fuente y se compilan los elementos agregados).

En este capitulo sólo se analizan las funciones definidas por el usuario. (Se utiliza el termino UDF para describir el **conjunto** de funciones **C/C++** relacionadas, correspondiente a una unica funcion MySQL. El termino funcion describe una unica funcion **C/C++**).

En este capitulo se abordan los siguientes temas:

- Funciones estandar definidas por el usuario
- Agregacion de funciones definidas por el usuario
- Funciones y parametros UDF

Funciones definidas por el usuario

La mayor parte de las UDF se escriben en C o C++. Puede utilizarlas con distribuciones binarias o fuente configuradas `con-with-mysqld-id flags=rdynamic`. Una vez agregadas, las UDF estarán siempre disponibles al reiniciar el sistema, a menos que se utilice la opción `-skip-grant-tables`.

Existen dos tipos de UDF: estándar y agregadas. Las UDF estándar son como funciones integradas ordinarias, como `POW()` y `SIN()`, y actúan sobre una única fila de datos, y las funciones agregadas son similares a las funciones integradas `SUM()` y `AVG()` que actúan sobre grupos.

Para implementar una UDF necesita realizar los siguientes pasos:

1. Escribir las funciones en C o C++ (puede utilizar otros lenguajes siempre y cuando pueda compilarlas en una biblioteca compartida de código nativo).
2. Compilar e instalar la biblioteca.

Tras agregar una UDF, los detalles se almacenan en la tabla `func` de la base de datos `mysql`. La tabla `func` presenta este aspecto:

```
mysql> SHOW COLUMNS FROM func;
+-----+-----+-----+-----+-----+
| Field | Type                | Null | Key | Default |
Extra |
+-----+-----+-----+-----+
| name  | char(64) binary     |      | PRI |          |
|      |                      |      |     |          |
| ret   | tinyint(1)          |      |     | 0        |
|      |                      |      |     |          |
| dl    | char(128)           |      |     |          |
|      |                      |      |     |          |
| type  | enum('function','aggregate') |      |     | function |
|      |                      |      |     |          |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

El campo `name` contiene el nombre de la UDF (y el nombre de la función C/C++ principal). El campo `ret` indica si la UDF puede devolver valores nulos, el campo `dl` indica el nombre de la biblioteca que contiene la UDF (muchas UDF se pueden agrupar en una biblioteca) y el campo `type` indica si la UDF es estándar o agregada.

NOTA: Si está actualizando una versión más antigua de MySQL, es posible que no aparezcan todas las columnas. Ejecute la secuencia de comandos `mysql_fix_privilege_tables` (en el directorio `bin`) para agregar las columnas que faltan.

En esta **sección**, comenzaremos por compilar e instalar la UDF de ejemplo que se incluye en una **distribución** de MySQL y escribiremos una propia. Antes de empezar, **Cree** y agregue registros a una pequeña tabla, que utilizaremos para **probar** las UDF una vez agregadas:

```
mysql> USE firstdb;
Database changed
mysql> CREATE TABLE words (id tinyint(4), word varchar(50));
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO words VALUES (1,'aeiou');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO words VALUES (2,'bro');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO words VALUES (3,'so');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO words VALUES (4,'kisso');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO words VALUES (5,'lassoo');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO words VALUES (2,'bro');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO words VALUES (3,'so');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO words VALUES (4,'kisso');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO words VALUES (4,'kisso');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO words VALUES (5,'lassoo');
Query OK, 1 row affected (0.00 sec)
```

MySQL incorpora cinco UDF de ejemplo dentro del archivo `udf_example.cc`, que suele almacenarse en el directorio `mysql/sql` de la **distribución** de código fuente. Necesitara compilar este archivo **como** archivo objeto susceptible de uso compartido. En **los** sistemas Unix, estos **archivos** suelen tener la extensión `.so` y en Windows la extensión suele ser `.dll`. El comando (en Unix) que se utilizara sera parecido **al** siguiente:

```
% gcc -shared -o udf_example.so udf_example.cc
```

Este libro no es una guía de programación y compilación, de **manera** que **aun-**que se expliquen **los** distintos temas en profundidad, no se cubren todas las combinaciones posibles. Es posible que necesite recurrir a su experiencia o solicitar **ayuda** en dichas áreas para sacar el máximo **partido** a este capítulo.

Para **buscar** las opciones correctas de compilador para su sistema, puede recurrir a la utilidad **make**, que comprueba las dependencias. Cada sistema diferirá **pero** tras ejecutar **make** obtendrá un resultado parecido **al** siguiente:

```
% make udf_example.o
g++ -DMYSQL_SERVER -DDEFAULT_MYSQL_HOME="/usr/local/"
-DDATADIR="/usr/local/var/" -DSHAREDIR="/usr/local/share/
mysql/"
```

```
-DHAVE_CONFIG_H -I../innobase/include -I../include -I../
regex
-I. -I../include -I. -O3 -DDEBUG_OFF -fno-implicit-templates
-fno-exceptions -fno-rtti -c udf_example.cc
```

Tome las opciones que se suministran y utilícelas para compilar la UDF. De nuevo, su sistema puede diferir: algunos deberán eliminar la opción `-c` y otros deberán mantenerla.

Deberá conocer bien su sistema, recurrir a algún especialista en él o tener la suficiente paciencia y probar diferentes alternativas si en los primeros intentos no logra los resultados deseados. El comando final puede presentar un aspecto como el que se presenta a continuación:

```
% gcc -shared -o udf_example.so udf_example.cc -I../innobase/include
-I../include -I../regex -I. -I../include -I.
```

NOTA: En algunos sistemas, necesitará aplicar dos pasos: en primer lugar, compilar `udf_example.cc` como `udf_example.o`, y, a continuación, crear la biblioteca compartida a partir de `udf_example.o` (utilizando `gcc -shared -o udf_example.so udf_example.o`).

Tras compilar la UDF, colóquela en el directorio utilizado habitualmente para compartir sus bibliotecas. En los sistemas Unix, se tratará de cualquier directorio con la secuencia `ld` (generalmente, `/usr/lib` o `lib`); también podemos establecer una variable para que apunte al directorio en el que almacenamos la biblioteca.

Si escribe la secuencia `man dlopen` obtendrá el nombre de la variable de entorno (por lo general, `LD_LIBRARY` o `LD_LIBRARY_PATH`), que se establece en la secuencia de comandos de inicio (`mysql.server` o `mysql_safe`). En los sistemas Windows, por lo general la UDF se ubica en el directorio `WINDOWS\System32` o en el directorio `WINNT\System32`. Copie el archivo compilado en la ubicación correcta, por ejemplo:

```
% cp udf_example.so /usr/lib
```

Tras colocar el archivo, puede que algunos sistemas necesiten crear vínculos (ejecutando `ldconfig`, por ejemplo) o reiniciar MySQL antes de poder cargar la función.

Para cargar la UDF desde la línea de comandos MySQL, utilice la instrucción `CREATE FUNCTION`. Su sintaxis es la siguiente:

```
CREATE [AGGREGATE] FUNCTION nombre_de_función RETURNS
 (STRING|REAL|INTEGER)
 SONAME nombre_de_biblioteca_compartida
```

Este ejemplo incluye una serie de funciones UDF (puede incluir más de una dentro de una biblioteca).

Por ahora, cargaremos simplemente tres de las funciones, de la siguiente forma:

```
mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME
"udf_example.so";
Query OK, 0 rows affected (0.03 sec)
mysql> CREATE FUNCTION myfunc_double RETURNS REAL SONAME
"udf_example.so";
Query OK, 0 rows affected (0.00 sec)
mysql> CREATE AGGREGATE FUNCTION avgcost RETURNS REAL SONAME
"udf_example.so";
Query OK, 0 rows affected (0.00 sec)
```

Ahora podemos probar la nueva UDF. Para ver que hacen las UDF, debe examinar el archivo `udf_example.cc`. La UDF `metaphon` (el nombre correcto de este algoritmo es en realidad `metaphone`) toma una cadena y devuelve un resultado en función de la forma en la que suene la cadena. Es similar al conocido algoritmo `soundex`, ajustado al inglés.

```
mysql> SELECT METAPHON(word) FROM words;
+-----+
| METAPHON(word) |
+-----+
| E               |
| BR              |
| S               |
| KS              |
| LS              |
| BR              |
| S               |
| KS              |
| KS              |
| LS              |
+-----+
10 rows in set (0.00 sec)
```

Este ejemplo no resulta especialmente útil, pero nos ha servido para aprender a agregar funciones. Utilice el siguiente código para probar esta función agregada:

```
mysql> SELECT AVG COST(id,1.5) FROM words;
+-----+
| AVG COST(id,1.5) |
+-----+
| 1.5000           |
+-----+
1 row in set (0.00 sec)
```

Sólo se genera un resultado ya que la función actúa con un grupo. Como no se utilizó la cláusula `GROUP BY`, el conjunto completo de resultados se toma con un único grupo.

Si **agrupa los resultados por los contenidos de id**, obtendremos cinco **resultados**, ya que **existen cinco valores id exclusivos**:

```
mysql> SELECT id,AVGCOST(id,1.5) FROM words GROUP BY id;
+----+-----+
| id  | AVGCOST(id,1.5) |
+----+-----+
|  1  |          1.5000 |
|  2  |          1.5000 |
|  3  |          1.5000 |
|  4  |          1.5000 |
|  5  |          1.5000 |
+----+-----+
```

Puede **eliminar una funcion UDF con la instrucción DROP FUNCTION**, por ejemplo:

```
mysql> DROP FUNCTION myfunc_double;
Query OK, 0 rows affected (0.01 sec)
```

Puede visualizar la lista de UDF disponibles examinando **los contenidos de la tabla func** de la base de datos `mysql` de la siguiente **forma**:

```
mysql> SELECT * FROM mysql.func;
+----+-----+-----+-----+
| name      | ret  | dl          | type      |
+----+-----+-----+-----+
| metaphon  | 0    | udf_example.so | function  |
| avgcost   | 1    | udf-example-so | aggregate |
+----+-----+-----+-----+
2 rows in set (0.01 sec)
```

El usuario que agregue o elimine la funcion necesita disponer de permisos `INSERT` o `DELETE` para la tabla `func` o **la base de datos mysql**. Por lo general **sólo se conceden a un administrador**, ya que **además del riesgo de seguridad de acceso a la base de datos mysql**, una UDF puede causar mucho daioo.

A **continuación**, crearemos una UDF desde el principio. En primer lugar, **vamos a crear una UDF estándar** (no agregada) llamada `count_vowels`.

Funciones UDF estándar

Una UDF **estándar** consta de una funcion principal, que se denomina de la misma **forma** que la UDF y es obligatoria, y dos funciones opcionales, que se denominan de **forma similar pero se les agrega las secuencias `_init` y `_deinit`** al final. Todas estas funciones **deben** incluirse en la misma **biblioteca**.

La funcion `init`

La funcion `init` es la funcion de inicializacion, que se invoca una vez **al comienzo del procesamiento de la UDF**. Esta funcion comprueba **los argumentos**

pasados a la UDF (por ejemplo, si son del tipo o número correcto) y especifica los detalles sobre el resultado (si puede ser `NULL`, cuantos decimales puede tener, etc.)

Se declara de la siguiente forma:

```
my-bool nombre_de_función_init(UDF_INIT *initid, UDF_ARGS
*args, char *message);
```

La función devuelve un tipo booleano, que se establece en `false` si la función no recoge ningún error o `true` si detecta alguno.

El parámetro `initid`

El parámetro `initid` es la estructura de datos principal de la UDF. Se pasa a las tres funciones. Todos los cambios que se apliquen a los parámetros predeterminados se realizan en esta función. La estructura contiene los siguientes miembros:

- **my-bool maybe-null:** Se trata de un valor booleano que especifica si la UDF puede devolver un valor `NULL` (si se establece en `true`) o no (si se establece en `false`). De manera predeterminada está establecido en `false`, a menos que cualquiera de los argumentos de la función pueda ser `NULL`.
- **unsigned int decimals:** Especifica el número máximo de decimales que se pueden devolver. De manera predeterminada toma el número máximo de decimales que pasa a la función principal cualquiera de los argumentos. Por lo tanto, si se pasan 203,2, 219,12 y 341,456, `decimals` será 3 (por los tres decimales del último argumento). Puede establecer un límite máximo en la función `init`.
- **unsigned int max_length:** Especifica la longitud máxima del resultado devuelto. Para funciones UDF de cadena, el valor predeterminado se corresponde con la longitud del argumento de cadena más largo. Para enteros, el valor predeterminado es 21 (incluyendo el signo). Para números reales, el valor predeterminado es 13 (incluyendo el signo y el punto decimal) más el número de decimales.
- **char *ptr:** Se trata de un puntero que pueden utilizar las UDF (por ejemplo, para pasar datos entre las tres funciones). Asigne la memoria en la función `init` si el puntero se utiliza para nuevos datos.

El parámetro `arg`

El segundo parámetro, `args`, es una estructura que contiene argumentos pasados desde la consulta. Contiene los siguientes elementos:

- **unsigned int arg_count:** Contiene el número de argumentos pasados desde la consulta. Si la UDF toma un conjunto de argumentos, compruebe este valor para controlar los errores.

- **enum Item-result *arg_type:** Contiene una matriz de tipos. Cada elemento se corresponde con uno de los argumentos, por lo que el numero total de elementos es el mismo que el valor de `arg_count`. Los tipos posibles son `STRING_RESULT`, `INT_RESULT` y `REAL_RESULT`. Utilícelo para **comprobar errores** o para convertir el argumento en el tipo específico que necesite.
- **char **args:** Contiene una matriz de los argumentos pasados desde la consulta. Si el argumento es constante, se puede acceder a él como `args->args[i]`, donde `i` es el numero del elemento del argumento. Para un argumento no constante `args->args[i]` es 0, ya que el valor actual de la fila se pasa a la funcion principal (esta circunstancia se comenta en una seccion posterior).
- **unsigned long *lengths :** Una matriz que contiene la longitud maxima posible de cadena para cada argumento **pasado** por la consulta. Difiere de la funcion principal (esta circunstancia se comenta en una seccion posterior).

El parametro message

El parámetro `message` contiene un puntero de caracter, que se usa para todos aquellos mensajes que tengan lugar durante la inicializacion. Es aconsejable asignarle siempre un valor cuando la funcion `init` devuelve `true`, de forma que indique un error. El bufer de caracter predeterminado es de 200 bytes, pero es aconsejable utilizar un mensaje de error de un tamaño inferior (80 caracteres es una longitud de terminal estandar). Además debería utilizar un byte nulo al final.

La funcion principal

La funcion principal es la unica obligatoria en una UDF estándar y se invoca una vez para cada fila recuperada por la consulta. El valor devuelto desde esta funcion es el mismo que el valor recuperado para toda la UDF y puede ser una cadena, un numero real o un entero. La funcion debería declararse de una de las siguientes formas en funcion del valor recuperado. Si la UDF devuelve una cadena:

```
char *nombre_de_función(UDF_INIT *initid, UDF_ARGS *args, char
*result,
    unsigned long *length, char *is-null, char *error);
```

Si la UDF es un real:

```
double nombre_de_función (UDF_INIT *initid, UDF_ARGS *args,
    char *is-null, char *error);
```

Si la función devuelve un entero

```
long long nombre_de_función (UDF_INIT *initid, UDF_ARGS *args,
    char *is-null, char *error);
```

Para los tipos numericos, el valor devuelto de la funcion principal es simplemente el valor. Si es de **tipo** cadena, el valor devuelto es un puntero **al** resultado, con la longitud almacenada en el argumento `length`. El bufer de resultado lleva asignado 255 bytes de **forma predeterminada**, de **manera** que si el resultado fuera inferior, el puntero deberia ser el puntero de resultado **pasado** en la funcion principal. Si fuera superior, deberia ser el puntero asignado en la funcion `init` (para asignar espacio se utiliza `malloc()` y para desasignar dicho espacio se utiliza la funcion `deinit`).

El parametro `initd`

Todos los atributos de esta estructura (comentados anteriormente) estan disponibles para la funcion principal. Por **regla** general, no es necesario modificar ninguno de estos valores en la funcion principal.

El parametro `args`

Los atributos de esta estructura se comentaron anteriormente. **Ahora** bien, en la funcion principal, la matriz `args` contiene los argumentos pasados desde **cada** fila a la funcion. Como estos pueden diferir en tipo, debe asignarles el apropiado. Para un argumento de tipo `INT_RESULT`, utilice el tipo `long long`, de la siguiente forma:

```
long long int_val;
int_val = *((long long*) args->args[i]);
```

Para un argumento de tipo `REAL_RESULT`, utilice el tipo `double`, de la siguiente forma:

```
double real_val;
real_val = *((double*) args->args[i]);
```

Para un argumento de tipo `STRING_RESULT`, la cadena esta disponible como `args->args[i]` y la longitud de la **cadena** como `args->length[i]` excluyendo cualquier valor **nulo** final (para tipos numericos, `args->length[i]` sigue conteniendo la longitud maxima asignada en la funcion `init`).

El parametro `length`

Se trata de un puntero a un entero que se establece con la longitud del valor devuelto (excluyendo los valores nulos finales).

El parametro `is null`

Establezcalo en 1 si la UDF devuelve un valor **nulo**. De lo contrario, mantenga su valor predeterminado, 0.

El parametro `result`

Se trata de un puntero a una matriz de caracteres y es el lugar en el que se coloca el valor predeterminado de la UDF. Lleva asignado 255 bytes, de **manera** que si el resultado tuviera mayor longitud, necesitara utilizar el parametro `ptr` desde la funcion `init`. Es necesario asignar y desasignar esta **memoria**.

La función deinit

Esta función libera la memoria asignada por la función `init` y se encarga del resto de operaciones de limpieza, en especial del puntero asignado en la función. Esta función se declara de la siguiente forma:

```
void nombre_función_deinit(UDF_INIT *initid)
```

Creación de una UDF estandar de ejemplo

Tras su presentación, vamos a crear una pequeña UDF llamada `count_vowels`. Ésta toma un argumento (una cadena) y devuelve el número de vocales. El listado 6.1 contiene la UDF.

Listado 6.1. `count_vowels.cc`

```
#ifndef STANDARD
#include <stdio.h>
#include <string.h>
#else
#include <my_global.h>
#include <my_sys.h>
#endif
#include <mysql.h>
#include <m_ctype.h>
#include <m_string.h>

/* Debe estar bien o mysqld no encontrará el símbolo */

extern "C" {
my_bool count_vowels_init(UDF_INIT *initid, UDF_ARGS *args,
char *message);
void count_vowels_deinit(UDF_INIT *initid);
long long count_vowels(UDF_INIT *initid, UDF_ARGS *args,
char *is_null, char *error);
}

/* Asegurese de que se pasa un argumento y que es una cadena.
*/
my_bool count_vowels_init(UDF_INIT *initid, UDF_ARGS *args,
char *message) {
    if (args->arg_count != 1 || args->arg_type[0] !=
STRING_RESULT) {
        strcpy(message, "You can only pass one argument, and it
must be a string");
        return 1;
    }
    return 0;
}

/* no es necesario utilizar un función deinit, ya que no se ha
asignado memoria adicional */
```

```

void count_vowels_deinit(UDF_INIT *initid){
}

/* cuente el numero de vocales de la cadena */
long long count_vowels(UDF_INIT *initid, UDF_ARGS *args,char
*is-null,?
char *error) {
long long num-vowels = 0; /* el mismo tipo que el resultado
de la funcion */
char *word = args->args[0]; /* puntero a la cadena */
int i = 0; /* recorrer la palabra */
char c; /* para contener la letra */
while ( ( c = word[ i++ ] ) != '\0' ) {
switch ( c ) {
case 'a':
case 'e':
case 'i':
case 'o':
case 'u':
num_vowels++; /* si la letra de c es una vocal,
incremente el contador */
}
}
return num-vowels;
}

```

Tras guardar el archivo, apliquele el comando make, compilelo y copielo en el directorio seleccionado para sus bibliotecas, como se comento anteriormente:

```

% make count-vowels.o
g++ -DMYSQL_SERVER -DDEFAULT_MYSQL_HOME="\usr/local/mysql\"
-DDATADIR="\usr/local/mysql/var\"
-DSHAREDIR="\usr/local/mysql/share/mysql\"
-DHAVE_CONFIG_H -I../innobase/include -I../include
-I../regex -I. -I../include -I. -O3 -DDEBUG_OFF
-fno-implicit-templates -fno-exceptions -fno-rtti -c
repeat_str.cc
% gcc -shared -o count_vowels.so count_vowels.cc -I../innobase/include
-I../include -I../regex -I. -I../include -I.

```

A continuación establezca una conexión a MySQL, cargue la función y realice una prueba:

```

mysql> CREATE FUNCTION count-vowels RETURNS INTEGER SONAME
"count_vowels.so";
Query OK, 0 rows affected (0.02 sec)
mysql> SELECT id,word,count-vowels(word) FROM words;
+----+-----+-----+
| id  | word  | count-vowels(word) |
+----+-----+-----+
| 1  | aeiou | 5 |
| 2  | bro   | 1 |

```

```

| 3 | so | 1 |
| 4 | kisso | 2 |
| 5 | lassoo | 3 |
| 2 | bro | 1 |
| 3 | so | 1 |
| 4 | kisso | 2 |
| 4 | kisso | 2 |
| 5 | lassoo | 3 |
+-----+
10 rows in set (0.00 sec)

```

Si pasamos un argumento que no sea de cadena como el campo `id` o mas de un argumento, obtendremos el mensaje de error especificado:

```

mysql> SELECT id,word,count_vowels(id) FROM words;
ERROR:
You can only pass one argument, and it must be a string

```

ADVERTENCIA: Si realizó un cambio en la UDF, asegúrese de eliminar la función en primer lugar de MySQL antes de volver a cargarla. De lo contrario, se arriesga a bloquear MySQL y a tener que reiniciar.

Análisis de las funciones agregadas

Las funciones agregadas son aquellas que se pueden utilizar con una cláusula `GROUP BY`, como `SUM()` y `AVG()`. Para crear una UDF agregada, se utilizan las mismas funciones que en una UDF estandar, a las que se suman otras dos de caracter obligatorio: las funciones `reset` y `add`. El comportamiento de estas funciones resulta diferente.

- La función `reset`: Esta función se invoca al comienzo de cada nuevo grupo. Los datos utilizados para agrupar calculos se reinician aqui. Esta función se declara de la siguiente forma:

```

char *xxx_reset(UDF_INIT *initid, UDF_ARGS *args,
               char *is-null, char *error);

```

- La función `Addn`: Esta función se invoca para cada fila del grupo a excepción de la primera. Es probable que desee invocarla para todas las filas, en cuyo caso neccsitara hacerlo desde el interior de la función de reinicio.
- La función principal: La función principal solo se invoca una vez en cada grupo de datos (al final) para realizar los calculos necesarios sobre el grupo completo de datos (por regla general se accede mediante `initd->ptr`).
- La función `init`: Se comporta de la misma forma que en una UDF estandar, con la salvedad de que el atributo `ptr` resulta mucho mas importante en estas funciones: almacena datos sobre cada grupo que se agrega a la fun-

cion `add`. Como resultado se puede acceder a la funcion principal para obtener los datos sobre el grupo entero.

- **La función `deinit`:** Desempeña el mismo papel que en las UDF estandar, con la excepci3n de que se utiliza casi siempre, ya que es necesario vaciar el atributo `ptr`.

Creaci3n de una UDF agregada de ejemplo

Para crear la funcion UDF agregada, vamos a realizar algunos cambios en la UDF `count_vowels` para que cuente grupos de vocales. Vamos a crear una estructura llamada `data` con un contador de elementos. El valor del contador de esta estructura se incrementara cada vez que encontremos una vocal en la funcion `add` (que se invoca en todas las filas) y restableceremos el valor en la funcion `reset` (que se invoca una vez por grupo). Como la funcion `add` no se invoca de manera explicita en la primera fila, lo haremos desde la funcion `reset` para asegurarnos de que tambien se cuenta la primera fila de cada grupo. El listado 6.2 contiene la UDF agregada.

Listado 6.2. `count_agg_vowels.cc`

```
#ifndef STANDARD
#include <stdio.h>
#include <string.h>
#else
#include <my_global.h>
#include <my_sys.h>
#endif
#include <mysql.h>
#include <m_ctype.h>
#include <m_string.h>           // Para obtener strmov()

#ifdef HAVE_DLOPEN

/* Debe estar bien o mysqld no encontrara el simbolo */

extern "C" {
my_bool count_agg_vowels_init( UDF_INIT* initid, UDF_ARGS*
args, char* message );
void count_agg_vowels_deinit( UDF_INIT* initid );
void count_agg_vowels_reset( UDF_INIT* initid, UDF_ARGS* args,
char* is-null, char *error );
void count_agg_vowels_add( UDF_INIT* initid, UDF_ARGS* args,
char* is-null, char *error );
long long count_agg_vowels( UDF_INIT* initid, UDF_ARGS* args,
char* is-null, char *error );
}

struct count_agg_vowels_data {
```

```

    unsigned long long count;
};

/* Cuente el numero de vocales */
my_bool
count_agg_vowels_init( UDF-INIT* initid, UDF-ARGS* args, char*
message ) {
    struct count_agg_vowels_data* data;

    if (args->arg_count != 1 || args->arg_type[0] != STRING-RESULT)
    {
        strcpy(message, "You can only pass one argument, and it
must be a string");
        return 1;
    }

    initid->max_length = 20;
    data = new struct count_agg_vowels_data;
    data->count = 0;
    initid->ptr = (char*)data;

    return 0;
}

/* libere la memoria asignada desde ptr */
void
count_agg_vowels_deinit( UDF-INIT* initid ) {
    delete initid->ptr;
}

/* invocada una vez al principio de cada grupo. Necesita invocar
la funcion add
asi como restablecer data->count a 0 para cada nuevo grupo */
void
count_agg_vowels_reset( UDF-INIT* initid, UDF-ARGS* args,
char* is-null, char* message ) {
    struct count_agg_vowels_data* data = (struct
count_agg_vowels_data*)initid->ptr;
    data->count = 0;

    *is_null = 0;
    count_agg_vowels_add( initid, args, is-null, message );
}

/* invocada en cada fila, a ade el numero de vocales a data->count
*/
void
count_agg_vowels_add( UDF-INIT* initid, UDF-ARGS* args, char*
is-null, char* message ) {
    struct count_agg_vowels_data* data =
(struct count_agg_vowels_data*)initid->ptr;
    char *word = args->args[0]; /* pointer to string */

```



```

int I = 0;
char c;

while ( ( c = word[ I++ ] ) != '\0' ) {
    switch ( c ) {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
            data->count++;
    }
    I
}
I
/* devuelve data->count o un valor nulo si no encuentra nada */
long long
count_agg_vowels( UDF_INIT* initid, UDF_ARGS* args, char*
is-null, char* error )
{
    struct count_agg_vowels_data* data = (struct
count_agg_vowels_data*)initid->ptr;
    if (!data->count)
    {
        *is_null = 1;
        return 0;
    }
    I

    *is-null = 0;
    return data->count;
}

#endif /* HAVE_DLOPEN */
% make count_agg_vowels.o
g++ -DMYSQL_SERVER -DDEFAULT_MYSQL_HOME="\usr/local/mysql\"
-DDATADIR="\usr/local/mysql/var\"
-DSHAREDIR="\usr/local/mysql/share/mysql\"
-DHAVE_CONFIG_H -I../innobase/include -I../include
-I../regex -I. -I../include -I. -O3 -DDEBUG_OFF
-fno-implicit-templates -fno-exceptions -fno-rtti -c
repeat_str.cc
% gcc -shared -o count_agg_vowels.so count_agg_vowels.cc -I../
innobase/include -I../include -I../regex -I. -I../include -I.

```

A continuación, establezca una conexión a MySQL, cargue la función y realice una prueba.

```

mysql> CREATE AGGREGATE FUNCTION count_agg_vowels RETURNS INTEGER
SONAME "count_vowels.so";
Query OK, 0 rows affected (0.02 sec)
mysql> SELECT count_agg_vowels(word) FROM words;

```

```

+-----+
| count-agg-vowels(word) |
+-----+
|                          21 |
+-----+
1 row in set (0.01 sec)
mysql> SELECT id,count_agg_vowels(word) FROM words GROUP BY id;
+-----+
| id | count-agg-vowels(word) |
+-----+
|  1 |                          5 |
|  2 |                          2 |
|  3 |                          2 |
|  4 |                          6 |
|  5 |                          6 |
+-----+
5 rows in set (0.00 sec)

```

Resolución de problemas de UDF

Las UDF puede que no funcionen por varias razones. Resulta **bastante** habitual que MySQL se bloquee si la UDF no se ha implementando correctamente. Por lo **tanto**, resulta aconsejable poner atención **al** implementar una UDF sin **probar** en un sistema en ejecución **activo**.

Aunque **cada** sistema presenta sus características **propias**, a **continuación** se recogen **los** problemas mas comunes:

Asegurese de eliminar todas las funciones preexistentes que tengan el mismo nombre (si **está** actualizando una funcion) antes de cargarlas o volver a cargarlas. Puede que necesite eliminar la funcion manualmente de la tabla `func` si ha agregado una funcion UDF que no puede eliminar de la **forma** habitual.

Puede **intentar** detener y reiniciar MySQL antes de **cargar** la funcion aunque no suele ser necesario.

- Asegurese de que el tipo devuelto **al** crear la funcion coincide con el tipo devuelto de la funcion principal de su codigo (cadena, real o entero).
- Puede que necesite configurar MySQL con `--with-mysqld-ldflags=-rdynamic` para poder implementar la UDF.

Resumen

MySQL **permite** agregar funciones definidas por el usuario (UDF). Estas funciones se utilizan **como** si se tratara de una funcion normal incluida dentro de una **consulta**. Existen dos tipos de UDF: agregadas y estandar. Las UDF agregadas

funcionan sobre grupos de datos y pueden usarse con la clausula `GROUP BY`. Las **UDF** estandar actuan sobre filas simples de datos.

Las **UDF** estandar se componen de tres funciones: la funcion principal, que es obligatoria y debe invocarse para cada **fila**, y las funciones de inicializacion y desinicializacion, que se invocan una vez **al principio** y **al final**, respectivamente. Las funciones agregadas tambien utilizan la funcion `add` (invocada para cada **fila**, en lugar de la funcion principal, que se invoca una vez **al principio** de cada grupo) y la funcion `reset` (invocada **al principio** de cada grupo).

Parte II
Diseño
de una base
de datos

7 Comprension de las bases de datos relacionales

Al igual que aceptamos como algo habitual el uso de efectos especiales en las películas hasta que los comparamos con su estado en épocas anteriores, es probable que no apreciemos el potencial de las bases relacionales sin examinar sus precedentes.

Las bases de datos relacionales permiten a cualquier tabla relacionarse con otra a través de campos comunes.

Se trata de un sistema bastante flexible y la mayor parte de las bases de datos actuales son de este tipo.

En este capítulo se abordan los siguientes temas:

- El modelo jerárquico de base de datos
- El modelo de base de datos en red
- El modelo de base de datos relacionales
- Términos básicos
- Claves de tabla y clave externas
- Vistas

Análisis de los modelos anteriores a las base de datos

Hasta el advenimiento de las bases de datos, la única forma de almacenar datos consistía en utilizar archivos sin relacionar. Los programadores necesitaban realizar grandes esfuerzos para extraer los datos y sus programas tenían que realizar complejas operaciones de análisis y relación de datos.

Lenguajes como Perl, gracias a sus potentes expresiones regulares resultaban ideales para procesar texto y facilitaron enormemente el trabajo; sin embargo, el acceso a los datos desde archivos sigue resultando todo un reto. Sin un forma estandar para acceder a los datos, los sistemas resultan mas propensos a errores, mas lentos de desarrollar y mas dificiles de mantener. La redundancia de los datos (que se produce cuando se duplican de manera innecesaria) y su falta de integración (los datos no se modifican en todas las ubicaciones necesarias, lo que da lugar a que se devuelvan datos erroneos o sin actualizar) suelen ser las consecuencias frecuentes de los metodos de almacenamiento de datos cuyo acceso se realiza a traves de archivos. Los sistemas de administración de bases de datos (DBMS) se desarrollaron para suministrar una forma estandar y fiable de acceder y actualizar datos. Estos sistemas brindan una capa intermedia entre la aplicación y los datos, lo que permite a los programadores centrarse en el desarrollo de la aplicacion en lugar de tener que preocuparse por aspectos relacionados con el acceso a los datos.

Un modelo de base de datos es un modelo lógico que se centra en la representación de los datos. Los diseñadores de bases de datos no necesitan preocuparse por el almacenamiento físico de los datos ya que el modelo de base de datos les permite dirigir su atención a un nivel mas alto y conceptual, con lo que se reduce el espacio entre el problema del mundo real para cuya solución se esta desarrollando la aplicacion y su implementación técnica.

Existen varios modelos de base de datos. En primer lugar, estudiaremos dos modelos comunes: el modelo jerarquico de base de datos y el modelo de base de datos en red. A continuación, examinaremos el que utiliza MySQL (junto con los DBMS mas modernos), el modelo relacional.

Modelo jerarquico de base de datos

El primer modelo es el modelo jerarquico de base de datos, que se parece a un árbol invertido. Los archivos se relacionan en forma de superior a inferior: cada elemento superior puede tener mas de un elemento dependiente pero cada elemento secundario sólo puede tener un elemento superior. La mayor parte de los lectores estara familiarizado con este tipo de estructura (es la forma en la que funcionan la mayor parte de los sistemas). Por regla general existe un directorio raiz, o de nivel superior, que contiene varios directorios y archivos. A su vez,

cada subdirectorio puede contener otros archivos y directorios. Cada archivo y directorio sólo puede incluirse en otro directorio (solo tiene un directorio superior). Como puede observar en la figura 7.1, A1 es el directorio raíz y sus directorios dependientes son B1 y B2. B1 contiene a su vez a C1, C2 y C3, que también tiene sus propios elementos dependientes.

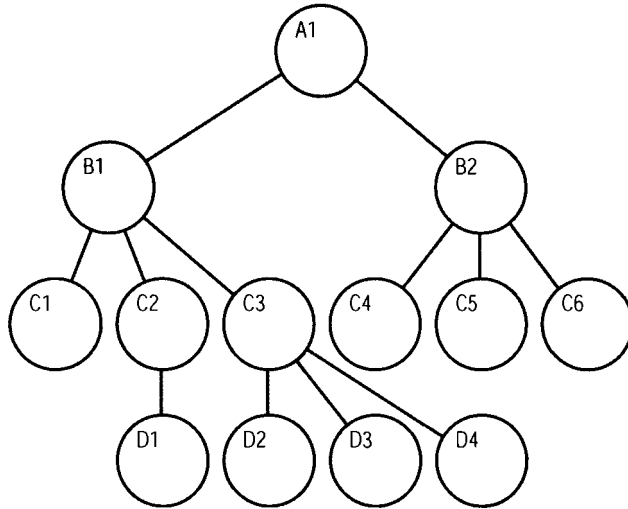


Figura 7.1. El modelo jerárquico de base de datos

Aunque este **modelo** supuso un gran **avance** para trabajar con archivos no relacionados, presenta serios inconvenientes. Las bases de datos jerárquicas representan bien las relaciones uno a varios (un directorio puede contener varios elementos secundarios; por ejemplo, una empresa puede tener varios **empleados**), **pero** presenta problemas en relaciones varios a varios. Las relaciones **como** las que se establecen entre un archivo de productos y un archivo de pedidos **resultan** difíciles de implementar en un **modelo** jerárquico. En **concreto**, un pedido puede contener varios productos y un **producto** puede aparecer en varios pedidos.

Así mismo, el **modelo** jerárquico no es flexible porque la agregación de nuevas relaciones puede dar lugar a grandes cambios en la estructura existente, lo que a su vez entraña la **modificación** de todas las aplicaciones existentes. Esta situación no resulta divertida cuando alguien olvida incluir un **tipo** de archivo y necesita agregarlo a la estructura justo antes del lanzamiento del proyecto.

El desarrollo de aplicaciones resulta también complejo porque el programador necesita conocer bien la estructura de los datos al recorrer el **modelo** para acceder a los datos deseados. Como hemos visto en los capítulos anteriores, para acceder a los datos situados en dos tablas relacionadas, **sólo** necesitamos **conocer** los campos pertinentes de las dos tablas. En el **modelo** jerárquico, es necesario conocer la cadena entera que une a ambos. Por ejemplo, para relacionar los datos de A1 y D4, necesitamos seguir la ruta que los une: A1, B1, C3 y D4.

Modelo de base de datos en red

El modelo de base de datos en red se desarrollo a partir del modelo jerarquico de base de datos y su objetivo era resolver algunos de los problemas de dicho modelo, en especial su falta de flexibilidad. En lugar de permitir que cada elemento secundario tuviera un unico elemento superior, este modelo admite que tengan varios (a los elementos secundarios se les denomina miembros y a los elementos superiores, propietarios). Este modelo permite modelar relaciones mas complejas, como las relaciones de varios a varios del ejemplo de pedidos y productos utilizado anteriormente. Como puede observar en la figura 7.2, A1 consta de varios miembros, B1 y B2. B1 es el propietario de C1, C2, C3 y C4. Sin embargo, en este modelo, C4 tiene dos propietarios, B1 y B2.

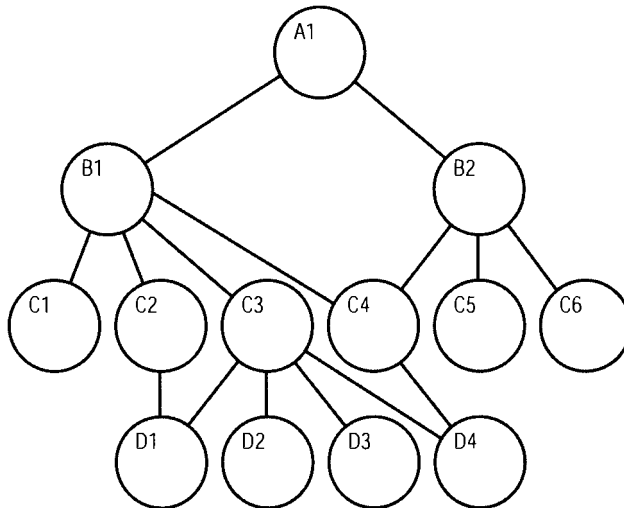


Figura 7.2. El modelo de base de datos en red

Obviamente este modelo presenta sus problemas o, de lo contrario, todo el mundo seguiria utilizandolo. Resulta mucho mas dificil de implementar y mantener, y aunque es mas flexible que el modelo jerarquico, sigue presentando problemas ya que no todas las relaciones se pueden satisfacer asignandoles otro propietario y el programador sigue teniendo que conocer bien las estructuras de datos para lograr que el modelo sea eficiente.

Modelo de base de datos relacional

El modelo de base de datos relacional supuso un gran avance con respecto al modelo de base de datos en red. En lugar de basarse en una relación de superior a inferior o de propietario y miembros, el modelo relacional permite que cualquier archivo se relacione con otro a traves de un campo comun. De repente, la com-

plejidad del diseño se redujo enormemente ya que los cambios se podían realizar sobre el esquema de la base de datos sin que ello afectara a la capacidad del sistema para acceder a los datos. Además, como el acceso no se realizaba a través de rutas que procedan o salgan de archivos, sino que se trata de una relación directa entre estos elementos, resulta sencillo agregar nuevas relaciones.

En 1970, cuando E. F. Codd desarrolló el modelo de base de datos relacional, se consideró inaplicable, ya que la facilidad de uso presentaba un coste importante de rendimiento, y el hardware en aquellos días no podía implementarlo. Desde entonces, los componentes informáticos han evolucionado enormemente y hoy en día incluso los equipos más sencillos son capaces de ejecutar sofisticados sistemas de administración de bases de datos relacionales.

Las bases de datos relacionales han evolucionado de forma paralela al desarrollo de SQL, tema analizado en la parte I de este libro. La simplicidad de SQL (que permite a cualquier persona sin experiencia aprender a realizar consultas básicas en un corto periodo de tiempo) explica en gran parte la popularidad del modelo relacional.

Las tablas 7.1 y 7.2 se relacionan entre sí a través del campo `stock_code`. Cualquiera de las dos tablas se puede relacionar mutuamente mediante un campo en común.

Tabla 7.1. La tabla PRODUCT

STOCK_CODE	DESCRIPTION	PRICE
A416	Nails, box	\$0.14
C923	Drawing pins, box	\$0.08

Tabla 7.2. La tabla INVOICE

INVOICE_CODE	INVOICE_LINE	STOCK_CODE	QUANTITY
3804	1	A416	10
3804	2	C923	15

Terminos basicos

El modelo relacional utiliza determinados terminos para describir sus componentes. Si ha leído la parte I del libro, estará familiarizado con muchos de ellos:

- Los *datos* son los valores que se almacenan en la base de datos. Por sí solos no tienen mucho significado. CA 684-2 13 es un ejemplo de una base de datos DMV (División de vehículos de motor).

- La *información* son los datos procesados. Por ejemplo, CA 684-213 es el número de registro de Lyndon **Manson** en una base de datos DMV.
- Una *base de datos* es un conjunto de tablas.
- Cada tabla se compone de *registros* (las **filas** horizontales de la tabla, que también se conocen como *tuplas*). Cada registro debe ser **exclusivo** y puede almacenarse en cualquier **orden** dentro de la tabla.
- Cada registro se compone de *campos* (las **columnas** verticales, que también se conocen como *atributos*). Básicamente, un registro es un hecho (por ejemplo, un cliente o una venta).
- Los campos puede ser de varios tipos. **MySQL** consta de una gran **cantidad** de tipos, como vimos en un capítulo anterior, **pero** por **regla** general se pueden clasificar en tres categorías: carácter, **numérico** y de fecha. Por ejemplo, el nombre de un cliente es un campo de tipo carácter, su fecha de nacimiento es de **tipo** fecha y su número de hijos es un campo de **tipo** numérico.
- El **rango** de valores permitidos para un campo se conoce como *dominio* (o *especificación del campo*). Por ejemplo, en un campo `tarjeta_de_crédito` se puede limitar los posibles valores a Mastercard, Visa y **Amex**.
- De un campo se dice que contiene un valor **nulo** cuando no incluye ningún valor. Los campos nulos pueden complicar los cálculos y generar **problemas** de exactitud en los datos. Por esta razón, muchos campos se **configuran** de **forma** que no puedan contener valores nulos.
- Una *clave* accede a registros especiales de una tabla.
- Un *índice* es un mecanismo que mejora el rendimiento de una base de datos. Los índices se suelen **confundir** con las claves. Estrictamente hablando, se integran en la estructura física mientras que las claves son parte de la **estructura lógica**. Sin embargo, estos términos se suelen utilizar indistintamente.
- Una *vista* es una tabla virtual compuesta de un subconjunto de tablas.
- Una **relación uno a uno** (1:1) es una **relación** en la que para cada **instancia** de la **primera** tabla de una **relación** sólo existe una instancia en la segunda. Por ejemplo, una cadena de **tiendas** en las que cada una tenga una máquina expendedora. Cada máquina expendedora **sólo** puede estar en una tienda y en cada tienda **sólo** puede haber una máquina **expendedora** (vease la figura 7.3).



Figura 7.3. Una relación uno a uno

- Una **relación uno a varios (1:V)** es una **relación** en la que para cada instancia de la **primera** tabla **existen** varias instancias en la segunda tabla. Este tipo de **relación** es muy habitual. Un ejemplo sería la **relación** entre un escultor y sus esculturas. Cada escultor puede crear varias esculturas, **pero** cada escultura ha sido creada por un solo escultor (vease la figura 7.4)

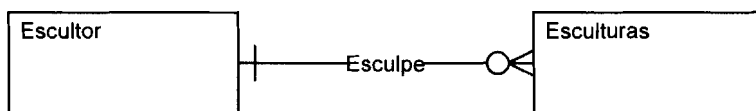


Figura 7.4. Una relación uno a varios

- Una **relación varios a varios (V:V)** es una **relación** que tiene lugar cuando para cada instancia de la **primera** tabla **existen** varias instancias en la segunda, y para cada instancia de la segunda **existen** varias instancias en la **primera**. Por ejemplo, un estudiante puede tener varios profesores y un profesor puede tener varios estudiantes (vease la figura 7.5).

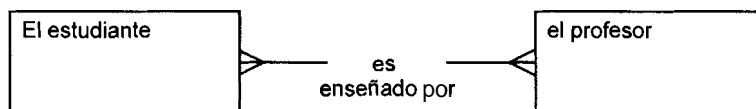


Figura 7.5. Una relación varios a varios

- Una **relación obligatoria** es una **relación** en la que para cada instancia de la **primera** tabla de la **relación** debe existir una o varias instancias en la segunda. Por ejemplo, para que exista un grupo de **música**, el grupo debe **constar** de **al menos** un músico.
- Una **relación opcional** es una **relación** en la que para cada instancia de la **primera** tabla de una **relación** pueden existir instancias en la segunda. Por ejemplo, si se puede incluir un autor que no haya escrito ningún libro en una base de datos (en otras palabras, un autor **potencial**), se establece una **relación** opcional. Lo **inverso** no tiene por que ser **cierto** necesariamente; por ejemplo, para poder incluir un libro en la base de datos, debe tener un autor.
- *La integridad de los datos* hace referencia a la **condición** de que los datos **sean** exactos, válidos y coherentes. Un ejemplo de **mala** integración sería si el número de teléfono de un cliente se almacenara de **forma** diferente en dos ubicaciones. Otro sería si el registro de un **curso** contuviera una referencia a un profesor que ya no estuviera en el **centro**. En un capítulo posterior aprenderemos una técnica que nos ayudara a minimizar **los** riesgos de este tipo de problemas: la **normalización** de base de datos.

Tras presentar algunos de los terminos basicos, en la siguiente sección se analizan las claves de tabla, un **aspecto** fundamental de las bases de datos relacionales.

Claves de tabla

Una *clave*, como indica el **término**, desbloquea el acceso a las tablas. Si **conocemos** la clave, sabremos como **identificar** sus registros asi como las relaciones entre las tablas. Una *clave candidata* es un campo o una combinacion de campos que identifiquen un registro de **manera** exclusiva. No puede contener un valor nulo y su valor debe ser exclusivo. (La existencia de duplicados impide la identificación de un registro exclusivo).

Una *clave primarin* es una clave candidata que ha sido designada para **identificar** de forma unica los registros de una tabla en la estructura completa de una tabla. Como **ejemplo**, la tabla 7.3 muestra la tabla `customer`.

Tabla 7.3. La tabla `CUSTOMER`

<code>CUSTOMER_CODE</code>	<code>FIRST_NAME</code>	<code>SURNAME</code>	<code>TELEPHONE_NUMBER</code>
1	John	Smith	448-2143
2	Charlote	Smith	448-2143
3	John	Smith	9231-5312

A **primera** vista, **existen** dos claves candidatas en esta tabla. Bastaría con **utilizar el campo** `customer_code` o **una combinación** de los otros tres campos. Siempre conviene asignar el **menor** numero posible de campos a las claves primarias; en este **caso**, seleccionaremos `customer_code`. Sin embargo, reflexionando un poco, nos daremos cuenta de que **existe** la posibilidad de que la segunda combinacion no resulte exclusiva. En teoria, la combinacion de los campos `first_name`, `surname` y `telephone` puede incluir campos duplicados, por ejemplo, un padre que tenga un **hijo** con el mismo nombre y que ambos **utilicen** el mismo telefono de contacto. Este sistema deberia excluir esta posibilidad de **manera expresa** para poder considerar la combinacion de los tres campos como clave primaria.

Para evitar la confusion de los nombres comunes se puede asignar a **cada** uno de ellos un nombre exclusivo. Tras crear una clave primaria, el resto de claves candidatas se etiquetan como claves alternativas.

Claves externas

Ya sabemos que una relación entre dos tablas se crea asignando un campo comun a dos tablas. Este campo comun debe ser la clave **primaria** de una de las

- Una relación **uno a varios** (1:V) es una relación en la que para cada instancia de la **primera** tabla **existen** varias instancias en la segunda tabla. Este tipo de relación es muy habitual. Un ejemplo sería la **relación** entre un escultor y sus esculturas. Cada escultor puede crear varias esculturas, **pero** cada escultura ha sido creada por un solo escultor (vease la figura 7.4).

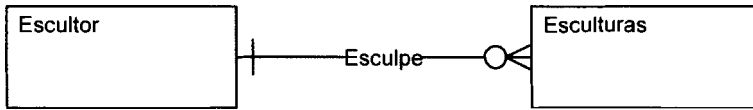


Figura 7.4. Una relación uno a varios

- Una relación **varios a varios** (V:V) es una relación que tiene lugar cuando para cada instancia de la **primera** tabla **existen** varias instancias en la segunda, y para cada instancia de la segunda **existen** varias instancias en la **primera**. Por ejemplo, una estudiante puede tener varios profesores y un profesor puede tener varios estudiantes (vease la figura 7.5).

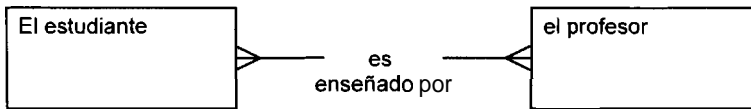


Figura 7.5. Una relación varios a varios

- Una relación **obligatoria** es una relación en la que para cada instancia de la **primera** tabla de la **relación** debe existir una o varias instancias en la segunda. Por ejemplo, para que exista un grupo de **música**, el grupo debe **constar** de **al menos** un músico.
- Una relación **opcional** es una relación en la que para cada instancia de la **primera** tabla de una **relación** pueden existir instancias en la segunda. Por ejemplo, si se puede incluir un autor que no haya escrito ningún libro en una base de datos (en otras palabras, un autor **potencial**), se establece una **relación** opcional. Lo **inverso** no tiene por que ser **cierto** necesariamente; por ejemplo, para poder incluir un libro en la base de datos, debe tener un autor.
- **La integridad de los datos** hace referencia a la **condición** de que los datos **sean** exactos, válidos y coherentes. Un ejemplo de **mala** integración sería si el número de teléfono de un cliente se almacenara de **forma** diferente en dos ubicaciones. Otro sería si el registro de un **curso** contuviera una referencia a un profesor que ya no estuviera en el **centro**. En un capítulo posterior aprenderemos una técnica que nos ayudara a minimizar **los** riesgos de este tipo de problemas: la **normalización** de base de datos.

Tras presentar algunos de los terminos básicos, en la siguiente sección se analizan las claves de tabla, un aspecto fundamental de las bases de datos relacionales.

Claves de tabla

Una *clave*, como indica el termino, desbloquea el acceso a las tablas. Si conocemos la clave, sabremos como identificar sus registros asi como las relaciones entre las tablas. Una *clave candidata* es un campo o una combinacion de campos que identifiquen un registro de manera exclusiva. No puede contener un valor nulo y su valor debe ser exclusivo. (La existencia de duplicados impide la identificación de un registro exclusivo).

Una *clave primaria* es una clave candidata que ha sido designada para identificar de forma unica los registros de una tabla en la estructura completa de una tabla. Como ejemplo, la tabla 7.3 muestra la tabla Customer.

Tabla 7.3. La tabla CUSTOMER

CUSTOMER_CODE	FIRST_NAME	SURNAME	TELEPHONE_NUMBER
1	John	Smith	448-2143
2	Charlote	Smith	448-2143
3	John	Smith	9231-5312

A primera vista, existen dos claves candidatas en esta tabla. Bastaría con utilizar el campo `customer_code` o una combinacion de los otros tres campos. Siempre conviene asignar el menor número posible de campos a las claves primarias; en este caso, seleccionaremos `customer_code`. Sin embargo, reflexionando un poco, nos daremos cuenta de que existe la posibilidad de que la segunda combinacion no resulte exclusiva. En teoria, la combinacion de los campos `first-name`, `surname` y `telephone` puede incluir campos duplicados, por ejemplo, un padre que tenga un hijo con el mismo nombre y que ambos utilicen el mismo telefono de contacto. Este sistema deberia excluir esta posibilidad de manera expresa para poder considerar la combinacion de los tres campos como clave primaria.

Para evitar la confusion de los nombres comunes se puede asignar a cada uno de ellos un nombre exclusivo. Tras crear una clave primaria, el resto de claves candidatas se etiquetan como claves alternativas.

Claves externas

Ya sabemos que una relación entre dos tablas se crea asignando un campo comun a dos tablas. Este campo comun debe ser la clave primaria de una de las

tablas. Considere una relación entre una tabla de clientes y una tabla de ventas. La relación no resultaría muy buena si en lugar de utilizar la clave primaria, `customer_code`, se utilizara otro en la tabla de ventas que no fuera exclusivo, como el nombre de cliente. De esta forma, no sabríamos nunca con seguridad qué cliente ha realizado la venta. Por ello, en la figura 7.6, `code_customer` se denomina *clave externa* de la tabla `sale`; en otras palabras, es la clave primaria de una tabla externa.

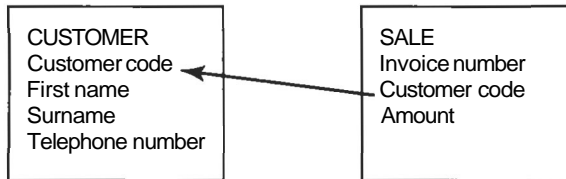


Figura 7.6. Establecimiento de las claves externas

Las claves externas permiten garantizar lo que se conoce como *integridad referencial*. Esto significa que si una clave externa contiene un valor, este valor hace referencia a un registro existente de la tabla relacionada. Por ejemplo, examine la tabla 7.4 y 7.5.

Tabla 7.4. La tabla LECTURER

CODE	FIRSTNAME	SURNAME
1	Anne	Cohen
2	Leonard	Clark
3	Vusi	Cave

Tabla 7.5. La tabla COURSE

COURSE_TITLE	LECTURER
Introduction to Programming	1
Information Systems	2
Systems Software	3

En este ejemplo existe integridad referencial ya que todos los profesores de la tabla COURSE existen en la tabla LECTURER. Ahora bien, supongamos que Anne Cohen abandona el centro y que se la elimina de la tabla de profesores. En caso de que la integridad referencial no se implemente de manera obligatoria, Ann Cohen se eliminaría de la tabla de profesores, pero no de la tabla COURSE, como se ilustra en la tabla 7.6 y 7.7.

Tabla 7.6. La tabla LECTURER

CODE	FIRSTNAME	SURNAME
2	Leonard	Clark
3	Vusi	Cave

Tabla 7.7. La tabla COURSE

COURSE_TITLE	LECTURER
Introduction to Programming	1
Information Systems	2
Systems Software	3

A partir de este momento, si se buscan los profesores de la asignatura *Introduction to Programming* se devolvera un registro no existente. Este hecho se conoce como **mala integración** de los datos.

Las claves **externas** tambien **permiten** realizar eliminaciones y actualizaciones en cascada. Por ejemplo, si Ann Cohen abandona el **centro**, **llevándose** consigo el **curso sobre introducción** a la programacion, se puede eliminar todas sus huellas de las tablas LECTURER y COURSE utilizando una **única instrucción**. El proceso de **eliminación** recorre las tablas pertinentes, suprimiendo todos los registros **precedentes**.

Desde la version 3.23.44, MySQL **admite** la comprobacion de clave externa en las tablas **InnoDB** y las eliminaciones en cascada se incorporaron en la version 4.0.0. Recuerde, sin embargo, que la implementacion de la integridad tiene un **coste de rendimiento**. Sin ella, la **responsabilidad** de mantener la integridad de los datos recae **sobre** la aplicacion.

Las claves **externas** pueden contener valores **nulos**, que indican que no existe relacion.

Introducción a las vistas

Las vistas son tablas virtuales. Son unicamente una estructura y no contiene datos. Su funcion es permitir que un usuario pueda ver un subconjunto de los datos reales.

Las vistas son una de las funciones MySQL mas solicitada y su implementacion esta prevista para la version 5.

Una vista puede componerse de un subconjunto de una tabla. Por ejemplo, la tabla 7.8 es un subconjunto de una tabla completa, que se ilustra en la figura 7.9.

tablas. Considere una relación entre una tabla de clientes y una tabla de ventas. La **relación** no resultaría muy buena si en lugar de utilizar la clave primaria, `customer_code`, se utilizara otro en la tabla de ventas que no fuera **exclusivo**, como el **nombre** de cliente. De esta **forma**, no sabríamos nunca con seguridad qué cliente ha realizado la venta. Por **ello**, en la **figura 7.6**, `code_customer` se denomina *clave externa* de la tabla `sale`; en otras **palabras**, es la **clave primaria** de una tabla externa.

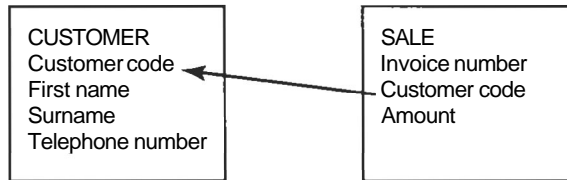


Figura 7.6. Establecimiento de las claves externas

Las **claves externas** permiten garantizar lo que se conoce como *integridad referencial*. Esto significa que si una **clave externa** contiene un valor, este valor hace referencia a un registro **existente** de la tabla relacionada. Por **ejemplo**, examine la **tabla 7.4 y 7.5**.

Tabla 7.4. La tabla LECTURER

CODE	FIRSTNAME	SURNAME
1	Anne	Cohen
2	Leonard	Clark
3	Vusi	Cave

Tabla 7.5. La tabla COURSE

COURSE_TITLE	LECTURER
Introduction to Programming	1
Information Systems	2
Systems Software	3

En este **ejemplo** existe integridad referencial ya que todos **los** profesores de la tabla `COURSE` **existen** en la tabla `LECTURER`. Ahora bien, supongamos que Anne Cohen abandona el **centro** y que se la elimina de la tabla de profesores. En caso de que la integridad referencial no se implemente de **manera** obligatoria, Ann Cohen se eliminaría de la tabla de profesores, pero no de la tabla `COURSE`, como se ilustra en la **tabla 7.6 y 7.7**.

Tabla 7.6. La tabla LECTURER

CODE	FIRSTNAME	SURNAME
	Leonard	Clark
3	Vusi	Cave

Tabla 7.7. La tabla COURSE

COURSE_TITLE	LECTURER
Introduction to Programming	1
Information Systems	2
Systems Software	3

A partir de este momento, si se buscan los profesores de la asignatura *Introduction to Programming* se devolvera un registro no existente. Este hecho se conoce como mala integración de los datos.

Las claves externas tambien permiten realizar eliminaciones y actualizaciones en cascada. Por ejemplo, si Ann Cohen abandona el centro, llevandose consigo el curso sobre introducción a la programacion, se puede eliminar todas sus huellas de las tablas LECTURER y COURSE utilizando una unica instruccion. El proceso de eliminación recorre las tablas pertinentes, suprimiendo todos los registros precedentes.

Desde la version 3.23.44, MySQL admite la comprobacion de clave esterna en las tablas InnoDB y las eliminaciones en cascada se incorporaron en la version 4.0.0. Recuerde, sin embargo, que la implementacion de la integridad tiene un coste de rendimiento. Sin ella, la responsabilidad de mantener la integridad de los datos recae sobre la aplicacion.

Las claves externas pueden contener valores nulos, que indican que no existe relacion.

Introducción a las vistas

Las vistas son tablas virtuales. Son unicamente una estructura y no contiene datos. Su funcion es permitir que un usuario pueda ver un subconjunto de los datos reales.

Las vistas son una de las funciones MySQL mas solicitada y su implementacion esta prevista para la version 5.

Una vista puede componerse de un subconjunto de una tabla. Por ejemplo, la tabla 7.8 es un subconjunto de una tabla completa, que se ilustra en la figura 7.9.

Tabla 7.8. La vista STUDENT

Vista STUDENT
First-name
Surname
Grade

Tabla 7.9. La tabla STUDENT

STUDENT
Student-id
First-name
Surname
Grade
Address
Telephone

Esta vista podría utilizarse para permitir que los estudiantes examinen las notas de otros sin permitir el acceso a la información personal.

Una vista puede consistir en una combinación de un número de tablas, como la que se muestra en la tabla 7.10. Es una combinación de las tablas 7.11, 7.12 y 7.13.

Tabla 7.10. La vista Student Grade

Vista STUDENT GRADE
First-name
Surname
Course description
Grade

Tabla 7.11. La tabla STUDENT

STUDENT
Student_id
First_name

STUDENT
Surname
Address
Telephone

Tabla 7.12. La tabla COURSE

COURSE
Course_id
Course description

Tabla 7.13. La tabla GRADE

GRADE
Student-id
Course-id
Grade

Las vistas también resultan útiles para cuestiones de seguridad. En organizaciones de gran tamaño, en la que trabajen varios desarrolladores en un proyecto, las vistas pueden limitar el acceso a determinados datos.

Se pueden ocultar los elementos no necesarios para realizar el trabajo, aunque se incluyan en la misma tabla, y evitar de esta forma que se vean o manipulen. Las vistas permiten además simplificar las consultas para los desarrolladores.

Por ejemplo, sin esta función, un desarrollador necesitaría recuperar los campos de la vista con el siguiente tipo de consulta:

```
SELECT first_name,surname,course_description,grade FROM student,
grade, course WHERE grade.student_id = student.student_id AND
grade.course_id = course.course_id;
```

Con la función de vista, un desarrollador lograría lo mismo con la siguiente secuencia:

```
SELECT first_name,surname,course_description,grade FROM
student-grade-view;
```

Esta secuencia resulta mucho más sencilla para un desarrollador con poca experiencia que todavía no haya aprendido a realizar combinaciones, pero también lo es para un desarrollador con experiencia.

Resumen

Antes del surgimiento de las bases de datos, **los** programadores almacenaban **los** datos en archivos. Sin embargo, el acceso a **los** datos a través de archivos resulta ineficiente para el programador, de ahí que se crearan las bases de datos.

Las bases de datos jerárquicas almacenan **los** datos en una estructura **descendente** de uno a varios. **Resultan** poco flexibles y su uso exige un gran trabajo para **los** programadores.

Las bases de datos en red **permiten** una mejor representación de las **relaciones** varios a varios, **pero resultan** difíciles de desarrollar y mantener.

Las bases de datos relacionales **permiten** que cualquier tabla se relacione con otra a través de campos comunes. Este **modelo** es muy flexible y todas las bases de datos **modernas adoptan** este modelo.

Las claves de tabla **permiten** acceder a **los** registros de una base de datos. Una clave **primaria** se compone de uno o varios atributos que identifican una fila de **forma** exclusiva. Una clave externa se compone de uno o varios atributos, que **forman** una clave **primaria** en otra tabla.

Las vistas de tablas son subconjuntos lógicos de tablas existentes. No **contienen** datos **pero** facilitan el trabajo de **los** desarrolladores, garantizan la seguridad, etc.

8

Normalización de bases de datos

Este capítulo presenta una potente herramienta para optimizar el diseño de las bases de datos: la normalización.

La normalización fue desarrollada en los años 70 por E.F. Codd y se ha convertido en un requisito estándar en la mayor parte de los diseños de base de datos.

Siguiendo los pasos que aquí se presentan logrará reducir las anomalías de los datos y simplificar su mantenimiento.

Se abordarán los siguientes temas:

- Primera forma normal
- Segunda forma normal
- Tercera forma normal
- Forma normal de Boyce-Codd
- Cuarta forma normal
- Quinta forma normal
- Desnormalización

Concepto de normalizacion

En la **primera parte** de este libro creamos algunas tablas en MySQL. Quizás haya utilizado MySQL en algunos proyectos pequeños en los que las bases de datos constaban de una o dos tablas. Sin embargo, a medida que vaya adquiriendo experiencia y aborde proyectos de mayor tamaño, ira descubriendo que las consultas **resultan cada vez** mas complejas y **menos** manejables, que surgen **problemas** de rendimiento o que empiezan a aparecer anomalias en **los datos**. Sin un conocimiento del diseio y normalizacion de bases de datos, estos problemas podrian resultar acuciantes y le impedirian avanzar en el dominio de MySQL. La normalizacion de bases de datos es una tecnica que puede ayudarle a evitar la aparicion de anomalias en **los datos** asi **como** otras cuestiones relacionadas con la **administración** de datos. Esta tecnica consiste en transformar una tabla en varias fases: **primera forma** normal, **segunda forma** normal, **tercera forma** normal y otras. El objetivo consiste:

- Eliminar las redundancias de datos (y por **tanto** utilizar **menos** espacio)
- Facilitar la tarea de realizar cambios en **los datos** y evitar las anomalias **al** hacerlo
- Facilitar la **implementación** de los requisitos de integridad referencial
- Generar una estructura facilmente comprensible muy parecida a la **situación** que representan **los datos** y que permita su crecimiento.

Comencemos por crear un **conjunto** de datos de ejemplo. En primer lugar **repasaremos** el proceso de normalizacion, sin preocuparnos por la teoria, para **comprender** las razones de la normalizacion. Tras **ello**, se presentara la teoria y revisaremos las distintas fases de la normalizacion, lo que le ayudara a utilizar este proceso cuando necesite aplicarlo de nuevo.

Imagine que estamos trabajando **sobre** un sistema que registra las **plantas** situadas en determinadas ubicaciones y las características del **suelo** asociadas a ellas.

La ubicacion:

```
Codigo de ubicacion: 11
Nombre de ubicacion: Kirstenbosch Gardens
```

contiene las siguientes tres **plantas**:

```
Codigo de planta: 431
Nombre de planta: Leucadendron
Categoria de suelo: A
Descripción de suelo: Arenisca
Codigo de planta: 446
Nombre de planta: Protea
Categoria de suelo: B
```

Descripcion del suelo: Arenisca/Caliza
 Codigo de planta: 482
 Nombre de planta: Erica
 Categoria de suelo: C
 Descripcion del suelo: Caliza

La ubicacion:

Codigo de ubicacion: 12
 Nombre de ubicacion: Karbonkelberg Mountains

contiene las siguientes dos plantas:

Codigo de planta: 431
 Nombre de planta: Leucadendron
 Categoria de suelo: A
 Descripcion del suelo: Arenisca

Codigo de planta: 449
 Nombre de planta: Restio
 Categoria de suelo: B
 Descripcion del suelo: Arenisca/Caliza

En los datos anteriores hay un problema. Las tablas de las bases de datos relacionales **tienen formato** de cuadrícula, o tabla, (**MySQL**, como la mayor **parte** de las bases de datos **modernas** son relacionales) en las que **cada fila** contiene un registro. Vamos a **intentar** reorganizar estos datos en **forma de informe** tabular (como se ilustra en la tabla 8.1).

Tabla 8.1. Datos sobre las plantas visualizados en formato tabular

CÓDIGO DE UBICACIÓN	NOMBRE DE UBICACIÓN	CÓDIGO DE PLANTA	NOMBRE DE PLANTA	CATEGORÍA DEL SUELO	DESCRIP. DEL SUELO
11	Kirstenbosch Gardens	431	Leuca-dendron	A	Arenisca
		446	Protea	B	Areniscal Caliza
		482	Erica	C	Caliza
	Karbonkel-berg Mountains	431	Leuca-dendron	A	Arenisca
		449	Restio	B	Areniscal Caliza

¿Cómo introducir estos datos en una tabla de la base de datos? Podemos **intentar** copiar la **distribución** anterior para generar una tabla parecida a la ilustrada en la tabla 8.2. Los campos nulos **reflejan** los campos en los que no se ha introducido ningún dato.

Tabla 8.2. Intento de creación de una tabla con los datos de las plantas

CÓDIGO DE UBICACIÓN	NOMBRE DE UBICACIÓN	CÓDIGO DE PLANTA	NOMBRE DE PLANTA	CATEGORÍA DEL SUELO	DESCRIP. DEL SUELO
11	Kirstenbosch Gardens	431	Leuca-dendron	A	Arenisca
NULL	NULL	446	Protea	B	Areniscal Caliza
NULL	NULL	482	Erica	C	Caliza
12	Karbonkel-berg Mountains	431	Leuca-dendron	A	Arenisca
NULL	NULL	449	Restio	B	Arenisca/ Caliza

Esta tabla no tiene mucho **sentido**. Las **primeras** tres filas **forman** en **realidad** un grupo, ya que todas pertenecen a la misma ubicación. Si **toma** la tercera fila, **los** datos no están **completos**, ya que no puede saber donde se encuentra Erica. Así mismo, con la tabla tal y como esta, no se puede utilizar el código de ubicación ni ningún otro campo como clave **primaria** (recuerde que una clave **primaria** es un campo o un **conjunto** de campos que identifican de **forma** exclusiva un registro). No tiene mucho **sentido** tener una tabla si no se puede identificar cada uno de sus registros de **manera** exclusiva. Por lo **tanto** la **solución** consiste en asegurarse de que cada fila de la tabla sea única y que no **forma** parte de un grupo o conjunto. Para **ello**, eliminaremos **los** grupos o conjuntos de datos y convertiremos cada fila en un registro **completo** con derecho propio, como se ilustra en la tabla 8.3.

Tabla 8.3. Cada registro separado

CÓDIGO DE UBICACIÓN	NOMBRE DE UBICACIÓN	CÓDIGO DE PLANTA	NOMBRE DE PLANTA	CATEGORÍA DEL SUELO	DESCRIP. DEL SUELO
11	Kirstenbosch Gardens	431	Leuca-dendron	A	Arenisca
11	Kirstenbosch Gardens	446	Protea	B	Arenisca/ Caliza
11	Kirstenbosch Gardens	482	Erica	C	Caliza
12	Karbonkel-berg Mountains	431	Leuca-dendron	A	Arenisca

<i>CÓDIGO DE UBICACIÓN</i>	NOMBRE DE UBICACIÓN	<i>CÓDIGO DE PLANTA</i>	NOMBRE DE PLANTA	CATEGORÍA DEL SUELO	DESCRIP. DEL SUELO
12	Karbonkel-berg Mountains	449	Restio	B	Arenisca/Caliza

NOTA: La clave primaria se muestra en cursiva en la tabla 8.3 y en las siguientes tablas.

Fijese en que el código de la ubicación no puede convertirse por sí mismo en clave primaria ya que no identifica las filas de datos de manera única. Por lo tanto, la clave primaria debe ser una combinación del código de la ubicación y del código de la planta. Nunca se agregará el mismo tipo de planta más de una vez a una ubicación dada. Una vez recogido el hecho que tiene lugar en dicha ubicación, bastará. Si desea registrar la cantidad de plantas situadas en una ubicación (en este ejemplo sólo estamos interesados en la distribución de las plantas) no necesita agregar un nuevo registro completo para cada planta sino simplemente un campo de cantidad.

Si por alguna razón, tuvieramos que agregar más de una instancia de una combinación de planta y ubicación, nos veríamos obligados a agregar algo más a la clave para que resulte única.

Por lo tanto, ahora los datos se pueden volcar en un formato de tabla, pero todavía presentan problemas. La tabla registra tres veces que el código 11 hace referencia a Kirstenbosch Gardens. Además del gasto de espacio que supone, esta situación presenta otro serio problema. Examine atentamente la tabla 8.4.

Tabla 8.4 Anormalia en los datos

<i>CÓDIGO DE UBICACIÓN</i>	NOMBRE DE UBICACIÓN	<i>CÓDIGO DE PLANTA</i>	NOMBRE DE PLANTA	CATEGORÍA DEL SUELO	DESCRIP. DEL SUELO
11	Kirstenbosch Gardens	431	Leuca-dendron	A	Arenisca
11	Kirstenbosch Gardens	446	Protea	B	Arenisca/Caliza
11	Kirstenbosch Gardens	482	Erica	C	Caliza
12	Karbonkel-berg Mountains	431	Leuca-dendron	A	Arenisca

CÓDIGO DE UBICACIÓN	NOMBRE DE UBICACIÓN	CÓDIGO DE PLANTA	NOMBRE DE PLANTA	CATEGORÍA DEL SUELO	DESCRIP. DEL SUELO
12	Karbonkel-berg Mountains	449	Restio	B	Arenisca/ Caliza

¿No observa algo extraño? Felicidades si lo hizo. La palabra *Kirstenbosch* esta mal escrita en el segundo registro. A continuación, imagine que tenemos que buscar este error en una tabla con miles de registros. Si utilizamos la estructura de la tabla 8.4, la posibilidad de incluir datos erroneos se dispara.

La solución es sencilla. Basta con eliminar la duplicación. Lo que estamos haciendo es buscar dependencias parciales o, lo que es lo mismo, campos que dependan de una parte de la clave y no de la clave entera. Como la clave se compone del código de la ubicacion y del codigo de la planta, buscaremos campos que dependen unicamente de uno u otro codigo.

Son varios los campos en los que ocurre lo mismo. El nombre de la ubicación depende del codigo de ubicacion (el codigo de la planta no se utiliza para determinar el nombre del proyecto) y el nombre de la planta, el codigo del suelo y el nombre del suelo dependen del numero de la planta. Por lo tanto, extraiga todos estos campos, como se ilustra en la tabla 8.5.

Tabla 8.5. Extracción de los campos que no dependen de la clave entera

CÓDIGO DE UBICACIÓN	CÓDIGO DE PLANTA
11	431
11	446
12	431
12	449

Obviamente no podemos quitar los datos y eliminarlos completamente de la base de datos. Lo que hacemos es extraerlos y se colocarlos en una nueva tabla, compuesta de campos que tengan dependencia parcial y de los campos de los que dependan. Para cada campo clave de la dependencia parcial, se crea una nueva tabla (en este caso, ambas forman parte ya de la clave primaria, pero este no tiene por que ser siempre el caso).

Por lo tanto, hemos identificado el nombre de la planta, la descripcion del suelo y la categoria del suelo como dependientes del codigo de planta. La nueva tabla quedara integrada por el codigo de planta como una clave asi como del nombre de la planta, de la categoria de suelo y de la descripcion del suelo, como se ilustra en la tabla 8.6.

Tabla 8.6. Creacion de una nueva tabla con los datos de las planta

CÓDIGO DE PLANTA	NOMBRE DE PLANTA	CATEGORÍA DEL SUELO	DESCRIPCIÓN DEL SUELO
431	Leucadendron	A	Arenisca
446	Protea	B	AreniscalCaliza
482	Erica	C	Caliza
449	Restio	B	AreniscalCaliza

Vamos a realizar el mismo proceso con los datos de ubicacion, como se ilustra en la tabla 8.7.

Tabla 8.7. Creacion de una nueva tabla con los datos de ubicacion

CODIGO DE UBICACIÓN	NOMBRE DE UBICACIÓN
11	Kirstenbosch Gardens
12	Karbonkelberg Mountains

Como puede observar, en estas tablas se ha eliminado el problema de **duplicacion** anterior. Ahora hay un solo registro con el valor `Kirstenbosch Gardens`, por lo que resulta mucho mas facil detectar un error ortografico. Además, se malgasta **menos espacio** al no almacenar el nombre en muchos registros **diferentes**. Fijese en que el campo de codigo de ubicacion y el codigo de planta se **repiten** en las dos tablas. Se trata de **los campos** que crean la **relación** y **permiten** asociar varias **plantas** con varias ubicaciones. Obviamente no hay forma de eliminar la **duplicación** de estos campos sin perder la **relación** complemente, pero resulta mucho mas eficiente almacenar un pequieii codigo de **manera** repetida que un gran **fragmento** de texto.

Sin embargo la tabla sigue sin ser **perfecta**. Todavia queda una posibilidad de que se deslicen anomalias. Examine la tabla 8.8 atentamente.

Tabla 8.8. Otra anomalía

CÓDIGO DE PLANTA	NOMBRE DE PLANTA	CATEGORÍA DEL SUELO	DESCRIPCIÓN DEL SUELO
431	Leucadendron	A	Arenisca
446	Protea	B	AreniscalCaliza
482	Erica	C	Caliza
449	Restio	B	AreniscalCaliza

El problema que presenta la tabla 8.8 es que Restio se ha asociado con arenisca, cuando de hecho si se tiene una categoría de suelo de tipo B, debería ser una combinación de arenisca y caliza. (La categoría de suelo determina la descripción del suelo del ejemplo). De nuevo estamos almacenando datos de manera redundante: la relación de la categoría de suelo y la descripción del suelo se está almacenando en su conjunto para cada planta. Como antes, la solución consiste en extraer los datos sobrantes y colocarlos en su propia tabla. En realidad, lo que estamos haciendo en esta fase es buscar las relaciones transitivas o las relaciones entre dos campos que no sean clave. Las descripción del suelo, aunque dependa en un sentido del código de la planta (parecía una dependencia parcial cuando la examinamos en el paso anterior) depende en realidad de la categoría de suelo. Por lo tanto, la descripción del suelo debe eliminarse. De nuevo, extraigala y colóquela en la nueva tabla, junto con su clave actual (categoría de suelo), como se muestra en la tabla 8.9 y 8.10.

Tabla 8.9. Datos de las plantas después de extraer la descripción del suelo

CÓDIGO DE PLANTA	NOMBRE DE PLANTA	CATEGORÍA DE SUELO
431	Leucadendron	A
446	Protea	B
482	Erica	C
449	Restio	B

Tabla 8.10. Creación de una nueva tabla con la descripción del suelo

CATEGORÍA DE SUELO	DESCRIPCIÓN DEL SUELO
A	Arenisca
B	Arenisca/Caliza
C	Caliza

Nuevamente, podemos reducir la posibilidad de que aparezcan anomalías. Ahora resulta imposible asumir de forma errónea que la categoría de suelo B está asociada a algo que no sea una combinación de arenisca y caliza. Las relaciones entre la descripción del suelo y la categoría de suelo se almacenan en un solo lugar: la nueva tabla de suelo, de cuya exactitud podemos dar fe.

A continuación, procederemos a examinar este ejemplo sin las tablas de datos que nos han servido de guía. Con frecuencia al diseñar un sistema no se tiene un conjunto completo de datos de prueba y no resultarán necesarios si se comprenden las relaciones entre los datos. En el ejemplo anterior se han utilizado tablas para poner de manifiesto las consecuencias de almacenar datos en tablas no normaliza-

Por ahora, el ejemplo de las plantas no lleva asignada ninguna clave y consta de grupos que se repiten. Para poder cumplir la primera forma normal, necesitamos definir la clave primaria y cambiar la estructura con el fin de que no se incluyan grupos repetitivos; en otras palabras, cada intersección de fila y columna debe contener un valor, y sólo uno. Sin ello, no podremos colocar los datos en la típica tabla bidimensional que exigen la mayor parte de las bases de datos. Como clave primaria seleccionamos el código de ubicación y el código de planta de forma conjunta (ninguno de los campos puede identificar por sí solo un registro de forma exclusiva) y sustituimos los grupos repetidos por un atributo de valor único. En la tabla 8.11 se recogen los resultados.

Tabla 8.11. Primera forma normal

Tabla PLANTA UBICACIÓN	
<i>Código de ubicación</i>	
Nombre de ubicación	
Código de planta	
Nombre de planta	
Categoría de suelo	
Descripción del suelo	

Esta tabla está en la primera forma normal. Pero ¿está en la segunda forma normal?

Segunda forma normal

Una tabla está en la segunda forma normal si sigue estas reglas:

- Si está en la primera forma normal
- Si no incluye dependencias parciales (cuando un atributo depende exclusivamente de parte de una clave primaria)

TRUCO : Para que un atributo dependa únicamente de parte de una clave primaria, dicha clave debe estar compuesta de más de un campo. Si la clave primaria contiene un solo campo, la tabla estará automáticamente en la segunda forma normal si está en la primera.

Examinemos todos los campos. El nombre de la ubicación depende únicamente del código de ubicación. El nombre de planta, la categoría de suelo y la descripción del suelo sólo dependen del código de planta. (Esto supone que cada planta

das. pero sin ellas tendra que basarse en las dependencias entre campos, que es la clave de la normalizacion de las bases de datos.

En primer lugar, la estructura de los datos era la siguiente:

- Codigo de utilizaci3n
- Nombre de ubicacion
- Numeros de plantas 1-n (1-n es una forma abreviada de indicar que existen varias ocurrencias de este campo: en otras palabras. que se trata de un grupo repetitivo).
- Nombre de plantas 1-n
- Categorias de suelo 1-n
- Descripciones de suelo 1-n

Ésta es una estructura sin ninguna normalizacion o lo que es lo mismo con *forma normal cero*. Por lo tanto, para iniciar el proceso de normalizacion. comenzamos por realizar el tr3nsito de la forma normal cero a la primera forma normal.

Primera forma normal

Las tablas que est3n en la primera forma normal siguen estas reglas:

- No incluyen grupos que se repitan
- Todos los atributos de clave est3n definidos
- Todos los atributos dependen de la clave primaria

Esto significa que los datos deben ser capaces de encajar en un formato tabular, en el que cada campo contiene un valor. En esta fase tambien se define la clave primaria. Algunos afirman que no es necesario definir la clave primaria para que una tabla cumpla la primera forma normal, pero esta operaci3n se suele realizar en esta fase y resulta necesaria para poder pasar a la siguiente fase. Dejando a un lado esta polemica, es aconsejable definir la clave primaria en este punto.

TRUCO: Aunque no siempre se considera como parte de la definici3n de la primera forma normal, el principio de la atomicidad se suele aplicar tambien en esta fase. Esto significa que todas las columnas deben contener las partes m3s pequeas o ser indivisibles. Un ejemplo habitual consiste en crear un campo para incluir el nombre completo de personas en lugar de un campo para el nombre y otro para el apellido. Por regla general, estas decisiones se lamentan posteriormente.

solo se desarrolla en un **tipo de suelo**, como se ha supuesto en este ejemplo). Por lo **tanto**, procedemos a extraer todos estos campos y a colocarlos en una tabla separada, con la clave que formaba **parte** de la original **sobre** la que dependen. El resultado son las tablas **8.12**, **8.13** y **8.14**.

Tabla 8.12. La tabla Planta Ubicacion sin las dependencias parciales

Tabla PLANTA UBICACIÓN
Código de planta
Codigo de ubicacion

Tabla 8.13. La tabla resultante de los campos dependientes del codigo de planta

Tabla PLANTA
Codigo de planta
Nombre de planta
Categoría de suelo
Descripción del suelo

Tabla 8.14. La tabla resultante de los campos dependientes del codigo de ubicación

Tabla UBICACIÓN
Código de ubicación
Nombre de ubicacion

Las tablas resultantes estan ahora en la segunda **forma normal**. Pero ¿están en la tercera **forma normal**?

Tercera forma normal

Una tabla esta en la tercera **forma normal** si cumple estas reglas:

- Si este en la segunda **forma normal**
- Si no contiene dependencias transitivas (cuando un atributo que no sea clave depende de una clave **primaria** a traves de otro atributo que no sea clave).

Como la unica tabla que tiene mas de un atributo que no sea clave es la tabla **PLANTA**, podemos ignorar el resto porque ya estan en la **tercera forma normal**. Todos los campos **dependen** de la **primera** clave de alguna **forma**, dado que las tablas estan en la segunda **forma normal**. Pero, ¿se realiza esta dependencia a traves de otro campo que no sea clave? El nombre de planta no depende de la

categoria de suelo ni de la descripcion del suelo. Ni la categoria de suelo depende de la descripcion del suelo ni del nombre de las plantas.

TRUCO: Si la tabla contiene un atributo que no sea clave es imposible, obviamente, que este dependa de otro atributo que tampoco sea clave. Todas las tablas de este tipo que están en la segunda forma normal, lo estarán automáticamente en la tercera forma normal

Sin embargo, la descripcion del suelo depende de la categoria de suelo. Volvemos a utilizar el procedimiento anterior para extraer el campo y colocarlo en su propia tabla con el atributo del que depende como clave. El resultado son las tablas 8.15, 8.16; 8.17 y 8.18.

Tabla 8.15. La tabla Ubicacion de planta no se modifica

Tabla UBICACIÓN DE PLANTA
Codigo de planta
Codigo de ubicacion

Tabla 8.16. La tabla Planta sin el campo Descripcion del suelo

Tabla PLANTA
<i>Código de planta</i>
Nombre de planta
Categoria de suelo

Tabla 8.17. La tabla Nuevo suelo

Tabla SUELO
<i>Categoria de suelo</i>
Descripcion del suelo

Tabla 8.18. La tabla Ubicacion no se modifica

Tabla UBICACIÓN
<i>Código de ubicación</i>
Nombre de ubicación

Todas estas tablas estan ahora en la tercera **forma normal**. La tercera **forma normal** suele ser suficiente para la mayor parte de las tablas porque evita los tipos mas comunes de anomalias en los datos. Es aconsejable ajustar todas las tablas a la tercera **forma normal** antes de implementarlas ya que con ello se cumpliran los objetivos de normalizacion expuestos al principio del capitulo en la gran **mayoria** de los casos. Las **formas normales** adicionales, como la **forma normal** de Boyce-Codd y la cuarta **forma normal**, no se suelen utilizar en aplicaciones de negocio. En la mayor parte de los casos, las tablas ajustadas a la tercera **forma normal** cumplen tambien con estas otras **formas normales**. **Ahora** bien, todos los expertos en bases de datos deberian conocer las excepciones y ser **capaces** de normalizar las tablas a otros niveles superiores si resultara necesario.

Forma normal de Boyce-Codd

E.F. Codd y R.F. Boyce, dos de las personas que mas han contribuido al desarrollo del **modelo** de base de datos, han sido distinguidos con el nombre de esta **forma normal**. E.F. Codd desarrollo y amplio el **modelo relacional**, además de desarrollar la normalizacion para los modelos relacionales en 1970, mientras que R.F. Boyce fue uno de los creadores del Lenguaje de **consulta** estructurado (por aquel entonces llamado SEQUEL).

No hay que confundir la **forma normal** de Boyce-Codd con la cuarta **forma normal**, pese a que algunas fuentes lo **hagan**. Vamos examinar un ejemplo con anomalias de datos presentes en la tercera **forma normal** que se resuelven con su **transformación** en la **forma normal** Boyce-Codd, antes de definirla (vease la tabla 8.19).

Tabla 8.19. Una tabla que contiene datos sobre las relaciones de estudiantes, cursos y profesores

Tabla ESTUDIANTE CURSO PROFESOR
Estudiante
Curso
Profesor

Supongamos que las siguientes afirmaciones son ciertas con respecto a la tabla 8.19:

- Cada profesor imparte un solo **curso**.
- Cada **curso** puede llevar asignado uno o varios profesores
- Cada estudiante **sólo** tiene un profesor por **curso**.
- Cada estudiante puede seguir uno o varios cursos.

¿Cuál sería la clave? Ninguno de estos campos sería suficiente por sí mismo para identificar de forma exclusiva un registro. Por lo tanto, tendremos que utilizar dos, pero ¿qué dos seleccionar?

Quizás la mejor opción serían el campo estudiante y el campo profesor, ya que este conjunto nos permitiría determinar el curso. También podríamos utilizar el campo estudiante y el campo curso, que determinarían al profesor. Por el momento vamos a utilizar esta última pareja como clave (véase la tabla 8.20).

Tabla 8.20. Uso del campo Estudiante y del campo Curso como clave

Tabla ESTUDIANTE CURSO PROFESOR
<i>Estudiante</i>
Curso
Profesor

¿En qué forma normal está la tabla? Está en la primera forma normal, ya que tiene una clave y no consta de grupos repetidos. También está en la segunda forma normal, ya que el profesor depende de los otros dos campos (los estudiantes tienen varios cursos y, por tanto, varios profesores y los cursos tienen varios profesores). Por último, también está en la tercera forma normal, ya que solo consta de un atributo que no sea clave.

Sin embargo, los datos presentan todavía algunas anomalías. En la tabla 8.21 se ilustra un ejemplo de la tabla con datos.

Tabla 8.21. Más anomalías en los datos

ESTUDIANTE	CURSO	PROFESOR
Conrad Peinaar	Biología	Nkosizana Asmal
Dingaan Fortune	Matemáticas	Kader Dlamini
Gerrie Jantjies	Ciencias	Helen Ginwala
Mark Thobela	Biología	Nkosizana Asmal
Conrad Pienaar	Ciencias	Peter Leon
Alicia Ncita	Ciencias	Peter Leon
Quinton Andrews	Matemáticas	Kader Dlamini

El hecho de que Peter Leon imparta el curso de ciencias está almacenado de manera redundante, como ocurre con Kader Dlamini con matemáticas y Nkosizana Asmal con biología. El problema es que el profesor determina el curso. O dicho de otra manera, el curso viene determinado por el profesor. La tabla se ajusta a las

reglas de la tercera forma normal porque existen atributos que no son clave que dependen de otros atributos que no sean clave. Sin embargo, un atributo clave depende de un atributo que no es clave. Podemos utilizar de nuevo el método consistente en eliminar este campo y colocarlo en una tabla nueva con su clave (vease la tabla 8.22 y la tabla 8.23).

Tabla 8.22. La tabla Estudiante Profesor tras eliminar el campo Curso

Tabla ESTUDIANTE PROFESOR
<i>Estudiante</i>
<i>Profesor</i>

Tras eliminar el campo de **curso**, la clave principal debe incluirse en los campos restantes para identificar un registro de **manera** exclusiva.

Tabla 8.23. La tabla Estudiante Profesor tras eliminar el campo Curso

Tabla PROFESOR CURSO
Profesor
Curso

Aunque seleccionamos el campo **Curso** como parte de la clave principal en la tabla original, el profesor determina el **curso**, que es la razón por la que lo convertimos en la clave **primaria** en esta tabla. Como puede observar, el problema de redundancia queda resuelto.

Por lo **tanto**, una tabla está en la **forma** de Boyce-Codd si cumple las siguientes condiciones:

- Si está en la tercera forma normal
- Si cada determinante es una clave candidata

Suena un tanto complicado, ¿verdad? Para la mayoría de la gente sin experiencia en el diseño de bases de datos, toda esta terminología resulta nueva. Si siguió este ejemplo, sin embargo, los términos le resultarán un poco más claros:

- Un *determinante* es un atributo que determina el valor de otro atributo.
- Una *clave candidato* es una clave o una clave alternativa (en otras palabras, el atributo puede ser una clave para dicha tabla).

El campo **Profesor** no es una clave candidata (ya que por sí solo no **identifica** de **manera** exclusiva un registro), pero determina el **curso**, por lo que la tabla no está en la **forma** normal de Boyce-Codd.

Examinemos el ejemplo de nuevo para ver que ocurría si seleccionamos el campo `Estudiante` y el campo `Profesor` como la clave, como muestra la tabla 8.24. ¿En que forma normal esta la tabla en esta ocasión?

Tabla 8.24. Uso de los campos `Estudiante` y `Profesor` como clave

Tabla ESTUDIANTE CURSO PROFESOR
Estudiante
Profesor
Curso

De nuevo, esta es la primera forma normal porque contiene una clave primaria y no incluye grupos repetidos. Sin embargo, no está en la segunda forma normal porque el `curso` viene determinado únicamente por una parte de la clave: el profesor. Si eliminamos el `curso` y su clave, el profesor, obtendremos los datos que se ilustran en las tablas 8.25 y 8.26.

Tabla 8.25. Eliminación del campo `Curso`

Tabla ESTUDIANTE PROFESOR
<i>Estudiante</i>
Profesor

Tabla 8.26. Creación de una nueva tabla con `Curso`

Tabla PROFESOR CURSO
Profesor
Curso

De cualquiera de las dos formas que lo hagamos, obtendremos las mismas dos tablas si nos aseguramos de que se ajustan a la forma normal de Boyce-Codd. Por regla general, suele ocurrir cuando existen campos alternativos entre los que seleccionar una clave; no importa los que se escojan al principio, porque tras el proceso de normalización obtendremos los mismos resultados en cada caso.

Cuarta forma normal

Examinemos una situación en la que pueden filtrarse redundancias aunque la tabla se encuentre en la forma normal de Boyce-Codd. Para esta demostración, retomaremos el ejemplo de los estudiantes, profesores y cursos utilizado en la

seccion anterior modificando una de las premisas iniciales. Esta vez, asumiremos que un estudiante pueda tener varios profesores asignados a un curso (vease la tabla 8.27).

Tabla 8.27. Datos de la tabla Estudiante
Curso Profesor, con varios profesores por curso

ESTUDIANTE	CURSO	PROFESOR
Conrad Peinaar	Biologia	Nkosizana Asmal
Dingaan Fortune	Matematicas	Kader Dlamini
Gerrie Jantjies	Ciencias	Helen Ginwala
Mark Thobela	Biologia	Nkosizana Asmal
Conrad Pienaar	Ciencias	Peter Leon
Alicia Ncita	Ciencias	Peter Leon
Quinton Andrews	Matematicas	Kader Dlamini
Dingaan Fortune	Matematicas	Helen Ginwala

Los datos son los mismos que en la seccion anterior con la excepci3n de que Helen Ginwala imparte clases de ciencias a Gerrie Jantjies adem3s de matem3ticas a Dignan Fortune y que Dingaan Fortune recibe clases de matematicas de Helen Ginwala y de Kader Dlamini.

La unica clave posible es una combinacion de los tres atributos, como muestra la tabla 8.28. Ninguna otra combinacion identifica de forma exclusiva un registro concreto.

Tabla 8.28. Tres atributos como clave

Tabla ESTUDIANTE CURSO PROFESOR
Estudiante
Profesor
Curso

Sin embargo, todavia presenta algunos comportamientos potencialmente anormales. El hecho de que Kader Dlamini de clases de matematicas se sigue almacenando mas de una vez asi como el hecho de que Dignan Fortune siga clases de matematicas. El autentico problema es que la tabla almacena mas de un tipo de circunstancias: una relaci3n estudiante-curso y una relaci3n estudiante-profesor. Podemos evitar esta situaci3n, como siempre, separando los datos en dos tablas, como se muestra en la tabla 8.29 y 8.30.

Tabla 8.29. Creacion de una tabla para la relación estudiante-profesor

Tabla ESTUDIANTE PROFESOR
Estudiante
Profesor

Tabla 8.30. Creacion de una tabla para la relación estudiante-curso

Tabla ESTUDIANTE CURSO
Estudiante
Curso

Esta **situación** se da cuando tenemos varias dependencias multivalor. Una **dependencia multivalor** se establece entre dos atributos cuando, para cada valor del primer atributo, existe uno o **varios** valores asociados al segundo atributo. Para cada valor de estudiantes, **existen** varios valores de **curso**.

Ésta es la **primera** dependencia multivalor. Seguidamente, para cada valor de estudiante, existe uno o **varios** valores de profesor. Ésta es la segunda **dependencia multivalor**.

Por lo **tanto**, una tabla esta en la cuarta **forma normal** si cumple los siguientes criterios:

- Si esta en la **forma normal** de Boyce-Codd
- Si no **contiene** mas de una dependencia multivalor

Quinta forma normal y otras formas

Existen otras **formas normales** de interes principalmente academico, ya que los problemas que solucionan apenas aparecen en la realidad. No vamos a **examinarlas** en detalle, pero para aquellos interesados, el siguiente ejemplo puede servirles de **aperitivo** (vease la tabla 8.31).

Tabla 8.31. Ejemplo Comercial

COMERCIAL	COMPAÑÍA	PRODUCTO
Felicia Powers	Exclusive	Libros
Afzal Igenesund	Wordsworth	Revistas
Felicia Powers	Exclusive	Revistas

Por regla general, estos datos se almacenarían en una tabla, ya que necesitamos los tres registros para ver las combinaciones válidas. Afzal Igenesund vende revistas para Wordsworth, pero no necesariamente libros. Felicia Powers vende libros y revistas para Exclusive. Pero vamos a agregar otra condición: si un comercial venden un determinado producto y lo vende para una empresa en concreto, entonces deben vender dicho producto para dicha empresa. En la siguiente tabla se incluyen datos adicionales que cumplen la condición expuesta (vease tabla 8.32).

Tabla 8.32. Tabla Comercial con mas datos

COMERCIAL	COMPAÑÍA	PRODUCTO
Felicia Powers	Exclusive	Libros
Felicia Powers	Exclusive	Revistas
Afzal Igenesund	Wordsworth	Revistas
Felicia Powers	Wordsworth	Libros
Felicia Powers	Wordsworth	Revistas

Con dependencia adicional, podríamos normalizar la tabla 8.32 en tres tablas separadas sin perder ningún dato, como se ilustra en las tablas 8.33, 8.34 y 8.35.

Tabla 8.33. Creación de una tabla con los campos Comercial y Producto

COMERCIAL	PRODUCTO
Felicia Powers	Libros
Felicia Powers	Revistas
Afzal Igenesund	Libros

Tabla 8.34. Creación de una tabla con los campos Comercial y Compañía

COMERCIAL	COMPAÑÍA
Felicia Powers	Exclusive
Felicia Powers	Wordsworth
Afzal Igenesund	Wordsworth

Tabla 8.35. Creación de una tabla con los campos Compañía y Producto

COMPAÑÍA	PRODUCTO
Exclusive	Libros

COMPAÑÍA	PRODUCTO
Exclusive	Revistas
Wordsworth	Libros
Wordsworth	Revistas

Basicamente, una tabla esta en la quinta **forma** normal, si no se puede dividir en tablas mas pequeias con diferentes claves (la mayor parte de las tablas se pueden dividir en tablas mas pequeias con la misma clave).

Mas alla de la quinta **forma** normal, entremos en el apasionante mundo de las **formas normales** de clave dominante, un **tipo** ideal teorico. Su uso practico para un diseiador de base de datos es similar **al concepto** de infinito para un contable: existe en teoria, **pero** no se utiliza en la practica. Ni el mas **corrupto** de los ejecutivos esperaria que su **contable** utilizara este concepto.

Para aquellos lectores interesados en **profundizar** en este tema de caracter **académico** y muy teorico, les sugiero obtener una copia de la **obra** *An Introduction to Database Systems* de C.J. Date (Addison-Wesley, 1999).

Concepto de desnormalizacion

La desnormalizacion es el proceso de invertir las transformaciones realizadas durante la normalizacion por razones de rendimiento. Se trata de un tema que suscita la **polémica** entre los expertos en bases de datos. Para algunos el **coste** es demasiado alto y nunca desnormalizan mientras otros alaban sus ventajas y **acostumbran** a desnormalizar.

Los defensores de la normalizacion siguen este proceso mental: la normalizacion crea mas tablas **al** avanzar **hacia formas normales** mas altas, **pero** un mayor numero de tablas significa un mayor numero de combinaciones **al** recuperar los datos, lo que contribuye a la ralentizacion de las consultas. Por esta razon, para mejorar la velocidad de determinadas consultas, se pueden anular las ventajas de la integridad de datos y devolver la estructura de los datos a una **forma** normal inferior.

En **materia** de desnormalizacion, es aconsejable adoptar un enfoque practico, teniendo en cuenta las limitaciones de SQL y de MySQL en particular, y ser prudente no desnormalizando de **manera** innecesaria. Los siguientes consejos le ayudaran a la hora de decidir:

- Si el uso de una estructura normalizada genera un rendimiento aceptable, no deberia desnormalizar.
- Si el rendimiento no resultara aceptable, asegurese de comprobar si el proceso de desnormalizacion lo convierte en aceptable. Es aconsejable buscar alternativas, **como** la **elección** de mejor hardware para evitar la

desnormalizacion. Resulta **difícil** deshacer los cambios estructurales posteriormente.

Asegurese de que prefiere una **menor** integridad de los datos a **cambio** de un mejor rendimiento.

Considere posible escenarios **futuros**, en los que las aplicaciones puede que **planteen** exigencias diferentes a los datos. El uso de la desnormalizacion para mejorar el rendimiento de una aplicacion obliga a la estructura de datos a depender de dicha aplicacion, cuando la **situación** ideal seria la contraria.

La tabla 8.36 introduce una estructura comun en la que puede que no le interese desnormalizar. **¿Puede** indicar en **qué forma** normal se **encuentra** la tabla?

Tabla 8.36. Tabla Cliente

Tabla CLIENTE

ID

Nombre

Apellido

Direccion 1

Direccion 2

Ciudad

Codigo postal

La tabla 8.36 debe estar en la **primera forma** normal porque tiene una clave **primaria** y no ser **repiten** grupos. Debe estar en la segunda **forma** normal porque consta de una **sola** clave, por lo que no puede haber dependencias parciales. **¿Está** en la tercera **forma**? **¿Contiene** alguna dependencia transitiva? Parece que si. El campo **Código postal** viene probablemente determinado por el atributo de la ciudad. Para convertir esta tabla en la tercera **forma** normal, deberia sacar el codigo postal y colocarlo en una tabla separada con la ciudad como clave. En la mayor parte de los casos, sin embargo, no es aconsejable realizar esta **operación**. Aunque la tabla no este en la tercera **forma** normal, no merece la pena separar esta tabla. Cuanto mayor sea el numero de tablas, mayor sera el numero de **combinaciones** que se necesitan establecer, lo que ralentiza el sistema. La razon para normalizar una tabla es reducir su **tamaño** eliminando campos redundantes (que a **menudo** aumentan la velocidad el sistema). Pero tambien debemos tener en cuenta la **forma** en la que se utilizan las tablas. Por **regla** general, el campo **Ciudad** y el campo **Código postal** se devolveran siempre de **forma** conjunta, como parte de la **dirección**. En la mayor parte de los casos, la **pequeña cantidad** de espacio

que se ahorra al eliminar **las** uniones de ciudad y código postal no compensarán la ralentización del sistema debido a **las** combinaciones extra. En algunas situaciones, puede que resulte útil, por ejemplo si necesita ordenar direcciones en función de códigos postales o ciudades para miles de clientes y la **distribución** de los datos implica que una **consulta** sobre una tabla nueva más pequeña puede devolver los resultados de manera significativamente más rápida. Los diseñadores de base de datos experimentados suelen **probar** otras posibilidades, siguiendo estrictamente **las reglas**, a medida que van entendiendo la **forma** en la que se utilizan los datos. Pero esto es algo que solo nos puede enseñar la **experiencia**. La normalización no es más que un conjunto de pasos que **suelen** generar una estructura de tabla más eficiente, no un conjunto de reglas para el diseño de bases de datos.

TRUCO : Existen diseños de bases de datos terribles, prácticamente siempre debidos a una falta de normalización en lugar de a un exceso de celo normativo. Por lo tanto, en caso de duda, normalice.

Resumen

La normalización de bases de datos es un proceso aplicado a **las** tablas para evitar los distintos tipos de anomalías que afectan a **los** datos:

- La primera **forma** normal no contiene grupos repetidos y garantiza que todos los atributos dependan de una clave primaria.
- La segunda **forma** normal no contiene dependencias parciales.
- La tercera **forma** normal no contiene dependencias transitivas.
- En la práctica, la tercera **forma** normal suele ser suficiente; de hecho, una normalización **excesiva** puede dar lugar a problemas de rendimiento, ya que es probable que la **base** de datos tenga que realizar demasiadas combinaciones.
- La **forma** normal de Boyce-Codd garantiza que **cada** determinante sea una clave candidata.
- La cuarta **forma** normal no contiene más que una dependencia multivalor.
- La quinta **forma** normal garantiza que **las** tablas no puedan reducirse a tablas de menor tamaño con **claves** diferentes.
- La **forma** normal de clave dominante es un ideal teórico que queda fuera del alcance de este libro.

9

Diseño de bases de datos

Las bases de datos **existen** por la necesidad de convertir **los datos** en **información**. Los datos son la **materia** prima. La información se obtiene procesando **los datos** para convertirlos en algo **útil**. Por ejemplo, **los millones** de nombres y números de teléfono de una guía de teléfonos son datos. La información es el número de teléfono del departamento de bomberos cuando nuestra casa se está quemando.

Una base de datos es un gran almacén de hechos, diseñado de tal **forma** que el procesamiento de dichos hechos resulte sencillo. Si la guía de teléfonos estuviera estructurada de **forma menos** práctica, por ejemplo, con **los nombres** y **los números** clasificados en **orden** cronológico en función de la asignación de teléfonos, la conversión de **los datos** en información resultaría mucho más **difícil**. Si no **sabemos** cuándo fue asignado el número de teléfono del departamento de bomberos, podríamos pasarnos horas buscándolo. Es probable que nuestra casa estuviera reducida a cenizas para cuando diéramos con él. Por **ello** debemos alegrarnos de que la guía de teléfonos esté diseñada **como** lo está.

Una base de datos es mucho más flexible. Un **conjunto** de datos similar **al** que contiene su guía de teléfonos se puede ordenar en **MySQL** por el nombre, el número de teléfono, la **dirección** o cronológicamente. Sin embargo, las bases de datos **resultan** mucho más complejas, ya que contienen muchos tipos diferentes de información. Se pueden combinar nombres de personas, trabajos y productos de una compañía para devolver información compleja. Pero esta complejidad **hace**

que el diseño de las bases de datos resulte también más complicado. Los malos diseños pueden ralentizar las consultas o incluso impedir el acceso a determinado tipo de información. **Ahora** analizaremos el diseño correcto de las bases de datos.

En este capítulo se abordan los siguientes temas:

- Ciclo de vida de la base de datos
- **Modelo** entidad-relación
- **Errores** comunes en el diseño de bases de datos
- Ejemplo real: **creación** de un sistema de seguimiento de publicaciones
- Control de simultaneidad mediante transacciones

Ciclo de vida de las bases de datos

Como todo, las bases de datos **tienen** una vida **finita**. Nacen en un arrebato de **optimismo**, y su vida discurre cosechando **fama**, **fortuna** y notoriedad, o un anonimato **tranquilo** según los **casos**, antes de extinguirse. Incluso las bases de datos más **aclamadas** acaban siendo sustituidas con el tiempo por otras estructuras más flexibles y actualizadas, y la vida comienza de nuevo. Aunque su **definición exacta** puede variar, por **regla** general el ciclo de vida de una base de datos consta de seis fases.

- **Análisis:** En la fase de **análisis** se entrevista a los accionistas y se examinan todos los sistemas existentes para identificar los problemas, las posibilidades y los límites. En esta fase se determinan los objetivos y el ámbito del nuevo sistema.
- **Diseño:** En la fase de diseño se crea el diseño conceptual a partir de las necesidades determinadas previamente. También se crea un diseño **lógico** y físico para preparar la implementación de la base de datos.
- **Implementación:** En la fase de implementación se **instala** el sistema de **administración** de la base de datos (DBMS), se crea la base de datos y se cargan o **importan** los datos.
- **Pruebas:** En la fase de pruebas se examina la base de datos y se ajusta, por lo general junto a las aplicaciones asociadas.
- **Puesta en marcha:** En esta fase la base de datos opera normalmente, produciendo información para sus usuarios.
- **Mantenimiento:** En la fase de mantenimiento se introducen cambios en la base de datos en respuesta a las nuevas necesidades o se modifican las condiciones operativas (como una carga más pesada).

El desarrollo de la base de datos no es independiente al desarrollo de los sistemas. De hecho, se **suele** considerar como uno de los componentes del proceso más

amplio de desarrollo de sistemas. Las fases del desarrollo de sistemas coinciden basicamente con las fases del ciclo de vida de una base de datos, con la diferencia de su alcance. Mientras el diseiio de las bases de datos se centra en el diseiio del sistema para almacenar los datos, el diseiio de sistemas se ocupa además de los procesos que incidiran en los datos.

Fase 1: Análisis

El sistema actual ya no da mas de si. Es hora de cambiar. Quizás el sistema de papel que se utiliza en la actualidad genera demasiados errores o el antiguo guion de Perl basado en archivos planos ya no es capaz de procesar la carga. O puede que la base de datos de noticias utilizada en un sitio Web presente problemas por su popularidad y necesite actualizarse. En esta fase se revisa el sistema existente.

En función del tamaño del proyecto, la responsabilidad del diseiio recaera en una sola persona, encargada de la implementación y de la codificación de la base de datos, o en un equipo completo de analistas. Utilizaremos el termino *diseñador* para hacer referencia a ambos casos. Estos son los pasos que constituyen la fase de analisis:

1. Análisis de la organizacion
2. Definicion de todos los problemas, posibilidades y limites
3. Definicion de los objetivos
4. Acuerdo sobre el ambito

Al revisar un sistema, el diseiador necesita adoptar una vision general, teniendo en cuenta no sólo el hardware o las estructuras de tabla existentes, sino la situación completa de la organizacion. Por ejemplo, un gran banco con un sistema de gestion centralizado tendra una estructura diferente y una forma especial de operar que una empresa de comunicacion descentralizada en el que todo el mundo puede remitir noticias a un sitio Web. Puede que parezca obvio, pero resulta vital comprender la organizacion para la que estamos desarrollando la base de datos para que el diseiio resulte satisfactorio. Las mismas exigencias del banco y de la empresa de comunicaci3n darán lugar a diferentes diseiios porque el distinto caracter de las organizaciones. En otras palabras, una soluci3n desarrollada para un banco no se puede implementar si un examen previo en una empresa de comunicacion, aunque las situaciones parezcan similares. El banco puede adoptar una cultura de control centralizado consistente en que las noticias enviadas a su sitio Web deban ser moderadas y autorizadas por el departamento central de administraci3n o puede exigir al diseiador que mantenga un control de cambios detallado sobre que se modifica, por que personas y cuando tienen lugar los cambios. Por otra parte, la empresa de comunicacion puede aplicar una política menos intervencionista y aceptar que las noticias las modifique cualquier editor autorizado. La comprension de la cultura de la organizacion ayudara al diseiador a plantear las preguntas correctas. Puede que el cliente no solicite un control de cambios, sino que lo de por supuesto simplemente

y que cuando llegue la fase de **implementación** necesitemos cubrir dicha necesidad, lo que exigira mas tiempo y recursos.

Una vez entendida la estructura de la organizacion, puede preguntar a los usuarios de los sistemas existentes cuales son sus problemas y necesidades, que limites **existen** actualmente y cuales son objetivos de la nueva base de datos, asi como cuales son los limites que existiran entonces. Debe consultar a diferentes miembros de la organizacion, ya que **cada** uno aiiadira nuevas puntos de vista **sobre** las necesidades de la base de datos. Por ejemplo, el departamento de marketing de la empresa de comunicacion puede que desee hacer un seguimiento de los movimientos de un articulo de noticias a otro dentro de su sitio Web y que el departamento editorial desee estadisticas detalladas **sobre** las horas a las que se suelen leer determinados articulos. Puede que tambien se le avise **sobre** posibles necesidades futuras. Por ejemplo, puede que el departamento editorial este planeando **ampliar** el sitio Web, que permitira a la plantilla vincular de forma entrecruzada los articulos Web. Si tenemos presente esta necesidad **futura**, es probable que resulte mas sencillo agregar la funcion de entrecruzar enlaces Web cuando llegue el **momento**.

Las restricciones puede referirse al hardware ("tenemos que utilizar nuestro servidor de base de datos existente, un AMD Duron a 900 MHz") o a los recursos humanos ("solo disponemos de una persona para capturar los datos por turno"). Las restricciones tambien pueden hacer referencia a limites **sobre** valores. Por ejemplo, que las **notas** de un estudiante en una base de datos de una universidad no puedan superar el 10 o que las tres categorias de asientos de la base de datos de un teatro sean **pequeiio**, **mediano** y **grande**.

Por **regla** general, no basta con que un nivel de **dirección**, o un individuo, indique los objetivos y los problemas actuales, salvo en organizaciones de **pequeiio** tamaio. Puede que las instancias de gestion mas altas corran con los gastos del diseiio de la base de datos, **pero** los niveles inferiores necesitaran utilizarlos y es probable que los datos que estos indiquen **sean** los mas importantes para lograr un diseiio satisfactorio.

Por supuesto, aunque **todo** resulta posible si no se ponen limites temporales o monetarios, este escenario es muy poco objetivo. La **determinación** del ambito y su **formalización** es una parte importante del proyecto. Si el presupuesto se establece para un mes de trabajo **pero** la **solución** ideal requiere tres, el diseiador debe dejar claras estas circunstancias y llegar a un acuerdo con los propietarios del proyecto **sobre** los aspectos que no se van a implementar.

Fase 2: Diseño

La fase de diseiio es aquella en la que se **utilizan** las necesidades identificadas en la fase anterior como base para desarrollar el sistema de noticias. **O** lo que es lo mismo, la conversion al plano tecnico de las estructuras de datos recogidas en el plano de los negocios. Los "ques" ("¿**Qué** datos se necesitan? ¿**Qué** problemas

se **deben** resolver?") se convierten en "cómos" ("¿Cómo se estructurarán los datos? ¿Cómo se accederá a los datos?").

Esta fase consta de tres partes: el diseño conceptual, el diseño lógico y el diseño físico. Algunas metodologías **funden** la fase de diseño lógica dentro de las otras dos fases. El objetivo de este capítulo no es el de realizar un estudio definitivo **sobre** las metodologías de diseño de bases de datos (**existen** libros enteros dedicados a analizar esta **materia**) sino el de **servir** de **introducción** a este tema.

Diseño conceptual

La finalidad de la fase de diseño conceptual consiste en desarrollar un **modelo** conceptual **basado** en las necesidades identificadas anteriormente y que resulte cercano **al modelo** físico final. El **modelo** conceptual más útil y común se denomina *modelo entidad-relación*.

Entidades y atributos

Las *entidades* son básicamente gente, **lugares** o cosas **sobre** las que deseamos mantener información. Por ejemplo, el sistema de una biblioteca consta de las entidades **libro**, **biblioteca** y **cliente**. Para aprender a identificar las entidades, su número y **los** atributos de una identidad, se necesita práctica **pero existen** algunas reglas generales. Las siguientes preguntas le ayudarán a identificar si un **elemento** es una **entidad**:

- ¿Puede variar en número independientemente de otras identidades? Por ejemplo, no es probable que **altura** de **persona** sea una **entidad**, ya que no puede modificarse de **forma** independiente de **persona**. No es fundamental, por lo que no es una **entidad** en este caso.
- ¿Resulta lo suficientemente importante como para garantizar el esfuerzo de mantenimiento? Por ejemplo, **cliente** puede que no sea importante para una tienda de comestibles y que, consecuentemente, no sea una **entidad** en este caso, **pero** que si sea importante para una videoteca y que sea **entidad**, **por tanto**.
- ¿Se puede dividir en categorías? Por ejemplo, un establecimiento dedicado **al** alquiler de vehículos puede utilizar diferentes criterios y almacenar las necesidades según el tipo de vehículos. Puede que **vehículo** no sea una **entidad** ya que se puede dividir en **coche** y **barco**, que son entidades.
- ¿Enumera un tipo de cosa, no una instancia? El videojuego **blow-em-up 6** no es una **entidad**, sino más bien una instancia de la **entidad** **juego**.
- ¿Lleva muchos hechos asociados? Si **sólo** contiene un atributo, es poco probable que sea una **entidad**. Por ejemplo, **ciudad** puede ser una **entidad** en algunos **casos**, **pero si sólo** contiene un atributo, **nombre de ciudad**, es más probable que sea un atributo de otra **entidad**, como **cliente**.

A **continuación** se incluyen ejemplos de entidades relacionadas con una **universidad** con sus posibles atributos entre parentesis:

- Curso(nombre, cbdigo, requisitos previos del curso)
- Estudiante(nombre, apellido, dirección, edad)
- Libro (título, ISBN, precio, cantidad en existencias)

Una instancia de una **entidad** es una ocurrencia concreta de dicha **entidad**. Por ejemplo, el estudiante Rudolf **Sono** es una instancia de la **entidad** estudiante. Es probable que haya muchas instancias. Si **sólo** existiera una, deberíamos **considerar** si se justifica la existencia de la **entidad**. El nivel superior de una instancia no justifica su **condición** de **entidad**. Por ejemplo, si el sistema se esta **desarrollando** para una universidad dada, **universidad** no sera una **entidad** porque el sistema **completo** es para dicha universidad. Sin embargo, si se desarrolla el sistema para controlar el proceso de **matriculación** en todas las universidades del **país**, **universidad** seria una **entidad** valida.

Relaciones

Las entidades se relacionan de determinadas **formas**. Por ejemplo, un usuario puede estar asociado a una biblioteca y puede sacar libros. Un libro puede **encontrarse** en una biblioteca dada. La **compresion** de **los** datos que se estan **almacenando** y su **relación** le guiara en gran **parte** de la tarea de **implementación** física en la base de datos. **Existen** varias relaciones posibles:

- **Obligatoria:** Para cada instancia de la **entidad** A, debe haber una o varias instancias de la **entidad** B. Esto no **significa** necesariamente que para cada **instancia** de la **entidad** B deba existir **una** o varias instancias de la **entidad** A. Las relaciones son opcionales u obligatorias en **una sola dirección**, de **manera** que la **relación** A-a-B puede ser opcional mientras que la **relación** B-a-A es obligatoria.
- **Opcional:** Para cada instancia de la **entidad** A, puede que existan o que no existan instancias de B.
- **Relaciones uno a uno (1:1):** En estas relaciones para cada **entidad** A, existe una instancia de B y viceversa. Si la **relación** es opcional, puede existir una o varias instancias y si la instancia es obligatoria, existe una y sólo una instancia de la **entidad** asociada.
- **Relación uno a varios (1:V):** Para cada instancia de la **entidad** A, existen varias instancias de la **entidad** B, mientras que para cada instancia de la **entidad** B sólo existe una instancia de la **entidad** A. De nuevo, las relaciones pueden ser opcionales u obligatorias.
- **Relaciones varios a vario (V:V):** Para cada instancia de la **entidad** A, existen varias instancias de la **entidad** B y viceversa. Estas relaciones pueden ser opcionales u obligatorias.

Existen varias formas de mostrar estas relaciones. La figura 9.1 muestra las entidades **student** y **courses**. En este caso, cada estudiante debe haberse registrado en un **curso** como minimo, pero un **curso** no tiene por que tener ningun estudiante registrado. La **relación estudiante-a-curso** es obligatoria y la **relación curso-a-estudiante** es opcional.

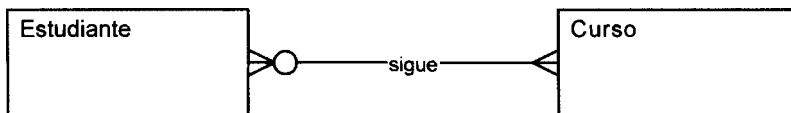


Figura 9.1. Una relación varios a varios

La figura 9.2 muestra las entidades **línea de factura** y **producto**. Cada línea de factura debe **constar** de un producto como minimo (pero no mas de un producto). Sin embargo, cada producto puede aparecer en varias **líneas** de producto o en ninguna. La **relación línea de factura-a-producto** es obligatoria mientras que la **relación producto-a-línea** de factura es opcional

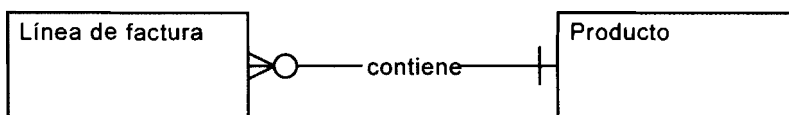


Figura 9.2. Una relación uno a varios

La figura 9.3 muestra las entidades **marido** y **mujer**. Cada marido debe tener una mujer, y solamente una, y cada mujer debe tener un marido, y solamente uno. **Ambas** relaciones son obligatorias.



Figura 9.3. Una relación uno a uno

Una **entidad** tambien puede estar relacionada consigo misma. Estas entidades se conocen como entidades recursivas. Tomemos un **entidad persona**: si esta interesado en almacenar **los datos sobre** las personas que son hermanos, tendremos una **relación "es hermano de"**. En este caso, la **relación** sera **V:V**.

Por el contrario, una **entidad debil** es una **entidad** que no puede existir sin otra **entidad**. Por ejemplo, en una escuela, la **entidad** **alumno** esta relacionada con la **entidad debil "padre/tutor"**. Sin el **alumno**, el padre o tutor no pueden existir en el sistema. Las entidades debiles suelen derivar su clave primaria, en parte o en su totalidad, de la **entidad** asociada. **Padre/tutor** puede **tomar** la clave **primaria** de la tabla **alumno** como parte de su clave **primaria** (o la clave entera si el sistema solo **almacena** un padre/tutor por alumno).

El término *conectividad* hace referencia a la clasificación de las relaciones (1:1, 1:V o V:V).

El término *cardinalidad* hace referencia al número específico de instancias posibles para una relación. Los *límites de cardinalidad* establecen el número máximo y el mínimo de ocurrencias de la entidad asociada. En el ejemplo del padre y la mujer, el límite de cardinalidad es (1,1) y en el caso de un estudiante que puede matricularse de uno a ocho cursos, el límite de cardinalidad se representaría como (1,8).

Desarrollo de un diagrama entidad-relación

Un diagrama de entidad-relación establece el modelo en el que las entidades se relacionen mutuamente. Se compone de varias relaciones, como los vistos en las figuras 9.1, 9.2 y 9.3. En general, estas entidades acaban convirtiéndose en tablas de base de datos.

El primer paso para desarrollar el diagrama consiste en identificar todas las entidades del sistema. En la fase inicial, no es necesario identificar los atributos, pero puede ayudarnos a aclarar las cosas si no estamos seguros de las entidades. Tras establecer las entidades, se identifican las relaciones entre ellas y se modelan en función de su tipo: uno a varios, opcional, etc. Existen muchos paquetes de software que pueden ayudarnos a dibujar un diagrama de entidad-relación, pero bastará con cualquier paquete básico.

Tras dibujar el diagrama entidad-relación inicial, se suele presentar a los accionistas. Los diagramas entidad-relación resultan sencillos de entender para personas sin conocimientos técnicos, en especial si se les guía a través del proceso. Este paso puede ayudarnos a identificar cualquier error que haya podido pasar inadvertido. Parte de la razón de utilizar modelos es que resultan mucho más sencillos de entender que el texto y es mucho más probable que los accionistas los comprendan, lo que contribuye a reducir los errores que puedan pasar inadvertidamente a las siguientes fases, en las que resultaran mucho más difíciles de resolver.

TRUCO: Es importante recordar que no existe una respuesta correcta o incorrecta. Cuanto más compleja sea la situación, mayor será el número de diseños que funcionarán. El diseño de bases de datos requiere práctica y los diseñadores más experimentados sabrán cuándo algo funciona y los posibles problemas que se pueden presentar en una fase posterior, ya que habrán recorrido el proceso anteriormente.

Una vez aprobado el diagrama, el siguiente paso consiste en sustituir las relaciones varios a varios por relaciones uno a varios. Un DBMS no puede implementar relaciones varios a varios directamente, por lo que se descomponen en relaciones más pequeñas. Para lograrlo, tendrá que crear una intersección o un tipo de entidad compuesta. Como las entidades de intersección son menos reales que las

entidades ordinarias, a veces **resultan** difíciles de designar. En este caso, puede nombrarlas en función de las dos entidades que se cruzan. Por ejemplo, puede intersecar la **relación** varios a varios establecida entre **estudiante** y **curso** en la **entidad estudiante-curso** (vease figura 9.4).

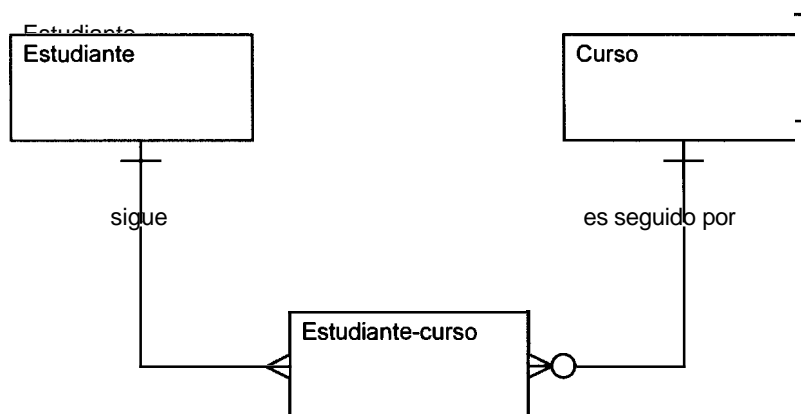


Figura 9.4. Creación de la entidad de intersección estudiante-curso

Lo mismo se puede aplicar incluso si la **entidad** es recursiva. La **entidad persona** con una **relación** varios a varios "es hermano de" también necesita una **entidad** de intersección. En este caso una buena idea como nombre para la intersección sería hermano. Esta **entidad** contiene dos campos, uno para cada persona de la **relación** hermano, en otras palabras, la clave **primaria** del primer hermano y la clave **primaria** del segundo hermano (vease figura 9.5)

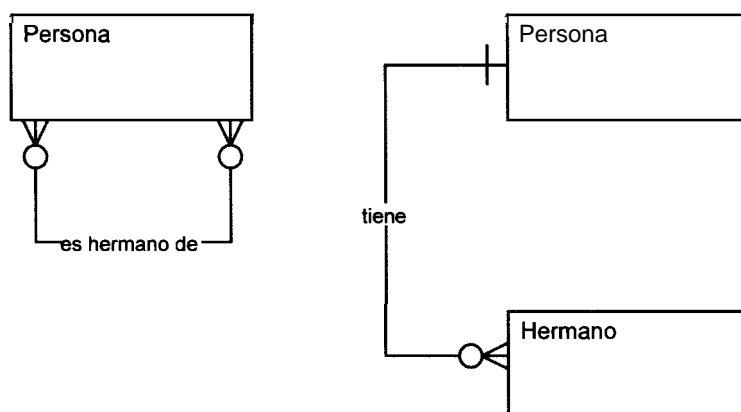


Figura 9.5. Creación de la entidad de intersección hermano

Diseño lógico y físico

Tras finalizar el diseño conceptual, **llega el momento** de convertirlo en diseño lógico y físico. Por **regla** general, en esta fase se selecciona el DBMS, en función

de los requisitos y la complejidad de las estructuras de datos. Estrictamente hablando, el diseño lógico y el diseño físico son dos fases diferentes que se suelen fundir en una. Se **solapan** porque la mayor parte de los DBMS actuales (incluyendo MySQL) hacen corresponder los registros lógicos y los registros físicos en el disco sobre una base de 1 a 1.

Cada entidad se convertirá en una tabla de base de datos y cada atributo se convertirá en un campo de esta tabla. Las claves externas se pueden crear si el DBMS las admite y el diseñador decide implementarlas. Si la relación es obligatoria, la clave externa debe definirse como NOT NULL y si fuera opcional, la clave externa puede admitir valores nulos. Por ejemplo, debido a la relación línea de factura-a-producto del ejemplo anterior, el campo del código de producto es una clave externa en la tabla de la línea de factura. Como la línea de factura contiene un producto, el campo debe definirse como NOT NULL. En la actualidad, las tablas InnoDB admiten restricciones de clave externa y las tablas MyISAM no admiten claves externas en la versión 4 pero sí en la versión 4.1. Un DBMS que admita claves externas utiliza cláusulas ON DELETE CASCADE y ON DELETE RESTRICT en sus definiciones. ON DELETE RESTRICT significa que los registros no se pueden eliminar a menos que todos los registros asociados a la clave externa se eliminen. En la relación línea de factura-a-producto del ejemplo anterior, el uso de la cláusula ON DELETE RESTRICT sobre la tabla de línea de factura significa que si el producto se elimina, la eliminación no tendrá lugar a menos que también se eliminen todas las líneas de factura asociadas al producto. De esta forma se evita la posibilidad de que exista una línea de factura que apunte a un producto que no exista. ON DELETE CASCADE logra un efecto similar pero de manera más automática (y más peligrosa). Si la clave externa se declara con ON DELETE CASCADE, las líneas de factura asociadas se eliminarán automáticamente si se elimina el producto. La cláusula ON UPDATE CASCADE se parece a la cláusula ON DELETE CASCADE en que todas las referencias a la clave externa asociadas a una clave primaria se actualizan al actualizar esta última.

La normalización de las tablas es un paso importante durante el proceso de diseño de la base de datos. Este proceso contribuye a evitar la redundancia de los datos y a mejorar su integridad.

Los diseñadores de bases de datos noveles suelen cometer una serie de errores comunes. Si ha identificado las entidades y los atributos con atención y ha normalizado los datos, es probable que se eviten dichos errores. Sin embargo, los diseñadores que se precipitan durante el proceso de diseño a menudo terminan con grandes tablas de datos sin relacionar. Si aplica los consejos que se incluyen a continuación evitara parte de los errores más frecuentes.

- Mantenga los datos sin relacionar en tablas diferentes. La gente acostumbrada a utilizar hojas de cálculo a menudo comete este tipo de errores porque están acostumbrados a ver todos los datos en una tabla bidimensional. Las bases de datos relacionales son mucho más potentes.

- No almacene valores que pueda calcular. Digamos que esta interesado en tres numeros: A, B y el **producto** de A y B ($A * B$). No almacene el **producto**. Malgastara espacio y el calculo se puede realizar con facilidad si lo necesita. **Además**, dificultara el mantenimiento de la base de datos: si **modifica** A, necesitara tambien cambiar todos los **productos**. **¿Por que desperdiciar** recursos de su base de datos en algo que puede calcular cuando lo necesite?
- **¿Responde** su diseio a todas las condiciones analizadas? **Al** crear un **diagrama** de relacion-entidad a toda prisa, podemos olvidarnos de incluir una condicion. Los diagramas de entidad-relacion suelen resultar mejores para conseguir que los **accionistas detecten** una **regla mal** formulada que una que no se haya incluido. La **lógica** de negocios es tan importante como la **lógica** de las bases de datos y es mas probable que se pase por alto. Por ejemplo, resulta sencillo detectar que no se puede tener una venta sin un cliente asociado, **pero ¿ha** incorporado ya la **regla** que no **permite** aprobar a un cliente por una venta inferior a 500 dolares si no ha sido recomendado por otro cliente?
- **¿Los** atributos, que estan a **punto** de convertirse en los nombres de campo, estan bien elegidos? Los campos **deben** llevar asignados nombres que resulten claros. Por ejemplo, si utiliza f1 y f2 en lugar de **apellido** y **nombre**, el tiempo que ahorrara **al** escribir el nombre lo perdera si el campo esta correctamente escrito o en detectar los **errores producto** de la confusion de f1 con el nombre y f2 con el apellido. De **manera** similar, intente evitar el uso de nombres iguales para identificar campos diferentes. **Si** se utiliza **código** para designar la clave **primaria** de seis tablas, estaremos complicando demasiado las cosas. En su lugar utilice terminos mas descriptivos, **como** `código_ventas` o `código_cliente`.
- No **cree** demasiadas relaciones. Practicamente todas las tablas de un **sistema** se pueden relacionar en un alarde de **imaginación** **pero** no hay **necesidad**. Por ejemplo, un **jugador** de **tenis** pertenece a un club de **tenis**. Un club de **tenis** pertenece a una region. Por lo **tanto**, los jugadores de **tenis** **pertene-** **cen** a una region, **pero** esta **relación** se puede derivar a traves del club de deporte por lo que no hay necesidad de agregar otra clave externa (a **menos** que desee obtener ventajas para determinados tipos de consultas). La **normalizacion** puede ayudarle a evitar este tipo de problemas (e incluso si esta intentando optimizar la base de datos para hacerla mas rapida, resulta mejor normalizar y, seguidamente, desnormalizar que no normalizar).
- **¿Ha** tenido en cuenta todas las relaciones? **¿Se** incluyen todas las entidades de su diagrama de entidad-relacion, como campos comunes, dentro de sus estructuras de tabla? **¿Ha** cubierto todas las relaciones? **¿Están** todas las relaciones varios a varios divididas en relaciones uno a varios, con una **entidad de intersección**?

- ¿**Ha** enumerado todas las restricciones? **Ejemplos** de restricciones son limitar un genero a hombre o **mujer**, establecer la edad maxima de **los** escolares en 20 años o obligar a utilizar el simbolo **@** y **al menos** un **punto** en las **direcciones** de correo electronico. No de nunca por **sentado** estos limites. En alguna fase, el sistema necesitara implementarlos, y puede olvidarse de hacerlo o tener que dar **marcha** atras para recopilar **los** datos si no lo hizo **al** principio.
- ¿**Está** planeando almacenar demasiados datos? ¿**Obliga** a sus clientes a **suministrar** el color de sus ojos, su pescado favorito y el nombre de sus abuelos para poder registrarse en un **boletín informativo**? En ocasiones **los accionistas** quieren demasiada informacion **sobre** sus clientes. Si el usuario queda **fuera** de la **organización**, es probable que no participe en el proceso de **diseño**, **pero** debe tenerse siempre presente. Considere **además** la dificultad y el tiempo que conlleva capturar todos esos datos. Si un operador de telefono necesita **tomar** toda esta informacion para poder realizar una venta, imagine lo lenta que resultara la transaccion. **Además**, es necesario tener en cuenta la incidencia de **los** datos en la velocidad de la base de datos. El acceso a las tablas de gran tamaño suelen resultar mas lento y el uso de campos BLOB, TEXT y VARCHAR puede dar **lugar** a la **fragmentación** de registros y tablas.
- ¿**Ha** combinado campos que deberían estar separados? La **combinación** de nombres y apellidos en un solo campo suele ser un error habitual. Mas **adelante** descubriera que el proceso de ordenacion de nombres **alfabéticamente** resulta complicado si se almacenado **como** John Ellis y Alfred Ntombela. Mantenga separados **los** datos de distinto tipo.
- ¿**Constan** todas las tablas de **al menos** una clave primaria? Es necesario tener una buena razon para no incluir una clave primaria. ¿**Cómo** vamos a identificar un registro unico rapidamente sin **ellas**? Considere el hecho de que **los** indices agilizan el acceso tremendamente y agregan poca carga si se **controla** su tamaño. Asi mismo, resulta **mejor** crear un nuevo campo para asignar una clave **primaria** que seleccionar campos existentes. Puede que el nombre y el apellido **resulten exclusivos** en el **conjunto** de datos actual **pero** que no lo **sean** siempre.
- Examine el resto de **los** indices. ¿**Qué** campos es probable que se utilicen en la **condición** de acceso a la tabla? Siempre puede crear otros **posteriormente** **al probar** el sistema, **pero** conviene agregar todos aquellos que **necesite** en esta fase.
- ¿**Están** bien asignadas sus claves externas? En una **relación** uno a varios, la clave externa aparece en la tabla "varios" y la clave **primaria** se asocia a la tabla "uno". Si se asignan de **manera** errónea, pueden surgir errores.
- ¿**Está** garantizada la integridad **referencial**? Las claves **externas** no deberían estar relacionadas con una clave primaria en otra tabla que ya no exista.

- ¿Se han cubierto todos los conjuntos de caracteres necesarios? Las letras del alfabeto alemán, por ejemplo, tienen un conjunto de caracteres expandido, y si la base de datos debe contemplar la inclusión de usuarios alemanes, tendremos que tener este hecho en cuenta. De manera similar, las fechas y los formatos de monedas deben considerarse atentamente si el sistema será de carácter internacional.
- ¿Basta con el sistema de seguridad implementado? Recuerde asignar los permisos mínimos que pueda. No permita que nadie pueda ver una tabla si no lo necesitan. El hecho de permitir que usuarios malintencionados vean los datos, aunque no puedan modificarlos, suele ser el paso previo a un ataque.

Fase 3: Implementación

La fase de implementación es en la que se instala el DBMS en los equipos necesarios, se optimiza la base de datos para obtener el mejor resultado en función de dichos equipos y de la plataforma de software que se utilice, se crea la base de datos y se carga con datos. Los datos iniciales pueden ser nuevos datos capturados directamente o datos ya existentes importados desde una base de datos MySQL u otro DBMS. En esta fase también se establece la seguridad de la base de datos y se concede a los diferentes usuarios identificados acceso en función de sus necesidades. Por último, también se inician los planes para realizar copias de seguridad.

Los siguientes pasos forman parte de la fase de implementación:

1. Instalación del DBMS
2. Ajuste de las variables en función del hardware, software y condiciones de uso
3. Creación de las bases de datos y las tablas
4. Carga de los datos
5. Establecimiento de los usuarios y los parámetros de seguridad
6. Implementación del régimen de volcados

Fase 4: Pruebas

Esta fase es en la que se prueba el rendimiento, la seguridad y la integridad de los datos. Por regla general estas operaciones se realizan en combinación con las aplicaciones que se han desarrollado. El rendimiento se prueba bajo diferentes condiciones de carga para ver cómo procesa la base de datos varias conexiones simultáneas o altos volúmenes de procesos de actualización y lectura. ¿Se gene-

ran los informes lo suficientemente rápido? Por ejemplo, una aplicación diseñada con tablas **MyISAM** puede resultar demasiado lenta porque se ha desestimado la incidencia de las actualizaciones. Puede que resulte necesario modificar el tipo de tabla a **InnoDB** en respuesta.

También debe probarse la integridad de los datos ya que la aplicación puede presentar fallos lógicos que den lugar a la pérdida de transacciones u otro tipo de inexactitudes. Además, es necesario probar la seguridad para garantizar que los usuarios disponen de acceso y sólo pueden cambiar los datos pertinentes.

Puede que resulte necesario modificar el diseño lógico o el diseño físico de la base de datos. Quizás se necesiten nuevos índices (que el responsable de las pruebas puede descubrir tras usar meticulosamente la instrucción **EXPLAIN** de **MySQL**, explicada en un capítulo anterior) o desnormalizar determinadas tablas por razones de rendimiento (vease un capítulo anterior).

El proceso de prueba y ajuste es de carácter interactivo, durante el cual se realizan varias pruebas y se implementan cambios.

A continuación se enumeran los pasos de la fase de prueba:

1. Comprobación de la seguridad
2. Comprobación la integridad de datos
3. Ajuste de los parámetros o modificar el diseño lógico o físico en respuesta a las pruebas

Fase 5: Puesta en marcha

Esta fase tiene lugar cuando se completan las pruebas y la base de datos queda lista para su funcionamiento diario. Los usuarios del sistema comienzan a utilizarlo, a cargar datos, a leer informes, etc. Es inevitable que surjan problemas. El diseñador necesita gestionar el ámbito de la base de datos con cuidado durante esta fase, ya que los usuarios esperan que se atiendan todos sus deseos. Los responsables de bases de datos mal diseñadas pueden verse obligados a extender el proyecto más allá del plazo de tiempo inicial previsto y la situación puede convertirse en desagradable si el ámbito no quedó definido y acordado claramente en un primer momento. Los propietarios del proyecto pueden sentirse perjudicados si no se da respuesta a sus necesidades y los diseñadores de la base de datos se sentirán cargados de trabajo y mal pagados. Incluso en el caso de que el ámbito se haya determinado correctamente, siempre surgirán nuevas necesidades. Éstas nos llevan a la siguiente fase.

Existen varias estrategias para realizar la primera implantación de un proyecto. El enfoque discreto suele ser el que mejor resultados ofrece. En este enfoque se utiliza un número pequeños de usuarios en la fase inicial para que utilicen el desarrollo, lo que facilita la solución de los problemas que puedan surgir. Las puestas en marcha a gran escala anunciadas a bombo y platillo suelen acabar sacando los colores a los propietarios ya que los usuarios encuentran siempre un

fallo imprevisto, que resulta aconsejable solucionar fuera de la luz pública. La puesta en **marcha** se puede realizar también de **manera** distribuida. Para ello, se selecciona una sucursal o una oficina **piloto** y cuando el sistema ha demostrado su estabilidad, se implementa en el resto de las sucursales.

A **continuación**, se enumeran los pasos de la fase de puesta en **marcha**:

1. Entrega de los mandos de la base de datos a los usuarios
2. **Realización** de todos los cambios finales necesarios en función de los problemas descubiertos por los usuarios

Fase 6: Mantenimiento

Esta fase incluye operaciones de mantenimiento general sobre la base de datos, como el mantenimiento de los índices, la **optimización** de las tablas, las operaciones de agregar y eliminar usuarios y la **modificación** de contraseñas, así como la **realización** de volcados y su restauración en caso de fallo. (En un capítulo posterior encontrará más **información** sobre el tema de mantenimiento.) En esta fase también surgirán otras necesidades que pueden dar lugar a la **creación** de nuevos campos o tablas.

A medida que el sistema y la organización van cambiando, la base de datos existente tendrá cada vez más problemas para dar respuesta a las necesidades de la organización. Por ejemplo, la empresa de comunicación puede fundirse con otras empresas extranjeras del sector, lo que exigirá la **integración** de una gran **cantidad** de fuentes de datos o los volúmenes y la plantilla puede aumentar (o reducirse) de **manera** espectacular. Finalmente, llegará un **momento**, ya sea 10 meses o 10 años después de su desarrollo, en el que será necesario sustituir la base de datos. En **concreto**, el mantenimiento de la base de datos existente comenzará a absorber más recursos cada día y los esfuerzos por crear un nuevo diseño equivaldrán a los empleados **actualmente** en funciones de mantenimiento. Este **punto** marca el comienzo del **final** de la base de datos y el nacimiento de nuevo proyecto en su fase de **análisis**. La fase de mantenimiento consta de los siguientes pasos:

1. Mantenimiento de los índices (por ejemplo, con el comando ANALYZE de MySQL)
2. Mantenimiento de las tablas (con el comando OPTIMIZE de MySQL)
3. Mantenimiento de los usuarios (con los comandos GRANT y REVOKE de MySQL)
4. Cambio de contraseñas
5. Volcados
6. Restauración de volcados
7. Cambio del diseño con el surgimiento de nuevas necesidades

Un ejemplo del mundo real: un sistema de seguimiento de publicaciones

A continuacion vamos a recorrer el proceso de diseiio con **ayuda** de un ejemplo desarrollado **paso a paso**. Poet's Circle es una editorial dedicada a la publicacion de poesia y antologias **poéticas** e interesada en desarrollar un sistema para realizar el seguimiento de **poetas**, **poemas**, antologias y **ventas**. En la siguiente **sección** se muestran los pasos seguidos desde el **análisis inicial** a **creación** de la base de datos final.

Fase 1 de la base de datos de Poet's Circle: Analisis

La siguiente informacion se ha **recabado** tras hablar con varios propietarios de la editorial: su **intención** es desarrollar un sistema de base de datos que se **encar-**gue de realizar el seguimiento de **los poetas** que **tienen** registrados, de **los poemas** que han escrito y de las publicaciones en las que aparecen, asi como las **ventas** generadas por estas publicaciones.

El diseiador planteo otras preguntas para obtener informacion mas detallada, como "**¿Qué** se considera como un poeta en lo que **al** sistema se **refiere**?

¿Se realiza el seguimiento de **los poetas** aunque no **hayan** escrito o publicado **poemas**? **¿Se** registran las publicaciones antes de que tengan ningun poema **asociado**? **¿Las** publicaciones se componen de un solo poema o de varios? **¿Se** guardan los detalles de los clientes potenciales? A continuacion se **resumen** las respuestas:

- Poet's Circle es una editorial que basa sus publicaciones en una comunidad activa de poesia basada en su sitio Web. Si una parte importante de la comunidad quiere que se publique un poema, Poet's Circle lo **publica**.
- Un poeta puede ser cualquiera que desee ser poeta, no necesariamente alguien que tenga un poema capturado en el sistema o que haya escrito un poema.
- Los **poemas** se puede enviar a traves de una interfaz Web, por correo electronico o en **papel**.
- Todos los **poemas** capturados estan escritos por un poeta asociado, cuyos detalles se encuentran ya incluidos en el sistema. No puede haber **poemas** remitidos y almacenados sin un **conjunto completo** de detalles sobre su autor.
- Una publicacion puede estar constituida por un unico poema, por una antologia de poesias o por un trabajo de **critica** literaria.

- Los clientes pueden registrarse a través de la interfaz Web y pueden realizar pedidos de publicaciones en dicho momento o indicar si desean recibir actualizaciones para realizar compras en un momento posterior.
- Las ventas de publicaciones se realizan a los clientes cuyos detalles estén almacenados en el sistema. No hay ventas anónimas.
- Una venta puede consistir en una sola publicación o en varias. Si hay varios clientes implicados en la venta, Poet's Circle lo considera como varias ventas. Cada venta se asigna a un cliente.
- No todas las publicaciones crean ventas, algunas son ediciones especiales y otras no venden ninguna copia.

Fase 2 de la base de datos de Poet's Circle: Diseño

En función de esta información, podemos iniciar el diseño lógico y deberíamos ser capaces de identificar las entidades iniciales:

- Poet
- Poem
- Publication
- Sale
- Customer

Poet's Circle no es una entidad ni siquiera una instancia de la entidad publisher. Solo si el sistema fuera desarrollado por varias editoriales entonces sería una entidad válida.

Ni website ni poetry community son entidades. Sólo hay un sitio Web y, además, un sitio Web es simplemente una forma de procesar los datos para completar la base de datos con datos. Así mismo, sólo hay una comunidad de poesía en lo que a este sistema se refiere y no hay mucho que almacenar sobre ella.

A continuación necesitamos determinar las relaciones que existen entre estas entidades. Se pueden identificar las siguientes:

- Un poeta puede escribir muchos poemas. El análisis identificó el hecho de que un poeta se puede almacenar en el sistema aunque no tenga poemas asociados. Los poemas se pueden capturar en un momento posterior y el poeta puede seguir siendo un poeta potencial. Por el contrario, aunque varios poetas puedan escribir un poema, el poema tiene que estar escrito al menos por uno.

- Una publicación puede contener muchos poemas (una antología) o uno solo. También puede no contener ninguno (una crítica del poema, por ejemplo).
Un poema puede aparecer o no aparecer en una publicación.
- Una venta debe incluir al menos una publicación pero también varias. Una publicación puede obtener ventas o no obtenerlas.
- Un cliente puede realizar varias compras o ninguna. Una venta se realiza para un solo cliente, y nada más que uno.

Puede identificar los siguientes atributos:

- Poet: first name, surname, address, telephone number
- Poem: poem title, poem contents
- Publication: title, price
- Sales: date, amount
- Customer: first name, surname, address, telephone number

En función de estas entidades y relaciones, se puede construir el diagrama de entidad-relación que se muestra en la figura 9.6.

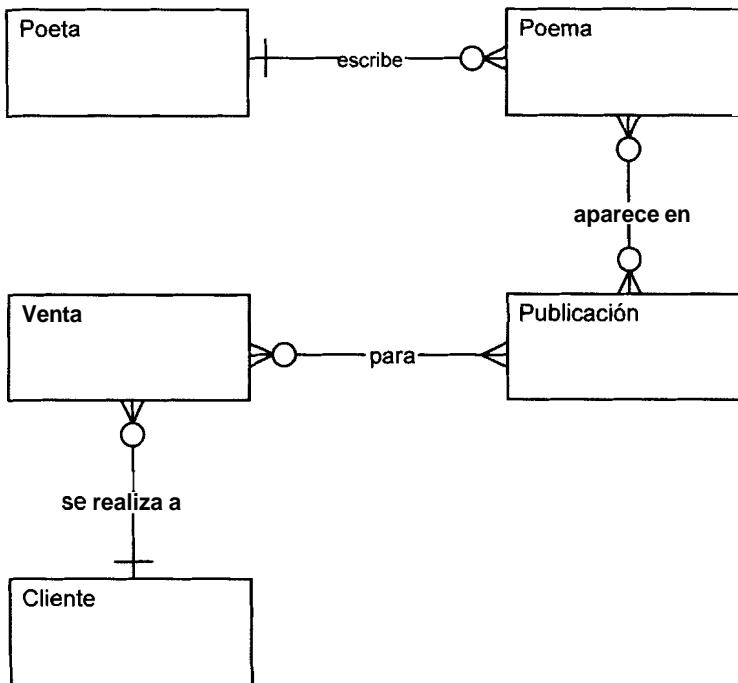


Figura 9.6. Diagrama de entidad-relación de Poet's Circle

Como se muestra en la figura 9.6, existen dos relaciones varios a varios. Éstas necesitan convertirse en relaciones uno a varios antes de poder implementarlas en un DBMS. El resultado se recoge en la figura 9.7, con las entidades de **intersección** *poem-publication* y *sale-publication*.

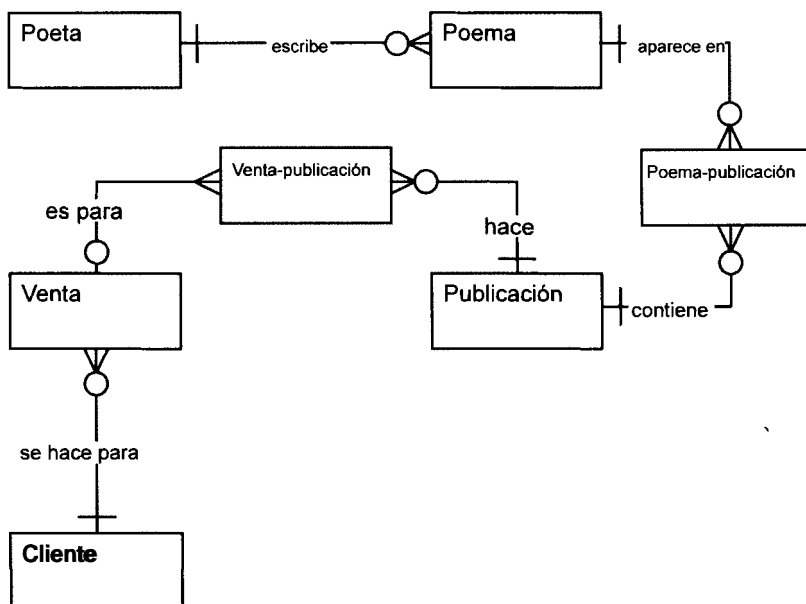


Figura 9.7. Diagrama de entidad-relación de Poet's Circle, con varias relaciones varios a varios eliminadas

A continuación, para comenzar el diseño lógico y físico, necesitamos agregar atributos que pueden crear la **relación** entre las entidades y especificar las claves primarias. Procedemos de la **forma** que suele resultar mas adecuada y creamos claves primarias nuevas y exclusivas. Las tablas 9.1 a 9.7 muestran las **estructuras** de las tablas para **cada** una de las entidades.

Tabla 9.1. Tabla Poet

Campo	Definición
poet code	clave primaria, entero
first name	caracter (30)
surname	caracter (40)
address	caracter(1 00)
postcode	caracter (20)
telephone number	caracter (30)

Tabla 9.2. Tabla Poem

Campo	Definición
poem code	clave primaria, entero
poem title	caracter (50)
poem contents	texto
poet code	clave externa, entero

Tabla 9.3. Tabla Poem-Publication

Campo	Definición
poem code	clave primaria combinada, clave externa, entero
publication code	clave primaria combinada, clave externa, entero

Tabla 9.4. Tabla Publication

Campo	Definición
publication code	clave primaria, entero
title	caracter (100)
price	numérico (5,2)

Tabla 9.5. Tabla Sale Publication

Campo	Definición
sale code	clave primaria combinada, clave externa, entero
publication code	clave primaria combinada, clave externa, entero

Tabla 9.6. Tabla Sale

Campo	Definición
sale code	clave primaria, entero
date	fecha
amount	numérica (10.2)
customer code	clave externa, entero

Tabla 9.7. Tabla Customer

Campo	Definición
customer code	clave primaria, entero
first name	caracter (30)
surname	caracter (40)
address	caracter(1 00)
postcode	caracter (20)
telephone number	caracter (30)

Fase 2 de la base de datos Poet's Circle: Implementación

Con el diseiio terminado, ha llegado el momento de instalar MySQL y ejecutar las instrucciones CREATE, de la siguiente forma:

```
mysql> CREATE TABLE poet (poet-code INT NOT NULL, first-name
VARCHAR(30),
  surname VARCHAR(40), address VARCHAR(100), postcode
VARCHAR(20),
  telephone-number VARCHAR(30), PRIMARY KEY(poet-code));
Query OK, 0 rows affected (0.02 sec)
mysql> CREATE TABLE poem(poem_code INT NOT NULL, title
VARCHAR(50),
  contents TEXT, poet-code INT NOT NULL, PRIMARY KEY(poem_code),
  INDEX(poet_code), FOREIGN KEY(poet-code) REFERENCES
poem(poet-code))
  type=InnoDB;
Query OK, 0 rows affected (0.00 sec)
mysql> CREATE TABLE publication(publication_code INT NOT NULL,
title VARCHAR(100),price MEDIUMINT UNSIGNED,
  PRIMARY KEY(publication-code)) type=InnoDB;
Query OK, 0 rows affected (0.05 sec)
mysql> CREATE TABLE poem-publication(poem-code INT NOT NULL,
publication-code INT NOT NULL, PRIMARY KEY(poem_code,
publication-code), INDEX(poem-code), INDEX(publication-code),
FOREIGN KEY(poem-code) REFERENCES poem(poem-code),
FOREIGN KEY(publication-code) REFERENCES
publication(publication_code)) TYPE=InnoDB;
Query OK, 0 rows affected (0.09 sec)
mysql> CREATE TABLE sales_publication(sales_code INT NOT NULL,
publication-code INT NOT NULL,PRIMARY KEY(sales_code,
publication-code)) TYPE =InnoDB;
Query OK, 0 rows affected (0.07 sec)
mysql> CREATE TABLE customer(customer_code INT NOT NULL,
first_name
```



```

    VARCHAR(30), surname VARCHAR(40), address VARCHAR(100),
postcode
    VARCHAR(20), telephone_number VARCHAR(30), PRIMARY
KEY(customer-code))
    TYPE=InnoDB;
Query OK, 0 rows affected (0.06 sec)
mysql> CREATE TABLE sale(sale-code INT NOT NULL, sale-date
DATE,
    amount INT UNSIGNED, customer-code INT NOT NULL, PRIMARY
KEY(sale-code), INDEX(customer-code), FOREIGN
KEY(customer-code)
    REFERENCES customer(customer-code)) TYPE = InnoDB;
Query OK, 0 rows affected (0.08 sec)

```

Fase 4 a 6 de la base de datos Poet's Circle: Prueba, puesta en marcha y mantenimiento

Una vez que la base de datos queda lista y que los programas de la aplicación se han presentado públicamente, ha llegado el momento de empezar con las pruebas. Mientras el resto de las fases de ciclo de vida de una base de datos pueden tener lugar de manera razonablemente independiente de los procesos de desarrollo de sistemas, parte de la fase de pruebas consiste en comprobar que todos los componentes del sistema funcionan conjuntamente.

Las pruebas de carga pueden indicar que MySQL no está todavía preparado para procesar las 600 conexiones simultáneas que se esperan y resulta necesario modificar el archivo de configuración. Otras pruebas pueden indicar que en determinadas circunstancias, se reciben errores de clave duplicada debido a que el mecanismo de bloqueo no se implementa de forma uniforme y que la aplicación no procesa correctamente los bloqueos. La aplicación necesita arreglos. Se deben probar además los volcados de seguridad, así como la posibilidad de realizar una operación correctamente a partir de un volcado reduciendo al mínimo el tiempo de inactividad del sistema.

ADVERTENCIA: La fase de pruebas es una de las que más fases más vitales del desarrollo de una base de datos y la que más se suele descuidar. Se puede tildar de incompetente al diseñador o coordinador que no prevea el tiempo necesario para esta fase. Independientemente del tamaño de su desarrollo, asegúrese de asignar tiempo suficiente para realizar pruebas completas así como para solucionar los inevitables fallos.

Tras completar la fase de pruebas, el sistema se puede poner en marcha públicamente. Nos decantamos por la puesta en marcha discreta en la que sólo se concede acceso a un grupo de poetas al sitio Web para que carguen sus poemas. Durante esta fase descubrimos otros problemas: una serie de navegadores presentan incompatibilidades que dan lugar al envío de poemas indescifrables. En sentido estricto

este problema no entra en el dominio del programador de bases de datos, **pero** es el **tipo de situación** que las pruebas revelaran una vez que todos los elementos del sistema **funcionen** conjuntamente. Insistimos en que los usuarios utilicen navegadores que puedan representar las **páginas** correctamente y que se prohíba el uso de los navegadores para operaciones de carga que no se ajusten a estos estandares.

Poco despues, el sistema echa a rodar completamente. La tarea de mantenimiento, sin embargo, es interminable y con la **gran cantidad** de actualizaciones y eliminaciones que se realizan, la base de datos tiende a fragmentarse. El **administrador** ejecuta instrucciones OPTIMIZE de **manera** regular y, por supuesto, un **fallo** inevitable del disco obliga a realizar una sesion nocturna de restauracion y a expresar un **enorme** agradecimiento por la facilidad de uso de `mysqldump`.

Control de simultaneidad mediante transacciones

Las peticiones a bases de datos **tienen** lugar de **forma** lineal, una detras de otra. Cuando varios usuarios acceden a una base de datos o uno dispone de un **conjunto** relacionado de peticiones que ejecutar, resulta importante garantizar la coherencia de los resultados. Para **ello**, se utilizan las transacciones, que son grupos de peticiones a bases de datos que se procesan de **manera** conjunta. Dicho de otra **manera**, son unidades lógicas de trabajo.

Atomicidad

La atomicidad significa que debe completarse toda la transaccion. De lo contrario, se anulara toda la transaccion. De esta **forma** se garantiza que la base de datos nunca incluire transacciones completadas de **manera** parcial, lo que mermara la integridad de los datos. Si extraemos dinero de una cuenta bancaria, por ejemplo, **pero** falla la segunda **peticion** y el sistema no logra colocar el dinero en otra cuenta bancaria, **ambas** peticiones fallaran. El dinero no puede perderse sin mas ni se puede extraer de una cuenta si ir a parar a otra.

Coherencia

La coherencia **hace** referencia al estado en el que se encuentran los datos cuando **tienen** lugar determinadas condiciones. Por ejemplo, una **regla** puede ser que **cada** factura este asociada con un cliente de la tabla de clientes. Estas reglas se pueden incumplir durante el **curso** de una transaccion si, por ejemplo, la factura se inserta sin un cliente asociado, que se agregara en una fase posterior de la transaccion. Estas violaciones temporales no **resultan** visibles desde fuera de la transaccion y se resolveran siempre antes de que la transaccion se complete.

Aislamiento

El aislamiento significa que todos los datos utilizados durante el procesamiento de una transacción no pueden ser utilizados por otra transacción hasta que no se haya completado la **primera**. Por ejemplo, si dos personas depositan 100 dólares en una cuenta con un saldo de 900 dólares, la **primera** transacción debe agregar 100 dólares a 900 dólares y la segunda debe agregar 100 dólares a 1.000 dólares. Si la segunda transacción lee 900 dólares antes de que se complete la **primera**, **ambas** transacciones parecerán haberse llevado a **cabo** satisfactoriamente, **pero** han desaparecido 100 dólares. La segunda transacción debe esperar hasta que pueda acceder a los datos en solitario.

Durabilidad

La durabilidad **hace** referencia al hecho de que una vez que se han confirmado los datos de una transacción, sus efectos permanecerán, incluso tras un **fallo** del sistema. Mientras una transacción este en proceso, los efectos son permanentes. Si la base de datos dejara de **funcionar**, los volcados deberían restaurarla a un estado coherente anterior **al** inicio de la transacción. Ninguna de las acciones que efectúe una transacción debería modificar este hecho.

Resumen

Los buenos diseños de bases de datos garantizan un sistema duradero y eficiente. La inversión de tiempo en el proceso de diseño evita la mayor parte de los **errores** más comunes que aquejan a muchas bases de datos actuales.

El ciclo de vida de las bases de datos se puede definir de muchas **formas**, pero siempre se reduce a los mismos pasos principales. En primer lugar esta la fase de **análisis** que es en la que se recoge la **información** y se **examina** el sistema **existente** para identificar los problemas actuales, posibles soluciones, etc. **A** continuación viene la fase de diseño en la que se perfila atentamente el nuevo sistema, **primero** de **manera** conceptual para su **presentación** a los accionistas y, después, **lógica** y físicamente para su implementación.

Seguidamente viene la fase de la implementación en la que se implanta la base de datos, antes de que la fase de pruebas saque a la luz **los** posibles problemas. Tras la fase de pruebas, el sistema se pone en **marcha** para su uso diario y da comienzo casi de **forma** inmediata la fase de mantenimiento. **A** medida que entran peticiones de **modificación**, resulta necesario realizar optimizaciones y volcados de **forma** periódica.

Finalmente, cuando las operaciones de mantenimiento comienzan a hacerse demasiado intensivas, se inicia un nuevo ciclo en el desarrollo de una base de datos para sustituir **al** sistema obsoleto.

Las transacciones garantizan la coherencia de la base de datos a lo largo de toda su existencia. Son cuatro los principios en los que se apoyan. La atomicidad establece que todas las peticiones de una transacción se realicen satisfactoriamente o se **cancelen** en su conjunto. La coherencia garantiza que las bases de datos devuelvan siempre un estado coherente entre transacciones. El aislamiento **garantiza** que todas las peticiones procedentes de una transacción no se completen antes de que se permita el procesamiento de la siguiente transacción que afecta a los mismos datos. Y la durabilidad mantiene la coherencia de la base de datos incluso en caso de fallo.

Parte III

Administración

de MySQL

Administración básica

Aunque **MySQL** es un aplicación sencilla de mantener y administrar, no se vale por si misma. En este capítulo se analizan las **tareas** administrativas **básicas**, **parte** de las cuales se examinarán más detenidamente en los siguientes capítulos.

En concreto, aprenderemos a iniciar y a detener el servidor, de **manera** manual y automática.

También veremos algunas de las herramientas indispensables para un administrador de **MySQL**, aprenderemos a utilizar los registros y a configurar **MySQL**.

En este capítulo se abordan los siguientes temas:

- Utilidades de **MySQL**
- Inicio y cierre de **MySQL**
- Inicio automático de mysqld al arrancar el sistema
- Configuración de **MySQL**
- Operaciones de registro
- Alternancia de registros
- Optimización, comprobación, análisis y reparación de tablas

Uso de MySQL como administrador

Como administrador, necesitara saber mucho mas sobre el funcionamiento de MySQL que si estuviera ejecutando consultas simplemente. **Deberá familiarizarse** con las utilidades que incorpora la distribucion de MySQL, asi como con su **forma** de configurar la aplicacion. A **continuación** se describen las principales utilidades dirigidas a **los administradores**:

- **mysqldadmin**: Se trata probablemente de la utilidad administrativa mas util. **Permite** crear y eliminar bases de datos, detener el servidor, visualizar variables de servidor, visualizar y anular procesos de MySQL, establecer contraseñas y vaciar archivos de registros, entre otras cosas.
- **mysqld**: No se trata de una utilidad en realidad, ya que es el servidor de MySQL. Es probable que se tope con terminos como *las variables de mysqld*, que no son otra cosa que variables de servidor.
- **mysqlimport**: **Importa** archivos de texto a tablas de base de datos.
- **mysqlcheck**: Comprueba, analiza y **repara** bases de datos.
- **mysqlhotcopy**: Una secuencia de comandos de **Perl** que **hace** volcados de tablas de base de datos rapidamente.
- **myisampack**: Comprime tablas MyISAM.

La **primera** pregunta que **deberíamos** plantearnos sobre un nuevo sistema es **bastante** elemental: ¿**dónde** se almacenan los datos? La ubicacion predeterminada suele **ser** `/usr/local/var` para una distribucion **fuentes** de Unix, `/usr/local/mysql/data` para una distribucion **binaria** de Unix y `C:\mysql\data` en Windows. **Al** compilar MySQL, se selecciona un directorio de datos, **pero** este se puede establecer en cualquier otra ubicacion especificando el nuevo directorio en el archivo de configuracion. En una **sección** posterior, analizaremos los pormenores del archivo de configuracion, Por el **momento**, basta con saber que suele llamarse **my.cnf** en Unix o **my.ini** en Windows y que contiene los datos de configuracion de MySQL. Si desea que MySQL use otra ubicacion, utilice una secuencia parecida a la siguiente:

```
datadir = C:/mysqldata
```

Para determinar los **directorios** de datos utilizados en una instalacion existente, puede utilizar la opcion de variables de **mysqldadmin**. Los usuarios noveles de MySQL pueden sentirse un poco abrumados por los resultados que devuelve esta opcion debido a la gran **cantidad** de variables incluidas (cuyo numero parece aumentar con **cada** nueva **actualización** de MySQL), aunque se enumeren alfabeticamente:

```
% mysqldadmin -uroot -pg00r002b variables;  
+-----+  
| Variable-name      | Value  
|
```

```

+-----+
| back-log          | 50
|
| basedir           | /usr/local/mysql-max-4.0.1-alpha-pc-
|                   | linux-gnu-i686/
|
| bdb_cache_size   | 8388600
|
| bdb_log_buffer_size | 32768
|
| bdb_home          | /usr/local/mysql/data/
|
| bdb_max_lock      | 10000
|
| bdb_logdir        |
|
| bdb_shared_data   | OFF
|
| bdb_tmpdir        | /tmp/
|
| bdb_version       | Sleepycat Software: Berkeley DB
3 2.9a:           |
|                   | (December 23, 2001)
|
| binlog-cache-size | 32768
|
| character-set      | latin1
|
| character-sets     | latin1 big5 czech euc_kr gb2312 gbk
|                   | latin1_de sjis tis620 ujis dec8 dos
|                   | german1 hp8 koi8_ru latin2 swe7 usa7
|                   | cp1251 danish hebrew win1251 estonia
|                   | hungarian koi8_ukr win1251ukr greek
|                   | win1250 croat cp1257 latin5
|
| concurrent-insert | ON
|
| connect-timeout   | 5
|
| datadir           | /usr/local/mysql/data/
|

```

Tambien puede seleccionar una variable concreta utilizando gred (Unix) o find (Windows), de la siguiente forma (primero en Unix y luego en Windows):

```

% mysqladmin -uroot -pg00r002b variables | grep 'datadir'
| datadir           | /usr/local/mysql/data/

```



```
C:\mysql\bin>mysqladmin variables | find "datadir"
| datadir | C:\mysql\data\
```

El directorio de datos en este caso es `/usr/local/mysql/data`, el predeterminado para la mayor parte de las instalaciones binarias de Unix, y `C:\mysql\data` para el ejemplo de Windows.

El directorio de datos suele contener los archivos de registro (se colocan en esta ubicacion de manera predeterminada, aunque puede seleccionar otra) asi como los datos. En el caso de las tablas MyISAM (las predeterminadas), cada base de datos consta de su propio directorio y dentro de dicho directorio cada tabla incluye tres archivos: un archivo `.MYD` para los datos, un archivo `.MYI` para los indices y un archivo `.frm` para la definicion. Las tablas BDB tambien se almacenan en el mismo directorio, pero se componen de un archivo `.db` y de un archivo de definicion `.frm`. Las tablas InnoDB incluyen su archivo de definicion `.frm` en el directorio de la base de datos, pero los datos se almacenan en un nivel superior, el mismo que el utilizado por los directorios de la base de datos.

Los clientes de MySQL pueden acceder al servidor de tres formas:

- **Sockets de Unix:** Éstos se utilizan en equipos Unix al establecer una conexión a un servidor en el mismo equipo. El archivo de socket se coloca en la ubicacion predeterminada (por regla general, `/tmp/mysql.sock` o `/var/lib/mysql.sock`), a menos que se especifique otra cosa.
- **Canalizaciones con nombre:** Éstas se utilizan en equipos con Windows NT/2000/XP en los que se usa la opción `-enable-named-pipe` con un ejecutable que permite canalizaciones con nombre (`mysqld-max-nt` o `mysqld-nt`).
- **TCP/IP a través de un puerto:** Éste es el método más lento pero la única forma de establecer una conexión a un servidor con Windows 95/98/Me o de establecer una conexión remota a un equipo Unix.

Como iniciar y cerrar MySQL

MySQL se ejecuta en la mayor parte de los sistemas operativos, pero los procedimientos utilizados varian de uno a otro. A continuación, se incluyen secciones separadas para Windows y Unix.

Como iniciar y cerrar MySQL en Unix

La forma mas elemental de iniciar MySQL consiste en ejecutar `mysqld` directamente. Sin embargo, la gran mayoría de las distribuciones incluyen una secuen-

cia de comandos de iniciación llamado `mysqld_safe` (en las versiones más antiguas se denomina `safe-mysld`), que debería utilizarse en lugar de abrir MySQL manualmente. Esta secuencia de comandos incluye varias funciones adicionales de seguridad, como el registro de errores y la reiniciación automática del servidor en caso de error. Para iniciar MySQL, regístrese como usuario raíz y ejecute el siguiente comando desde el directorio en el que tenga instalado MySQL (por regla general, `/usr/local/mysql`):

```
% bin/mysqld_safe -user=mysql &
```

Fíjese en que el usuario está establecido como `mysql`. Es importante ejecutar MySQL como usuario `mysql` para evitar problemas de **permiso** y seguridad.

ADVERTENCIA: Si no está familiarizado con el sistema, en especial si no ha realizado su instalación, hable con su administrador de sistemas antes de intentar iniciar el servidor de esta forma. Un servidor se puede iniciar de varias formas e intentar hacerlo de la errónea puede generar problemas. La mayor parte de los sistemas de producción utilizan una secuencia de comandos para iniciar MySQL al arrancar el sistema y es aconsejable utilizarla si existe.

Algunas distribuciones incorporan una secuencia de comandos llamada `mysql.server`, que puede incluso que se instale automáticamente (en ocasiones recibe el nombre de `mysql` simplemente). Ésta suele ubicarse en un directorio en el que los procesos se activan automáticamente al iniciar el sistema. Si este fuera el caso, deberíamos utilizar la mencionada secuencia de comandos para iniciar el servidor. `mysql.server` toma opciones de inicio y cierre. A continuación se muestra una sesión de inicio común en un sistema Red Hat Linux:

```
% /etc/rc.d/init.d/mysql start
% Starting mysqld daemon with databases from /usr/local/mysql/
data
```

En FreeBSD, el archivo se coloca en `/usr/local/etc/rc.d`, en cuyo caso se utilizaría la siguiente secuencia:

```
% /usr/local/etc/rc.d/mysql.sh start
```

Para cerrar el sistema, puede utilizar `mysqladmin`:

```
% mysqladmin shutdown -uroot -p
Enter password:
020706 16:56:02 mysqld ended
```

o la opción equivalente desde la secuencia de comandos de inicio, como la siguiente:

```
% /etc/rc.d/init.d/mysql stop
```

```

Killing mysqld with pid 2985
Wait for mysqld to exit\c
.\c
.\c
.\c
.\c
.\c
.\c
.\c
020706 17:07:49 mysqld ended

```

Como iniciar MySQL automaticamente al arrancar el sistema

En sistemas de **producción**, si no es un obseso del control, es probable que desee que MySQL se ejecute durante el proceso de arranque del sistema. Para **ello**, si no se ha implementado automaticamente, necesitara saber como inicia o detiene los procesos su version de Unix **al** arrancar y apagar el sistema. Esta **operación** puede variar ostensiblemente de un sistema a otro. Si no esta seguro del proceso, especialmente si no **instaló** MySQL, lo mejor es dirigirse a su **administrador** de sistemas. **Éste** deberia conocer los detalles de su sistema mejor que de lo podriamos explicar en este libro.

El siguiente ejemplo se ha **tomado** de un sistema con Red Hat Linux:

```
% cp /usr/share/mysql/mysql.server /etc/rc.d/init.d
```

Esta linea copia la secuencia de comandos `mysql.server` en el directorio de inicializacion.

Las siguientes dos **lineas** garantizan que MySQL se inicia **al** arrancar el sistema, que alcanza el **modo** multiusuario (nivel 3 de ejecucion) y que se cierra con el sistema (nivel 0 de ejecucion). Se crea un **vínculo** desde el nivel pertinente a la secuencia de comandos de `mysql.server`:

```
% ln -s /etc/rc.d/init.d/mysql.server /etc/rc.d/rc3.d/S99mysql
% ln -s /etc/rc.d/init.d/mysql.server /etc/rc.d/rc0.d/S01mysql
```

El siguiente ejemplo se ha **tomado** de una version reciente de **FreeBSD**, en la que **sólo** necesitamos copiar las secuencias de comandos de inicio en el directorio `rc.d` y asignarles la extension `.sh`:

```
% cp /usr/local/mysql/support-files/mysql.server /usr/local/etc/rc.d/mysql.sh
```

Asegurese de que su secuencia de comandos es ejecutable y que no resulta accesible sin autorizacion con ayuda de la siguiente secuencia:

```
% chmod 700 /usr/local/etc/rc.d/mysql.sh
```

Algunos sistemas antiguos pueden utilizar `/etc/rc.local` para iniciar secuencias de comandos, en cuyo **caso** deberiamos agregar una **instrucción** como la siguiente **al** archivo:

```
/bin/sh -c 'cd /usr/local/mysql ; ./bin/safe_mysqld -user=mysql &'
```

Como evitar problemas comunes al iniciar MySQL en Unix

Los problemas mas comunes estan relacionados con los permisos. Si no se ha registrado como usuario raiz e intenta iniciar MySQL, obtendra un error como el siguiente:

```
% /usr/local/mysql/bin/mysqld_safe -user=mysql &
% The file /usr/local/mysql/libexec/mysqld doesn't exist or is
not executable
Please do a cd to the mysql installation directory and restart
this script from there as follows:
./bin/mysqld_safe.
```

Para solucionarlo, inicie la sesion como usuario raiz:

```
% su
Password:
% /usr/local/mysql/bin/mysqld_safe -user=mysql &
[1] 24756
% Starting mysqld daemon with databases from
  /usr/local/mysql-max-4.0.2-alpha-unknown-freebsdelf4.6-i386/
data
```

Para resolver otros problemas, utilice el registro de errores de MySQL. En una sección posterior analizaremos el registro de errores.

Como iniciar y cerrar MySQL en Windows

La distribución de MySQL para Windows incluye varios archivos ejecutables (vease la tabla 10.1). La ejecucion de uno u otro dependera de como vayamos a utilizar MySQL.

Tabla 10.1. Los ejecutables de MySQL

Ejecutables	Descripción
mysqld	Un binario que incorpora funciones de depuracion, comprobacion de asignación de memoria automáticamente, tablas transaccionales (InnoDB y BDB) y vínculos simbólicos.
mysqld-opt	Un binario optimizado que no incorpora compatibilidad para tablas binarias (InnoDB o BDB).
mysqld-nt	Un binario optimizado que incorpora compatibilidad para canalizaciones con nombre (para su uso con NT/2000/XP). Puede ejecutarse en sistemas 95/98/Me, pero no se crearan canalizaciones con nombre, ya que estos sistemas no las admiten.

Ejecutables	Descripción
mysqld-max	Un binario optimizado que admite tablas transaccionales (InnoDB y BDB) así como canalizaciones con nombre al ejecutar NT/2000/XP y vínculos simbólicos.

Para iniciar MySQL, basta con ejecutar el archivo deseado, por ejemplo:

```
C:\> c:\mysql\bin\mysqld-max
020706 18:53:45 InnoDB: Started
```

Sustituya `\mysql\bin\` por el directorio en el que haya instalado MySQL, si fuera diferente. Muchos usuarios de Windows prefieren utilizar la siguiente secuencia:

```
C:\> c:\program\mysql\bin\mysqld-max
```

También puede usar la utilidad `winmysqladmin` que se incluye en las distribuciones de Windows para iniciar MySQL.

NOTA: Si utiliza Windows 95, asegúrese de comprobar que tiene instalado Winsock 2. Las versiones más antiguas de Windows 95 no incluyen este componente lo que impedirá que MySQL pueda ejecutarse. Para descargarlo diríjase a www.microsoft.com.

Como iniciar MySQL automáticamente

En Windows 95/98/Me, cree un acceso directo al archivo ejecutable `winmysqladmin` dentro de la carpeta Inicio. Este archivo se almacena en el mismo lugar que otros ejecutables, en otras palabras, dentro de `c:\mysql\bin` de manera predeterminada.

Asegurese de que su archivo `my.ini` contiene el ejecutable deseado. Puede ejecutar `mysqld-max` manualmente y, a continuación, hacerlo automáticamente con `winmysqladmin`.

Ahora bien, si su archivo contiene la siguiente secuencia:

```
[WinMySQLAdmin]
Server=C:/PROGRAM FILES/MYSQL/bin/mysqld-opt.exe
```

no podrá utilizar las funciones transaccionales que cabría esperar. Puede editar el archivo `my.ini` manualmente o utilizar `winmysqladmin` para modificarlo, seleccionando `my.ini setup` para cambiar el archivo `mysqld` (vease figura 10.1).

Con NT/2000/XP, instale MySQL como servicio de la siguiente forma:

```
C:\> c:\mysql\bin\mysqld-max-nt -install
```

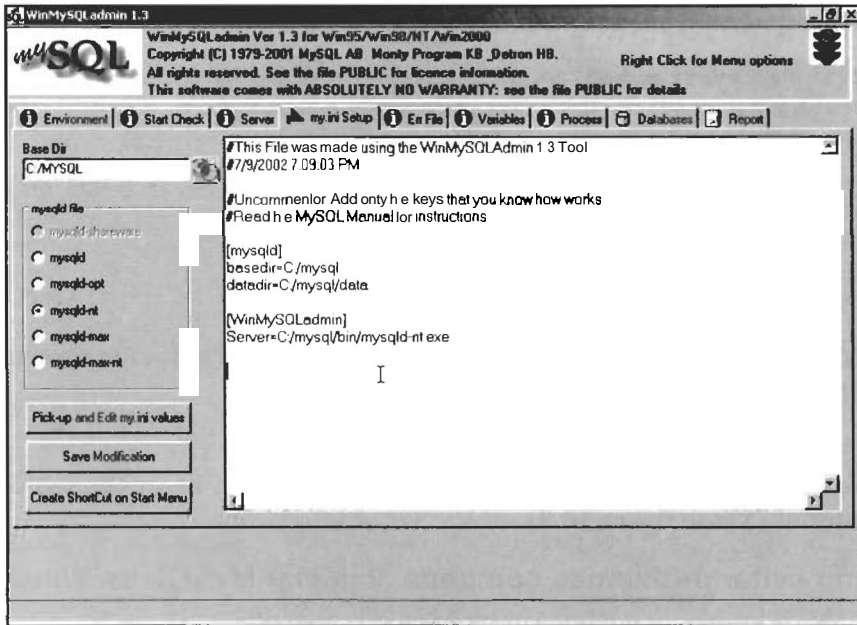


Figura 10.1. Como usar winmysqladmin para actualizar el archivo de configuración my.ini

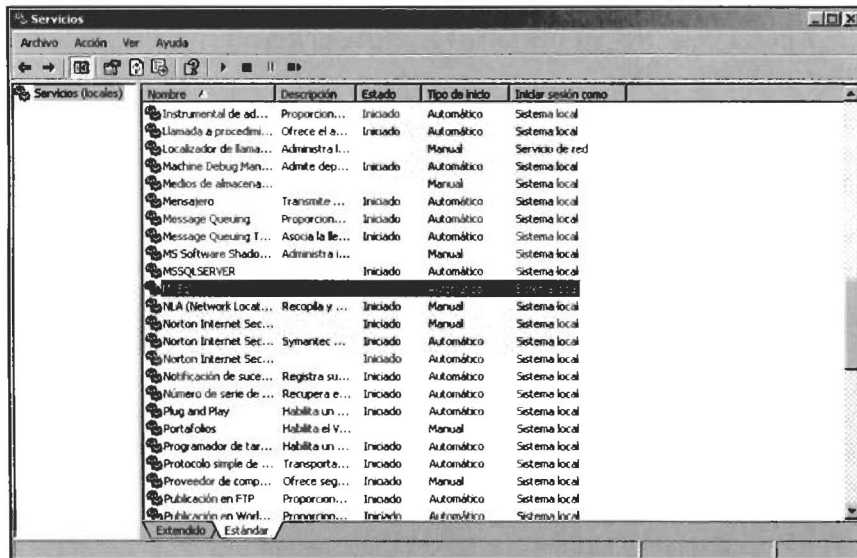


Figura 10.2. Cómo iniciar MySQL como un servicio en Windows 2000

Si no desea que MySQL se inicie automáticamente, pero quiere seguir utilizándolo como servicio, ejecute el mismo comando con la opción manual :

```
c:\mysql\bin> mysqld-max-nt -install-manual
```

A continuación puede iniciar el servicio con la siguiente secuencia:

```
C:\> net start mysql
The MySQL service is starting.
The MySQL service was started successfully.
```

Y detenerlo con la instrucción `mysqladmin shutdown` o esta otra:

```
C:\> net stop mysql
The MySQL service is stopping.....
The MySQL service was stopped successfully.
```

Para eliminar el servicio, ejecute `mysqld` con la opción de eliminación, como se indica a continuación:

```
C:\> c:\mysql\bin\mysqld-max-nt -remove
```

También puede utilizar el panel de control Servicios y hacer clic sobre Iniciar o Detener (Véase figura 10.2).

Como evitar problemas comunes al iniciar MySQL en Windows

Un problema común al iniciar MySQL en Windows tiene lugar cuando se instala en un directorio no predeterminado (como `c:\Archivos de programa\MySQL\bin`). En ocasiones las ubicaciones no se reflejan correctamente en el archivo de configuración `my.ini`. Por ejemplo, si instaló MySQL en `c:\Archivos de programa\MySQL`, entonces `my.ini` debería contener algo así como:

```
[mysqld]
basedir=C:/Program Files/mysql
datadir=C:/Program Files/data

[WinMySQLAdmin]
Server=C:/Program Files/mysql/bin/mysqld-max-nt.exe
```

NOTA: Los nombres de ruta de Windows se especifican con barras inclinadas no con las barras invertidas que suelen utilizarse en Windows, en los archivos de opción. Si desea utilizar barras invertidas, tendrá que utilizar secuencias de conversión de escape (otra barra invertida) ya que la barra invertida es un carácter especial de MySQL, por ejemplo: `Server=C:\\Archivos de programa\\mysql\\bin\\mysqld-opt.exe`.

Puede que haya utilizado espacios en su nombre de archivo y haya probado con esta secuencia:

```
C:/program files/mysql
```

en lugar de:

```
C:/progra~1/mysql
```

Si se utiliza winmysqladmin para iniciar el sistema puede que se cree un archivo `my.ini` que interfiera con una configuración existente. **Pruebe** a quitar el archivo `my.ini` recién creado (restaurando el original si fuera necesario). Si el problema no queda solucionado, pruebe a **examinar** el archivo de registro para comprobar si existe una razón obvia. El registro de error se incluye en `C:\MySQL\data\mysql.err` de manera predeterminada. También puede iniciar MySQL en modo independiente (`mysqld - standalone`), que puede generar resultados más útiles o, como último recurso, utilizar el modo de depuración, que generará un archivo de rastreo (por lo general en `C:\mysqld.trace`) que puede resultar de alguna utilidad.

configuración de MySQL

Para que MySQL se ejecute sin problemas de la forma deseada, necesitará aplicar determinados parámetros de configuración, como seleccionar **InnoDB** como tipo de tabla predeterminada o mostrar mensajes de error en un idioma determinado. Puede establecer la mayor parte de las opciones de tres formas: desde la línea de comandos, desde un archivo de configuración o desde una variable de entorno predeterminada. El uso de la línea de comandos para establecer opciones resulta útil para funciones de prueba, pero no conviene utilizarla si se desea mantener dichas opciones durante un largo período de tiempo. Las variables de entorno no se utilizan casi nunca. El método más útil y habitual es recurrir a un archivo de configuración.

En **Unix**, el archivo de configuración de inicio se suele denominar `my.cnf` y se puede colocar en las ubicaciones que se muestran a continuación. MySQL lo lee de arriba abajo, de manera que las posiciones situadas más abajo llevarán asignada una prioridad más alta (vease tabla 10.2).

Tabla 10.2. Precedencia de los archivos de configuración en Unix

Archivo	Descripción
<code>/etc/my.cnf</code>	Opciones globales que se aplican a todos los servidores y usuarios. Si no está seguro de donde colocar un archivo de configuración, hagalo aquí.
<code>DIRECTORIO_DE_DATOS/my.cnf</code>	Opciones específicas del servidor que almacena sus datos en el directorio de datos especificado. Por regla general suele ser <code>usr/local/mysql/data</code> para binarios o <code>usr/local/var</code> para instalaciones fuente. Tenga presente que no tiene que coincidir necesariamente con la opción <code>-datadir</code> especificada para <code>mysqld</code> .
<code>defaults-extra-file</code>	Opciones específicas de utilidades de servidor o de clientes iniciadas con la opción de línea de comandos <code>defaults-extra-file=nombre de archivo</code> .

Archivo	Descripción
~/my.cnf	Opciones específicas del usuario.

En Windows, el archivo de configuración se suele llamar `my.ini` o `my.cnf`, según su ubicación (vease tabla 10.3).

Tabla 10.3. Precedencia de los archivos de configuración en Windows

Archivo	Descripción
C:\CARPETA_DE_SISTEMA_DE_WINDOWS\my.ini	Opciones globales que se aplican a todos los servidores y usuarios. La carpeta de sistema de Windows suele ser <code>C:\WINNT\System32</code> , <code>C:\WINNT</code> o <code>C:\WINDOWS</code> .
C:\my.cnf	Opciones globales que se aplican a todos los servidores y usuarios (se podría utilizar el archivo <code>my.ini</code> en su lugar).
C:\DIRECTORIO_DE_DATOS\my.cnf	Opciones específicas del servidor que se almacenan en el directorio de datos especificado (que por regla general suele ser <code>c:\mysql\data</code>).
<code>defaults-extra-file=nombre de archivo</code>	Opciones específicas de utilidades de servidor o de clientes iniciadas con la opción de línea de comandos <code>defaults-extra-file=nombre de archivo</code> .

En Windows, si la unidad C no es la unidad de arranque o se puede utilizar la utilidad `winmysqladmin`, **deberá** usar el archivo de configuración `my.ini` (situado en la **carpeta** del sistema de Windows).

NOTA: Windows no dispone de un archivo de configuración para las opciones específicas del usuario.

A continuación, se incluye un archivo de **configuración** de ejemplo:

```
# Las siguientes opciones se pasarán a todos los clientes MySQL
[client]
#password          = su_contraseña
port               = 3306
socket             = /tmp/mysql.sock

# El servidor MySQL
[mysqld]
port              = 3306
```

```

socket                = /tmp/mysql.sock
skip-locking
set-variable          = key_buffer=16M
set-variable          = max_allowed_packet=1M
set-variable          = table_cache=64
set-variable          = sort_buffer=512K
set-variable          = net_buffer_length=8K
set-variable          = myisam_sort_buffer_size=8M
#set-variable         = ft_min_word_length=3
log-bin
server-id             = 1

[mysqldump]
quick
set-variable          = max_allowed_packet=16M

[mysql]
no-auto-rehash
# Elimine el símbolo de comentario siguiente si no esta
# familiarizado con SQL
#safe-updates

[myisamchk]
set-variable          = key_buffer=20M
set-variable          = sort_buffer=20M
set-variable          = read_buffer=2M
set-variable          = write_buffer=2M

[mysqlhotcopy]
interactive-timeout

```

El símbolo # indica un comentario y los corchetes ([]) son marcadores de sección. Los términos incluidos dentro de los corchetes indican a que programa afectarán los parámetros que siguen. En este ejemplo, el parámetro `interactive-timeout` se aplicará únicamente al ejecutar el programa `mysqlhotcopy`. Las opciones establecidas en un **marcador** de sección se aplican a la sección establecida previamente hasta el siguiente **marcador** de sección.

En el ejemplo anterior, el primer puerto se aplica a los clientes y el segundo al servidor MySQL. Por **regla** general se utiliza el mismo puerto para ambos, **pero** no necesariamente (por ejemplo, si se están **ejecutando** varios servidores MySQL en el mismo equipo). Las opciones pueden ser de tres tipos:

- opción=valor (como `port=3306`)
- opción (como `log-bin`). Se trata de opciones booleanas que no se establecen si la opción no está presente (en este **caso** se utiliza el valor predeterminado) y se establecen si la opción está presente.
- `set-variable = variable = valor` (como `set-variable = write_buffer=2M`). Este **parámetro** permite establecer variables de servidor MySQL.

ADVERTENCIA: Los parámetros de configuración de ejemplo incluyen una opción de contraseña para los clientes sin carácter de comentario. Puede que resulte práctico establecer la conexión de esta forma, pero no es aconsejable hacerlo en la mayor parte de los casos por razones de seguridad, ya que cualquiera que pueda leer este archivo dispondrá de acceso a MySQL.

Los siguientes programas admiten archivos de opción: `myisamchk`, `myisampack`, `mysql`, `mysql.server`, `mysqladmin`, `mysqlcheck`, `mysqld`, `mysqld_safe`, `mysqldump`, `mysqlimport` y `mysqlshow`.

En general, prácticamente cualquier opción que se use con un programa de MySQL desde la línea de comandos también se puede establecer en un archivo de configuración.

Una parte importante del proceso de dominio de MySQL consiste en conseguir la configuración que se ajuste a nuestra situación. En una sección posterior, examinaremos el significado de las opciones de servidor y como configurarlas para obtener el máximo rendimiento de MySQL.

La mayor parte de las distribuciones de MySQL incorporan cuatro configuraciones de ejemplo:

- **MY-HUGE.CNF:** Para sistemas con más de 1GB de memoria dedicada a MySQL.
- **MY-LARGE.CNF:** Para sistemas con al menos 512MB de memoria dedicadas a MySQL.
- **MY-MEDIUM.CNF:** Para sistemas con al menos 32MB de memoria dedicadas completamente a MySQL o al menos 128MB en un equipo con varias funciones (como un servidor dual Web/base de datos).
- **MY-SMALL.CNF:** Para sistemas con menos de 64MB de memoria en los que MySQL no puede utilizar demasiados recursos.

Algunas distribuciones incluyen un solo ejemplo: `my-example.cnf`.

Examine los archivos para comprobar la última documentación; 512MB no se considerara como sistema "grande" durante mucho tiempo.

Conviene copiar el que más se aproxime a las necesidades del directorio en el que vaya a almacenarlo y, a continuación, realizar las modificaciones pertinentes.

TRUCO: Realice un volcado de su archivo de configuración. En caso de que el archivo original falle, podría perder bastante tiempo configurando de nuevo el servidor.

Registro

El análisis de los archivos de registros puede que no coincida con su idea de pasarlo bien un viernes por la noche, **pero** puede convertirse en una ayuda inestimable no sólo para identificar los problemas surgidos sino también para detectar situaciones que, si se dejaran sin examinar, podrían obligarle a perder más de un viernes por la noche. MySQL integra varios archivos de registro diferentes:

- **El archivo de errores:** Éste es el lugar al que dirigirse para buscar problemas relacionados con el inicio, la ejecución o la detención de MySQL.
- **El archivo de consultas:** Aquí se almacenan todas las instrucciones SQL que modifican datos.
- **El archivo de consulta lenta:** Aquí se almacenan todas las consultas cuya ejecución lleva más tiempo que el establecido en `long_query_time` o que no utilizaron ninguno de los índices.
- **El archivo de actualización:** Este archivo ha quedado obsoleto y debería sustituirse con el archivo de **actualización binaria** en todas las instancias. Almacena instrucciones SQL que modifican datos.
- **El archivo ISAM:** Registra todos los cambios realizados sobre tablas ISAM. Se utiliza únicamente para depurar código ISAM.

El archivo de errores

En Windows, el archivo de errores de MySQL se denomina `mysql.err` y en Unix `nombredehost.err` (por ejemplo, `text.mysqlhost.co.za.err`). Se ubica en el directorio de datos (por regla general, `C:\MySQL\data` en Windows, `/usr/local/mysql/data` para instalaciones binarias de Unix, `/usr/local/var` para instalaciones fuente de Unix o `/var/lib/mysql` para variantes de Red Hat).

Este archivo contiene información de inicio y cierre del sistema así como errores vitales que tengan lugar durante el proceso de ejecución. Registra si el servidor deja de funcionar y se reinicia automáticamente o si MySQL detecta que una tabla necesita comprobarse o repararse automáticamente. El registro también puede contener un rastreo de pila cuando MySQL deja de funcionar. A continuación se recoge un ejemplo de registro de errores:

```
010710 19:52:43  mysqld started
010710 19:52:43  Can't start server: Bind on TCP/IP port:
Address already in use
010710 19:52:43  Do you already have another mysqld server
running on port: 3306 ?
010710 19:52:43  Aborting
```

```

010710 19:52:43 /usr/local/mysql-3.23.39-pc-linux-gnu-i686/
bin/mysqld:
  Shutdown Complete

010710 19:52:43 mysqld ended

010710 19:55:23 mysqld started
/usr/local/mysql-3.23.39-pc-linux-gnu-i686/bin/mysqld: ready
for connections
010907 17:50:38 /usr/local/mysql-3.23.39-pc-linux-gnu-i686/
bin/mysqld:
  Normal shutdown

010907 17:50:38 /usr/local/mysql-3.23.39-pc-linux-gnu-i686/
bin/mysqld:
  Shutdown Complete

010907 17:50:38 mysqld ended

```

En este ejemplo, se registra una **situación** mencionada anteriormente que tiene lugar cuando MySQL se inicia de **manera** incorrecta. Tras **ello**, **al intentar** iniciar MySQL de **manera correcta**, no podremos porque ya se ha iniciado otro proceso. Para solucionar este problema, tendremos que finalizar el proceso **incorrecto** (por ejemplo, ejecutando `kill -s 9 PID` o `kill -9 PID` en Unix o utilizar el Administrador de tareas de Windows).

El registro de consultas

Para iniciar el registro de consultas puede utilizar la siguiente **opción**:

```
log =[nombre_de_archivo_del_registro_de_consultas]
```

en `my.cnf`. Si no especifica `nombre_de_archivo_del_registro_de_consultas`, se le asignará el nombre de anfitrión.

Registrará todas las conexiones y las consultas ejecutadas. Puede resultar **útil** para determinar quien esta conectado (y cuando) para cuestiones de seguridad, **asi como** para operaciones de depuracion con el fin de comprobar si el servidor recibe correctamente las consultas.

Este **tipo** de registros afecta **al rendimiento**, por lo que deberiamos desactivarlos si esta **cuestión** resulta importante. Puede que convenga utilizar en su lugar el registro de **actualización** binario (que registra unicamente consultas de **actualización**): **A continuación** se recoge un registro de consultas de ejemplo:

```

/usr/local/mysql-max-4.0.1-alpha-pc-linux-gnu-i686/bin/mysqld,
Version:
  4.0.1-alpha-max-log, started with:
Tcp port: 3306  Unix socket: /tmp/mysql.sock
Time           Id Command      Argument
020707  1:01:29      1 Connect     root@localhost on
020707  1:01:35      1 Init DB     firstdb

```

```

020707 1:01:38      1 Query      show tables
020707 1:01:51      1 Query      select * from innotest
020707 1:01:54      1 Quit

```

El registro de actualización binario

El registro de actualización binario se activa cuando la opción `log-bin` se utiliza con el archivo de configuración `my.cnf` o `my.ini`, de la siguiente forma:

```
-log-bin[=nombre_de_archivo_de_registro_binario]
```

Se eliminara cualquier extensión, ya que MySQL agrega sus **propias extensiones** al registro binario. Si no se especifica ningún nombre de archivo, se tomara el nombre del anfitrión para designar al registro binario, adjuntandole la secuencia `-bin`. MySQL también crea un archivo de **índice** binario con el mismo nombre y con la extensión `.index`. Al **índice** se le puede asignar un nombre (y una **ubicación**) diferentes de la siguiente forma:

```
-log-bin-index=nombre_archivo_indice_del_registro_binario
```

Los registros de actualización **binaria** contienen todas las instrucciones SQL que actualizan los datos, así como el tiempo que tarda la consulta en ejecutarse y una marca de tiempo que establece cuando se procesara la consulta. Las **instrucciones** se registran en el mismo **orden** en el que se ejecutan (después de que la consulta y antes de que se completen las transacciones o se eliminen los **bloques**). Las actualizaciones que no se **hayan** confirmado todavía se colocaran en una cache primero.

El registro de actualización binario también resulta útil para restaurar **volcados** (en un capítulo posterior se examinara el tema de los volcados de bases de datos) así como para duplicar una base de datos esclava desde una principal (en un capítulo posterior se examinara la **duplicación** de bases de datos).

Los registros de actualización **binaria** comienzan por la extensión 001. Cada vez que el servidor se reinicia o se ejecuta alguna de las instrucciones `mysqladmin refresh`, `mysqladmin flush-logs` o `FLUSH LOGS` se crea uno nuevo incrementando dicho número en una unidad. También se crea un nuevo registro binario (y se **incrementa** su **numeración**) cuando el registro binario alcanza el máximo tamaño. El valor `max_binlog_size` se establece en el **archivo** `my.cnf` o `my.ini` de la siguiente forma:

```
set-variable      = max_binlog_size = 1000M
```

Puede observar el tamaño, en bytes, que se le asigna de **manera predeterminada** examinando las variables:

```
% mysqladmin -u root -pg00r002b variables | grep 'max_binlog_size'
```

```
max_binlog_size
```

```
| 1073741824
```

El **archivo de índice** de actualización binario contiene una lista de todos los registros utilizados hasta la fecha. **A continuación** se recoge un ejemplo:

```
./test-bin.001  
./test-bin.002  
./test-bin.003  
./test-bin.004
```

Si vaciamos los registros, el **índice** de actualización binario se adjuntará al nuevo registro binario:

```
o mysqladmin -u root -pg00r002b flush-logs
```

El ejemplo contiene ahora lo siguiente:

```
./test-bin.001  
./test-bin.002  
./test-bin.003  
./test-bin.004  
./test-bin.005
```

Puede eliminar todos los registros de actualización binarios sin utilizar con el comando `RESET MASTER`:

```
mysql> RESET MASTER;  
Query OK, 0 rows affected (0.00 sec)
```

El **índice** de actualización binario refleja ahora la existencia de un único registro de actualización binario:

```
./test-bin.006
```

ADVERTENCIA: No elimine los registros de actualización binarios hasta que esté seguro de que no los va a necesitar. Ponga especial cuidado si utiliza funciones de duplicación (en un capítulo posterior se ampliará este tema). Si utiliza registros binarios para restaurar volcados, asegúrese de no eliminar aquellos que sean más recientes que los últimos volcados.

No todas las actualizaciones de bases de datos necesitan registrarse. En muchos casos, puede que solo necesite almacenar actualizaciones de determinadas bases de datos.

Para ello, puede utilizar opciones `binlog-do-db` y `binlog-ignore-db` de los archivos de configuración `my.cnf` y `my.ini`. La primera permite establecer las actualizaciones de base de datos que se desea registrar. Por ejemplo, la siguiente instrucción:

```
binlog-do-db = firstdb
```

actualizara solamente la base de datos firstdb; sin embargo, esta:

```
binlog-ignore-db = test
```

actualizara todas **las** bases de datos **excepto** la base de datos test. Puede agregar varias **líneas** si desea registrar mas de una base de datos:

```
binlog-do-db = test  
binlog-do-db = firstdb
```

Cuando se necesita registrar actualizaciones que **formen parte** de una transacción, **MySQL** crea un bufer del tamaño especificado en `binlog-cache-size` en el archivo de **configuración** (el valor predeterminado es **32KB** o **32.768** bytes). Cada subproceso puede crear uno de estos bufer.

Para evitar el uso de demasiados bufer a la vez, tambien se puede establecer la variable `max-binlog-cache-size`. El tamaño máximo predeterminado es de **4GB**, o **4.294.967.295** bytes.

Como el archivo de actualización binario es un archivo binario, los datos se almacenaran de **manera** mas eficiente en el antiguo registro de actualización de texto.

Sin embargo, esta **opción** no **permite** ver los datos con un editor de texto. La utilidad `mysqlbinlog` soluciona este problema:

```
C:\Archivos de programa\MySQL\data>.\bin\mysqlbinlog test-  
bin.002  
# at 4  
#020602 18:40:02 server id 1 Start: binlog v 2, server v  
4.0.1-alpha-max-log  
created 020602 18:40:02  
# at 79  
#020602 18:41:27 server id 1 Query thread_id=3  
exec_time=0  
error_code=0  
use firstdb;  
SET TIMESTAMP=1023036087;  
CREATE TABLE customer(id INT);  
# at 146  
#020602 18:41:40 server id 1 Query thread_id=3  
exec_time=0  
error_code=0  
SET TIMESTAMP=1023036100;  
INSERT INTO customer(id) VALUES(1);  
# at 218  
#020602 18:43:12 server id 1 Query thread_id=5  
exec_time=0  
error_code=0  
SET TIMESTAMP=1023036192;  
INSERT INTO customer VALUES(12);  
# at 287  
#020602 18:45:00 server id 1 Stop
```


Para utilizar un archivo de **actualización** binario para actualizar los contenidos de un servidor MySQL, basta con canalizar los resultados **hacia** el servidor pertinente, por ejemplo:

```
% mysqlbinlog ..\data\test-bin.022 | mysql
```

NOTA: Una de las consecuencias de utilizar el registro de actualización binario es que no funcionarán las inserciones simultáneas con **CREATE INSERT** o **INSERT SELECT**. En las inserciones simultáneas MySQL permite que tengan lugar lecturas y escrituras a la vez en tablas MyISAM, pero si se habilitan con estos dos tipos de instrucciones, el registro de actualización binario no podrá utilizarse de manera fiable para restablecer volcados o realizar operaciones de duplicación.

El registro de consultas lentas

El registro de consultas **lentas** se inicia con la siguiente opción:

```
log-slow-queries[=nombre_de_archivo_de_registro_de_consultas_lentas]
```

en el archivo de **configuración**. Si no se suministra el parametro `nombre_de_archivo_de_registro_de_consulta_lento`, se asignara el nombre del equipo **anfitrión** al registro de consultas **lentas** y se le adjuntara la secuencia `-slow.log` (por ejemplo, `test.mysqlhost.co.za-slow.log`).

Se registraran todas las instrucciones SQL cuya ejecucion lleve mas tiempo que el establecido en `long_query_time`.

Este valor se establece en los **archivos** de **configuración** `my.cnf` o `my.ini` de la siguiente **forma**:

```
set-variable      = long-query-time = 20
```

Se mide en segundos (aunque MySQL tiene previsto cambiar a microssegundos, por lo que es aconsejable examinar la **documentación** mas reciente).

Si esta establecida la opción `log-long-format`, se registraran todas las consultas que no utilicen un **índice**.

Para **ello**, coloque la siguiente **línea**:

```
log-long-format
```

en su archivo `my.cnf` o `my.ini`.

Este registro resulta util; su incidencia en **materia** de rendimiento no es alta (asumiendo que la mayor parte de las consultas no sean **lentas**) y destaca las consultas que necesitan una mayor atencion (aquellas en las que falten indices o su uso no **esté** optimizado).

A continuación se muestra un ejemplo de registro de consultas lentas:

```
/usr/local/mysql-max-4.0.1-alpha-pc-linux-gnu-i686/bin/mysqld,
Version:
 4.0.1-alpha-max-log, started with:
Tcp port: 3306 Unix socket: /tmp/mysql.sock
Time          Id Command      Argument
# Time: 020707 13:57:57
# User@Host: root[root] @ localhost []
# Query-time: 0 Lock-time: 0 Rows-sent: 8 Rows-examined: 8
use firstdb;
select id from sales;
# Time: 020707 13:58:47
# User@Host: root[root] @ localhost []
# Query-time: 0 Lock-time: 0 Rows-sent: 6 Rows-examined: 8
```

En este registro, aparece la consulta `select id from sales` porque no utilizo ningún índice. La consulta **podría** haber utilizado un **índice sobre el campo id** (los índices se analizaron en un capítulo anterior).

También puede recurrir a la utilidad `mysqldumpslow` para mostrar los resultados de un registro de consultas lentas:

```
% mysqldumpslow test-slow.log

Reading mysql slow query log from test-slow.log
Count: 1 Time=0.00s (0s) Lock=0.00s (0s) Rows=0.0 (0),
root[root]@localhost
# Query-time: N Lock-time: N Rows-sent: N Rows-examined: N
use firstdb;
select id from sales

Count: 1 Time=0.00s (0s) Lock=0.00s (0s) Rows=0.0 (0),
root[root]@localhost
# Query-time: N Lock-time: N Rows-sent: N Rows-examined: N
DELETE FROM sales WHERE id>N

Count: 1 Time=0.00s (0s) Lock=0.00s (0s) Rows=0.0 (0),
root[root]@localhost
# Query-time: N Lock-time: N Rows-sent: N Rows-examined: N
select id from sales where id<N
```

Rotación de registros

Los archivos de registros, aunque **resultan** extremadamente útiles **tienen** una naturaleza cancerígena, ya que crecen sin parar hasta **agotar** el espacio disponible. Al final, no queda más remedio que eliminar los registros excesivos y la mejor **opción** es recurrir a alguna secuencia de comandos que se encargue de realizar automáticamente la **tarea**. Para los registros que no **resulten** vitales, **bastará** con utilizar la siguiente secuencia (asumiendo que **partamos** del directorio que contiene los archivos de registro).

En un sistema Unix:

```
mv logfile backup_directory/logfile.old
mysqladmin flush-logs
```

Y en un sistema Windows

```
move logfile backup_directory\logfile.old
mysqladmin flush-logs
```

El vaciado de los registros (que también se puede realizar durante la conexión al servidor con la instrucción de SQL FLUSH LOGS) cierra y vuelve a abrir los archivos de registro que no se incrementen en secuencia (como el registro de consultas lentas). O, en caso de que los registros se incrementen (el registro de consultas de actualización), el vaciado de los registros crea un nuevo archivo de registro con una extensión incrementada en una unidad a partir de la anterior y obliga a MySQL a utilizar el nuevo archivo.

El archivo de registro antiguo se puede retirar para su volcado o suprimir directamente si no se pudiera utilizar para nada más. Las consultas que se procesen entre las dos instrucciones no se registrarán, ya que no existe un registro de consulta para dicho momento en el tiempo. La operación de registro sólo se vuelve a crear cuando se vacían los registros. Por ejemplo, si asumimos que el registro de consulta se llama `querylog`, el siguiente conjunto de comandos muestra una forma de alternar registros. Necesitará tener dos ventanas abiertas. La ventana 1 conectada a su intérprete o línea de comandos y la ventana 2 a MySQL:

En primer lugar desde la ventana 1:

```
% mv querylog querylog.old
```

Seguidamente, ejecute una consulta desde la ventana 2 (conectada a MySQL):

```
mysql> SELECT * FROM sales;
```

Compruebe si se ha registrado la consulta desde la ventana 1:

```
% tail querylog_ _ _ _ _ _ _ _ _ _ _
tail: querylog: No such file or directory
```

Hasta que no vacie los registros, no existirá ningún registro de archivo y no se registrará ninguna consulta:

```
% mysqladmin -uroot -pg00r002b flush-logs
```

Ejecute otra consulta desde la ventana 2:

```
mysql> SELECT * FROM customer;
```

Esta vez se ha agregado al registro de consulta, como puede ver en la ventana 1:

```
% tail querylog
```

```

/usr/local/mysql-max-4.0.1-alpha-pc-linux-gnu-i686/bin/mysqld,
Version:
  4.0.1-alpha-max-log, started with:
Tcp port: 3306  Unix socket: /tmp/mysql.sock
Time           Id Command      Argument
020707 20:45:23    5 Quit
020707 20:45:26    4 Query      select * from customer

```

Esta técnica no se puede utilizar con archivos de registro vitales (como el registro de actualización **binaria**) ya que no se pueden permitir **las pérdidas** de consultas si se necesitan para operaciones de duplicación o para la restauración de volcados. Por esta razón, se crea un nuevo registro de actualización binario cuando los registros se vacían, con una extensión cuya numeración se **incrementa** en una unidad por **cada operación** de vaciado. Los registros solo se pueden agregar **al último** registro, lo que significa que podemos mover los antiguos sin tener que preocuparnos por **las consultas que faltan**. Pruebe a utilizar la siguiente secuencia; en **ella** se asume que el registro de actualización binario se denomina **gmbinlog** y que se parte de un registro de actualización binario:

```

C:\Program Files\MySQL\data>dir *-bin*

Volume in drive C has no label
Volume Serial Number is 2D20-1303
Directory of C:\Program Files\MySQL\data

GMBINLOG 001           272  07-07-02  8:50p gmbinlog.001
GMBINL-1 IND           0  07-07-02  8:48p gmbinlog.index
      2 file(s)                398 bytes
      0 dir(s)                 33,868.09 MB free
C:\Program Files\MySQL\data>..\bin\mysqladmin flush-logs
C:\Program Files\MySQL\data>dir *-bin*

Volume in drive C has no label
Volume Serial Number is 2D20-1303
Directory of C:\Program Files\MySQL\data

GMBINLOG 001           272  07-07-02  8:50p gmbinlog.001
GMBINL-1 IND           0  07-07-02  8:48p gmbinlog.index
GMBINLOG 002           0  07-07-02  8:50p gmbinlog.001
      3 file(s)                398 bytes
      0 dir(s)                 33,868.09 MB free
C:\Program Files\MySQL\data> move gmbinlog.001
D:\backup_directory\gmbinlog001.old

```

ADVERTENCIA: Si esta **realizando** operaciones de duplicación, no elimine **los** archivos de registro **antiguos hasta** que no este **seguro** de que **ningún servidor** esclavo **los** necesitara. **Consulte** un **capítulo** posterior para obtener más detalles.

MySQL para Red Hat Linux incluye una secuencia de comandos de **rotación** de registros. Si su distribución no la incorpora puede utilizar esta como base para crear una propia:

```
# This logname is set in mysql.server.sh that ends up in /etc/
# rc.d/init.d/mysql
#
# If the root user has a password you have to create a
# /root/.my.cnf configuration file with the following
# content:
#
# [mysqladmin]
# password = <secret>
# user= root
#
# where "<secret>" is the password.
#
# ATTENTION: This /root/.my.cnf should be readable ONLY
# for root !

/usr/local/var/mysqld.log {
    # create 600 mysql mysql
    notifempty
    daily
    rotate 3
    missingok
    compress
    postrotate
    # just if mysqld is really running
    if test -n "$(ps acx|grep mysqld)"; then
        /usr/local/bin/mysqladmin flush-logs
    fi
    endscript
}
```

Optimization, analisis, comprobacion y reparacion de tablas

Una parte del trabajo del administrador de bases de datos consiste en realizar labores de mantenimiento preventivo así como en reparar elementos que **hayan** salido mal. A pesar de los esfuerzos, pueden surgir errores, debido, por ejemplo, a un **corte** en el suministro eléctrico durante una **operación** de escritura. Por **regla** general, estos **errores resultan bastante** sencillos de resolver. Las operaciones de comprobación y reparación implican cuatro **tareas** principales:

- Optimización de tablas
- **Análisis** de tablas (se analiza y almacena la distribución de claves de las tablas MyISAM y BDB)

- Comprobación de tablas (se verifican las tablas en busca de errores y, en el caso de las tablas MyISAM, se actualizan las estadísticas clave)
- Reparación de tablas (se reparan tablas MyISAM dañadas)

Optimización de tablas

Las tablas que contienen campos BLOB y VARCHAR necesitan optimizarse con el tiempo. Como estos tipos de campo varían en longitud, al actualizar, insertar o eliminar registros, no siempre ocupan el mismo espacio, por lo que se fragmentan y los espacios vacíos permanecen. Al igual que ocurre con un disco fragmentado, esta situación ralentiza el rendimiento por lo que para mantener MySQL en buena forma, resultara necesario desfragmentarlo. En concreto, debemos optimizar la tablas, operación que se puede realizar de varias formas: mediante la instrucción OPTIMIZE TABLE, la utilidad mysqlcheck (si el servidor está en ejecución) o la utilidad mysiamchk (si el servidor no se está ejecutando o no hay interacción con la tabla.).

La optimización sólo funciona en la actualidad con las tablas MyISAM y parcialmente con las tablas BDB. En el caso de las tablas MyISAM, la optimización realiza las siguientes tareas:

- Desfragmenta las tablas en las que las filas aparecen divididas o han sido eliminadas
- Ordena los índices si no están ordenados
- Actualiza las estadísticas de índice si no han sido actualizadas

En el caso de las tablas BDB, la operación de optimización analiza la distribución de claves (realiza la misma operación que el comando ANALIZE TABLE, que se verá en una sección posterior).

optimización de tablas con la instrucción OPTIMIZE

La instrucción OPTIMIZE es una instrucción de SQL utilizada al establecer una conexión a una base de datos MySQL. Su sintaxis es la siguiente:

```
OPTIMIZE TABLE nombre_de_tabla
```

También puede optimizar una gran cantidad de tablas a la vez, utilizando una coma para separarlas:

```
mysql> OPTIMIZE TABLE customer,sales;
+-----+-----+-----+-----+
| Table           | Op          | Msg_type | Msg_text
|
+-----+-----+-----+-----+
| firstdb.customer | optimize    | status   | Table is already up
to date |
```

```

| firstdb.sales      | optimize | status  | OK
|
+-----+-----+-----+-----+
2 rows in set (0.02 sec)

```

En este ejemplo se ha actualizado la tabla `customer`

Optirnizacion de tablas con mysqlcheck

`mysqlcheck` es una utilidad de línea de comandos que puede realizar varias **tareas** de comprobacion y **reparación** además de operaciones de optimizacion. En una seccion posterior se recoge una descripcion completa de todas **las** funciones de esta utilidad. Para poder utilizar `mysqlcheck`, el servidor debe estar en **ejecucion**. Para optimizar la tabla `customer` de la base de datos `firstdb`, utilice la **opción** `mysqlcheck`, de la siguiente forma:

```

% mysqlcheck -o firstdb customer -uroot -pg00r002b
firstdb.customer      Table is already up to date

```

`mysqlcheck` **permite** optimizar mas de una tabla a la vez para lo cual debe **incluirlas** tras el nombre de la base de datos:

```

% mysqlcheck -o firstdb customer sales -uroot -pg00r002b
firstdb.customer      Table is already up to date
firstdb.sales         Table is already up to date

```

Tambien puede optimizar la base de datos entera dejando fuera todas **las** **refe-**rencias de tabla como muestra la siguiente secuencia:

```

% mysqlcheck -o firstdb -uroot -pg00r002b

```

Optirnizacion de tablas con myisamchk

Por ultimo, puede utilizar la utilidad de linea de comandos `myisamchk` cuando el servidor no este en ejecucion o no este interactuando con él. (Vacie las tablas antes de ejecutar esta instruccion si el servidor esta en funcionamiento con `mysqladmin flush-tables`.)

No obstante, tendra que asegurarse de que el servidor no esta interactuando con la tabla para evitar que se produzcan **daños**.) Ésta es la **forma** antigua de verificar **las** tablas.

Debe ejecutar `myisamchk` desde la ubicacion **exacta** de la tabla o especificar la ruta hasta ella. En una seccion posterior se recoge una descripcion completa de **las** funciones de esta utilidad.

El equivalente de una instruccion de optimizacion es el siguiente:

```

myisamchk -quick -check-only-changed -sort-index -analyze
nombre_de_tabla

```

o como el siguiente:

```

myisamchk -q -C -S -a nombre_de_tabla

```

Por ejemplo:

```
% myisamchk -quick -check-only-changed -sort-index -analyze
customer
- check key delete-chain
- check record delete-chain
- Sorting index for MyISAM-table 'customer'
```

La opción `-r` repara la tabla, pero también elimina el espacio malgastado:

```
% myisamchk -r sales
- recovering (with sort) MyISAM-table 'sales'
Data records: 8
- Fixing index 1
- Fixing index 2
```

Si especifica la ruta hasta el archivo de índice de tabla y no está en el directorio correcto, obtendrá el siguiente error:

```
, myisamchk -r customer
myisamchk: error: File 'customer' doesn't exist
Para corregirlo basta con especificar la ruta completa hasta el archivo .MYI:
myisamchk -r /usr/local/mysql/data/firstdb/customer
- recovering (with keycache) MyISAM-table '/usr/local/mysql/
data/firstdb/customer'
Data records: 0
```

ADVERTENCIA: Las tablas se bloquean durante la optimización, por lo que no conviene ejecutar esta operación en las horas de máximo tráfico. Así mismo, asegúrese de que dispone de cantidad de espacio libre suficiente en el sistema al ejecutar `OPTIMIZE TABLE`. Si intenta ejecutar este comando en un sistema que esté a punto de quedarse sin espacio de disco o ya haya llegado a ese punto, puede que MySQL no logre completar el proceso de optimización, lo que dejará la tabla inutilizable.

La optimización es una parte importante de cualquier rutina administrativa para el mantenimiento de bases de datos que contengan tablas MyISAM y es aconsejable aplicarla de manera regular.

Análisis de tablas

El análisis de tablas mejora el rendimiento actualizando la información de una tabla para que MySQL pueda tomar una mejor decisión sobre cómo combinar las tablas.

La distribución de varios elementos de índice se almacena para su uso posterior. (La operación de análisis sólo funciona con tablas MyISAM y BDB.)

Existen tres formas de analizar una tabla:

- Mediante la instrucción **ANALYZE TABLE** con una conexión a **MySQL** establecida
- Mediante la utilidad de línea de comandos `mysqlcheck`
- Mediante la utilidad de línea de comandos `mysiamcheck`

Los análisis regulares de tablas pueden contribuir a mejorar el rendimiento y deberían incluirse dentro de cualquier rutina de mantenimiento.

Análisis de tablas con ANALYZE TABLE

ANALYZE TABLE es una instrucción que se utiliza al establecer una conexión a una base de datos en el servidor. Su sintaxis es la siguiente:

```
ANALYZE TABLE nombre_de_tabla
```

Por ejemplo:

```
mysql> ANALYZE TABLE sales;
+-----+-----+-----+-----+
| Table           | Op       | Msg_type | Msg_text |
+-----+-----+-----+-----+
| firstdb.sales  | analyze  | status   | OK       |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

El tipo de mensaje (`Msg_type`) puede ser de estado, error, información o aviso. A continuación, puede apreciar que ocurrirá si falta el índice e intentamos analizar la tabla:

```
mysql> ANALYZE TABLE zz;
+-----+-----+-----+-----+
| Table           | Op       | Msg_type | Msg_text |
+-----+-----+-----+-----+
| firstdb.zz     | analyze  | error    | Table 'firstdb.zz' doesn't
exist |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

La tabla sólo se analizara de nuevo si se ha modificado desde la última vez que se analizó:

```
mysql> ANALYZE TABLE sales;
+-----+-----+-----+-----+
| Table           | Op       | Msg_type | Msg_text |
+-----+-----+-----+-----+
| firstdb.sales  | analyze  | status   | Table is already up to
date |
+-----+-----+-----+-----+
```

```
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Análisis de las tablas con mysqlcheck

La utilidad de línea de comandos `mysqlcheck` se comenta detenidamente en una sección posterior. El servidor necesita estar en ejecución para poder utilizar `mysqlcheck` y sólo se puede aplicar a tablas **MyISAM**. Para analizar tablas se utiliza con la opción `-a`:

```
% mysqlcheck -a firstdb sales -uroot -pg00r002b
firstdb.sales                                     OK
```

También podemos analizar varias tablas de una base de datos incluyéndolas tras el nombre de esta:

```
% mysqlcheck -a firstdb sales customer -uroot -pg00r002b
firstdb.sales                                     Table is already up to date
firstdb.customer                                 Table is already up to date
```

Si intenta analizar una tabla que no admita análisis (como las tablas de tipo **InnoDB**), la operación fallará sin que se ocasione ningún daño. Por ejemplo:

```
% mysqlcheck -a firstdb innotest -uroot -pg00r002b
firstdb.innotest
error      : The handler for the table doesn't support check/
repair
```

También podemos analizar todas las tablas de la base de datos dejando fuera todos los nombres de tabla:

```
% mysqlcheck -a firstdb innotest -uroot -pg00r002b
```

Análisis de tablas con myisamchk

La utilidad de línea de comandos `myisamchk` se comenta detalladamente en una sección posterior. Para poder utilizar la función `myisamchk` el servidor no debe estar en ejecución o debe asegurarse de que no existe ninguna interacción con las tablas con las que estamos trabajando. Si la opción `-skip-external-locking` no está activa, puede utilizar `myisamchk` con garantías, aunque el servidor esté en ejecución. Las tablas se bloquearán, lo que afectará a su acceso, pero no se producirán informes de error. Si se utiliza `-skip-external-locking`, deberá vaciar las tablas antes de iniciar el análisis (con `mysqladmin flush-tables`) y asegurarse de que no hay acceso. Puede que obtenga resultados no válidos si `mysqld` u otra instancia accede a la tabla mientras se está ejecutando esta utilidad. Para analizar tablas, utilice la opción `-a`:

```
% myisamchk -a /usr/local/mysql/data/firstdb/sales
Checking MyISAM file: /usr/local/mysql/data/firstdb/sales
Data records:      9   Deleted blocks:      0
- check file-size
```

- check key delete-chain
- check record delete-chain
- check index reference
- check data record references index: 1
- check data record references index: 2

Comprobacion de tablas

Pueden surgir errores cuando los índices no están sincronizados con los datos. Las caídas del sistema o los cortes de suministro eléctrico pueden dañar las tablas. Por regla general, sólo se ven afectados los archivos de índice, no los datos, lo que puede resultar difícil de detectar, aunque puede notar que la información se devuelve lentamente o que no se encuentran datos que deberían estar. Cuando sospeche que ocurre un error, lo primero que debe hacer es comprobar las tablas. Entre los síntomas que anuncian la existencia de tablas dañadas con errores se pueden citar los siguientes:

- Finales de archivo inesperados
- El archivo de registros está dañado
- nombre_de_tabla.frm bloqueado ante cambios.
- No se puede encontrar el archivo nombre_de_tabla.MYI (Código de error:###)
- Obtención del error ### procedente de un descriptor de tabla. La utilidad perror devuelve más información sobre el número de error. Basta con ejecutar perror que se almacena en el mismo directorio que otros binarios como mysqladmin) y el número de error. Por ejemplo:

```
% perror 126
126 = Index file is crashed / Wrong file format
```

A continuación se recogen otros errores comunes:

```
126 = Index file is crashed / Wrong file format
127 = Record-file is crashed
132 = Old database file
134 = Record was already deleted (or record file crashed)
135 = No more room in record file
136 = No more room in index file
141 = Duplicate unique key or constraint on write or update
144 = Table is crashed and last repair failed
145 = Table was marked as crashed and should be repaired
```

Una vez establecida la conexión al servidor MySQL, puede ejecutar un comando CHECK TABLE, usar la utilidad mysqlcheck (cuando el servidor está en ejecución) o la utilidad myisamchk cuando el servidor está detenido. La operación de comprobación actualiza las estadísticas de índices y busca errores.

Si se detectan errores, será necesario reparar la tabla (operación que se explica en una sección posterior). Los errores graves impiden el uso de la tabla hasta su reparación.

TRUCO: Compruebe siempre las tablas tras un corte en el suministro eléctrico o una caída del sistema. Por regla general, podrá solucionar cualquier error antes de que los usuarios detecten ningún problema.

Comprobación de las tablas con CHECK TABLES

La sintaxis del comando CHECK TABLES es la siguiente:

```
CHECK TABLE nombre_de_tabla [opción]
```

Por ejemplo:

```
mysql> CHECK TABLE customer;
+-----+-----+-----+
| Table           | Op   | Msg_type | Msg_text |
+-----+-----+-----+
| firstdb.customer | check | status   | OK       |
+-----+-----+-----+
1 row in set (0.01 sec)
```

Existen cinco opciones en función del nivel de comprobación que desee aplicar, como se describe en la tabla 10.4.

Tabla 10.4. Opciones de CHECK TABLES

Opción	Descripción
QUICK	Ésta es la forma más rápida de comprobación. No examina las filas para verificar vínculos erróneos.
FAST	Solo comprueba las tablas que no se han cerrado correctamente.
CHANGED	Solamente comprueba las tablas que no se han cerrado correctamente o que se han modificado desde la última verificación.
MEDIUM	Se trata de la opción predeterminada. Examina las filas para comprobar que los vínculos eliminados son correctos. También calcula una suma de comprobación de clave para las filas y lo verifica con una suma de comprobación para las claves.
EXTENDED	Se trata del método más lento, pero comprueba las tablas de forma exhaustiva para lo cual realiza un examen completo de claves para cada índice asociado a cada fila.

La opción **QUICK** resulta útil para examinar aquellas tablas sobre las que no se tengan sospechas de error. Si se devuelve un error o un aviso, debería **intentar** reparar la tabla. Puede comprobar más de una tabla simultáneamente **incluyéndolas** unas detrás de otras, por ejemplo:

```
mysql> CHECK TABLE sales, customer;
+-----+-----+-----+
| Table           | Op    | Msg_type | Msg_text |
+-----+-----+-----+
| firstdb.sales   | check | status   | OK       |
| firstdb.customer | check | status   | OK       |
+-----+-----+-----+
2 rows in set (0.01 sec)
```

Comprobación de tablas con mysqlcheck

La utilidad de línea de comandos **mysqlcheck** se puede utilizar con el servidor en ejecución y funciona únicamente con tablas **MyISAM**. Esta utilidad se **examina** detalladamente en una **sección** posterior. En la tabla 10.5 se describen sus opciones. La **sintaxis** es la siguiente:

```
mysqlcheck [opciones] nombre_de_base_de_datos nombre_de_tabla
[s]
```

Por ejemplo:

```
% mysqlcheck -c firstdb customer -uroot -pg00r002b
firstdb.customer OK
```

Tabla 10.5. Las opciones de **mysqlcheck** que se aplican a la comprobación de tablas

Opción	Descripción
-auto-repair	Se utiliza en combinación con una de las opciones de comprobación. Comienza automáticamente a reparar las tablas dañadas una vez completada la comprobación.
-c, -check	Comprueba tablas
-C, -check-only-changed	Comprueba las tablas que hayan cambiado desde la última operación de comprobación o que no se cerraron correctamente.
-F, -fast	Comprueba tablas que no se hayan cerrado correctamente.
-e, -extended	Ésta es la forma más lenta de comprobación, pero garantiza una operación completa. También puede utilizar esta opción para reparar tablas, aunque no suele resultar necesario.

Opción	Descripción
<code>-m, -medium-check</code>	Esta opción resulta mucho mas rapida que la comprobacion extendida y busca la gran mayoría de errores.
<code>-q, -quick</code>	Se trata de la comprobacion mas rapida. No examina las filas de las tablas al realizar la operación de comprobacion. Solo repara el arbol del índice.

Puede comprobar varias tablas incluyendolas tras el nombre de la base de datos:

```
% mysqlcheck -c firstdb sales customer -uroot -pg00r002b
firstdb.sales OK
firstdb.customer OK
```

Puede comprobar todas las tablas de la base de datos especificando unicamente el nombre de la base de datos:

```
% mysqlcheck -c firstdb -uroot -pg00r002b
```

Comprobacion de tablas con myisamchk

Cuando el servidor esta apagado o no existe **interacción** con las tablas que se estan comprobando, puede utilizar la opcion de linea de comandos myisamchk (que se describe detalladamente en una **sección** posterior). Si la opcion `-skip-external-locking` no esta activa, puede utilizar myisamchk con garantías, aunque el servidor este en ejecucion. Las tablas se bloquearan, lo que afectara a su acceso, **pero** no se produzcan informes de error. Si se utiliza `-skip-external-locking`, **deberá** vaciar las tablas antes de iniciar el **análisis** (con `mysqladmin flush-tables`) y asegurarse de que no hay acceso. Puede que obtenga resultados no validos si mysqld u otra instancia accede a la tabla mientras se esta ejecutando esta utilidad.

Su **sintaxis** es la siguiente:

```
myisamchk [opciones] nombre_de_tabla
```

El equivalente a la **instrucción** CHECK TABLE es la opcion intermedia:

```
myisamchk -m nombre_de_tabla
```

La opcion predeterminada para myisamchk es la opcion de comprobacion habitual (`-c`). Tambien existe una opcion de comprobacion rapida (`-F`), que sólo comprueba las tablas que no se han cerrado correctamente. Esta opcion es distinta a la opcion `-f`, que fuerza el proceso de comprobacion aunque se detecte un error. Tambien puede utilizar la comprobacion intermedia(`-m`), que resulta ligeramente mas lenta **pero** es mas completa. La opcion mas extrema es la opcion `-e`

(que realiza una comprobación extendida); se trata de la opción más completa y **la más lenta**.

Suele ser también un signo de desesperación; utilízela únicamente cuando el resto de las opciones **hayan fallado**. Si **incrementa** el valor de la variable `key_buffer_size` puede acelerar la comprobación extendida (pero debe **disponer de memoria** suficiente). En la tabla 10.6 se recoge una **descripción** de todas estas opciones.

Tabla 10.6. Opciones de comprobación de `myisamchk`

Opción	Descripción
<code>-c, -check</code>	Comprobación habitual. Se trata de la opción predeterminada.
<code>-e, -extend-check</code>	Se trata del tipo de comprobación más lento pero más completo . Si está utilizando <code>-extended-check</code> y dispone de una gran cantidad de memoria , debería incrementar el valor de la variable <code>key_buffer_size</code> .
<code>-F, -fast</code>	Comprobación rápida. Solo verifica que las tablas no se hayan cerrado incorrectamente.
<code>-C, -check-only-changed</code>	Comprueba únicamente las tablas que se hayan modificado desde la última operación de comprobación.
<code>-f, -force</code>	Ejecuta la opción de reparación si se encuentra algún error en la tabla.
<code>-i, -information</code>	Muestra estadísticas sobre la tabla que se está comprobando.
<code>-m, -medium-check</code>	Comprobación intermedia. Resulta más rápida que la extendida y es adecuada para la mayor parte de los casos .
<code>-U, -update-state</code>	Mantiene información sobre cuando fue comprobada la tabla y si ha sufrido algún bloqueo, información que resulta útil para la opción <code>-c</code> . Conviene no utilizarla cuando se está usando la tabla y la opción <code>-skip-external-locking</code> está activa .
<code>-T, -read-only</code>	No etiqueta la tabla como comprobada (resulta útil para ejecutar <code>myisamchk</code> cuando el servidor está activo y se está utilizando la opción <code>-skip-external-locking</code>).

A continuación se incluye un ejemplo de los resultados devueltos por la utilidad `myisamchk` cuando detecta errores:

```
% myisamchk largetable.MYI
Checking MyISAM file: Hits.MYI
Data records: 2960032 Deleted blocks: 0
myisamchk: warning: 1 clients is using or hasn't closed the
table
properly
- check file-size
myisamchk: warning: Size of datafile is: 469968400 Should be:
469909252
- check key delete-chain
- check record delete-chain
- check index reference
- check data record references index: 1
- check data record references index: 2
- check data record references index: 3
myisamchk: error: Found 2959989 keys of 2960032
- check record links
myisamchk: error: Record-count is not ok; is 2960394 Should be:
2960032
myisamchk: warning: Found 2960394 parts Should be: 2960032
parts
```

Reparación de tablas

Si ha comprobado las tablas y se han detectado errores, tendremos que repararlas. Existen varias opciones de reparación posibles, que dependen del método utilizado, pero es posible que no den buenos resultados. Si el disco ha fallado, o si no funciona ninguna de las opciones, tendremos que recurrir a una copia de seguridad para restaurarlas. La reparación de tablas puede absorber una gran cantidad de recursos, tanto en cuestión de disco como de memoria.

- Por regla general, la reparación de tablas absorbe el doble de espacio de disco que el archivo de datos original (en el mismo disco). La opción de reparación rápida (veanse las opciones en las siguientes secciones) es una excepción porque el archivo de datos no se modifica.
- Se necesita espacio para el nuevo archivo de índice (en el mismo disco que el original). El índice antiguo se elimina al principio, de forma que este espacio no resultara significativo a menos que el disco este prácticamente lleno.
- La opción estándar y la opción `– sort–recover` crean un bufer de ordenación. Éste absorbe la siguiente cantidad de espacio: $\text{clave_mas_larga} + \text{longitud_del_puntero_de_fila}) * \text{número_de_filas} * 2$. Puede trasladar una parte o todo este espacio a la memoria (y aumentar la velocidad del proceso) incrementando el

tamaño de la variable `sort_buffer_size` si dispone de memoria. De lo contrario, se creará según se especifique en la variable de entorno `TMPDIR` o en la opción `-t` de `mysamchk`.

- El uso de memoria viene determinado por las variables de `mysqld` o las opciones establecidas en la línea de comandos `mysamchk`.

Si surge un error como resultado de falta de espacio para la tabla y el tipo de tabla es InnoDB, tendremos que ampliar el espacio de la tabla InnoDB. Las tablas MyISAM tienen un límite de tamaño teórico (ocho millones de terabytes), pero de manera predeterminada sólo se asigna 4GB a los punteros. Si la tabla alcanza este límite, puede ampliarlo utilizando los parámetros `MAX_ROWS` y `AVG_ROW_LENGTH` `ALTER TABLE`. En el siguiente fragmento se prepara la tabla denominada `limited` para acoger una gran cantidad de datos (actualmente solo consta de tres registros):

```
mysql> ALTER TABLE limited MAX_ROWS=999999999999
AVG_ROW_LENGTH=100;
Query OK, 3 rows affected (0.28 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

Esta secuencia asigna punteros para una cantidad mucho mayor de registros. Se utiliza `AVG_ROW_LENGTH` cuando los campos `BLOB` y `TEXT` están presentes para dar una idea a MySQL sobre el tamaño promedio de un registro, que a continuación se puede utilizar para fines de optimización.

Reparación tablas de tipo diferente a MyISAM

Los tres métodos de reparación que se comentan en las siguientes secciones sólo sirven para tablas MyISAM. Algunas de las opciones que se indicaran pueden aplicarse ocasionalmente a tablas BDB, pero no están diseñadas para su funcionamiento con ellas. En la actualidad, la única forma de reparar tablas BDB e InnoDB consiste en restaurarlas a partir de una copia de seguridad.

Reparación de tablas con REPAIR TABLE

Puede ejecutar la instrucción `REPAIR TABLE` al conectarse al servidor MySQL. Actualmente sólo funciona con tablas MyISAM. En la tabla 10.7 se recogen las instrucciones. Su sintaxis es la siguiente:

```
REPAIR TABLE nombre_de_tabla[s] opcion[s]
```

Tabla 10.7. Opciones de la instrucción `REPAIR TABLE`

Opción	Descripción
QUICK	Se trata de la forma más rápida de reparación porque el archivo de datos no se modifica. Además,

Opción	Descripción
EXTENDED	utiliza mucho menos espacio de disco porque no se modifica el archivo de datos.
USE_FRM	Intenta recuperar todas las filas posibles del archivo de datos. Esta opción debería utilizarse como último recurso porque puede generar filas inservibles.
	Esta opción es la que se utiliza si el archivo <code>.MYI</code> no está o tiene un encabezado dañado . Los índices se reconstruirán a partir de las definiciones del archivo de definición de tabla <code>.frm</code> .

A continuación se incluye un ejemplo de uso de la instrucción REPAIR en caso de que falte un archivo `.MYI`. Vamos a eliminar el archivo `.MYI` de una tabla existente, `t4`.

```
% ls -l t4.*
-rw-rw- 1 mysql mysql 10 Jun 14 02:00 t4.MYD
-rw-rw- 1 mysql mysql 4096 Jun 14 02:00 t4.MYI
-rw-rw- 1 mysql mysql 8550 Jun 8 10:46 t4.frm
% rm t4.MYI
% mysql -uguru2b -pg00r002b firstdb
```

Una instrucción REPAIR normal no funcionaría:

```
mysql> REPAIR TABLE t4;
+-----+-----+-----+-----+
| Table          | Op      | Msg_type | Msg_text
+-----+-----+-----+-----+
| firstdb.t4    | repair  | error    | Can't find file: 't4.MYD'
(errno: 2) |
+-----+-----+-----+-----+
1 row in set (0.47 sec)
```

El mensaje de error (4.0.3) indica que no se puede encontrar el archivo `.MYD` cuando en realidad el archivo desaparecido es `.MYI`. Es probable que este mensaje de error haya sido corregido para cuando lea estas líneas. Para reparar esta tabla, necesitamos recurrir a la opción `USE_FRM`, que utiliza el archivo de **definición** `.frm` y volver a crear el archivo de **índice** `.MYI`:

```
mysql> REPAIR TABLE t4 USE_FRM;
+-----+-----+-----+-----+
| Table          | Op      | Msg_type | Msg_text
+-----+-----+-----+-----+
| firstdb.t4    | repair  | warning  | Number of rows changed from
0 to 2 |
```

```
| firstdb.t4 | repair | status | OK
+-----+-----+-----+-----+
2 rows in set (0.46 sec)
```

Reparación de las tablas con `mysqlcheck`

La utilidad de línea de comandos `mysqlcheck` se utiliza con el servidor en ejecución y funciona únicamente con las tablas **MyISAM** (en una sección posterior se analiza detenidamente). Para reparar tablas, utilice la opción `-r`:

```
% mysqlcheck -r firstdb customer -uroot -pg00r002b
firstdb.customer OK
```

Puede reparar varias tablas a la vez incluyendo sus nombres tras el de la base de datos:

```
% mysqlcheck -r firstdb customer sales -uroot -pg00r002b
firstdb.customer OK
firstdb.sales OK
```

Si por alguna razón todas las tablas de la base de datos están dañadas, puede repararlas en su conjunto indicando el nombre de la base de datos:

```
% mysqlcheck -r firstdb -uroot -pg00r002b
```

Reparación de tablas con `myisamchk`

Puede utilizar la utilidad de línea de comandos `myisamchk` (descrita en detalle en una sección posterior) para reparar tablas (vease tabla 10.8).

El servidor no debería estar en ejecución o deberíamos asegurarnos de que no tiene lugar ninguna **interacción** con las tablas con las que estamos trabajando, como sería el caso si hubiéramos empezado a trabajar con la opción `skip-external-locking`. Si la opción `skip-external-locking` está activa, sólo podrá utilizarse `myisamchk` para reparar tablas si está seguro de que no se produzcan accesos simultáneos. Utilice o no la opción `skip-external-locking`, necesitará vaciar las tablas antes de iniciar el proceso de **reparación** (con `mysqladmin flush-tables`) y asegurarse de que no se produce ningún acceso. Puede que obtenga resultados erróneos (con tablas **marcadas** como dañadas incluso si no lo están) si `myslqd` u otra instancia accede a la tabla mientras `myisamchk` está en ejecución. Su **sintaxis** es la siguiente:

```
myisamchk [opciones] [nombres_de_tablas]
```

Tabla 10.8. Reparación de tablas con `myisamchk`

Opción	Descripción
<code>-D #, -data-file-length=#</code>	Especifica la longitud máxima del archivo de datos al recrearlos.

Opción	Descripción
<code>-e, --extend-check</code>	Intenta recuperar todas las filas posibles desde el archivo de datos. Debería utilizar esta opción como último recurso porque puede generar filas inservibles.
<code>-f, --force</code>	Sobrescribe los viejos archivos temporales (con extensión <code>.TMD</code>) en lugar de anular la operación si encuentra uno ya existente.
<code>-k #, --keys-used=#</code>	Especifica que clave utilizar, lo que puede agilizar el proceso. Cada bit binario equivale a una clave que comience por 0 para la primera clave.
<code>-r, --recover</code>	Repara la mayor parte de los daños y debería ser la primera opción en utilizarse. Puede incrementar el valor de <code>sort_buffer_size</code> para que el proceso de recuperación resulte más rápido, si dispone de memoria. Esta opción no sirve para recuperar tablas en el extraño caso en el que la clave exclusiva deje de serlo.
<code>-o, --safe-recover</code>	Se trata de una opción de reparación más completa aunque más lenta que <code>-r</code> que solo se debería utilizar si fallara <code>-r</code> . Lee todas las filas y vuelve a generar los índices en función de ellas. También utiliza una menor cantidad de espacio de disco que <code>-r</code> porque no se crea un búfer de ordenación. Puede incrementar el tamaño <code>key_buffer_size</code> para que el proceso de recuperación resulte más rápido.
<code>-n, --sort-recover</code>	Obliga a MySQL a utilizar la operación de ordenación para resolver índices, aunque el resultado sean archivos temporales de mayor tamaño.
<code>--character-sets-dir=..</code>	El directorio que contiene el conjunto de caracteres.
<code>--set-character-set=nombre</code>	Especifica un nuevo conjunto de caracteres para el índice.
<code>-t, --tmpdir=ruta</code>	Especifica una nueva ruta para almacenar los archivos temporales si no desea que se utilice el valor especificado por la variable de entorno <code>TMPDIR</code> .

Opción	Descripción
-q, -quick	La opción de recuperación mas rapida porque los datos no se modifican. Si se especifica q dos veces (-q -q) se modificara el archivo de datos en caso de que haya claves duplicadas. Además utiliza mucho menos espacio de disco porque el archivo de datos no se modifica.
-u, -unpack	Descomprime un archivo comprimido con la utilidad myisampack.

Debe ejecutar myisamchk desde el directorio que contenga los archivos .MY I o suministrar su ruta.

Los siguientes ejemplos muestran un proceso de reparacion en acción, en el que MySQL decide si utilizar una operación de ordenacion o una cache de clave:

```
% myisamchk -r customer
- recovering (with keycache) MyISAM-table 'customer.MYI'
Data records: 0
% myisamchk -r sales
- recovering (with sort) MyISAM-table 'sales.MYI'
Data records: 9
- Fixing index 1
- Fixing index 2
```

Si dispone de una gran cantidad de memoria, además de aumentar el tamaño de los parametros sort_buffer_size y key_buffer_size como se describió anteriormente, también puede establecer otras variables para que myisamchk trabaje de forma más rápida (en una sección posterior se analiza detalladamente esta utilidad).

Como usar mysqlcheck

La utilidad mysqlcheck ha sido como un regalo llovido del cielo para los usuarios mas recientes de MySQL ya que antes gran parte de la funcionalidad de reparacion y comprobacion sólo se podia utilizar con el servidor apagado. Por suerte esta limitación pertenece al pasado.

mysqlcheck utiliza las instrucciones CHECK, REPAIR, ANALYZE y OPTIMIZE para realizar estas tareas desde la linea de comandos, lo que resulta util para el mantenimiento automatizado de las bases de datos (vease la tabla 10.9).

Su sintaxis es la siguiente:

```
mysqlcheck [opciones] nombre_de_base_de_datos
[nombres_de_tablas]
```

o la siguiente:

```
mysqlcheck [opciones] -databases nombre_de_base_de_datos1  
[nombre_de_base_de_datos2 nombre_de_base_de_datos3 ...]
```

o la siguiente:

```
mysqlcheck [opciones] -all-databases
```

Tabla 10.9. Opciones de mysqlcheck

Opción	Descripción
-A, -all-databases	Comprueba todas las bases de datos disponibles.
-1, -all-in-1	Combina consultas de tablas en una consulta de base de datos (en lugar de una por tabla). Las tablas se incluyen en una lista separada por comas.
-a, -analyze	Analiza la lista de tablas.
-auto-repair	Repara automáticamente las tablas si están dañadas (tras comprobar todas las tablas de la consulta).
-#, -debug=...	Genera un registro de depuración.
-character-sets-dir=...	Especifica el directorio en el que se encuentra el conjunto de caracteres.
-C, -check	Comprueba las tablas.
-C, -check-only-changed	Comprueba las tablas que se han modificado desde la última comprobación o que no se cerraron correctamente.
-compress	Utiliza la compresión en el protocolo cliente/servidor .
-?, -help	Muestra el mensaje de ayuda y sale.
-B, -databases	Enumera una lista de bases de datos que comprobar (se verificarán todas las tablas de las bases de datos).
-default-character-set=...	Establece el conjunto de caracteres pre- determinado.
-F, -fast	Comprueba las tablas que no se han cerrado correctamente.
-f, -force	Obliga al proceso a continuar aunque encuentre un error.

Opción	Descripción
-e, <code>—extended</code>	Se trata de la forma mas lenta de comprobacion pero garantiza la coherencia completa de la tabla. Tambien puede utilizar esta opcion para labores de reparacion, aunque no suele resultar necesario.
-h, <code>—host=...</code>	Nombre del anfitrión al que conectarse.
-m, <code>—medium-check</code>	Mucho mas rapida que la comprobacion extendida y detecta la mayor parte de los errores.
-o, <code>—optimize</code>	Optimiza las tablas.
-p, <code>—password[=...]</code>	La contraseña con la que establecer la conexion.
-P, <code>—port=...</code>	El puerto que utilizar para la conexion.
-q, <code>—quick</code>	La comprobacion mas rapida. No comprueba las filas de tabla al realizar la comprobacion. Cuando se utiliza para operaciones de reparacion, solo repara el arbol de índice.
-r, <code>—repair</code>	Repara la mayor parte de los errores, a excepcion de las claves exclusivas que contienen duplicados.
-s, <code>—silent</code>	No genera mensajes de salida a excepcion de los de error.
-S, <code>—socket=...</code>	Especifica el archivo de socket que se utilizara al establecer la conexion.
<code>—tables</code>	Lista de tablas que comprobar. Con la opcion -B, tendra prioridad.
-u, <code>—user=#</code>	Especifica como que usuario conectarse.
-v, <code>—verbose</code>	Genera una gran cantidad de mensajes de salida sobre el proceso.
-V, <code>—version</code>	Muestra la información de version y sale.

La utilidad `mysqlcheck` tambien consta de una funcion que permite ejecutarse en diferentes modos sin especificar las opciones. Basta con crear una copia de `mysqlcheck` con uno de los siguientes nombres para que adopte el comportamiento predeterminado:

- `mysqlrepair`: La opcion predeterminada es `-r`
- `mysqlanalyze`: La opcion predeterminada es `-a`.
- `mysqloptimize`: La opcion predeterminada es `-o`.

La opcion predeterminada cuando se utiliza `mysqlcheck` con nombre es `-c`. Todos estos archivos con nombres diferentes constan de la misma funcionalidad que `mysqlcheck`, lo unico que varia es su comportamiento predeterminado.

Uso de myisamchk

La utilidad myisamchk es la herramienta que se utilizaba antes y que esta disponible desde los primeros días de MySQL. También se utiliza para analizar, comprobar y reparar tablas, **pero** es necesario tener cuidado **al** utilizarla si el servidor esta en ejecución. En la tabla 10.10 se describen las opciones generales de myisamchk, en la 10.11 se describen sus opciones de comprobación, en la 10.12 se describen las opciones de reparación y en la 10.13 se describen el resto de las opciones.

Para utilizar myisamchk el servidor no debería estar en ejecución o deberíamos asegurarnos de que no se produce ninguna **interacción** con las tablas con las que estemos trabajando, **como** cuando iniciamos MySQL con la opción `skip-external-locking`. Con esta opción sin activar, **sólo** se puede utilizar myisamchk para reparar tablas si estamos seguros de que no se va a producir ningún acceso simultáneo. Utilicemos o no esta opción, debemos vaciar las tablas antes de **iniciar** el proceso de reparación (con `mysqladmin flush-tables`) y asegurarnos de que no existe ningún acceso.

Es aconsejable utilizar una de las siguientes opciones si el servidor esta en ejecución. Su **sintaxis** es la siguiente:

```
myisamchk [opciones] nombre_de_tabla[s]
```

Debe ejecutar myisamchk desde el directorio en el que se encuentran ubicados los **archivos de índice .MYT** a menos que especifique la ruta **hacia** ellos, ya que de lo contrario obtendrá el siguiente error:

```
% myisamchk -r sales.MYI
myisamchk: error: File 'sales.MYI' doesn't exist
```

Especificación de la ruta que resuelve el problema:

```
% myisamchk -r /usr/local/mysql/data/firstdb/sales.MYI
- recovering (with sort) MyISAM-table '/usr/local/mysql/data/
firstdb/sales.MYI'
Data records: 9
- Fixing index 1
- Fixing index 2
```

El nombre de la tabla se puede especificar con o sin la extensión **.MYT**.

```
% myisamchk -r sales
- recovering (with sort) MyISAM-table 'sales'
Data records: 9
- Fixing index 1
- Fixing index 2
% myisamchk -r sales.MYI
- recovering (with sort) MyISAM-table 'sales.MYI'
Data records: 9
- Fixing index 1
- Fixing index 2
```


ADVERTENCIA: Un error común consiste en ejecutar `myisamchk` en el archivo de datos `.MYD`. Utilice **siempre** el archivo de índice `.MYI`.

Puede utilizar el comodín para buscar todas las tablas de un directorio de base de datos (`*.MYI`) o incluso todas las tablas de todas las bases de datos:

```
% myisamchk -r /usr/local/mysql/data/*/*.MYI
```

Tabla 10.10. Opciones generales de `myisamchk`

Opción	Descripción
<code>-#, -debug=opciones_de_depuracion</code>	Genera un registro de depuración. Una opción de depuración habitual es <code>d:t:o</code> , <code>nombre_de_archivo</code> .
<code>-?, -help</code>	Muestra un mensaje de ayuda y sale.
<code>-O var=opcion, -set-variable var=opcion</code>	Establece el valor de una variable. Las variables posibles y sus valores predeterminados para <code>myisamchk</code> se pueden examinar con <code>myisamchk -help</code> .
<code>-s, -silent</code>	Solo muestra mensajes de error. Se puede utilizar una segunda <code>s</code> para que no muestre ningún mensaje.
<code>-v, -verbose</code>	Muestra más información de la habitual. Como ocurre con la opción <code>silent</code> , si se utilizan varias <code>v</code> se generará más información (<code>-vv</code> o <code>-vvv</code>).
<code>-V, -version</code>	Muestra los detalles de la versión <code>myisamchk</code> y sale.
<code>-w, -wait</code>	Si la tabla está bloqueada, <code>-w</code> espera a que la tabla se desbloquee en lugar de salir con un error. Si <code>mysqld</code> se está ejecutando con la opción <code>-skip-external-locking</code> , la tabla sólo se puede bloquear utilizando otro comando <code>myisamchk</code> .

Si ejecuta el comando `myisamchk -help`, además de las opciones generales, podremos ver las variables modificadas con la opción `-O` y los parámetros actuales:

```
% myisamchk -help
```

```
Possible variables for option -set-variable (-O) are:
key-buffer-size          current value: 520192
myisam-block-size       current value: 1024
```

```

read-buffer-size      current value: 262136
write-buffer-size    current value: 262136
sort-buffer-size     current value: 2097144
sort-key-blocks      current value: 16
decode-bits          current value: 9
ft_min_word_len      current value: 4
ft_max_word_len      current value: 254
ft_max_word_len_for_sort  current value: 20

```

El espacio asignado a la variable `key_buffer_size` se utiliza al realizar una comprobación extendida o al insertar índices fila a fila (utilizando la opción `safe-recover`). La variable `sort_buffer_size` se utiliza en la operación de reparación predeterminada cuando los índices se ordenan en la reparación.

Para lograr que el proceso de reparación resulte más rápido, asigne a la variable `sort_buffer_size` un cuarto del tamaño de memoria total disponible. Sólo se utiliza una de las dos variables a la vez, por lo que no necesitará preocuparse por cuestiones de memoria si asigna valores altos a ambas.

NOTA: Dentro del archivo `my.cnf` (o `my.ini`), hay secciones separadas para `mysqld` y `myisamchk`. Resulta bastante sencillo asignar un valor alto a la variable `sort_buffer_size` para operaciones de reparación y mantener dicho valor bajo si su sistema presenta otras necesidades de memoria para el día a día.

Tabla 10.11. Las opciones de comprobación de `myisamchk`

Opciones	Descripción
<code>-c, --check</code>	Comprobación habitual. Se trata de la opción predeterminada.
<code>-e, --extend-check</code>	Se trata del tipo de comprobación más lento pero más completo. Si está utilizando <code>--extended-check</code> y dispone de una gran cantidad de memoria, debería incrementar el valor de la variable <code>key_buffer_size</code> .
<code>-F, --fast</code>	Comprobación rápida. Solo verifica que las tablas no se hayan cerrado incorrectamente.
<code>-C, --check-only-changed</code>	Comprueba únicamente las tablas que se hayan modificado desde la última operación de verificación.
<code>-f, --force</code>	Ejecuta la opción de reparación si se encuentra algún error en la tabla.

Opciones	Descripción
<code>-i, -information</code>	Muestra estadísticas sobre la tabla que se esta comprobando.
<code>-m, -medium-check</code>	Comprobacion intermedia. Resulta mas rapida que la extendida y es adecuada para la mayor parte de los casos .
<code>-U, -update-state</code>	Mantiene información sobre cuando fue comprobada la tabla y si ha sufrido algún bloqueo, información que resulta util para la opcibn <code>-C</code> . Conviene no utilizarla cuando se esta usando la tabla y la opcion <code>-skip-external-locking</code> está activa .
<code>-T, -read-only</code>	No etiqueta la tabla como comprobada (resulta útil para ejecutar <code>mysamchk</code> cuando el servidor esta activo y se esta utilizando la opcion <code>-skip-external-locking</code>).

Tabla 10.12. Las opciones de reparación de `mysamchk`

Opciones	Descripción
<code>-D #, -data-file-length=#</code>	Especifica la longitud maxima del archivo de datos al recrearlos.
<code>-e, -extend-check</code>	Intenta recuperar todas las filas posibles desde el archivo de datos. Deberia utilizar esta opcion como último recurso porque puede generar filas inservibles.
<code>-f, -force</code>	Sobrescribe los viejos archivos temporales (con extension <code>.TMD</code>) en lugar de anular la operación si encuentra uno ya existente.
<code>-k #, keys-used=#</code>	Especifica que clave utilizar, lo que puede agilizar el proceso. Cada bit binario equivale a una clave que comience por 0 para la primera clave (por ejemplo, 1 es el primer índice y 10 es el segundo).
<code>-r, -recover</code>	Repara la mayor parte de los dafios y deberia ser la primera opcibn en utilizarse. Puede incrementar el valor de <code>sort_buffer_size</code> para que el proceso de recuperación resulte mas rapido si dispone de memoria . Esta opcion no sirve para recuperar tablas en el extraño caso en el que la clave exclusiva deje de serlo.

Opciones	Descripción
<code>-o, --safe-recover</code>	Se trata de una opción de reparación mas completa aunque mas lenta que <code>-r</code> que solo se debería utilizar si fallara <code>-r</code> . Lee todas las filas y vuelve a generar los índices en función de las filas. También utiliza una menor cantidad de espacio de disco que <code>-r</code> porque no se crea un bufer de ordenación. Puede incrementar el tamaño <code>key_buffer_size</code> para que el proceso de recuperación resulte mas rapido.
<code>-n, --sort-recover</code>	Obliga a MySQL a utilizar la operación de ordenación para resolver índices, aunque el resultado sean archivos temporales de mayor tamaño .
<code>--character-sets-dir=...</code>	El directorio que contiene el conjunto de caracteres .
<code>--set-character-set=nombre</code>	Especifica un nuevo conjunto de caracteres para el índice .
<code>-t, --tmpdir=ruta</code>	Especifica una nueva ruta para almacenar los archivos temporales si no desea utilizar el valor especificado por la variable de entorno <code>TMPDIR</code> .
<code>-q, --quick</code>	La opción de recuperación mas rapida porque los datos no se modifican. Si se especifica <code>q</code> varias veces se modificara el archivo de datos en caso de que haya claves duplicadas. Ade- mas utiliza mucho menos espacio de disco porque el archivo de datos no se modifica.
<code>-u, --unpack</code>	Descomprime un archivo comprimido con la utilidad <code>mysampack</code> .

Tabla 10.13. Otras opciones de `mysamchk`

Opciones	Descripción
<code>-a, --analyze</code>	El análisis de las tablas mejora el rendimiento al actualizar la información de los índices de una tabla lo que ayuda a MySQL a tomar un decision mejor sobre cómo combinar las tablas. La distribucion de varios elementos de índice se almacena para su uso posterior. Se trata de la misma opción que <code>ANALIZE TABLE</code> .

Opciones	Descripción
-d, --description	Muestra una descripción de la tabla.
-A, --set-auto-increment [=valor]	Asigna el valor especificado al contador AUTO_INCREMENT (o lo incrementa en una unidad si no se indica ningun valor).
-S, --sort-index	Ordena el índice tres bloques en orden descendente , lo que mejora el rendimiento de las búsquedas y el examen de las tablas por claves.
-R, --sort-records=#	Ordena los registros en funcion del índice especificado (la numeración de los índices comienza en 1; puede utilizar la funcion SHOW INDEX para ver una lista ordenada). De esta forma puede agilizar las consultas sobre este índice asi como selecciones por rangos. Si va a ordenar una tabla de gran tamaño por primera vez, es probable que la operacion resulte muy lenta.

Si ejecutamos myisamchk con la **opción -d** se genera el siguiente resultado:

```
% myisamchk -d customer
MyISAM file:          customer
Record format:       Packed
Character set:       latin1 (8)
Data records:        3 Deleted blocks:          0
Recordlength:        75
table description:
Key Start Len Index  Type
1         2         4    unique   long
```

Resumen

MySQL dispone de **todo** un **conjunto** de herramientas para que la **administracion** del servidor de bases de datos resulte lo mas sencilla posible. Cuando mas vitales **sean** sus datos y mayor **tamaño** tengan las tablas, mas importante resultara resolver **los** problemas de **manera** competente y rapida cuando tengan lugar. **Puede** detener e iniciar el servidor MySQL de varias **formas**, por ejemplo utilizando mysqld directamente, **pero** resulta mucho mas aconsejable utilizar una secuencia de comandos que se encargue de la tarea, como la secuencia de comandos mysqld-safe que se incluye en todas las distribuciones.

Windows y Unix utilizan diferentes metodos para la **automatización** del **proceso** de arranque del sistema, **pero** resulta **bastante** sencillo implementarlos una vez que sabemos como hacerlo. Los **archivos** de **configuración** son una **forma** flexible de controlar el **comportamiento** del servidor y aprenderemos mas **sobre** ellos en un capitulo posterior.

Cuando tiene lugar un desastre, **los archivos** de registro pueden tener un valor incalculable para ayudarnos a identificar el problema, ya se trate de una **consulta** o de otro **tipo** de error no previsto. También **resultan** útiles para restablecer el sistema a partir de copias de seguridad así como para otras **tareas** más mundanas, como la **identificación** de consultas **lentas** en **labores** de optimización. Obviamente, pueden crecer rápidamente y convertirse en inmanejables por lo que la tarea de alternarlos para evitar el problema de que crezcan indefinidamente resulta **importante**.

El mantenimiento regular de las tablas es la mejor receta para evitar problemas. Al optimizar **las** tablas se desfragmentan, lo que también resulta útil para actualizar la información de clave para ayudar a **MySQL** a **tomar** mejores decisiones **sobre** la **combinación** de consultas en función de información actualizada **sobre los datos**. Pero a pesar de **los mejores esfuerzos**, **los picos** de tensión eléctrica imprevistos, **los fallos** de **los componentes** de hardware o **los errores** humanos pueden terminar **dañando** los datos (especialmente **los índices**) en cuyo **caso** las distintas opciones de **reparación** **tienen** un valor incalculable. La antigua utilidad `myisamchk` resulta más útil cuando el servidor no está en ejecución mientras que la utilidad `mysqlcheck` puede realizar **tareas** de mantenimiento con el servidor en ejecución (como las instrucciones SQL relacionadas).

En **los siguientes capítulos** investigaremos la seguridad de base de datos, la **duplicación** y la **configuración** en mayor detalle.

Copias de seguridad de bases de datos

La creación de copias de seguridad es una de las tareas más importantes asociada a una base de datos y una de las que más se descuida. Los sucesos imprevistos pueden resultar desastrosos debido a su carácter. Muchos han aprendido la lección de la **forma** más dura: **pagando** el precio que **supone** la pérdida completa de **los datos** al posponer la realización de un volcado para una fecha futura, que nunca **parecía** llegar. Cuanto más importantes **sean los datos** y más frecuentes **sean los cambios** que se realizan **sobre ellos**, mayor debería ser la frecuencia a la que hacer copias de seguridad. En una base de datos de noticias en la que **los cambios tienen** lugar de **manera** continua, resulta aconsejable **hacer** un volcado diario y activar la función de registro para permitir la **recuperación** del trabajo diario. En un sitio Web pequeño en el que **los datos** se modifiquen semanalmente, el **sentido** común **dictaría** la realización de un volcado **cada** semana. Sea cual sea el tamaño de su base de datos, no hay excusa. Las copias de seguridad son un **elemento** vital en cualquier sistema de almacenamiento de datos.

Es aconsejable mantener también una copia de seguridad de su archivo de configuración (`my.cnf` o `my.ini`) ya que merece la pena guardar el trabajo realizado para ajustar el **servidor**. Personalmente, he **pasado** por la desagradable **situación** de respirar aliviado tras recuperar un sistema debido a un **fallo** en el disco para descubrir que mi querido y cuidado archivo de configuración se había perdido.

En este capítulo se muestran varias **formas** de realizar copias de seguridad y de restablecer el sistema con MySQL. Una vez conocidos los entresijos y las posibilidades que brinda el proceso de volcado de seguridad, podremos seleccionar con conocimiento de causa la estrategia que mejor se adapte a nuestra situación.

En este capítulo se abordan los siguientes temas:

- Los comandos BACKUP y RESTORE
- Realización de volcados de seguridad copiando archivos directamente
- mysqldump
- mysqlhotcopy
- Uso de SELECT INTO para realizar volcados
- Uso de LOAD DATA INFILE para restablecer volcados
- Aspectos de seguridad basados en LOAD DATA LOCAL
- Uso del registro de actualización binario
- Volcado de tablas InnoDB
- Duplicación como medio de volcado

Volcados de seguridad de tablas MyISAM con BACKUP

Una de las **formas** más sencillas de realizar un volcado de seguridad es utilizar el comando BACKUP. Este comando sólo se puede utilizar con tablas MyISAM. Su **sintaxis** es la siguiente:

```
BACKUP TABLE nombre_de_tabla TO '/db_backup_path';
```

La ruta de volcado debe ser la ruta completa al directorio en el que queremos guardarlo y no debe ser un nombre de archivo. Esta **instrucción** realiza una copia de los archivos **.frm** (definición) y **.MYD** (datos), pero no del archivo **.MYI** (índice). Puede volver a generar el índice una vez restaurada la base de datos. Al trabajar con archivos, debe prestar atención a la **cuestión** de los permisos. MySQL no muestra mensajes de error muy agradables si, al realizar la copia de seguridad, no disponemos de los permisos adecuados para utilizar todos los archivos y directorios.

Uso de BACKUP en Unix

El siguiente ejemplo se ejecuta en un equipo Unix, en el que el usuario que realiza las operaciones es el usuario raíz (en la siguiente sección se muestra un ejemplo en un sistema Windows):

```
% cd /  
% mkdir db_backups
```


Esta secuencia crea el directorio en el que queremos colocar los volcados dentro del directorio raíz. Establezca una conexión a la base de datos `firstdb` y ejecute el comando `BACKUP` de la siguiente forma:

```
% mysql firstdb
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13 to server version: 4.0.1-alpha-
max-log
mysql> BACKUP TABLE sales TO '/db_backups';
+-----+-----+-----+-----+
| Table          | Op      | Msg_type | Msg_text      |
+-----+-----+-----+-----+
| firstdb.sales | backup  | error    | Failed copying .frm file:
errno = 13 |
| firstdb.sales | backup  | status   | Operation failed
+-----+-----+-----+-----+
```

El problema en este ejemplo es que MySQL no dispone de permisos para escribir archivos en el directorio `/db_backups`. Tendremos que salir de MySQL y convertir al usuario de `mysql` en el titular del directorio desde la línea de comandos:

```
mysql> exit
Bye
% chown mysql db_backups/
```

ADVERTENCIA: Es necesario disponer de los permisos correctos para poder realizar esta tarea. Asegúrese de que el usuario utilizado para trabajar dispone de los permisos pertinentes. En este ejemplo, se trata del usuario raíz, por lo que no hay problemas, pero puede que no este trabajando como tal. En caso de que se le presenten problemas de este tipo, es probable que necesite recurrir a su administrador de sistema en busca de ayuda.

Ahora la instrucción `BACKUP` se ejecuta correctamente:

```
% mysql firstdb;
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 15 to server version: 4.0.1-alpha-
max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql> BACKUP TABLE sales TO '/db_backups';
+-----+-----+-----+-----+
| Table          | Op      | Msg_type | Msg_text      |
+-----+-----+-----+-----+
| firstdb.sales | backup  | status   | OK            |
```

```

+-----+-----+-----+-----+
1 row in set (0.00 sec)
mysql> exit
Bye

```

En esta ocasión el volcado ha resultado satisfactorio y podemos ver los archivos recién creados utilizando la línea de comandos de nuevo:

```

: ls -l db_backups/
total 10
-rw-rw-  1 mysql  mysql      136 May 26 14:07 sales.MYD
-rw-rw-  1 mysql  mysql     8634 May 26 14:07 sales.frm

```

Se han creado dos archivos.

TRUCO: Si tiene **algún** problema a **la** hora de realizar un volcado con este **método**, es probable que se deba a **los** permisos de **archivo**. Pida **ayuda** a su administrador **si** no dispone de **permiso** para crear **los** archivos o si **no está** seguro. Así **mismo**, **recuerde** que el **comando** BACKUP **sólo funciona** en la **actualidad** con tablas **MyISAM** (**de todos** modos, examine la **última docu-**mentación ya que puede que no sea así cuando lea estas **líneas**).

BACKUP coloca un bloqueo de lectura **sobre** la tabla antes de realizar una copia de seguridad de ella para asegurarse de que la tabla de volcado es coherente. También puede volcar más de una tabla simultáneamente incluyendo sus nombres:

```

mysql> BACKUP TABLE sales,sales_rep,customer TO '/db_backups';
+-----+-----+-----+-----+
| Table                | Op      | Msg_type | Msg_text |
+-----+-----+-----+-----+
| firstdb.sales        | backup  | status   | OK       |
| firstdb.sales_rep    | backup  | status   | OK       |
| firstdb.customer    | backup  | status   | OK       |
+-----+-----+-----+-----+
3 rows in set (0.05 sec)

```

El bloqueo se coloca en una tabla por vez. Primero sobre `sales`; cuando se ha realizado el volcado de `sales`, se pasa a `sales_rep`, etc. De esta forma queda garantizada la coherencia de las tablas, pero si **desea** realizar una **instantánea** coherente de todas las tablas a la vez, tendrá que aplicar sus propios bloqueos sobre ellas:

```

mysql> LOCK TABLES customer READ,sales READ,sales_rep READ;
Query OK, 0 rows affected (0.00 sec)

mysql> BACKUP TABLE sales,sales_rep,customer TO '/db_backups';
+-----+-----+-----+-----+
| Table                | Op      | Msg_type | Msg_text |
+-----+-----+-----+-----+

```

```

| firstdb.sales      | backup | status | OK      |
| firstdb.sales_rep | backup | status | OK      |
| firstdb.customer  | backup | status | OK      |
+-----+-----+-----+
3 rows in set (0.00 sec)

```

Fijese en que no puede bloquear las tablas de manera individual:

```

mysql> LOCK TABLE sales READ;
Query OK, 0 rows affected (0.00 sec)

```

```

mysql> LOCK TABLE sales-rep READ;
Query OK, 0 rows affected (0.00 sec)

```

```

mysql> LOCK TABLE customer READ;
Query OK, 0 rows affected (0.00 sec)

```

```

mysql> BACKUP TABLE sales,sales_rep,customer TO '/db_backups';

```

```

+-----+-----+-----+
| Table          | Op      | Msg_type | Msg_t      |
+-----+-----+-----+
| firstdb.sales  | backup  | error    | Table 'sales' was not locked with LOCK TABLES |
| firstdb.sales_rep | backup  | error    | Table 'sales-rep' was not locked with LOCK TABLES |
| firstdb.customer | backup  | status   | OK         |
+-----+-----+-----+

```

```

3 rows in set (0.00 sec)

```

El comando `LOCK TABLE` libera automáticamente todos los bloqueos implmentados por el mismo subproceso, por lo que el unico que se mantiene en el momento de realizar el volcado es sobre la tabla `customer`.

NOTA: Para poder bloquear una tabla, necesitara disponer del privilegio `LOCK TABLES` y del **privilegio SELECT sobre la tabla que está intentando el bloqueo.**

```

mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)

```

Uso de BACKUP con Windows

Deberia leer el ejemplo anterior aunque no utilice un sistema Unix, ya que en él se explican algunos de los conceptos relacionados con la instrucción `BACKUP`. El

siguiente ejemplo se centra en un problema específico de Windows. Esta plataforma no presenta la misma complejidad en cuanto a los permisos de archivo, pero tiene sus propios problemas. Examine el siguiente ejemplo e intente detectar el origen del problema:

```
C:\MySQL\bin>cd \
C:\>mkdir db_backups
C:\>mysql firstdb
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 4.0.1-alpha-
max

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql> BACKUP TABLE sales TO 'c:\db_backups';
+-----+-----+-----+-----+
| Table           | Op      | Msg_type | Msg_text
+-----+-----+-----+-----+
| firstdb.sales   | backup  | error     | Failed copying .frm file:
errno = 2 |
| firstdb.sales   | backup  | status    | Operation failed
+-----+-----+-----+-----+
2 rows in set (0.33 sec)
```

Por desgracia, el mensaje de error no resulta muy claro.

El problema es que la barra invertida (\) es el carácter de escape que utiliza MySQL para poder utilizar otros caracteres especiales como las comillas simples y las comillas dobles.

Para utilizar la barra invertida en la ruta de Windows, es necesario utilizar otro carácter de escape:

```
mysql> BACKUP TABLE sales TO 'c:\\db_backups';
+-----+-----+-----+-----+
| Table           | Op      | Msg_type | Msg_text |
+-----+-----+-----+-----+
| firstdb.sales   | backup  | status    | OK       |
+-----+-----+-----+-----+
1 row in set (0.55 sec)
```

Restauración de tablas MyISAM con RESTORE

Lo contrario de BACKUP es RESTORE que restaura las tablas MyISAM creadas previamente con BACKUP. También vuelve a crear los índices, lo que puede llevar cierto tiempo si las tablas son de gran tamaño.

Su sintaxis es la siguiente:

```
RESTORE TABLE nombre_de_tabla FROM '/db_backup_path'
```

No se puede restaurar **sobre** una tabla existenc. Si intenta restaurar la tabla sales cuya copia de seguridad realizamos anteriormenc, obtendremos el siguiente resultado:

```
mysql> RESTORE TABLE sales FROM '/db_backups';
+-----+-----+-----+-----+
| Table | Op      | Msg_type | Msg_text                                     |
+-----+-----+-----+-----+
| sales | restore | error    | table exists, will not overwrite         |
|       |         |          | on restore |
+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

Al menos este mensaje resulta mas claro que el utilizado para indicar que la operaci3n de volcado no se completo correctamente.

Para probar este volcado, tenemos que realizar un acto de fe y eliminar la tabla sales.

ADVERTENCIA: Puede que resulte obvio, pero no pruebe a realizar un volcado de una tabla activa eliminando la tabla original. Intente realizar el proceso de restauraci3n de una base de datos o servidor diferente. Si ya es demasiado tarde y falla el proceso de restauraci3n, no diga que sac3 los datos de este libro.

A continuaci3n, elimine la tabla e intente restaurarla:

```
mysql> DROP TABLE sales;
Query OK, 0 rows affected (0.01 sec)

mysql> RESTORE TABLE sales FROM '&-backups';
+-----+-----+-----+-----+
| Table | Op      | Msg_type | Msg_text                                     |
+-----+-----+-----+-----+
| sales | restore | error    | Failed copying .frm file |
+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

Que no cunda el panico. ¿Ve el error en el codigo anterior? La ruta no es correcta. Uno de los peores enemigos cuando algo sale mal es el panico.

Tras obtener el resultado anterior en una situaci3n de crisis, puede que le entren ganas de salir corriendo maldiciendo a MySQL. Cuando algo no funciona a menudo existe una sencilla explicaci3n, como el error tipografico del ejemplo anterior.

Si corregimos la ruta, la tabla se restaurara perfectamente. como muestra este ejemplo de Unix:

```
mysql> RESTORE TABLE sales FROM '/&-backups ';
```

```

+-----+-----+-----+-----+
| Table           | Op       | Msg_type | Msg_text |
+-----+-----+-----+-----+
| firstdb.sales  | restore  | status   | OK       |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

Y esta sería la **instrucción correcta** en Windows:

```

mysql> RESTORE TABLE sales FROM 'c:\\db_backups';
+-----+-----+-----+-----+
| Table           | Op       | Msg_type | Msg_text |
+-----+-----+-----+-----+
| firstdb.sales  | restore  | status   | OK       |
+-----+-----+-----+-----+
1 row in set (0.66 sec)

```

Y para calmar a los más paranoicos, vamos a comprobar si la tabla `sales` se restauró correctamente:

```

mysql> SELECT * FROM sales;
+-----+-----+-----+-----+
| code | sales-rep | id | value |
+-----+-----+-----+-----+
| 1    | 1         | 1  | 2000  |
| 2    | 4         | 3  | 250   |
| 3    | 2         | 3  | 500   |
| 4    | 1         | 4  | 450   |
| 5    | 3         | 1  | 3800  |
| 6    | 1         | 2  | 500   |
| 7    | 2         | NULL | 670  |
| 8    | 3         | 3  | 1000  |
+-----+-----+-----+-----+
8 rows in set (0.00 sec)

```

Volcados de seguridad de tablas de MyISAM mediante la copia de archivos directamente

Las tablas MyISAM se almacenan como archivos (`.frm` para la definición, `.MYD` para los datos y `.MYI` para los índices) dentro de un directorio denominado como la base de datos. Por lo tanto, una forma sencilla de realizar volcados de datos consiste en copiar los archivos.

A diferencia de `BACKUP`, la copia directa de los archivos no bloquea automáticamente las tablas, por lo que deberá hacerlo por sí mismo para obtener un resultado coherente.

Otra opción consiste en realizar la copia mientras el servidor no está en ejecución. Tras bloquear las tablas, debería vaciarlas para asegurarse de que todos los índices no escritos se escriben en el disco.

Para este ejemplo, necesitaremos tener dos ventanas abiertas:
Bloquee y vacíe las tablas en la Ventana 1:

```
mysql> LOCK TABLES sales READ,sales-rep READ,customer READ;
Query OK, 0 rows affected (0.00 sec)
mysql> FLUSH TABLES sales,sales-rep,customer;
Query OK, 0 rows affected (0.02 sec)
```

A continuación copie las tablas desde la ventana 2 (en Unix):

```
cd /usr/local/mysql/data/firstdb
\ cp sales.* /db_backups
% cp sales-rep.* /db_backups
\ cp customer.* /db_backups
```

O en Windows (desde la ventana 2):

```
C:\MySQL\data\firstdb>copy customer.* c:\db_backup
customer.frm
customer.MYI
customer.MYD
          3 file(s) copied
C:\MySQL\data\firstdb>copy sales.* c:\db_backup
sales.frm
sales.MYI
sales.MYD
          3 file(s) copied
C:\MySQL\data\firstdb>copy sales-rep.* c:\db_backup
sales_rep.frm
sales-rep.MYI
sales-rep.MYD
          3 file(s) copied
```

Tras copiar los archivos, puede liberar los bloqueos desde la ventana 1, de la siguiente forma:

```
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)
```

ADVERTENCIA: Durante el proceso de volcado, con los bloqueos aplicados, no podrá agregar registros nuevos a las tablas además de experimentar bajadas de rendimiento. Intente no realizar los volcados en horas con mucho tráfico.

De nuevo para probar el volcado, vamos a eliminar una tabla
Desde la ventana 1, elimine la tabla:

```
mysql> DROP TABLE sales;
Query OK, 0 rows affected (0.00 sec)
```

Copie las tablas desde la ventana 2 (este ejemplo es de Unix):

```
% cp /db_backups/sales.* .
```

Y nuestra tabla se restaura. Puede verificarlo utilizando la siguiente secuencia desde la ventana 2:

```
mysql> SELECT * FROM sales;
+----+-----+-----+
| code | sales-rep | id | value |
+----+-----+-----+
| 1 | 1 | 1 | 2000 |
| 2 | 4 | 3 | 250 |
| 3 | 2 | 3 | 500 |
| 4 | 1 | 4 | 450 |
| 5 | 3 | 1 | 3800 |
| 6 | 1 | 2 | 500 |
| 7 | 2 | NULL | 670 |
| 8 | 3 | 3 | 1000 |
+----+-----+-----+
8 rows in set (0.00 sec)
```

Tambien existe la posibilidad de que los permisos de Unix vuelvan a plantear problemas. Si no realiza los volcados como usuario mysql (por regla general esta tarea se realiza como usuario raiz) es probable que vea aparecer el siguiente mensaje de error:

```
mysql> SELECT * FROM sales;
ERROR 1017: Can't find file: './firstdb/sales.frm' (errno: 13)
```

El problema es que se ha copiado el archivo pero que el usuario mysql no dispone de acceso el. El siguiente fragmento de codigo, generado en la ventana 1, muestra que se han volcado los archivos como usuario raiz:

```
[root@test firstdb]# ls -l
total 183
...
-rw-r-- 1 root root 153 May 27 22:27 sales.MYD
-rw-r-- 1 root root 3072 May 27 22:27 sales.MYI
-rw-r-- 1 root root 8634 May 27 22:27 sales.frm
-rw-rw- 1 mysql mysql 156 May 22 21:50
sales_rep.MYD
-rw-rw- 1 mysql mysql 3072 May 22 21:50
sales_rep.MYI
-rw-rw- 1 mysql mysql 8748 May 22 21:50
sales_rep.frm
...
```

Para restaurar el permiso a mysql, cambie el titular de la tabla a `les` a mysql desde la ventana 1:

```
% chown mysql sales.*
%
```


Ahora todo debería funcionar correctamente, como puede observar si ejecutamos una consulta en la ventana 2:

```
mysql> SELECT * FROM sales;
+----+-----+-----+-----+
| code | sales-rep | id | value |
+----+-----+-----+-----+
| 1 | 1 | 1 | 2000 |
| 2 | 4 | 3 | 250 |
| 3 | 2 | 3 | 500 |
| 4 | 1 | 4 | 450 |
| 5 | 3 | 1 | 3800 |
| 6 | 1 | 2 | 500 |
| 7 | 2 | NULL | 670 |
| 8 | 3 | 3 | 1000 |
+----+-----+-----+-----+
8 rows in set (0.00 sec)
```

Las tablas MyISAM son ajenas a plataformas por lo que se pueden transferir a MySQL en un equipo con hardware diferente o con otro sistema operativo. Las tablas MyISAM creadas en la versión 3 de MySQL se pueden usar en la versión 4, pero no al revés. Para pasar datos creados con la versión 4 de MySQL a la versión 3, es necesario utilizar un método como mysqldump, que se comenta en la siguiente sección.

Realización de volcados con mysqldump

Los dos métodos anteriores copian archivos directamente y sólo funcionan con tablas MyISAM. (Las tablas InnoDB no se almacenan como archivos, por lo que no pueden hacer uso de los métodos anteriores.)

Otro método, mysqldump, realiza un volcado de las instrucciones SQL necesarias para crear las tablas que se desean copiar. Este método permite portar la base a otros sistemas de base de datos (teniendo en cuenta que no todas las funciones de MySQL se pueden aplicar entre bases de datos).

Si necesita portar su base de datos a otro DBMS, prepárese para un proceso largo y laborioso.

Para realizar una copia de seguridad de la tabla `customer`, ejecute el siguiente comando desde el intérprete de comandos de Unix:

```
% mysqldump firstdb customer > /db_backups/
customer_2002_11_12.sql
```

O este otro desde la línea de comandos de Windows:

```
C:\MySQL\bin>mysqldump firstdb customer >
c:\db_backups\customer_2002_11_12.sql
```

```
C:\MySQL\bin>
```

Recuerde especificar la ruta, el nombre de usuario y la contraseña si lo necesitara. Esta instrucción crea un archivo en el directorio `db_backups` con las instrucciones de SQL necesarias para recrear la tabla `customer`. Puede ver este archivo en cualquier editor de texto, como el Bloc de notas o vi. La primera parte del archivo contiene lo siguiente:

```
# MySQL dump 8.14
#
# Host: localhost      Database: firstdb
#-----
# Server version      4.0.1-alpha-max-log
```

Las líneas precedidas del símbolo `#` son simples comentarios, información sobre versiones, etc. En una parte posterior del archivo se incluyen las instrucciones SQL importantes y necesarias para volver a crear las distintas tablas. Este fragmento es el que permite volver a crear la tabla `customer`:

```
#
# Estructura de tabla de la tabla 'customer'
#

CREATE TABLE customer (
  id int(11) NOT NULL auto-increment,
  first-name varchar(30) default NULL,
  surname varchar(40) default NULL,
  initial varchar(5) default NULL,
  PRIMARY KEY (id),
  KEY surname (surname,initial,first-name)
) TYPE=MyISAM;

#
# Volcado de datos de la tabla 'customer'
#

INSERT INTO customer VALUES (1, 'Yvonne', 'Clegg', 'X');
INSERT INTO customer VALUES (2, 'Johnny', 'Chaka-Chaka', 'B');
INSERT INTO customer VALUES (3, 'Winston', 'Powers', 'M');
INSERT INTO customer VALUES (4, 'Patricia', 'Mankunku', 'C');
INSERT INTO customer VALUES (5, 'Francois', 'Papo', 'P');
INSERT INTO customer VALUES (7, 'Winnie', 'Dlamini', NULL);
INSERT INTO customer VALUES (6, 'Neil', 'Beneke', NULL);
INSERT INTO customer VALUES (10, 'Breyton', 'Tshabalala', 'B');
```

ADVERTENCIA El uso de los resultados de una instrucción `mysqldump` con valores predeterminados para restaurar una base de datos puede requerir mucho tiempo. Como el bufer de índice se vacía tras cada instrucción `INSERT`, las tablas de gran tamaño pueden tardar mucho tiempo en restaurarse. Examine las opciones de `mysqldump` para ver cómo se puede agilizar este proceso.

Restauracion de una base de datos volcada con mysqldump

Puede probar su copia de seguridad eliminando los datos y volviendolos a crear:

```
mysql> DROP TABLE customer;
Query OK, 0 rows affected (0.31 sec)
mysql> exit
Bye
```

ADVERTENCIA: Nuevamente, le aconsejamos no realizar esta prueba con una base de datos activa. Aqui solo simulamos la pérdida de una base de datos.

Para restaurar la tabla en un equipo Unix, ejecute la siguiente instruccion:

```
% mysql firstdb < /db_backups/customer_2002_11_12.sql
```

O desde un equipo Windows:

```
C:\MySQL\bin>mysql firstdb < c:\db_backups\customer_2002_11_12.sql
```

La tabla ha quedado restaurada.

En la tabla 11.1 se describen las opciones de mysqldump

Tabla 11.1. Opciones de mysqldump

Opción	Descripción
-add-locks	Coloca una instruccion LOCK TABLES antes y una instruccion UNLOCK TABLE despues de cada volcado. Como resultado, las instrucciones INSERT se procesaran de manera mucho mas rapida (ya que el bufer de claves solo se vacia tras la instruccion UNLOCK TABLE).
-add-drop-table	Agrega una instruccion DROP TABLE tras cada instruccion CREATE TABLE. Si la tabla ya existiese, podría interferir con el proceso de restauracion, por lo que esta opcion garantiza una limpieza completa de la tabla restablecida.
-A, -all-databases	Vuelca todas las bases de datos existentes. Equivale a la opcion -B O - databases con todas las bases de datos enumeradas.
-a, -all	Incluye todas las CREATE especificas de MySQL.

Opción	Descripción
-allow-keywords	Por regla general los nombres de columna no pueden coincidir con una palabra clave. Esta opción si lo permite, anteponiendo al nombre de cada columna el nombre de la tabla.
-c, -complete-insert	Utiliza instrucciones de inserción completas; en otras palabras <code>INSERT INTO nombre_de_tabla(x,y,z) VALUES (a,b,c)</code> en lugar de <code>INSERT INTO nombre_de_tabla VALUES (a,b,c)</code>
-C, -compress	Comprime los datos transferidos entre el cliente y el servidor si ambos admiten compresión.
-B, -databases	Vuelca varias bases de datos. No se pueden especificar nombres de tablas con esta opción ya que se vuelcan toda la base de datos. La salida coloca una instrucción <code>USE nombre_de_base_de_datos</code> delante de cada nueva base de datos.
-delayed	Inserta filas con el comando <code>INSERT DELAYED</code> en lugar de simplemente con <code>INSERT</code> .
-e, -extended-insert	Utiliza sintaxis la <code>INSERT</code> multilinea. El resultado es mas compacto y se ejecuta de manera mas rapida porque el búfer de índice se vacia solamente tras cada instruccidn <code>INSERT</code> .
-#, -debug[=cadena_de_opciones]	Realiza el seguimiento del uso del programa para fines de depuración.
- help	Muestra un mensaje de ayuda y sale.
-fields-terminated-by=...	Igual que las opciones <code>LOAD DATA INFILE</code> . Examine las secciones dedicadas a <code>SELECT INTO</code> y a <code>LOAD DATA INFILE</code> .
-fields-enclosed-by=...	Igual que las opciones <code>LOAD DATA INFILE</code> . Examine las secciones dedicadas a <code>SELECT INTO</code> y a <code>LOAD DATA INFILE</code> .
-fields-optionally-enclosed-by=...	Igual que las opciones <code>LOAD DATA INFILE</code> . Examine las secciones dedicadas a <code>SELECT INTO</code> y a <code>LOAD DATA INFILE</code> .
- fields-escaped-by=...	Igual que las opciones <code>LOAD DATA INFILE</code> . Examine las secciones dedicadas a <code>SELECT INTO</code> y a <code>LOAD DATA INFILE</code> .
-lines-terminated-by=...	Igual que las opciones <code>LOAD DATA INFILE</code> . Examine las secciones dedicadas a <code>SELECT INTO</code> y a <code>LOAD DATA INFILE</code> .

Opción	Descripción
-F, -flush-logs	Vacia el archivo de registro antes de iniciar el volcado.
-f, -force	Continúa con la operación de volcado aunque surjan errores de MySQL.
-h, -host=...	Vuelca datos desde el servidor MySQL encontrados en el anfitrión con nombre. El anfitrión predeterminado es localhost.
-l, -lock-tables	Bloquea todas las tablas antes de iniciar el volcado. Las tablas se bloquean con READ LOCAL, lo que permite inserciones simultáneas en las tablas MyISAM.
-K, -disable-keys	Los índices se deshabilitan antes de las instrucciones INSERT y se habilitan después, lo que agiliza los procesos de inserción.
-n, -no-create-db	No se incluirá una instrucción CREATE DATABASE en el resultado. Se suele hacer si se utiliza la opción -databases 0 - all-databases.
-t, -no-create-info	No incluye la instrucción CREATE TABLE, por lo que se asume que las tablas ya existen.
-d, -no-data	Solo devuelve la estructura de tabla y no incluye ninguna instrucción INSERT para la tabla.
-opt	Equivale a -quick -add-drop-table -add-locks -extended-insert -lock-tables. Esta opción agiliza el proceso de restauración.
-ppassphrase, -password [=contraseña]	Especifica que contraseña utilizar al establecer la conexión al servidor. Como de costumbre, si no se especifica la contraseña, se le pedirá que la haga, lo cual resulta más seguro.
-P número de puerto, -port=número de puerto	Especifica el número de puerto TCP/IP que utilizar al establecer la conexión al anfitrión. No se aplica en caso de conexiones al sistema local. Véase la opción -S.
-q, -quick	No almacena en búfer la consulta, sino que la vuelca directamente en stdout. Para ello, utiliza mysql_use_result(). Puede que necesite utilizar esta opción con tablas de gran tamaño.
-Q, -quote-names	Coloca los nombres de las tablas y de las columnas dentro de comillas simples.
-r, -result-file=...	Dirige los resultados directamente a un archivo dado. Esta operación resulta útil en DOS porque

Opción	Descripción
	evita que los caracteres <code>\n</code> de nueva línea de Unix se conviertan en una nueva línea así como el carácter de retorno del carro <code>\n\r</code> .
<code>-S /ruta/a/socket,</code> <code>- socket=/ ruta</code> <code>/a/socket</code>	Especifica el archivo de socket que utilizar al establecer la conexión al sistema local (el predeterminado).
<code>- tables</code>	Anula la opción <code>-B o - databases</code> .
<code>-T, -tab=ruta-a</code> <code>-algún-directorio</code>	Crea dos archivos para cada tabla: <code>nombre_de_tabla.sql</code> , que contiene las instrucciones <code>CREATE</code> , y <code>nombre_de_tabla.txt</code> , que contiene los datos. Esta opción solo funciona cuando <code>mysqldump</code> se ejecuta en el servidor.
<code>-u nombre_de_usuario,</code> <code>- user= nombre_de_usuario</code>	Especifica que nombre de usuario utilizar al establecer la conexión al servidor. El valor predeterminado es su nombre de inicio de sesión de Unix.
<code>-O var=opción,</code> <code>- set-variable</code> <code>var= opcion</code>	Establece el valor de una variable.
<code>-v, -verbose</code>	Hace que MySQL muestre más información sobre el proceso de <code>mysqldump</code> .
<code>-V, -version</code>	Muestra información de versión y sale.
<code>-w, - where=</code> <code>'condicion- where'</code>	Muestra únicamente los registros que satisfacen la condición <code>where</code> . Esta condición debe encerrarse entre comillas.
<code>-X, -xml</code>	Vuelca la base de datos como XML bien formado.
<code>-x, -first-esclavo</code>	Bloquea todas las tablas de todas las bases de datos.
<code>-O net_buffer_length=n</code>	Crea filas de tamaño <code>n</code> al crear instrucciones de inserción multifila (con las opciones <code>-e o - opt</code>). <code>n</code> debe ser inferior a 16MB y la variable <code>max_allowed_packet</code> de <code>mysqld</code> debe ser mayor que <code>n</code> .

Puede utilizar `mysqldump` de tres formas principalmente.

```
% mysqldump [OPCIONES] database [tabla]
```

o

```
% mysqldump [OPCIONES] --databases [OPCIONES] DB1 [DB2 DB3...]
```

O

```
% mysqldump [OPCIONES] --all-databases [OPCIONES]
```

Los siguientes ejemplos muestran algunas de las opciones disponibles. En concreto, el siguiente ejemplo vuelca todas las tablas en la base de datos `firstdb`.

```
mysqldump firstdb > /db_backups/firstdb_2002_11-12.sql
```

La opción `-v` aumenta la **cantidad de información** que se devuelve a lo largo del proceso, lo cual puede resultar útil para la **operación de depuración** si surgen problemas:

```
% mysqldump -v firstdb customer > /db_backups/customer_2002_11-12.sql
# Connecting to localhost...
# Retrieving table structure for table customer...
# Sending SELECT query...
# Retrieving rows...
# Disconnecting from localhost...
-
```

El siguiente ejemplo utiliza `where` para limitar el volcado de los registros con `id` mayor que 5:

```
% mysqldump --where='id>5' firstdb customer > /db_backups/customer_2002_11-12.sql
```

El resultado presentará este aspecto:

```
#
# Dumping data for table 'customer'
# WHERE: id>5
#
INSERT INTO customer VALUES (7,'Winnie','Dlamini',NULL);
INSERT INTO customer VALUES (6,'Neil','Beneke',NULL);
INSERT INTO customer VALUES (10,'Breyton','Tshabalala','B');
```

La opción `-e` permite inserciones más rápidas:

```
% mysqldump -e firstdb customer > /db_backups/customer_2002_11-12.sql
```

Este ejemplo utiliza la **instrucción INSERT multilínea**, como se puede observar examinando el **archivo de texto**:

```
#
# Dumping data for table 'customer'
#
INSERT INTO customer VALUES
(1,'Yvonne','Clegg','X'),
(2,'Johnny','Chaka-Chaka','B'),
(3,'Winston','Powers','M'),
```

```
(4, 'Patricia', 'Mankunku', 'C'),
(5, 'Francois', 'Papo', 'P'),
(7, 'Winnie', 'Dlamini', NULL),
(6, 'Neil', 'Beneke', NULL), (10, 'Breyton', 'Tshabalala', 'B');
```

Como solo existe una instruccion INSERT, el bufer de índice se vacia únicamente una vez, operación que resulta mas rapida que tener que hacerlo tras cada inserción.

Copias de seguridad con SELECT INTO

Otra forma de realizar una copia de seguridad consiste en utilizar SELECT INTO. Esta instruccion resulta similar a mysqldump en que crea un archivo que se utiliza para volver a crear la tabla de volcado. Resulta además opuesta a la instruccion LOAD DATA INTO. El archivo resultante sólo se puede crear en el servidor MySQL, no en ningun otro sistema. Su sintaxis es la siguiente:

```
SELECT INTO OUTFILE 'ruta_y_nombre_de_archivo'
```

Se puede utilizar cualquier instruccion SELECT para crear un archivo.

Para crear una copia de seguridad de la tabla `customer`, necesitara utilizar la siguiente secuencia, primero en Unix:

```
mysql> SELECT * FROM customer INTO OUTFILE '/db_backups/
customer.dat';
Query OK, 8 rows affected (0.00 sec)
```

y despues en Windows:

```
mysql> SELECT * FROM customer INTO OUTFILE
'c:\\db_backups\\bdb.dat';
Query OK, 8 rows affected (0.33 sec)
```

De nuevo, necesitara poner atencion al hacerlo. A continuación se recoge un error habitual:

```
mysql> SELECT * FROM customer INTO OUTFILE '/db_backups/
customer.dat';
ERROR 1086: File '/db_backups/customer.dat' already exists
```

No puede sobrescribir un archivo existente (lo que brinda cierto grado de seguridad, ya que un sistema mal configurado puede permitir que se sobrescriban archivos vitales).

Otro error habitual es el siguiente, desde Windows:

```
mysql> SELECT * FROM customer INTO OUTFILE
'c:\\db_backups\\customer.dat';
ERROR 1: Can't create/write to file 'c:db_backupscustomer.dat'
(Errorcode: 2)
```


El mensaje de error resulta **bastante** claro en este caso: MySQL no puede escribir en este directorio porque hemos olvidado introducir los caracteres \ de escape. En Windows, \ es el caracter de escape y como tambien **forma parte** de la ruta de Windows, **se necesita** acompaiiar del caracter de escape cuando se utiliza en dicho contexto. En Unix, un error similar es habitual:

```
mysql> SELECT * FROM customer INTO OUTFILE
'\db_backups\customer.dat';
Query OK, 8 rows affected (0.18 sec)
```

En este caso, sin embargo, MySQL ni siquiera nos avisa del error. Alguien procedente del **entorno** de Windows **podría** facilmente colocar las barras en el **sentido** erroneo y no lograr la copia de seguridad. Verifique siempre que se ha creado la copia de seguridad.

Si examinamos el **archivo** en cualquier editor de texto (como vi o el Bloc de **notas**), veremos la siguiente secuencia:

```
1      Yvonne  Clegg    X
2      Johnny Chaka-Chaka    B
3      Winston Powers  M
4      Patricia          Mankunku    C
5      Francois         Papo      P
7      Winnie  Dlamini  \N
6      Neil    Beneke   \N
10     Breyton Tshabalala    B
```

Se utilizan tabuladores para separar los campos y nuevas **líneas** para separar los registros, que son las mismas opciones predeterminadas utilizadas por LOAD DATA INTO. Podemos cambiar estos valores agregando opciones al final de la **instrucción**. A **continuación** se muestra el **conjunto completo** de opciones de SELECT INTO (y LOAD DATA INTO):

```
[FIELDS
  [TERMINATED BY '\t']
  [[OPTIONALLY] ENCLOSED BY '']
  [ESCAPED BY '\\']
]
[
  [LINES TERMINATED BY '\n']
]
```

Seguidamente se ilustran algunos ejemplos de uso de opciones no **predetermi-** nadas con SELECT INTO y los **archivos** de texto resultantes:

```
mysql> SELECT * FROM customer INTO OUTFILE '/db_backups/
customer2.dat'
  FIELDS TERMINATED BY 'zz';
Query OK, 8 rows affected (0.00 sec)
1zzYvonnezzCleggzzX
2zzJohnnyzzChaka-ChakazzB
3zzWinstonzzPowerszzM
4zzPatriciazzMankunkuzzC
5zzFrancoiszzPapozzP
```

```
7zzWinniezzDlaminizz\n
6zzNeilzzBenekezz\n
10zzBreytonzzTshabalalazzB
```

El caracter de tabulacion predeterminado se ha sustituido por los caracteres zz entre cada campo.

ADVERTENCIA: Los caracteres zz se utilizan aquí para crear un punto. Resulta peligroso utilizar caracteres ordinarios como éstos para funciones de terminación. Si el texto contiene la frase zz z, los campos quedarán mal alineados, ya que MySQL creará que los dos primeros son terminadores. Utilice caracteres convencionales como tabuladores, nuevas líneas o barras verticales (|) como terminadores.

La siguiente instrucción crea una línea larga:

```
mysql> SELECT * FROM customer INTO OUTFILE '/db_backups/
customer3.dat'
  FIELDS TERMINATED BY '|' LINES TERMINATED BY '[end]';
Query OK, 8 rows affected (0.00 sec)
```

Los datos se visualizan de la siguiente forma:

```
1|Yvonne|Clegg|X[end]2|Johnny|Chaka-Chaka|B[end]?
3|Winston|Powers|M[end]4|Patricia|Mankunku|C[end]?
5|Francois|Papo|P[end]7|Winnie|Dlamini|\N[end]?
6|Neil|Beneke|\N[end]10|Breyton|Tshabalala|B[end]
```

Los saltos de línea se sustituyen por caracteres [end]. En el siguiente ejemplo, la palabra clave ENCLOSED encierra todos los campos con los caracteres especificados:

```
mysql> SELECT * FROM customer INTO OUTFILE '/db_backups/
customer4.dat'
  FIELDS TERMINATED BY '|' ENCLOSED BY '"' LINES TERMINATED BY
'\n';
Query OK, 8 rows affected (0.00 sec)
```

Los datos se visualizan de la siguiente forma:

```
"1"|"Yvonne"|"Clegg"|"X"
"2"|"Johnny"|"Chaka-Chaka"|"B"
"3"|"Winston"|"Powers"|"M"
"4"|"Patricia"|"Mankunku"|"C"
"5"|"Francois"|"Papo"|"P"
"7"|"Winnie"|"Dlamini"|\N
"6"|"Neil"|"Beneke"|\N
"10"|"Breyton"|"Tshabalala"|"B"
```

La palabra clave OPTIONALLY sólo encierra campos de carácter (al igual que se encierran campos de carácter entre comillas simples al agregar registros pero no al añadir campos numéricos).

Por ejemplo:

```
mysql> SELECT * FROM customer INTO OUTFILE '/db_backups/
customer5.dat'
  FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES
TERMINATED
  BY '\n';
Query OK, 8 rows affected (0.00 sec)
```

El primer campo de datos (de tipo INT) no aparece encerrado entre comillas:

```
1|"Yvonne"|"Clegg"|"X"
2|"Johnny"|"Chaka-Chaka"|"B"
3|"Winston"|"Powers"|"M"
4|"Patricia"|"Mankunku"|"C"
5|"Francois"|"Papo"|"P"
7|"Winnie"|"Dlamini"|\N
6|"Neil"|"Beneke"|\N
10|"Breyton"|"Tshabalala"|"B"
```

Tambien puede hacer una copia de seguridad de un subconjunto de datos, utilizando una condición en la instrucción SELECT:

```
mysql> SELECT * FROM customer WHERE id<10 INTO OUTFILE
'/db_backups/customer6.dat' FIELDS TERMINATED BY '|' LINES
TERMINATED
  BY '\n';
Query OK, 7 rows affected (0.01 sec)
```

En el archivo sólo aparecen los siete registros aplicables:

```
1|Yvonne|Clegg|X
2|Johnny|Chaka-Chaka|B
3|Winston|Powers|M
4|Patricia|Mankunku|C
5|Francois|Papo|P
7|Winnie|Dlamini|\N
6|Neil|Beneke|\N
```

Restauracion de una tabla con LOAD DATA

Para restaurar una tabla creada con SELECT INTO, se utiliza la instrucción LOAD DATA. Tambien puede utilizar esta instrucción para agregar datos creados de otra forma, quizás una aplicacion o una hoja de calculo. Se trata de la forma mas rapida de agregar datos, en especial grandes cantidades de ellos. Su sintaxis es la siguiente:

```
LOAD DATA [LOW-PRIORITY | CONCURRENT] [LOCAL] INFILE
'nombre_de_archivo'
  [REPLACE | IGNORE]
  INTO TABLE nombre_tbl
  [FIELDS
```

```

[TERMINATED BY '\t']
[[OPTIONALLY] ENCLOSED BY '']
[ESCAPED BY '\\']
]
[LINEs TERMINATED BY '\n']
[IGNORE numero LINES]
[(nombre_col,...)]

```

Vamos a eliminar los datos de la tabla `customer` y a restaurarlos utilizando `LOAD DATA`:

```

mysql> TRUNCATE customer;
Query OK, 0 rows affected (0.02 sec)

```

Para restaurar la tabla en Unix, utilice:

```

mysql> LOAD DATA INFILE '/db_backups/customer.dat' INTO TABLE
customer;
Query OK, 8 rows affected (0.01 sec)
Records: 8 Deleted: 0 Skipped: 0 Warnings: 0

```

Y para restaurarla en Windows, utilice:

```

mysql> LOAD DATA INFILE 'c:\\db_backups\\customer.dat' INTO
TABLE customer;
Query OK, 8 rows affected (0.01 sec)
Records: 8 Deleted: 0 Skipped: 0 Warnings: 0

```

Como puede observar, los datos se han restaurado de manera satisfactoria:

```

mysql> SELECT * FROM customer;
+-----+-----+
| id | first-name | surname      | initial |
+-----+-----+
| 1 | Yvonne     | Clegg        | X       |
| 2 | Johnny     | Chaka-Chaka | B       |
| 3 | Winston    | Powers       | M       |
| 4 | Patricia   | Mankunku     | C       |
| 5 | Francois   | Papo         | P       |
| 7 | Winnie     | Dlamini      | NULL    |
| 6 | Neil       | Beneke       | NULL    |
| 10 | Breyton    | Tshabalala   | B       |
+-----+-----+
8 rows in set (0.00 sec)

```

¿Qué ocurriría si algo sale mal?

Algo puede salir mal por varias razones:

- Si esta intentando utilizar `LOAD DATA` sin éxito, es probable que no disponga de permisos para leer un archivo del servidor. El usuario que realice la operación de cargar los datos necesita disponer del privilegio `FILE` (consulte un capítulo posterior) y es necesario que el archivo se

encuentre en el directorio de la base de datos o que **todo** el mundo pueda leerlo.

- Un error común consiste en no utilizar **los** mismos terminadores y **caracteres** de cierre. **Deben** ser exactamente iguales a **los** utilizados en el archivo de datos (o especificados en la **instrucción** `SELECT INTO`). De lo contrario **todo** pareciera funcionar, **pero** la tabla se generara sin datos o llena de valores `NULL`. Consulte la siguiente **sección** para obtener mas **información**.
- Si esta utilizando la **palabra** clave `LOCAL` y ha iniciado **MySQL** con la **opción** `-local-infile=0`, no funcionara (consulte un capítulo posterior).
- Si el nombre de la ruta y el del archivo no se ha especificado correctamente (recuerde utilizar el caracter de escape para nombres de ruta de Windows).

Uso de `LOAD DATA` con opciones

Vamos a restaurar copias de seguridad utilizando otras opciones:

```
mysql> LOAD DATA INFILE '/db_backups/customer2.dat' INTO TABLE
customer;
Query OK, 8 rows affected (0.01 sec)
Records: 8 Deleted: 0 Skipped: 0 Warnings: 32
```

Aunque parece funcionar, no restaura los datos correctamente:

```
mysql> SELECT * FROM customer;
+----+-----+-----+-----+
| id | first-name | surname | initial |
+----+-----+-----+-----+
| 1 | NULL      | NULL    | NULL    |
| 2 | NULL      | NULL    | NULL    |
| 3 | NULL      | NULL    | NULL    |
| 4 | NULL      | NULL    | NULL    |
| 5 | NULL      | NULL    | NULL    |
| 7 | NULL      | NULL    | NULL    |
| 6 | NULL      | NULL    | NULL    |
| 10 | NULL     | NULL    | NULL    |
+----+-----+-----+-----+
8 rows in set (0.00 sec)
```

El problema esta en que los terminadores no coinciden. Recuerde que `customer2.dat` se creó con la opción `FIELDS TERMINATED BY 'zz'`. Por lo tanto tendremos que restaurarlo de la misma forma:

```
mysql> TRUNCATE customer;
Query OK, 0 rows affected (0.00 sec)
mysql> LOAD DATA INFILE '/db_backups/customer2.dat' INTO TABLE
customer
FIELDS TERMINATED BY 'zz';
```

```

Query OK, 8 rows affected (0.01 sec)
Records: 8 Deleted: 0 Skipped: 0 Warnings: 0
mysql> SELECT * FROM customer;
+----+-----+-----+-----+
| id | first-name | surname      | initial |
+----+-----+-----+-----+
| 1 | Yvonne     | Clegg        | X       |
| 2 | Johnny     | Chaka-Chaka | B       |
| 3 | Winston    | Powers       | M       |
| 4 | Patricia   | Mankunku     | C       |
| 5 | Francois   | Papo         | P       |
| 7 | Winnie     | Dlamini      | NULL    |
| 6 | Neil       | Beneke       | NULL    |
| 10 | Breyton    | Tshabalala   | B       |
+----+-----+-----+-----+
8 rows in set (0.00 sec)

```

Lo mismo se aplica a la cláusula LINES TERMINATED BY utilizada para crear customer3.dat:

```

mysql> TRUNCATE customer;
Query OK, 0 rows affected (0.00 sec)
mysql> LOAD DATA INFILE '/db_backups/customer3.dat' INTO TABLE
customer
FIELDS TERMINATED BY '|' LINES TERMINATED BY '[end]';
Query OK, 8 rows affected (0.00 sec)
Records: 8 Deleted: 0 Skipped: 0 Warnings: 0
mysql> SELECT * FROM customer;
+----+-----+-----+-----+
| id | first-name | surname      | initial |
+----+-----+-----+-----+
| 1 | Yvonne     | Clegg        | X       |
| 2 | Johnny     | Chaka-Chaka | B       |
| 3 | Winston    | Powers       | M       |
| 4 | Patricia   | Mankunku     | C       |
| 5 | Francois   | Papo         | P       |
| 7 | Winnie     | Dlamini      | NULL    |
| 6 | Neil       | Beneke       | NULL    |
| 10 | Breyton    | Tshabalala   | B       |
+----+-----+-----+-----+
8 rows in set (0.00 sec)

```

La cláusula ENCLOSED BY también necesita agregarse si se ha utilizado, como en customer4.dat:

```

mysql> TRUNCATE customer;
Query OK, 0 rows affected (0.00 sec)
mysql> LOAD DATA INFILE '/db_backups/customer4.dat' INTO TABLE
customer
FIELDS TERMINATED BY '|' ENCLOSED BY '"' LINES TERMINATED BY
'\n';
Query OK, 8 rows affected (0.01 sec)
Records: 8 Deleted: 0 Skipped: 0 Warnings: 0

```

```
mysql> SELECT * FROM customer;
+----+-----+-----+-----+
| id | first-name | surname      | initial |
+----+-----+-----+-----+
| 1  | Yvonne     | Clegg        | X       |
| 2  | Johnny     | Chaka-Chaka | B       |
| 3  | Winston    | Powers       | M       |
| 4  | Patricia   | Mankunku     | C       |
| 5  | Francois   | Papo         | P       |
| 7  | Winnie     | Dlamini      | NULL    |
| 6  | Neil       | Beneke       | NULL    |
| 10 | Breyton    | Tshabalala   | B       |
+----+-----+-----+-----+
8 rows in set (0.00 sec)
```

Y por supuesto lo mismo se aplica a la cláusula `OPTIONALLY ENCLOSED`, utilizada para crear `customer5.dat`:

```
mysql> TRUNCATE customer;
Query OK, 0 rows affected (0.00 sec)
mysql> LOAD DATA INFILE '/db_backups/customer5.dat' INTO TABLE
customer
  FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '' LINES
  TERMINATED BY '\n';
Query OK, 8 rows affected (0.01 sec)
Records: 8 Deleted: 0 Skipped: 0 Warnings: 0
mysql> SELECT * FROM customer;
+----+-----+-----+-----+
| id | first-name | surname      | initial |
+----+-----+-----+-----+
| 1  | Yvonne     | Clegg        | X       |
| 2  | Johnny     | Chaka-Chaka | B       |
| 3  | Winston    | Powers       | M       |
| 4  | Patricia   | Mankunku     | C       |
| 5  | Francois   | Papo         | P       |
| 7  | Winnie     | Dlamini      | NULL    |
| 6  | Neil       | Beneke       | NULL    |
| 10 | Breyton    | Tshabalala   | B       |
+----+-----+-----+-----+
8 rows in set (0.00 sec)
```

También puede realizar una actualización desde un archivo volcado parcialmente, `customer6.dat`:

```
mysql> TRUNCATE customer;
Query OK, 0 rows affected (0.00 sec)
mysql> LOAD DATA INFILE '/db_backups/customer6.dat' INTO TABLE
customer
  FIELDS TERMINATED BY '|' LINES TERMINATED BY '\n';
Query OK, 7 rows affected (0.01 sec)
Records: 7 Deleted: 0 Skipped: 0 Warnings: 0
mysql> SELECT * FROM customer;
+----+-----+-----+-----+
| id | first-name | surname      | initial |
+----+-----+-----+-----+
| 1  | Yvonne     | Clegg        | X       |
| 2  | Johnny     | Chaka-Chaka | B       |
| 3  | Winston    | Powers       | M       |
| 4  | Patricia   | Mankunku     | C       |
| 5  | Francois   | Papo         | P       |
| 7  | Winnie     | Dlamini      | NULL    |
| 6  | Neil       | Beneke       | NULL    |
| 10 | Breyton    | Tshabalala   | B       |
+----+-----+-----+-----+
```

```

| id | first-name | surname      | initial |
+----+-----+-----+-----+
| 1 | Yvonne     | Clegg       | X       |
| 2 | Johnny     | Chaka-Chaka| B       |
| 3 | Winston    | Powers      | M       |
| 4 | Patricia   | Mankunku    | C       |
| 5 | Francois   | Papo        | P       |
| 7 | Winnie     | Dlamini     | NULL    |
| 6 | Neil       | Beneke      | NULL    |
+----+-----+-----+-----+
7 rows in set (0.00 sec)

```

¿Qué ocurría si nos damos cuenta de que hemos **cometido** un error y queremos restaurar la tabla entera? Si cargamos los datos inmediatamente desde un archivo que contenga un volcado **completo**, nos encontraríamos con el siguiente problema:

```

mysql> LOAD DATA INFILE '/db_backups/customer.dat' INTO
      TABLE customer;
ERROR 1062: Duplicate entry '1' for key 1
mysql> SELECT * FROM customer;
+----+-----+-----+-----+
| id | first-name | surname      | initial |
+----+-----+-----+-----+
| 1 | Yvonne     | Clegg       | X       |
| 2 | Johnny     | Chaka-Chaka| B       |
| 3 | Winston    | Powers      | M       |
| 4 | Patricia   | Mankunku    | C       |
| 5 | Francois   | Papo        | P       |
| 7 | Winnie     | Dlamini     | NULL    |
| 6 | Neil       | Beneke      | NULL    |
+----+-----+-----+-----+
7 rows in set (0.00 sec)

```

Tenemos un error de clave duplicada y el archivo deja de procesarse en dicho punto. Podríamos haber vaciado simplemente la tabla primero, **como** hemos estado **haciendo** hasta ahora con todas las restauraciones, **pero** si estamos intentando restaurar una tabla que ya contiene registros, es probable que no queramos borrar **todo** y empezar de nuevo. Las opciones clave a las que dirigir nuestra atención son REPLACE e IGNORE. La **última** ignora todas **las filas** que dupliquen una **fila** existente en un **índice exclusivo** o clave primaria. **Por lo tanto**, IGNORE resulta útil cuando sabemos que **los** registros no se han modificado y no queremos eliminar y restaurar todos **los** registros de nuevo:

```

mysql> LOAD DATA INFILE '/db_backups/customer.dat' IGNORE INTO
      TABLE customer;
Query OK, 1 row affected (0.00 sec)
Records: 8 Deleted: 0 Skipped: 7 Warnings: 0

```

Como puede ver, de las **ocho** filas, siete se han **saltado** y **sólo** se ha insertado el registro que faltaba. En un archivo de mayor tamaño, lograríamos ahorrar una

gran cantidad de tiempo y evitar el inconveniente de no disponer de los datos temporalmente. Todos los registros estan ahora presentes de nuevo:

```
SELECT * FROM customer;
+-----+-----+-----+
| id | first-name | surname      | initial |
+-----+-----+-----+
| 1 | Yvonne     | Clegg       | X       |
| 2 | Johnny     | Chaka-Chaka | B       |
| 3 | Winston    | Powers      | M       |
| 4 | Patricia   | Mankunku    | C       |
| 5 | Francois   | Papo        | P       |
| 7 | Winnie     | Dlamini     | NULL    |
| 6 | Neil       | Beneke      | NULL    |
| 10 | Breyton    | Tshabalala  | B       |
+-----+-----+-----+
8 rows in set (0.00 sec)
```

La palabra clave REPLACE resulta útil cuando los valores de los registros han cambiado y queremos restablecer los registros existentes en el disco. Para mostrar su uso, vamos a cometer el error habitual de actualizar todos los registros cuando solo queriamos actualizar uno, con lo que todos los apellidos pasan a tener el valor de Fortune:

```
mysql> UPDATE customer SET surname='Fortune';
Query OK, 8 rows affected (0.00 sec)
Rows matched: 8  Changed: 8  Warnings: 0
```

Nos damos cuenta del error al examinar la tabla:

```
mysql> SELECT * FROM customer;
+-----+-----+-----+
| id | first-name | surname      | initial |
+-----+-----+-----+
| 1 | Yvonne     | Fortune     | X       |
| 2 | Johnny     | Fortune     | B       |
| 3 | Winston    | Fortune     | M       |
| 4 | Patricia   | Fortune     | C       |
| 5 | Francois   | Fortune     | P       |
| 7 | Winnie     | Fortune     | NULL    |
| 6 | Neil       | Fortune     | NULL    |
| 10 | Breyton    | Fortune     | B       |
+-----+-----+-----+
8 rows in set (0.00 sec)
```

A continuación, vamos a restaurar la tabla utilizando la palabra clave REPLACE:

```
mysql> LOAD DATA INFILE '/db_backups/customer.dat' REPLACE INTO
TABLE customer;
Query OK, 16 rows affected (0.00 sec)
Records: 8 Deleted: 8 Skipped: 0 Warnings: 0
mysql> SELECT * FROM customer;
```

```

+-----+-----+-----+
| id | first-name | surname      | initial |
+-----+-----+-----+
| 1 | Yvonne     | Clegg       | X       |
| 2 | Johnny     | Chaka-Chaka | B       |
| 3 | Winston    | Powers      | M       |
| 4 | Patricia   | Mankunku    | C       |
| 5 | Francois   | Papo        | P       |
| 7 | Winnie     | Dlamini     | NULL    |
| 6 | Neil       | Beneke      | NULL    |
| 10 | Breyton    | Tshabalala  | B       |
+-----+-----+-----+
8 rows in set (0.00 sec)

```

LOAD DATA LOCAL es una opción que **permite cargar los contenidos** de un archivo que existe en el equipo cliente de MySQL al servidor de la base de datos.

LOW PRIORITY obliga al proceso de agregación de datos a esperar hasta que no quede ningún cliente leyendo la tabla (**como hace con una instrucción INSERT habitual**).

La **palabra clave CONCURRENT resulta útil** si seguimos queriendo que la tabla se lea. Permite que otros subprocesos lean la tabla MyISAM (pero ralentiza el proceso de LOAD DATA).

Aspectos de seguridad relacionados con LOAD DATA LOCAL

La posibilidad de realizar restauraciones desde un equipo cliente puede **resultar práctica pero entraña un riesgo de seguridad**. Alguien **podría utilizar LOAD DATA LOCAL para leer cualquier archivo al que tenga acceso** el usuario que este utilizando para establecer la conexión.

Se puede **hacer** creando una tabla y realizando una **operación** de lectura tras **cargar los datos**. Si estuviera estableciendo la **conexión** utilizando el mismo **usuario** que el servidor Web y tuviera derecho de acceso para ejecutar consultas, la **situación resultaría peligrosa**.

De **manera predeterminada**, MySQL **permite** el uso de LOAD DATA LOCAL. Para evitar el **peligro** y **deshabilitar a11** LOAD DATA LOCAL, **inicie** el servidor MySQL con la opción `-local-infile=0`. También podríamos compilar MySQL sin la opción `-enable-local-infile`.

Uso de mysqlimport en lugar de LOAD DATA

En lugar de LOAD DATA, que se **ejecuta** desde MySQL, puede **utilizar su equivalente de línea de comandos**, mysqlimport.

Su **sintaxis** es la siguiente:

```

% mysqlimport [opciones] nombre_de_la_base_de_datos
nombre_de_archivo1 [nombre_de_archivo2 ...]

```

Muchas de las opciones son las mismas que las disponibles para LOAD DATA. La tabla a la que importar los datos viene determinada por el nombre de archivo.

Para ello, mysqlimport elimina la extensión del nombre de archivo, de manera que customer.dat se importa dentro de la tabla customer.

Utilice mysqlimport para restablecer los datos de clientes, de la siguiente forma:

```
mysql> SELECT * FROM customer;
+----+-----+-----+-----+
| id | first-name | surname      | initial |
+----+-----+-----+-----+
| 1  | Yvonne     | Clegg        | X       |
| 2  | Johnny     | Chaka-Chaka | B       |
| 3  | Winston    | Powers       | M       |
| 4  | Patricia   | Mankunku     | C       |
| 5  | Francois   | Papo         | P       |
| 7  | Winnie     | Dlamini      | NULL    |
| 6  | Neil       | Beneke       | NULL    |
| 10 | Breyton    | Tshabalala   | B       |
+----+-----+-----+-----+
8 rows in set (0.00 sec)
mysql> TRUNCATE customer;
Query OK, 0 rows affected (0.01 sec)
mysql> exit
Bye
% mysqlimport firstdb /db_backups/customer.dat
firstdb.customer: Records: 8 Deleted: 0 Skipped: 0 Warnings: 0
[root@test data]# mysql firstdb;
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7 to server version: 4.0.1-alpha-
max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

```
mysql> SELECT * FROM customer;
+----+-----+-----+-----+
| id | first-name | surname      | initial |
+----+-----+-----+-----+
| 1  | Yvonne     | Clegg        | X       |
| 2  | Johnny     | Chaka-Chaka | B       |
| 3  | Winston    | Powers       | M       |
| 4  | Patricia   | Mankunku     | C       |
| 5  | Francois   | Papo         | P       |
| 7  | Winnie     | Dlamini      | NULL    |
| 6  | Neil       | Beneke       | NULL    |
| 10 | Breyton    | Tshabalala   | B       |
+----+-----+-----+-----+
8 rows in set (0.00 sec)
```

Y los datos se restauran.

La tabla 11.2 describe las opciones disponibles para mysqlimport.

Tabla 11.2. Opciones de mysqlimport

Opciones	Descripción
-c, -columns=...	Toma una lista separada por comas de los nombres de campo como argumento. Éstos se utilizan para crear un comando <code>LOAD DATA INFILE</code> .
-character-sets-dir=nombre	Indica a MySQL el directorio en el que se encuentran los conjuntos de caracteres.
-C, -compress	Comprime los datos transferidos entre el cliente y el servidor si ambos admiten compresión.
-#, -debug[=cadena_de_opciones]	Rastrea el uso de programas con fines de depuración.
-d, -delete	Vacía la tabla antes de importar el archivo de texto.
-fields-terminated-by=...	Igual que la opción <code>LOAD DATA INFILE</code> .
-fields-enclosed-by=...	Igual que la opción <code>LOAD DATA INFILE</code> .
-fields-optionally-enclosed-by=...	Igual que la opción <code>LOAD DATA INFILE</code> .
-fields-escaped-by=...	Igual que la opción <code>LOAD DATA INFILE</code> .
-lines-terminated-by=...	Igual que la opción <code>LOAD DATA INFILE</code> .
-f, -force	Continúa aunque MySQL detecte errores durante el volcado.
-help	Muestra un mensaje de ayuda y sale.
-h nombre_de_anfitrión, -host= nombre_de_anfitrión	Importa datos al servidor MySQL descubiertos en el anfitrión designado. El anfitrión predeterminado es el sistema local.
-i, -ignore	Los registros agregados que originen un error de clave duplicada se ignoran. Por regla general dan lugar a un error y obligan al proceso a detenerse en dicho punto.
-l, -lock-tables	Bloquea todas las tablas para escribir antes de procesar cualquier archivo de texto, que mantiene las tablas sincronizadas sobre el servidor.

Opciones	Descripción
<code>-L, -local</code>	Lee los archivos de entrada desde el equipo del cliente. Si se conecta al sistema local (opcion predeterminada), se asume que los archivos de texto estaran en el servidor.
<code>-ppassphrase, -password [=contraseña]</code>	Especifica que contraseña utilizar al establecer la conexión al servidor. Como de costumbre, si no se especifica la contraseña, se le pedira que lo haga, lo cual resulta mas seguro.
<code>-P numero de puerto, -port=número de puerto</code>	Especifica el número de puerto TCP/IP que utilizar al establecer la conexión al anfitrión. No se aplica en caso de conexiones al sistema local. Vease la opcion <code>-s</code> .
<code>-r, -replace</code>	Un registro que debe agregarse dara lugar a una clave duplicada y sustituirá al registro original de la misma clave. Por regla general, esto generara un error y obliga al proceso a detenerse en dicho punto.
<code>-s, -silent</code>	Visualiza mensajes solo cuando tienen lugar errores.
<code>-S /ruta/a/socket, -socket= / ruta /a/socket</code>	Especifica que archivo de socket utilizar al establecer la conexión al sistema local (el predeterminado).
<code>-u nombre_de_usuario, -user=nombre_de_usuario</code>	Especifica que nombre de usuario utilizar al establecer la conexión al servidor. El valor predeterminado es su nombre de inicio de sesion de Unix.
<code>-v, -verbose</code>	Hace que MySQL muestre mas información sobre el proceso de mysqldump.
<code>-V, -version</code>	Muestra información de version y sale.

Uso de mysqlhotcopy para realizar copias de seguridad

La utilidad `mysqlhotcopy` es una secuencia de comandos de Perl que facilita la creación de copias de seguridad. Todavía se encuentra en fase beta (consulte la documentación mas reciente para averiguar si todavía sigue en dicha fase al leer

estas líneas), por lo que es probable que no funcione correctamente en todas las situaciones. Resulta rápida y sencilla de utilizar y funciona cerrando y vaciando las tablas y copiando los archivos en el directorio especificado (vease tabla 11.3). Solo puede copiar los archivos a otro lugar del servidor. Su **sintaxis** es la siguiente:

```
% mysqlhotcopy databasename backup_directory_path
```

En la tabla 11.3 se describen las opciones de mysqlhotcopy.

Tabla 11.3. Opciones de mysqlhotcopy

Opciones	Descripción
-?, -help	Muestra una pantalla de ayuda y sale.
-u, -user=#	El nombre de usuario para establecer la conexión al servidor .
-p, -password=#	Contraseña para establecer la conexión al servidor .
-P, -port=#	Puerto que utilizar al establecer la conexión al servidor local .
-S, -socket=#	Socket que utilizar al establecer la conexión al servidor local .
-allowold	Si los archivos ya existen , mysqlhotcopy suele anular la operación. Esta opción adjunta la secuencia <code>_old</code> a los nombres de archivo y continua con la operación.
-keepold	Los archivos con nombres cambiados por la opción <code>-allowold</code> se suelen eliminar tras la operación. Esta opción los mantiene.
-noindices	Esta opción no incluye los archivos de índice en el volcado, lo que agiliza el proceso. Tras restaurar los archivos, los índices pueden volver a generarse con <code>myisamchk -rq</code> .
-method=#	Permite especificar si utilizar <code>cp</code> o <code>scp</code> para copiar los archivos.
-q, -quiet	Solo se muestran mensajes de error
-debug	Permite labores de depuración.
-n, -dryrun	Genera mensajes pero no realiza acciones.
-regex=#	Copia todas las bases de datos con nombres que coinciden con la expresión regular.
-suffix=#	Asigna un sufijo a los nombres de las bases de datos copiadas.

Opciones	Descripción
<code>-checkpoint=#</code>	Inserta entradas de punto de comprobacion en tabla de base de datos especificadas.
<code>-flushlog</code>	Vacia los registros una vez bloqueadas todas las tablas.
<code>-tmpdir=#</code>	Permite especificar un directorio temporal.

mysqlhotcopy obtiene sus opciones del cliente y los grupos mysqlhotcopy los agrupa en archivos de opcion.

Para restablecer una copia de seguridad realizada con mysqlhotcopy, sustituya los archivos en el directorio de datos, como si hubiera hecho las copias directamente.

Deben cumplirse una serie de requisitos para poder ejecutar mysqlhotcopy:

- Necesita poder ejecutar las secuencias de comandos Perl en su servidor de base de datos.
- mysqlhotcopy depende de las siguientes clases de Perl para poder ejecutarse:

```
Getopt::Long, Data::Dumper, File::Basename, File::Path, DBI y Sys::Hostname.
```

- Necesita escribir el acceso al directorio en el que esta intentando realizar la copia de seguridad.
- Necesita seleccionar los privilegios sobre la base de datos que esta volcando.
- Para vaciar la tabla, necesita volver a cargar los privilegios.

Uso del registro de actualizacion binario para restablecer la base de datos a su posición mas reciente

El registro de actualizacion binario es una forma ideal de restaurar la base de datos a un punto lo mas cercano posible a aquel en el que tuvo lugar el desastre (vease un capitulo anterior). El registro de actualizacion binario registra todos los cambios realizados sobre la base de datos. Este registro esta habilitado cuando se inicia MySQL con la opcion `-log-bin`. Puede especificar un nombre con `-log-bin = nombre de archivo`; de lo contrario el nombre predeterminado sera el nombre del equipo servidor, al que se adjunta `-bin`. Se crea un nuevo

archivo de registro **cada** vez que se reinicia el servidor, que se vacian los registros, que se actualiza el servidor o que se alcanza su tamaño máximo (que se establece en la variable `max_bin_log_size`).

Tras realizar una copia de seguridad con `mysqldump`, reinicie MySQL con la opción `-log-bin`.

Cuando llegue el momento indicado, restaure el archivo de `mysqldump` y, a continuación, utilice los archivos de registro binarios para devolver a la base de datos a su estado más reciente.

Por ejemplo, imagine que realizamos una última copia de seguridad del archivo `customer.dat`, que la restaura a los 10 registros que se muestran a continuación:

```
mysql> SELECT * FROM customer;
+----+-----+-----+-----+
| id | first-name | surname      | initial |
+----+-----+-----+-----+
| 1  | Yvonne     | Clegg       | X       |
| 2  | Johnny    | Chaka-Chaka | B       |
| 3  | Winston   | Powers      | M       |
| 4  | Patricia  | Mankunku    | C       |
| 5  | Francois  | Papo        | P       |
| 7  | Winnie    | Dlamini     | NULL    |
| 6  | Neil      | Beneke      | NULL    |
| 10 | Breyton   | Tshabalala  | B       |
+----+-----+-----+-----+
8 rows in set (0.00 sec)
```

Llegados a este punto (justo después de realizar la copia de seguridad), inicie el servidor con la función de registro binario habilitada si no lo ha hecho todavía:

```
C:\MySQL\bin> mysqladmin shutdown
020601 23:59:01 mysqld ended
```

Si no la tiene todavía, incluya la siguiente opción dentro de su archivo `my.cnf` o `my.ini` para habilitar el registro binario.

```
log-bin
```

A continuación reinicie el servidor:

```
C:\MySQL\bin> mysqld-max
020602 18:58:21 InnoDB: Started
C:\MySQL\bin> mysql firstdb;
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 4.0.1-alpha-max-log
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> INSERT INTO customer VALUES(11,'Robin','McKenzie',NULL);
Query OK, 1 row affected (0.00 sec)
```


A continuacion, vamos a simular un desastre para lo cual detendremos el servidor y eliminaremos los archivos de datos y de indices de clientes:

```
mysql> exit
Bye
C:\MySQL\bin> del c:\MySQL\data\firstdb\customer.*
```

Puede que no disponga de permisos para eliminar los archivos si no cierra el servidor o inicia la sesion como usuario raiz.

Si elimina los archivos con la conexión todavía activa e intenta realizar una consulta sobre la tabla de clientes, es posible que siga obteniendo resultados debido a que esten almacenados en cache. Pero si cierra el servidor y vuelve a iniciarlo, no encontrara ningún dato sobre clientes:

```
C:\MySQL\bin> mysqladmin shutdown
020601 23:59:01 mysqld ended
C:\MySQL\bin> mysqld-max
020602 18:58:21 InnoDB: Started
C:\MySQL\bin> mysql firstdb;
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 4.0.1-alpha-
max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> SELECT * FROM customer;
ERROR 1146: Table 'firstdb.customer' doesn't exist
mysql> exit
Bye
```

A continuacion restaure la copia de seguridad realizada anteriormente:

```
C:\MySQL\bin> copy c:\db_backups\customer.*
c:\MySQL\data\firstdb
```

Con ayuda de una consulta, descubriera que se han perdido los datos mas recientes agregados tras realizar la copia de seguridad:

```
C:\MySQL\bin> mysql firstdb;
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 4.0.1-alpha-
max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> SELECT * FROM customer;
+----+-----+-----+-----+
| id | first-name | surname      | initial |
+----+-----+-----+-----+
| 1 | Yvonne     | Clegg        | X       |
| 2 | Johnny    | Chaka-Chaka | B       |
| 3 | Winston   | Powers       | M       |
| 4 | Patricia  | Mankunku    | C       |
| 5 | Francois  | Papo         | P       |
```

```

| 7 | Winnie      | ( Dlamini      | NULL | | |
| 6 | Neil        | | Beneke        | NULL | |
| 10 | Breyton     | | Tshabalala    | B    | |
+---+-----+-----+-----+
8 rows in set (0.00 sec)

```

Para restaurar la copia de seguridad, necesitamos utilizar el registro de actualización binario. En primer lugar, vamos a examinar que hay en el registro de actualización binario. No se trata de un archivo de texto, por lo que no podemos utilizar un editor de textos ordinario. Sin embargo, MySQL incorpora una utilidad, `mysqlbinlog`.

Si ejecuta esta utilidad en uno de los archivos de registro binario se generaran los contenidos del archivo.

Su **sintaxis** es la siguiente:

```
mysqlbinlog ruta_al_registro_de actualización_binario
```

Veamos que contiene el registro:

```

C:\MySQL\bin>mysqlbinlog ..\data\speed_demon-bin.001
# at 4
#020602 18:58:21 server id 1 Start: binlog v 2, server v
#4.0.1-alpha-max-log
created 020602 18:58:21
# at 79
#020602 19:01:11 server id 1 Query thread_id=2 exec_time=0
#error_code=0
use firstdb;
SET TIMESTAMP=1023037271;
INSERT INTO customer VALUES(11,'Robin','McKenzie');
# at 167
#020602 19:01:48 server id 1 Stop

```

Si ha estado ejecutando la función de registro binario de actualización es probable que tenga muchos archivos de registro. Seleccione el segundo más reciente que haya capturado la **última instrucción** INSERT.

Obviamente, el resultado no se ve muy bien en pantalla. Puede dirigirlo **hacia** su base de datos correspondiente de la siguiente forma:

```
C:\MySQL\bin>mysqlbinlog ..\data\speed_demon-bin.001 | mysql
firstdb
```

A **continuación**, puede visualizar la tabla y ver los registros restaurados.

```

C:\MySQL\bin> mysql firstdb;
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 4.0.1-alpha-
max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> SELECT * FROM customer;

```

```

+-----+-----+-----+
| id | first-name | surname      | initial |
+-----+-----+-----+
| 1  | Yvonne     | Clegg       | X       |
| 2  | Johnny    | Chaka-Chaka | B       |
| 3  | Winston   | Powers      | M       |
| 4  | Patricia  | Mankunku   | C       |
| 5  | Francois  | Papo        | P       |
| 7  | Winnie    | Dlamini     | NULL    |
| 6  | Neil      | Beneke      | NULL    |
| 10 | Breyton   | Tshabalala | B       |
| 11 | Robin     | McKenzie    | NULL    |
+-----+-----+-----+
9 rows in set (0.00 sec)

```

**El registro se ha restaurado correctamente.
La tabla 11.4 describe las opciones de mysqlbinlog**

Tabla 11.4. Opciones de mysqlbinlog

Opción	Descripción
-?, -help	Muestra la ayuda y sale.
-d, -database=nombrebd	Solo enumera las entradas correspondientes a la base de datos especificada.
-s, -short-form	Muestra únicamente las consultas, ninguna otra información.
-o, -offset=N	Salta un serie de entradas comenzando por el principio y especificadas con N.
-h, -host=servidor	Obtiene el registro binario desde el servidor especificado.
-P, -port=puerto	Utiliza el puerto especificado para conectar al servidor remoto.
-u, -user=nombre_de_usuario	Nombre de usuario para establecer la conexión al servidor.
-p, -password=contraseña	Contraseña para establecer una conexión al servidor.
-r, -result-file=archivo	Coloca el resultado en un archivo especificado.
-j, -position=N	Comienza leyendo el registro binario en la posición N.
-t, -table=nombre	Obtiene el volcado de la tabla original utilizando COM_TABLE_DUMB.
-V, -version	Muestra la version y sale.

Copia de seguridad y restauracion de tablas InnoDB

Actualmente es imposible realizar una copia de seguridad online estandar de una tabla InnoDB mientras el servidor esta en ejecucion con la **distribución estándar**. La **situación** no tardara en cambiar, por lo que es aconsejable que consulte la **documentación de MySQL** de manera regular.

No obstante, puede adquirir una herramienta que **permite** registros online de tablas InnoDB, llamada **InnoDB HotBackup**. Si desea obtener **detalles al respecto**, dirijase a la **dirección** www.innodb.com/hotbackup.html.

Por **regla** general, para realizar una copia de seguridad, es necesario apagar el servidor o impedir el acceso a los clientes. **Existen dos formas** de realizar una copia de seguridad y en **caso** de que los datos **resulten** vitales deberia utilizar ambos metodos. Una consiste en utilizar `mysqldum` (la misma que se utiliza para las tablas **MyISAM**), sin **permiso** de acceso de escritura durante la **operación**. Este **método** crea un archivo de texto con las instrucciones necesarias para restaurar las tablas.

La segunda consiste en **hacer** copias de los archivos de base de datos binarios. Para **ello**, necesitara cerrar la base de datos sin **errores** y copiar los archivos de datos, los archivos de registro InnoDB, el archivo de configuracion (el archivo `my.cnf` o `my.ini`) y los archivos de **definición** (`.frm`) en un lugar seguro.

```
% mysqladmin shutdown
% ls -l
total 76145
drwx--- 2 mysql mysql 2048 Jun 1 21:01 firstdb
-rw-rw- 1 mysql mysql 25088 May 4 20:08
ib_arch_log_0000000000
-rw-rw- 1 mysql mysql 5242880 Jun 1 21:04
ib_logfile0
-rw-rw- 1 mysql mysql 5242880 May 4 20:08
ib_logfile1
-rw-rw- 1 mysql mysql 67108864 Jun 1 21:04 ibdata1
drwxrwx- 2 mysql mysql 1024 May 4 20:07 mysql
drwxrwx- 2 mysql mysql 1024 Dec 23 17:44 test
-rw-rw- 1 mysql mysql 98 May 19 15:03 test-
bin.001
-rw-rw- 1 mysql mysql 30310 Jun 1 21:04 test-
bin.002
-rw-rw- 1 mysql mysql 30 May 19 15:09 test-
bin.index
-rw-r-r- 1 mysql mysql 7292 Jun 1 21:04
test.dummysql.co.za.err
```

Deberia copiar todos los archivos desde el directorio de datos que comiencen por `ib`, ya que se trata de los registros y los datos InnoDB.

Por ejemplo:

```
% cd /usr/local/mysql/data/  
% cp ib*/db_backups/
```

A continuacion copie los archivos de configuracion (recuerde copiarlos todos si tiene mas de uno):

```
% cp /etc/my.cnf /db_backups/
```

A continuacion copie los archivos de definicion, en este caso `innotest` dentro del directorio `firstdb` (todos los archivos de definicion asi como los archivos de `indice` y de datos `MySQL` se incluyen dentro de un directorio con el mismo nombre de la base de datos):

```
% cp firstdb/innotest.frm /db_backups/
```

A continuacion, vamos a reiniciar el servidor para que un usuario `malintencionado` pueda destruir los datos:

```
% mysqld-max  
% Starting mysqld daemon with databases from /usr/local/mysql/  
data  
% mysql firstdb  
mysql> TRUNCATE innotest;  
Query OK, 11 rows affected (0.00 sec)
```

Todos los datos se han eliminado. Su telefono comenzara a sonar dentro de un momento. Ha llegado el momento de restaurar la copia de seguridad. Nuevamente, tendra que apagar el servidor para evitar interferencias:

```
% mysqladmin shutdown  
020601 21:20:34 mysqld ended  
% cp /db_backups/ib* /usr/local/mysql/data/  
cp: overwrite '/usr/local/mysql/data/ib_arch_log_0000000000'? y  
cp: overwrite '/usr/local/mysql/data/ib_logfile0'? y  
cp: overwrite '/usr/local/mysql/data/ib_logfile1'? y  
cp: overwrite '/usr/local/mysql/data/ibdata1'? y
```

En este caso no hay necesidad de restaurar los archivos de configuracion o de definicion, ya que no se han dañado. En caso de que tenga lugar un fallo de hardware, necesitara restaurar estos tambien.

```
% mysqld-max  
% Starting mysqld daemon with databases from /usr/local/mysql/  
data  
% mysql firstdb  
mysql> SELECT * FROM innotest;  
+----+----+  
| f1 | f2 |  
+----+----+  
| 1 | NULL |  
| 2 | NULL |  
| 3 | NULL |
```

```

|    4 | NULL |
|    5 | NULL |
|    6 | NULL |
|    7 | NULL |
|    8 | NULL |
|    9 | NULL |
|   10 | NULL |
+-----+

```

10 rows in set (0.12 sec)

Los datos se han restaurado correctamente. En caso de que se produzca una caída del servidor, para restaurar los datos InnoDB, sólo necesitará reiniciarlo. Si están activadas las funciones de registro y almacenamiento generales (lo que resulta recomendable), las tablas InnoDB se restaurarán automáticamente a partir de los registros MySQL (los registros MySQL son los registros ordinarios, no los registros InnoDB). Todas las transacciones confirmadas presentes en el momento de la caída se desharán. El resultado presentará un aspecto parecido al siguiente:

```

InnoDB: Database was not shut down normally.
InnoDB: Starting recovery from log files...
InnoDB: Starting log scan based on checkpoint at
InnoDB: log sequence number 0 24785115
InnoDB: Doing recovery: scanned up to log sequence number 0
24850631
InnoDB: Doing recovery: scanned up to log sequence number 0
24916167
InnoDB: 1 uncommitted transaction(s) which must be rolled back
InnoDB: Starting rollback of uncommitted transactions
InnoDB: Rolling back trx no 982
InnoDB: Rolling back of trx no 98 completed
InnoDB: Rollback of uncommitted transactions completed
InnoDB: Starting an apply batch of log records to the
database...
InnoDB: Apply batch completed
InnoDB: Started
mysqld: ready for connections

```

TRUCO: Los archivos InnoDB no son tan portables como los archivos MyISAM. Sólo se pueden utilizar en otras plataformas si el equipo consta del mismo formato de coma flotante que el equipo en el que se genera. Esto significa, por ejemplo, que puede mover los archivos entre equipos Intel x86, independientemente del sistema operativo que esté utilizando.

Duplication como medio de realizar copias de seguridad

La **duplicación** es otra forma de mantener una copia de seguridad (consulte un capítulo posterior). Se protege contra fallos de hardware en una de las bases

duplicadas, **pero** no contra la estupidez o las malas intenciones. Si un usuario elimina un **conjunto** de registros, el proceso se replicara en otros servidores duplicados a **menos** que existe una **forma** fiable de volcado. Si utiliza la duplicacion, se reduciran las preocupaciones por los fallos de hardware **pero** seguira necesitando otro **método** de volcado.

Resumen

Las copias de seguridad son una **parte** fundamental de la caja de herramientas del administrador de MySQL. Se pueden utilizar varios metodos para implementarlas:

- La instruccion **BACKUP** crea una copia de los archivos de **definición** y de datos de la tabla **MyISAM**. La instruccion **RESTORE** **restaura** los datos.
- **Copia directa** los archivos. Necesitara aplicar manualmente los bloqueos. La operacion de devolver los archivos de datos **al** directorio de datos **restablece** los datos.
- Uso de **mysqldump**, que crea un archivo de texto que contiene las instrucciones **SQL** necesarias para regenerar la tabla. El uso de archivos como entrada para el demonio de MySQL **restaura** los datos.
- **El** uso de instrucciones **SELECT INTO** crea un archivo de texto que se puede utilizar para restaurar los datos con el **comando** **LOAD DATA** o la utilidad **mysqlimport**.
- Uso de la utilidad **mysqlhotcopy**. Se trata de una secuencia de comandos de **Perl** que copia los archivos de datos a otro directorio. La operacion de devolver los archivos de datos **al** directorio de datos **restaura** los datos.
- Uso de la duplicacion, que **vuelca** los datos en otro equipo, **pero** tambien duplica su perdida entre equipos si viene causada por instrucciones **SQL**.

El registro de **actualización** binario, si esta habilitado, mantiene un registro de todos los cambios en la base de datos. La utilidad **mysqlbinlog** se puede utilizar para visualizar los contenidos del registro o usarse para restaurar actualizaciones a la base de datos realizadas a partir de una copia de seguridad. Las tablas **InnoDB** no se almacenan en archivos, como las tablas **MyISAM** por lo que requieren un cuidado especial. **Además** tambien **constan** de su propio mecanismo de registro.

12

Duplicación de base de datos

MySQL dispone de una característica denominada *duplicación* que permite reflejar automáticamente una o varias bases de datos de un servidor (denominado *principal*) en uno o varios servidores (denominados *esclavos*). La duplicación resulta muy útil como estrategia de **creación** de copias de seguridad y como técnica de mejora del rendimiento. En este capítulo veremos como funciona la duplicación y le mostraremos la **forma** de configurarla.

En este capítulo veremos:

- Como configurar la duplicación
- Como configurar **archivos** principales y esclavos
- Las instrucciones SQL principales y esclavas

Que es la duplicación

La duplicación **funciona** de la siguiente **forma**. El servidor esclavo se inicia con una copia **exacta** de los datos del servidor principal. Tras **ello**, se activa el registro binario en el principal y el esclavo se conecta a este periódicamente y comprueba los cambios efectuados en el registro binario desde la última vez que

se conecto. Seguidamente, el esclavo repite de **forma automática** estas **instrucciones** en su servidor. El archivo `master.info` del esclavo **permite** realizar el seguimiento del **punto** en el que se encuentra en el registro binario del principal. La **relación** entre el registro binario principal y el archivo `master.info` esclavo es de gran relevancia: si no están sincronizados, **los datos** no serán idénticos en ambos servidores. La duplicación puede resultar muy útil para crear copias de seguridad (en función de **errores de disco**, no **errores humanos**) y para acelerar el rendimiento. Es un **método** muy **práctico** para ejecutar varias bases de datos, **sobre todo** en entornos en los que **las instrucciones SELECT superan a las instrucciones INSERT o UPDATE** (también podemos **optimizar** esclavos solamente para instrucciones SELECT y **hacer** que el principal se **encargue** de las instrucciones INSERT y UPDATE).

Sin embargo, la duplicación no es la solución a todos los problemas de **rendimiento**. Es necesario realizar actualizaciones en el esclavo y, aunque se realizan de **manera** óptima, si sus tablas **MyISAM** también llevan a **cabo** actualizaciones y se bloquean con frecuencia, una mejor solución consiste en **convertirlas a InnoDB**. Al mismo tiempo, también hay un retraso entre las actualizaciones que se **duplican** en los esclavos, cuya longitud depende de la capacidad de su red y de los propios servidores de bases de datos.

Por esta razón, su aplicación no puede simplemente asumir que puede utilizar el principal o el esclavo como servidor de bases de datos. Puede que un registro del principal no aparezca inmediatamente en el esclavo, lo que **podría** generar problemas en la aplicación.

Normalmente, la duplicación se lleva a **cabo** de **forma** jerárquica (como se indica en las figuras 12.1 y 12.2) pero se puede **configurar** de **forma** circular (como mostramos en las figuras 12.3 y 12.4). Su código cliente debe cerciorarse de que no haya conflictos ya que, en **caso** de haberlos, la aplicación puede fallar debido a las irregularidades, razón por la que las estructuras mostradas en las figuras 12.3 y 12.4 son poco habituales.

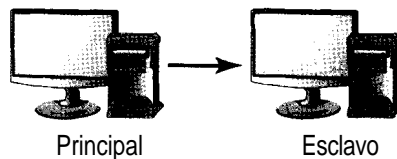


Figura 12.1. Un principal, un esclavo

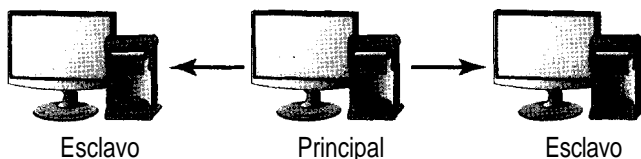


Figura 12.2. Un principal, varios esclavos

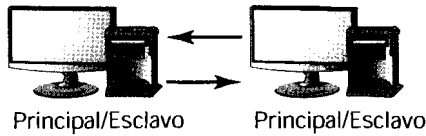


Figura 12.3. Relación circular principal/esclavo

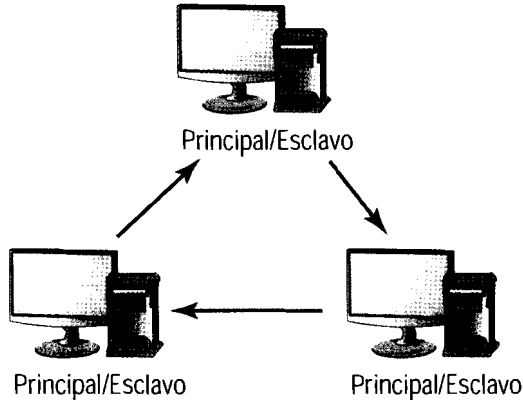


Figura 12.4. Cadena principal/esclavo

No es necesario que la **conexión** sea continua. Si se rompe el enlace por cualquier razón, el esclavo intentará volver a conectarse y, en el **momento** en que se restablezca el enlace, iniciará la **actualización** desde el **punto** en que se interrumpió.

Configuración de duplicación

Existen distintos tipos de configuración de una **relación** principal-esclavo, como veremos en los ejemplos descritos en este capítulo. Los pasos que debe seguir son los **mínimos** que necesita para iniciar la duplicación.

En el principal, siga los pasos descritos a **continuación**:

1. Defina un usuario de duplicación con el **permiso** **REPLICATION SLAVE**:

```
GRANT REPLICATION-SLAVE ON *.* TO replication-user IDENTIFIED BY
'replication-password';
```

2. Copie las tablas y los datos. Si la base de datos ya se ha utilizado y ya existe un registro binario (consulte un capítulo anterior), anote el desplazamiento inmediatamente después de la copia de seguridad (como veremos en un **apartado** posterior). La operación **LOAD DATA FROM MASTER** en el esclavo se encarga de este **paso**. Actualmente, esta operación **sólo** funciona con tablas **MyISAM** y es mejor utilizarla con pequeños conjuntos de datos o cuando los datos en el principal se puedan bloquear durante la operación. La versión 4.1 resuelve alguna de estas deficiencias.

3. Añada el siguiente código al archivo de configuración (`my.cnf` o `my.ini`). `log-bin` indica que el principal utilizará registros de actualizaciones binarias y `server-id` es un número exclusivo que identifica cada uno de los equipos principal y esclavos. Por convención, el principal se define como 1 y los esclavos desde 2 en adelante:

```
[mysqld]
log-bin
server-id=1
```

Realice estos pasos en el esclavo o esclavos:

1. Añada el siguiente código al archivo de configuración (`my.cnf` o `my.ini`). `master-hostname` es el nombre de anfitrión del principal y los valores `master-user` y `master-password` son el nombre de usuario y la contraseña, respectivamente, definidos en el principal para la duplicación (con el privilegio de duplicación esclavo). `master_TCP/IP` es el número de puerto con el que se comunica el principal (solamente se necesita si el puerto no es estándar) y el número exclusivo es un número que empieza por 2 hasta $2^{32}-1$:

```
[mysqld]
master-host=master_hostname
master-user=replication_user
master-password=replication_password
master-port=master_TCP/IP_port
server-id=unique_number
```

2. Copie los datos obtenidos del principal en el esclavo (si no ejecuta `LOAD DATA FROM MASTER`).
3. Guarde el servidor esclavo.
4. Si no ha obtenido los datos, utilice `LOAD DATA FROM MASTER` para acceder a los mismos.

Con los dos servidores en ejecución, ya puede iniciar la duplicación.

Opciones de duplicación

En la tabla 12.1 se describen las distintas opciones de duplicación disponibles para el principal y en la tabla 12.2, las que están disponibles para el esclavo.

Tabla 12.1. Opciones de archivo de configuración principal

Opción	Descripción
<code>log-bin=nombre_de_archivo</code>	Activa el registro binario. Es necesaria la presencia de esta opción en el principal para que tenga lugar la duplicación. El nombre de archivo es <code>op-</code>

Opción	Descripción
	cional. Para borrar el registro, ejecute <code>RESET MASTER</code> y no olvide ejecutar <code>RESET SLAVE</code> en todos los esclavos. De forma predeterminada, el registro binario se denomina <code>hostname.xxx</code> , siendo <code>xxx</code> un número que empieza en 001 y se incrementa en una unidad cada vez que se gira el registro.
<code>log-bin-index= nombre-de-archivo</code>	Especifica el nombre del archivo de índice del registro binario (que enumera en orden los archivos del registro binario, para que el esclavo siempre sepa cual está activo). La opción predeterminada es <code>hostname.index</code> .
<code>sql-bin-update-same</code>	Si se configura, al definir <code>SQL_LOG_BIN</code> como 1 o 0, automáticamente se define <code>SQL_LOG_UPDATE</code> con el mismo valor, y viceversa. <code>SQL_LOG_UPDATE</code> dejara de ser necesario, por lo que esta opción apenas se utiliza.
<code>binlog-do-db=nombre_de_base_de_datos</code>	Solamente registra las actualizaciones en el registro binario que provienen de la base de datos <code>nombre_de_base_de_datos</code> . Las bases de datos restantes se ignoran. También puede restringir las bases de datos del esclavo.
<code>binlog-ignore-db= nombre_de_base_de_datos</code>	Registra todas las actualizaciones en el registro binario a excepción de las que provienen de la base de datos <code>nombre_de_base_de_datos</code> . También puede configurar la base de datos para que ignore el esclavo.

Tabla 12.2. Opciones del archivo de configuración en el esclavo

Opción	Descripción
<code>master-host= anfitrión</code>	Especifica el nombre de anfitrión o dirección IP del principal al que se conecta. Es necesario configurarlo para iniciar la duplicación. Una vez iniciada, los datos <code>master.info</code> lo determinarán y será necesario utilizar una instrucción <code>CHANGE MASTER TO</code> para cambiarlo.
<code>master-user= nombre_de_usuario</code>	Especifica el nombre de usuario con el que el servidor esclavo se conecta al principal. El usuario debe tener permiso <code>REPLICATION SLAVE</code> en el principal. Una vez iniciada la duplicación, los datos <code>master.info</code> lo determinarán y será necesario

Opción	Descripción
	utilizar una instrucción <code>CHANGE MASTER TO</code> para cambiarlo.
<code>master-password=contrasefia</code>	Especifica la contraseña con la que el servidor esclavo se conecta al principal. El valor predeterminado es una cadena vacía. Una vez iniciada la duplicación , los datos <code>master.info</code> lo determinarán y será necesario utilizar una instrucción <code>CHANGE MASTER TO</code> para cambiarlo.
<code>master-port=número de puerto</code>	Especifica el número de puerto al que se conecta el servidor principal (de forma predeterminada es el valor de <code>MYSQL_PORT</code> , normalmente <code>3306</code>). Una vez iniciada la duplicación , los datos <code>master.info</code> lo determinarán y será necesario utilizar una instrucción <code>CHANGE MASTER TO</code> para cambiarlo.
<code>master-connect-retry=segundos</code>	Si la conexión entre el servidor principal y el esclavo se pierde. MySQL espera la cantidad de segundos establecida antes de intentar una nueva conexión (el valor predeterminado es <code>60</code>).
<code>master-ssl</code>	Determina que la duplicación utiliza SSL.
<code>master-ssl-key=nombre_de_clave</code>	Si se determina que se utilice SSL (la opción <code>master-ssl</code>), esta opción indica el nombre de archivo de clave SSL principal .
<code>master-ssl-cert=nombre_de_certificado</code>	Si se determina que se utilice SSL (la opción <code>master-ssl</code>), esta opción indica el nombre de certificado SSL principal .
<code>master-info-file=nombre_de_archivo</code>	Especifica el archivo de información principal (de forma predeterminada , <code>master.info</code> en el directorio de datos), que realiza el seguimiento del punto de duplicación en el que se encuentra el servidor esclavo en el registro binario.
<code>report-host</code>	Especifica el nombre de anfitrión o dirección IP con el que se presenta el servidor esclavo en el principal (se utiliza durante la instrucción <code>SHOW SLAVE HOSTS</code>). No aparece configurado de forma predeterminada .
<code>report-port</code>	Especifica el puerto que utiliza el puerto esclavo para conectarse al principal. Solamente necesita esta opción si el esclavo se encuentra en un puerto no estándar o si la conexión no se realiza de forma convencional .

Opción	Descripción
<code>replicate-do-table=nombre_de_bd.nombre_de_tabla</code>	Garantiza que el esclavo solamente duplica el nombre de la tabla especificada de la base de datos especificada. Puede utilizar esta opción varias veces para duplicar varias tablas.
<code>replicate-ignore-table=nombre_de_bd.nombre_de_tabla</code>	Indica al servidor esclavo que no duplique una instrucción que actualice la tabla especificada (incluso si hay otras tablas actualizadas por la misma instrucción). Puede especificar esta opción varias veces.
<code>replicate-wild-do-table=nombre_de_bd.nombre_de_tabla</code>	Indica al servidor esclavo que solamente duplique instrucciones que coincidan con una determinada tabla (similar a la opción <code>replicate-do-table</code>), pero tomando en cuenta cualquier comodín. Por ejemplo, cuando el nombre de una tabla es <code>db%.tb%</code> la coincidencia se aplica a cualquier base de datos que empiece por db y a cualquier tabla que empiece por tb .
<code>replicate-wild-ignore-table=nombre_de_bd.nombre_de_tabla</code>	Indica al servidor esclavo que no duplique una instrucción que actualice la tabla especificada, incluso si hay otras tablas actualizadas por la misma instrucción, similar a la opción <code>replicate-ignore-table</code> , a excepción de que se toman en consideración los comodines. Por ejemplo, cuando el nombre de la tabla es <code>db%.tb%</code> , no se realizará la duplicación cuando la base de datos comience por db y la tabla por tb . Puede especificar varias veces esta opción.
<code>replicate-ignore-db=nombre_de_base_de_datos</code>	Indica al servidor esclavo que no duplique ninguna instrucción cuando la base de datos actual sea <code>nombre_de_base_de_datos</code> . Puede utilizarse esta opción varias veces para ignorar múltiples bases de datos.
<code>replicate-do-db=nombre_de_base_de_datos</code>	Indica al subproceso esclavo que solamente duplique una instrucción cuando la base de datos sea <code>nombre_de_base_de_datos</code> . Puede utilizar varias veces esta opción para duplicar varias bases de datos.
<code>log-slave-updates</code>	Indica al esclavo que registre las actualizaciones en el registro binario. Opción no configurada de forma predeterminada. Si piensa utilizar el esclavo como principal en otro esclavo, tendrá que configurar esta opción.

Opción	Descripción
<code>replicate-rewrite-db=base_de_datos_principal-> base_de_datos_esclava</code>	Si la base de datos del esclavo tiene un nombre diferente a la del principal, tendrá que asignar la relación con esta opción.
<code>slave-skip-errors=[codigo_error1, codigo_error2,... all]</code>	Cuando la duplicación encuentra un error, se detiene (ya que un error implica que los datos no son consistentes y es necesario seguir una serie de pasos manuales). Esta opción indica a MySQL que prosiga con la duplicación si el error es uno de los que aparecen enumerados. Los códigos de error se identifican mediante un número (el mismo que aparece en el registro de errores) y aparecen separados por una coma. También puede utilizar la opción <code>all</code> para procesar cualquier posible error. Normalmente no debería utilizar esta opción ya que un uso incorrecto de la misma puede afectar a la sincronización de los datos con el principal. Si esto sucede, la única forma posible de volver a sincronizarlos es copiar de nuevo los datos principales .
<code>skip-slave-start</code>	Al configurar esta opción, la duplicación no comienza cuando se inicia el servidor. Puede iniciarla manualmente con el comando <code>SLAVE START</code> .
<code>slave_compressed_protocol=#</code>	Si se configura como 1, MySQL utiliza la compresión para transferir los datos entre el esclavo y el principal, en caso de ambos servidores admitan esta función.
<code>slave_net_timeout=#</code>	Determina los segundos que espera el principal antes de que se anule una lectura.

Comandos de duplicación

Es aconsejable que se familiarice con los comandos de duplicación, tanto en el servidor principal como en el esclavo. A continuación mostramos los comandos de duplicación en el esclavo:

- `SLAVE START` y `SLAVE STOP` **inician y detienen el proceso de duplicación, respectivamente.**
- `SHOW SLAVE STATUS` **devuelve información sobre el esclavo, incluyendo si esta conectado al principal** (`Slave_IO_Running`), **si la duplicación esta en marcha** (`SLAVE_SQL_Running`), **que registro binario se**

utiliza (`Master_Log_File` y `Relay_Master_Log_File`) y cuál es la **posición actual en el registro binario** (`Read_Master_Log_Pos` y `Exec_Master_Log_Pos`).

- La instrucción `CHANGE MASTER TO` es importante para **sincronizar** la duplicación o para **iniciarla** desde cero en el **punto exacto**. `MASTER_LOG_FILE` hace referencia **al** registro binario del principal desde el que **el esclavo** debe **iniciar** la duplicación y `MASTER_LOG_POS` la **posición** en dicho **archivo** (como veremos en **los ejemplos de capítulos** posteriores). Esta instrucción también se utiliza cuando falla el principal y es necesario cambiarlo por el principal **al** que se conecte el esclavo. El **conjunto completo** de opciones `CHANGE MASTER TO` es **el siguiente**:

```
CHANGE MASTER TO MASTER_HOST = 'master_hostname',
  MASTER_USER='replication_username',
  MASTER_PASSWORD='replication_user_password',
  MASTER_PORT='master_port',
  MASTER_LOG_FILE='master_binary_logfile',
  MASTER_LOG_POS='master_binary_log_position'
```

- La instrucción `RESET SLAVE` hace que **el esclavo olvide** su **posición** en los registros principales.
- `LOAD DATA FROM MASTER` copia **los datos** del principal y **los lleva** hasta el esclavo. En la actualidad, no resulta demasiado útil para conjuntos de datos de gran tamaño o en situaciones en las que el principal debe estar disponible durante largos periodos, ya que se produce un bloqueo de lectura global **al copiar los datos**. También actualiza el valor de `MASTER_LOG_FILE` y `MASTER_LOG_POS`. Actualmente **sólo funciona con tablas MyISAM**. Es muy **probable que**, en el **futuro**, esta **instrucción** se convierta en la **forma** estándar de preparar el esclavo, por lo que debe consultar la última **documentación** al respecto.
- La instrucción `GLOBAL SQL_SLAVE_SKIP_COUNTER=n` hace que **el esclavo ignore** las siguientes **n instrucciones del registro binario** del principal.

A continuación describimos los comandos de duplicación en el principal:

- La instrucción `SET SQL_LOG_BIN=n` desactiva el registro de **actualizaciones binarias** (si se **configura** como 0) o lo reactiva (si se configura como 1). Necesitará el **privilegio SUPER** para ejecutar esta instrucción.
- `RESET MASTER` **elimina** todos los registros binarios y empieza la **numeración** de nuevo desde 001.
- `SHOW MASTER STATUS` muestra el registro binario actual, la **posición** en el mismo y si se ha excluido alguna base de datos del registro binario.

`PURGE MASTER LOGS nombre_de_archivo_de_registro_binario` elimina todos los registros anteriores al registro binario especificado. Compruebe que ningún esclavo lo necesita antes de eliminarlo. En apartados posteriores veremos un ejemplo al respecto.

`SHOW MASTER LOGS` muestra la lista de archivos de registro binario disponibles. Normalmente se utiliza esta opción antes de aplicar la anterior.

- `SHOW SLAVE HOSTS` devuelve una lista de los esclavos registrados en el principal (de forma predeterminada, un esclavo no se registra a sí mismo, sino que requiere la configuración de la opción `report-host`).
- La instrucción `SHOW BINLOG EVENTS [IN 'nombre de registro'] [FROM pos] [LIMIT [desplazamiento,] filas]` lee instrucciones de los registros binarios.

Dificultades de la duplicación

A continuación enumeramos algunos de los aspectos fundamentales que debe tener en cuenta a la hora de configurar y ejecutar la duplicación:

- Las instrucciones `FLUSH` no se duplican, lo que le puede afectar si actualiza directamente las tablas de permiso en el principal y, tras ello, utiliza `FLUSH` para activar los cambios. Los cambios no serán efectivos hasta que también ejecute una instrucción `FLUSH` en dicho punto.
- Asegúrese de que tanto los principales como los esclavos tienen el mismo conjunto de caracteres.

La función `RAND()` no funciona correctamente cuando se pasa una expresión aleatoria como argumento. Puede utilizar algo como `UNIX_TIMESTAMP()`.

- La duplicación de consultas que actualizan datos y utilizan variables de usuario no es segura (aunque es probable que haya cambiado. Consulte su documentación).
- La duplicación suele funcionar con distintas versiones de MySQL, incluso entre la versión 3.23.x y la versión 4.0.x, pero hay excepciones (4.0.0, 4.0.1 y 4-02 no funcionan entre sí), razón por la que debe consultar en la documentación. En caso contrario, utilice las últimas versiones siempre que sea posible.

Duplicación de una base de datos

En este ejemplo, crearemos una nueva base de datos con una tabla y la duplicaremos en otro servidor. Para ello tendrá que disponer de dos servidores MySQL

operativos (preferiblemente de la misma version) y es necesario que ambos se comuniquen para poder **probar** este ejemplo.

En primer lugar, en el servidor principal, **cree** una base de datos con el nombre **replication_db**, una tabla con el nombre **replication-table** y aiiada datos a esta tabla, como mostramos a **continuación**:

```
mysql> CREATE DATABASE replication_db;
Query OK, 1 row affected (0.01 sec)
mysql> USE replication_db;
Database changed
mysql> CREATE TABLE replication-table(f1 INT,f2 VARCHAR(20));
Query OK, 0 rows affected (0.03 sec)
mysql> INSERT INTO replication-table (f1,f2) VALUES (1,'first');
Query OK, 1 row affected (0.03 sec)
```

Tras **ello**, conceda **permiso de duplicación** al servidor esclavo. El usuario esclavo sera **replication-user** y tendra la contraseiia **replicationpwd**:

```
mysql> GRANT REPLICATION SLAVE ON *.* TO replication-user
IDENTIFIED BY
' replication_pwd';
```

Cierre el servidor esclavo y aiiada el siguiente codigo al **archivo de configuracion** (my. cfg o my. ini). Sustituya el parametro **master-host** con la **direccion IP** de su servidor esclavo. **server_id** puede ser cualquier numero, siempre que no coincida con el del **server_id** del principal:

```
master-host      = 192.168.4.100
master-user      = replication-user
master_password = replication-pwd
server-id        = 3
replicate-do-db = replication_db
```

En el servidor esclavo, **cree** la base de datos **replication_db** y copie los datos **replication_table** desde el principal al esclavo (**como se trata de tablas MyISAM, los datos se encuentran en el directorio replication_db**). En capitulos anteriores **encontrará información sobre cómo hacerlo**. Cuando copie los **archivos** al servidor esclavo, compruebe que los permisos sean correctos (en Unix, **chown mysql.mysql ***, **chmod 700 ***). Al mismo tiempo, debe saber que si su servidor principal ya ha utilizado el registro binario, tendra que **restablecerlo con RESET MASTER** para que el esclavo pueda empezar la **operación de actualización** desde el principio del primer registro binario. Inicie el servidor esclavo y conectelo. Una vez conectado, compruebe su estado para ver si la **duplicacion se ha iniciado correctamente**:

```
mysql> SHOW SLAVE STATUS;
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

```

+-----+-----+
| Master-Host | Master-User | Master-Port |
| Connect-retry |
| Master-Log-File | Read-Master-Log-Pos | Relay-Log-File |
|
| Relay-Log-Pos | Relay-Master-Log-File | Slave-I0-Running |
| Slave_SQL_Running | Replicate-do-db | Replicate-ignore-db |
| Last-errno |
| Last-error | Skip-counter | Exec-master-log-pos |
| Relay-log-space |

```

```

+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| 192.168.4.100 | replication-user | 3306 | { 60 |
|
| g-bin.001 | 79 | | s-bin.002 |
| 124 | g-bin.001 | Yes | Yes |
| replication-db | | 0 |
| 0 |
| 79 | | 132 |

```

```

+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

```

mysql> INSERT INTO replication-table (f1,f2)
VALUES(2,'second');
Query OK, 1 row affected (0.06 sec)
mysql> SELECT * FROM replication-table;

```

```

+-----+-----+
| f1 | f2 |
+-----+-----+
| 1 | first |
| 2 | second |
+-----+-----+

```

```

2 rows in set (0.00 sec)

```

```

mysql> SHOW SLAVE STATUS;

```

```

+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| Master-Host | Master-User | Master-Port |
| Connect-retry |
| Master-Log-File | Read-Master-Log-Pos | Relay-Log-File |
|
| Relay-Log-Pos | Relay-Master-Log-File | Slave-I0-Running |
| Slave_SQL_Running | Replicate-do-db | Replicate-ignore-db |
| Last-errno |

```

```

| Last_error | Skip_counter | Exec_master_log_pos |
Relay_log_space |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| 192.168.4.100 | replication-user | 3306 | 60 |
|
g-bin.001 | 180 | s-bin.002 |
225 | g-bin.001 | Yes | Yes |
| replication-db | | 0 | |
| 0 |
| 180 | 233 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

En el servidor principal puede ejecutar instrucciones DELETE y UPDATE, acciones que se reflejarán en el esclavo. Por ejemplo:

```

mysql> DELETE FROM replication-table WHERE f1=1;
Query OK, 1 row affected (0.34 sec)
mysql> UPDATE replication-table SET f1=1;
Query OK, 1 row affected (0.05 sec)

```

Al revisar el esclavo, vera lo siguiente:

```

mysql> SELECT * FROM replication-table;
+-----+-----+
| f1 | f2 |
+-----+-----+| 1 | second |
+-----+-----+
1 row in set (0.01 sec)

```

No es necesario que el servidor esclavo este conectado al principal para estar en sincronizacion, siempre que los registros binarios sean correctos, como se demuestra en el siguiente ejemplo. En primer lugar, apague el servidor esclavo:

```

% /usr/local/mysql/bin/mysqladmin -uroot -pg00r002b shutdown
020821 17:25:37 mysqld ended

```

Seguidamente, añade otro registro al principal:

```

mysql> INSERT INTO replication-table (f1,f2) VALUES(3,'third');
Query OK, 1 row affected (0.03 sec)

```

Reinicie el servidor esclavo, conectelo a la base de datos replication_db y vera que se ha añadido el nuevo registro:

```

% bin/mysqld_safe &

```

```
[1] 1989
% /usr/local/mysql/bin/mysql -uroot -pg00r002b mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 4.0.2-alpha-
max-log
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> SELECT * FROM replication-table;
+----+-----+
| f1  | f2      |
+----+-----+
|    3 | third   |
|    1 | second  |
+----+-----+
2 rows in set (0.02 sec)
```

Es posible que el servidor principal tambien se **caiga** y que el esclavo intente volver a conectarse (en funcion de **los segundos** especificados en `master-connect-retry`, cuyo valor predeterminado es 60) hasta que vuelva a **funcionar** el principal. Debe prestar especial atencion **al modificar los registros binarios** ya que es lo unico que necesita el servidor esclavo para funcionar. En el siguiente **ejemplo** vemos **cómo se pueden perder los datos**. En primer lugar, cierre el **servidor esclavo**:

```
% /usr/local/mysql/bin/mysqladmin -uroot -pg00r002b shutdown
020821 17:25:37 mysqld ended
```

Como en el **caso anterior**, **añada otro registro al principal** pero, en esta **ocasión**, **ejecute despues la instrucción RESET MASTER** (para **eliminar los registros binarios antiguos** y **empezar de nuevo con el registro 1**):

```
mysql> INSERT INTO replication-table (f1,f2)
VALUES (4, 'fourth');
Query OK, 1 row affected (0.01 sec)
mysql> RESET MASTER;
Query OK, 0 rows affected (0.03 sec)
```

Tras **ello**, **al reiniciar el esclavo**, no mostrara el **cambio**:

```
% bin/mysqld_safe &
[1] 1989
% /usr/local/mysql/bin/mysql -uroot -pg00r002b mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 4.0.2-alpha-
max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> SELECT * FROM replication-table;
+----+-----+
| f1  | f2      |
+----+-----+
|    3 | third   |
```

```

|      1 | second |
+-----+-----+
2 rows in set (0.00 sec)

```

Puede ver la razon en el estado del servidor esclavo:

```

mysql> SHOW SLAVE STATUS;
+-----+-----+-----+-----+-----+
| Master-Host | Master-User | Master-Port |
Connect-retry | \Master-Log-File | Read-Master-Log-Pos |
Relay-Log-File |
Relay-Log-Pos | Relay-Master-Log-File | Slave-I0-Running |
Slave-SQL-Running | Replicate-do-db | Replicate-ignore-db |
Last-errno
| Last-error | Skip-counter | Exec-master-log-pos |
Relay_log_space |
+-----+-----+-----+-----+
| 192.168.4.100 | replication-user | 3306 | 60 |
|
g-bin.001 | 443 | s-bin.004 | 4 |
| g-bin.001 | Yes | Yes |
replication-db | | 0 |
| 0 |
| 443 | 500 |
+-----+-----+-----+-----+
|
+-----+-----+-----+-----+
-+-----+-----+-----+-----+
+-----+-----+
1 row in set (0.00 sec)

```

El registro principal deberia estar en la posicion 443. Puede compararlo con la posicion real en la que se encuentra en el servidor principal:

```

mysql> SHOW MASTER STATUS;
+-----+-----+-----+-----+
| File | Position | Binlog_do_db | Binlog_ignore_db |
+-----+-----+-----+-----+
| g-bin.001 | 79 | |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

Puede recuperar la sincronizacion si restablece el esclavo, como indicamos a continuación:

```
mysql> RESET SLAVE;
```

Query OK, 0 rows affected (0.01 sec)

En este caso, el estado del esclavo ha cambiado y de nuevo comienza al principio del registro binario 1 o en la posición 79:

```
mysql> SHOW SLAVE STATUS;
+-----+-----+-----+-----+
| Master-Host | Master-User | Master-Port |
Connect-retry |
Master-Log-File | Read-Master-Log-Pos | Relay-Log-File |
|
Relay-Log-Pos | Relay-Master-Log-File | Slave-I0-Running |
Slave-SQL-Running | Replicate-do-db | Replicate-ignore-db |
Last-errno
| Last-error | Skip-counter | Exec_master_log_pos |
Relay-log-space |
+-----+-----+-----+-----+
| 192.168.4.100 | replication-user | 3306 | 60 |
|
g-bin.001 | 79 | s-bin.002 |
124 | g-bin.001 | Yes | Yes |
| replication-db | | 0 | |
| 0
| 79 | 132 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Vuelva al servidor principal, añada de nuevo el registro y observe el estado del principal, en el que el registro binario se ha desplazado hasta la posición 180:

```
mysql> INSERT INTO replication-table (f1,f2)
VALUES (4,'fourth');
Query OK, 1 row affected (0.00 sec)
mysql> SHOW MASTER STATUS;
+-----+-----+-----+-----+
| File | Position | Binlog_do_db | Binlog_ignore_db |
+-----+-----+-----+-----+
| g-bin.001 | 180 | | |
|
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Y el esclavo ha recuperado de nuevo el registro:

```
mysql> SELECT * FROM replication-table;
+----+-----+
| f1  | f2      |
+----+-----+
|    3 | third   |
|    1 | second  |
|    4 | fourth  |
+----+-----+
3 rows in set (0.00 sec)
```

Si es observador, vera que el registro se ha aiiadido dos veces en el servidor principal:

```
mysql> SELECT * FROM replication-table;
+----+-----+
| f1  | f2      |
+----+-----+
|    3 | third   |
|    1 | second  |
|    4 | fourth  |
|    4 | fourth  |
+----+-----+
4 rows in set (0.00 sec)
```

Este ejemplo nos sirve de advertencia. Por el mero hecho de que funcione la duplicacion, no nos garantiza que los datos sean identicos en los dos servidores.

Con un buen diseiio (como el uso de una clave principal en la tabla), podríamos haber evitado este problema. No obstante, hemos aprendido a prestar especial atencion al estado del esclavo y del principal, y a la hora de trabajar con los registros binarios.

Duplicación con un registro binario activo en el principal

En este ejemplo veremos como controlar una **situación** en la que el servidor principal lleva un tiempo en ejecucion con el registro binario activado y queremos configurar una duplicacion.

En primer lugar, cierre el servidor esclavo para evitar cualquier **conflicto** del ejemplo anterior:

```
% /usr/local/mysql/bin/mysqladmin -uroot -pg00r002b shutdown
020821 23:40:49 mysqld ended.
```

Utilizaremos la misma tabla que en el ejemplo anterior.

En el servidor principal, elimine los registros binarios y restablezca los para empezar con uno nuevo antes de añadir registros, como indicamos a continuación:

```
mysql> DELETE FROM replication-table;
Query OK, 4 rows affected (0.09 sec)
mysql> RESET MASTER;
Query OK, 0 rows affected (0.02 sec)
mysql> INSERT INTO replication-table (f1,f2) VALUES(1,'first');
Query OK, 1 row affected (0.01 sec)
mysql> INSERT INTO replication-table (f1,f2)
VALUES(2,'second');
Query OK, 1 row affected (0.01 sec)
```

Tras ello, copie estos datos en el esclavo y compruebe el desplazamiento del registro binario en el principal. Asegurese de que no se ha escrito ningún dato en el servidor principal después de copiar los datos pero antes de comprobar el estado:

```
mysql> SHOW MASTER STATUS;
+-----+-----+-----+-----+
| File          | Position | Binlog_do-db | Binlog_ignore_db |
+-----+-----+-----+-----+
| g-bin.001    | 280      |               |                   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

En el servidor esclavo, realice las mismas operaciones que en el ejemplo anterior, es decir, copie los datos, defina las opciones de configuración (con el siguiente cambio), elimine el archivo `master.info` en caso de que exista (en el ejemplo anterior se habrá creado en el directorio de datos, por ejemplo en `C:\mysql\data` o `/usr/local/mysql/data`) y reinicie el servidor. La única diferencia es que el archivo de configuración debe incluir la opción `skip-slave-start`.

No empezaremos la duplicación del esclavo hasta que hayamos indicado el punto de inicio correcto. Seguidamente, añadimos nuevos registros al servidor principal:

```
mysql> INSERT INTO replication-table (f1,f2) VALUES(3,'third');
Query OK, 1 row affected (0.01 sec)
mysql> INSERT INTO replication-table (f1,f2)
VALUES(4,'fourth');
Query OK, 1 row affected (0.01 sec)
```

En el servidor esclavo, indique que debe comenzar con el archivo de registro binario correcto y con el desplazamiento adecuado. Para ello, configure `MASTER_LOG_FILE` como `g-bin.001` (o con el valor que aparezca cuando ejecute `SHOW MASTER STATUS`) y `MASTER_LOG_POS` como `280` (o, en su caso, el valor adecuado).

Tras ello, inicie la duplicación del esclavo y pruebe los resultados:

```
mysql> CHANGE MASTER TO MASTER_LOG_FILE='g-bin.001',
  MASTER_LOG_POS=280;
Query OK, 0 rows affected (0.00 sec)
mysql> SLAVE START;
query OK, 0 rows affected (0.00 sec)
mysql> SELECT * FROM replication-table;
+----+-----+
| f1  | f2      |
+----+-----+
| 1   | first   |
| 2   | second  |
| 3   | third   |
| 4   | fourth  |
+----+-----+
4 rows in set (0.01 sec)
```

Eliminación de registros binarios antiguos del servidor principal e inicio de la operación

Al utilizar la duplicación, la eliminación de registros binarios puede ser un riesgo ya que puede que un servidor esclavo no haya terminado con uno de los registros que pretenda eliminar.

En este ejemplo, tendremos que eliminar los datos y restablecer el servidor principal, para empezar desde cero y, tras ello, añadir nuevos datos:

```
mysql> DELETE FROM replication-table;
mysql> RESET MASTER;
mysql> INSERT INTO replication-table (f1,f2) VALUES(1,'first');
mysql> INSERT INTO replication-table (f1,f2)
VALUES(2,'second');
mysql> INSERT INTO replication-table (f1,f2) VALUES(3,'third');
```

Copie estos datos en un nuevo esclavo (si ha ejecutado ejemplos anteriores en el servidor esclavo, borre el archivo `master.info` e inicie el esclavo con la opción `skip-slave-start`).

Vacíe los registros del principal para imitar a un servidor que lleva un tiempo en ejecución:

```
mysql> FLUSH LOGS;
mysql> FLUSH LOGS;
```

Tras ello, al analizar el estado del principal, verá que ya se encuentra en su tercer registro binario:

```
mysql> SHOW MASTER STATUS;
+-----+-----+-----+-----+
| File          | Position | Binlog_do_db | Binlog_ignore_db |
```

```

+-----+-----+-----+-----+
| g-bin.003 | 4 | | |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

Inicie el servidor esclavo y comience la **duplicación** desde el **punto** correcto:

```

mysql> CHANGE MASTER TO MASTER_LOG_FILE='g-
bin.003',MASTER_LOG_POS=4;
Query OK, 0 rows affected (0.01 sec)
mysql> SLAVE START;
Query OK, 0 rows affected (0.00 sec)

```

Ahora, el esclavo comenzara desde el registro **correcto** del principal. Todavía tenemos otros dos registros binarios en el principal que ocupan espacio. Sera necesario empezar a mantener **archivos** de registros para que no se nos vayan de las **manos**.

Puede que quiera eliminar los registros uno y dos, **pero** no es seguro hacerlo ya que puede haber algun esclavo que tenga que utilizarlos.

Para verificarlo, tendra que comprobar el estado de todos los servidores esclavos. En este **caso**, solo hay uno:

```

mysql> SHOW SLAVE STATUS;
+-----+-----+-----+-----+-----+
| Master-Host | Master-User | Master-Port |
Connect-retry |
Master-Log-File | Read-Master-Log-Pos | Relay-Log-File |
|
Relay-Log-Pos | Relay-Master-Log-File | Slave-Io-Running |
Slave_SQL_Running | Replicate-do-db | Replicate-ignore-db |
Last-errno
| Last-error | Skip-counter | Exec-master-log-pos |
Relay-log-space |
+-----+-----+-----+-----+-----+
| 192.168.4.100 | replication-user | 3306 | 60 |
|
g-bin.003 | 4 | s-bin.003 |
830 | g-bin.003 | Yes | Yes |
| replication-db | | 0 | |
| 4 | 1885 | |

```

Comprara que el esclavo utiliza **g-bin.003** y que esta actualizado (posicion 4). Si todos los esclavos estan actualizados, puede eliminar sin riesgo los

registros binarios 1 y 2 del principal, por medio de la **instrucción** PURGE MASTER LOGS, como indicamos a **continuación**:

```
mysql> PURGE MASTER LOGS TO "g-bin.003";
Query OK, 0 rows affected (0.00 sec)
```

Si realizo un **listado** en el directorio de datos (o donde haya especificado la ubicacion de los registros binarios), vera que los dos primeros se han eliminado.

Esta instruccion fallara si un esclavo **activo** intenta leer un registro que hemos intentando suprimir y generara el siguiente error:

```
mysql> PURGE MASTER LOGS TO "g-bin.003";
ERROR:
A purgeable log is in use, will not purge
```

Si el esclavo no esta conectado y elimina un registro binario que todavia no se ha utilizado, ese esclavo no podra continuar con la duplicacion. En alguna fase, este proceso se puede automatizar **pero** por ahora tendra que verificar **manualmente cada** uno de los esclavos para ver su **posicion**. Veamos que sucede si no lo hacemos.

En primer lugar, detenga el servidor esclavo:

```
mysql> SLAVE STOP;
Query OK, 0 rows affected (0.00 sec)
```

Seguidamente, en el principal, vacie una vez mas los registros, aiiada un nuevo registro y, tras ello, borre los registros binarios:

```
mysql> FLUSH LOGS;
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO replication-table (f1,f2)
VALUES(4,'fourth');
Query OK, 1 row affected (0.00 sec)
mysql> FLUSH LOGS;
Query OK, 0 rows affected (0.00 sec)
mysql> SHOW MASTER STATUS;
+-----+-----+-----+-----+
| File          | Position | Binlog_do_db | Binlog_ignore_db |
+-----+-----+-----+-----+
| g-bin.005    | 4        |              |                   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
mysql> PURGE MASTER LOGS TO "g-bin.005";
Query OK, 0 rows affected (0.02 sec)
```

Si reinicia ahora el esclavo, no duplicara, ya que busca un registro binario que no existe:

```
mysql> SLAVE START;
Query OK, 0 rows affected (0.00 sec)
```

El problema es que la última instrucción `INSERT` en el principal no aparece en ningún registro ya que todos los registros "antiguos" se han limpiado. Si realizo una copia de seguridad de los registros binarios, los podrá restablecer fácilmente. En caso contrario, tendrá que volver a ejecutar manualmente la instrucción para que el esclavo utilice el registro más reciente, como se aprecia a continuación:

```
mysql> INSERT INTO replication-table (f1,f2)
VALUES(4,'fourth');
Query OK, 1 row affected (0.00 sec)
mysql> CHANGE MASTER TO MASTER-LOG-FILE='g-
bin.005',MASTER_LOG_POS=4;
Query OK, 0 rows affected (0.01 sec)
```

Si añada otro registro al principal:

```
mysql> INSERT INTO replication-table (f1,f2) VALUES(5,'fifth');
Query OK, 1 row affected (0.00 sec)
```

se duplicará sin problemas en el servidor esclavo:

```
mysql> SELECT * FROM replication-table;
+----+-----+
| f1  | f2      |
+----+-----+
| 1   | first   |
| 2   | second  |
| 3   | third   |
| 4   | fourth  |
| 5   | fifth   |
+----+-----+
5 rows in set (0.00 sec)
```

Este ejemplo muestra que la duplicación no es una copia exacta de los datos, sino un trasvase de las instrucciones de un servidor a otro. Esta operación genera una copia exacta de los datos pero si se modifica el archivo `master.info` o los registros binarios, MySQL no podrá seguir secuencialmente los comandos, lo que puede provocar la desincronización de los datos.

Como evitar un exceso de actualizaciones

Este ejemplo muestra lo que puede suceder si no iniciamos el servidor esclavo desde el punto correcto del registro binario del principal.

Comience con un servidor principal limpio, añada algunos registros y anote los detalles del registro binario inmediatamente después de realizar la copia:

```
mysql> DELETE FROM replication-table;
mysql> RESET MASTER;
mysql> INSERT INTO replication-table (f1,f2) VALUES(1,'first');
```

```
mysql> INSERT INTO replication-table (f1,f2)
VALUES (2,'second');
mysql> SHOW MASTER STATUS;
+-----+-----+-----+-----+
| File          | Position | Binlog_do_db | Binlog_ignore_db |
+-----+-----+-----+-----+
| g-bin.001    | 280      |               |                   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Copie los datos en el esclavo e inicie. El esclavo debe estar limpio (sin archivos master. info y datos en la base de datos replication_db) y debe contar con algo similar a un conjunto de opciones en su archivo de configuración, como mostramos a continuación:

```
master-host      = 192.168.4.100
master-user      = replication-user
master-ssl-key   = replication_pwd
server-id        = 3
replicate-do-db = replication-db
```

Inicie el servidor esclavo y revise el estado del mismo para comprobar si la duplicación ha empezado correctamente:

```
mysql> SHOW SLAVE STATUS;
+-----+-----+-----+-----+-----+
| Master-Host | Master-User | Master-Port | Connect-retry |
Master-Log-File | Read-Master-Log-Pos | Relay-Log-File |
Relay-Log-Pos | Relay-Master-Log-File | Slave-IO-Running |
Slave-SQL-Running | Replicate-do-db | Replicate-ignore-db |
Last-errno | Last-error | Skip-counter | Exec_master_log_pos |
Relay-log-space |
+-----+-----+-----+-----+-----+
| 192.168.4.100 | replication-user | 3306 | 60 |
| g-bin.001 | 280 | s-bin.003 |
325 | g-bin.001 | Yes | Yes |
| replication-db | | 0 | |
| 0 |
| 280 | 329 |
+-----+-----+-----+-----+-----+
```

```

+-----+-----+-----+
-+-----+-----+-----+-----+
+-----+-----+
1 row in set (0.00 sec)

```

Todo parece funcionar correctamente. La duplicación ha comenzado y el esclavo se encuentra en el mismo punto que el maestro, en el registro binario g-bin.001 y en la posición 280. Sin embargo, al examinar los datos en el servidor esclavo, nos encontramos con una desagradable sorpresa:

```

mysql> SELECT * FROM replication-table;
+-----+-----+
| f1  | f2      |
+-----+-----+
| 1   | first   |
| 2   | second  |
| 1   | first   |
| 2   | second  |
+-----+-----+
4 rows in set (0.00 sec)

```

El problema es que el esclavo ha iniciado la duplicación desde el principio del primer registro binario, lo que significa que ha repetido las dos instrucciones INSERT aunque la copia de los datos se realizó posteriormente. Hay dos soluciones. Puede ejecutar RESET MASTER en el servidor principal justo después de realizar la copia o ejecutar la instrucción CHANGE MASTER TO en el esclavo antes de comenzar la duplicación para configurar el punto de inicio correcto, como hicimos en un apartado anterior (lo que implica iniciar el servidor con la opción skip-slave-start).

Como evitar errores clave

El siguiente ejemplo muestra lo que sucede cuando los datos se actualizan en el esclavo, lo que genera un error clave. Este error se puede producir cuando el servidor principal y el esclavo realizan la duplicación de forma circular o cuando se realizan actualizaciones directamente en el esclavo.

Por esta razón, añadiremos una clave principal a la tabla replication_tb. Puede modificar la tabla existente, como se muestra a continuación:

```

mysql> ALTER TABLE replication-table MODIFY f1 INT NOT NULL,ADD
PRIMARY KEY(f1);
query OK, 0 rows affected (0.36 sec)

```

O puede crear una tabla nueva:

```

mysql> CREATE TABLE replication-table(f1 INT NOT NULL,f2
VARCHAR(20),
PRIMARY KEY(f1));
query OK, 0 rows affected (0.03 sec)

```

Añada algunos registros al servidor principal y reinicielo, para no repetir el error del ejemplo anterior:

```
mysql> INSERT INTO replication-table (f1,f2) VALUES(1,'first');
mysql> INSERT INTO replication-table (f1,f2)
VALUES(2,'second');
mysql> RESET MASTER;
```

Copie los datos en el servidor principal e inicie la duplicación del servidor. Añada un nuevo registro al principal:

```
mysql> INSERT INTO replication-table (f1,f2) VALUES(3,'third');
Query OK, 1 row affected (0.01 sec)
```

Los datos tendrán el siguiente aspecto en el esclavo:

```
mysql> SELECT * FROM replication-table;
+----+-----+
| f1 | f2      |
+----+-----+
|  1 | first   |
|  2 | second  |
|  3 | third   |
+----+-----+
3 rows in set (0.00 sec)
```

Hasta ahora **todo funciona** correctamente. El problema aparece si añadimos un registro al esclavo y, tras ello, añadimos el mismo registro al principal. En este ejemplo, hemos definido la misma clave principal a propósito, pero suele producirse cuando se **utilizan** campos AUTO_INCREMENT. Añada el siguiente registro primero al servidor esclavo y luego al principal:

```
mysql> INSERT INTO replication-table (f1,f2)
VALUES(4,'fourth');
Query OK, 1 row affected (0.01 sec)
```

Aunque si ha comprobado los datos en ambos servidores puedan parecer idénticos, hay un error en el esclavo, como si este hubiera intentado insertar el registro dos veces. A menos que haya utilizado la arriesgada opción `slave-skip-errors` en el archivo de configuración, la duplicación debería detenerse y el esclavo informara de la consulta errónea, como mostramos a continuación:

```
mysql> SHOW SLAVE STATUS;
+-----+-----+-----+-----+
| Master-Host | Master-User | Master-Port | Connect-retry |
+-----+-----+-----+-----+
```



```

| Master_Host      | Master-User      | Master-Port |
Connect_retry |
Master_Log_File   | Read-Master-Log-Pos | Relay-Log-File
|
Relay_Log_Pos | Relay-Master-Log-File | Slave-I0-Running |
Slave_SQL_Running | Replicate_do_db | Replicate_ignore_db |
Last-errno
| Last-error | Skip-counter | Exec_master_log_pos |
Relay_log_space |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| 192.168.4.100 | replication_user | 3306         | 60
|
g-bin.001 { 378 | s-bin.002 |
423 | g-bin.001 | Yes | Yes
| replication_db | | 0 |
| 378 | 431 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

Añada un registro al servidor principal:

```

mysql> INSERT INTO replication-table(f1,f2) values(5,'fifth');
Query OK, 1 row affected (0.00 sec)

```

Como en casos anteriores, se duplicara en el esclavo:

```

mysql> SELECT * FROM replication-table;
+-----+
| f1 | f2
+-----+
| 1 | first |
| 2 | second |
| 3 | third |
| 4 | fourth |
| 5 | fifth |
+-----+
5 rows in set (0.00 sec)

```

Resumen

La duplicación es una función muy útil que nos permite conservar una copia exacta de datos de un servidor en otro. La base de datos principal escribe en el

registro binario y una serie de servidores esclavos se conectan al principal, leen el registro binario de actualizaciones y duplican estas instrucciones en sus servidores.

La duplicación resulta de gran utilidad para realizar copias de seguridad y también para mejorar el rendimiento.

La **relación** entre el registro binario del principal y el **archivo master.info** del esclavo es fundamental para conservar la sincronización de la duplicación. Si se eliminan **los** registros binarios antes de que **los** esclavos **los** utilicen, la duplicación fallará.

Configuración y optimización de MySQL

Solamente podrá presumir de dominar MySQL si conoce los entresijos de las variables `mysqld` y como afectan a su rendimiento.

En este capítulo veremos la forma de configurar y optimizar MySQL y explicaremos como configurar las variables y los elementos necesarios para mejorar el rendimiento.

Nos centraremos en los siguientes aspectos:

- Las opciones y variables `mysqld`
- Un análisis detallado de la optimización de las variables `table_cache`, `key_buffer_size`, `delayed_queue`, `back_log` y `sort_buffer`
- Como procesar mas conexiones
- Como cambiar variables sin reiniciar el servidor
- Configuración de tablas InnoDB
- Como mejorar el hardware
- Como ejecutar análisis comparativos
- Ejecucion de MySQL en modo ANSI
- Como utilizar distintos idiomas y conjuntos de caracteres

Optimización de las variables mysqld

Para saber cuales son los valores existentes de las variables mysqld, puede utilizar mysqladmin en la linea de comandos:

```
% mysqladmin -uroot -pg00r002b variables;
```

o cuando se conecte a MySQL:

```
mysql> SHOW VARIABLES
+-----+-----+
| Variable-name,          | Value                               |
+-----+-----+
| back-log                | 50                                  | |
| basedir                  | /usr/local/mysql-max-4.0.1-       |
|                          | alpha-pc-linux-gnu-i686          |
| bdb_cache_size          | 8388600                             |
| bdb_log_buffer_size     | 32768                               |
| bdb_home                 | /usr/local/mysql/data/           |
| bdb_max_lock            | 10000                               |
| bdb_logdir              |                                     |
| bdb_shared_data         | OFF                                  |
| bdb_tmpdir              | /tmp/                               |
| bdb_version              | Sleepycat Software: Berkeley DB   |
|                          | 3.2.9a: (December 23, 2001)      |
| binlog-cache-size       | 32768                               |
| character-set            | latin1                              |
| character-sets          | latin1 big5 czech euc_kr gb2312   |
|                          | gbk latin1_de sjis tis620 ujis    |
|                          | dec8 dos german1 hp8 koi8_ru      |
|                          | latin2 swe7 usa7 cp1251 danish     |
|                          | hebrew win1251 estonia            |
|                          | hungarian koi8_ukr win1251ukr     |
|                          | greek win1250 croat cp1257        |
|                          | latin5                             |
| concurrent-insert       | ON                                  |
| connect-timeout         | 5                                    |
| datadir                 | /usr/local/mysql/data/           |
| delay-key-write         | ON                                  |
| delayed-insert-limit    | 100                                 |
| delayed_insert_         |                                     |
| timeout                 | 300                                 |
| delayed-queue-size     | 1000                                |
| flush                   | OFF                                  |
| flush-time              | 0                                    |
| ft_min_word_len         | 4                                    |
| ft_max_word_len         | 254                                 |
| ft_max_word_len_for_   |                                     |
| sort                    | 20                                  |
| ft_boolean_syntax       | + -><()~*:""&|                   |
| have-bdb                | YES                                  |
| have-innodb             | YES                                  |
```

```

| have_isam | YES |
| have-raid | NO |
| have-symlink | YES |
| have-openssl | NO |
| init_file | |
| innodb_additional_ | |
| mem_pool_size | 1048576 |
| innodb_buffer_pool | |
| size | 8388608 |
| innodb_data_file_path | ibdata1:64M |
| innodb_data_home_dir | |
| innodb_file_io_ | |
| threads | 9 |
| innodb_force_recovery | 0 |
| innodb_thread_ | |
| concurrency | 8 |
| innodb_flush_log_at_ | |
| trx_commit | OFF |
| innodb_fast_shutdown | OFF |
| innodb_flush_method | |
| innodb_lock_wait_ | |
| timeout | 11073741824 |
| innodb_log_arch_dir | |
| innodb_log_archive | OFF |
| innodb_log_buffer_ | |
| size | 1048576 |
| innodb_log_file_size | 15242880 |
| innodb_log_files_in_ | |
| group | 2 |
| innodb_log_group_ | |
| home-dir | ./ |
| innodb_mirrored_log_ | |
| groups | 1 |
| interactive-timeout | 28800 |
| join-buffer-size | 1131072 |
| key-buffer-size | 116773120 |
| language | /usr/local/mysql-max-4.0.1-alpha |
| | -pc-linux-gnu-i686/share/mysql/ |
| | english/ |
| large-files-support | ION |
| locked-in-memory | OFF |
| log | ON |
| log-update | OFF |
| log-bin | ON |
| log-slave-updates | OFF |
| log-long-queries | ON |
| long-query-time | 20 |
| low-priority-updates | OFF |
| lower-case-table- | |
| names | 0 |
| max_allowed_packet | 11047552 |
| max_binlog_cache_size | 14294963200 |
| max_binlog_size | 11073741824 |

```

```

| max_connections          |100
| max_connect_errors      |10
| max_delayed_threads     |20
| max_heap_table_size    |16777216
| max_join_size           |4294967295
| max_sort_length         |1024
| max_user_connections    |0
| max_tmp_tables          |32
| max_write_lock_count    |4294967295
| myisam-bulk-insert-    |
| tree-size               |8388608
| myisam_max_extra_      |
| sort-file-size          |256
| myisam_max_sort_       |
| file-size               |2047
| myisam_recover_        |
| options                 |OFF
| myisam-sort-buffer-    |
| size                    |8388608
| net-buffer-length       |7168
| net-read-timeout        |30
| net-retry-count         |10
| net-write-timeout       |60
| open-files-limit        |0
| pid_file                |/usr/local/mysql/data/host.pid
| port                    |3306
| protocol-version        |10
| record-buffer           |131072
| record_rnd_buffer       |131072
| rpl_recovery_rank       |0
| query_buffer-size       |0
| query-cache-limit       |1048576
| query-cache-size        |0
| query_cache_startup_    |
| type                    |1
| safe-show-database     |OFF
| server-id               |1
| slave-net-timeout       |3600
| skip-external-locking   |ON
| skip-networking         |OFF
| skip-show-database      |OFF
| slow_launch_time        |2
| socket                  |/tmp/mysql.sock
| sort-buffer             |1524280
| sql_mode                |0
| table-cache             |64
| table-type              |MYISAM
| thread-cache-size       |0
| thread-stack            |165536
| transaction-isolation   |READ-COMMITTED
| timezone                |SAST
| tmp_table_size          |133554432
| tmpdir                  |/tmp/

```

```
| version                | 4.0.1-alpha-max-log |
| wait-timeout          | 128800               |
```

Otro aspecto de gran importancia es la información proporcionada por el propio servidor. Podemos ver estos datos desde la línea de comandos con la siguiente construcción:

```
% mysqladmin extended-status
```

o cuando se conecte a MySQL:

```
mysql> SHOW STATUS
+-----+-----+
| Aborted-clients          | 142 |
| Aborted-connects        | 5   |
| Bytes-received           | 9005619 |
| Bytes-sent               | 15444786 |
| Connections              | 794 |
| Created-tmp-disk-tables | 1   |
| Created-tmp-tables      | 716 |
| Created_tmp_files        | 0   |
| Delayed-insert-threads  | 0   |
| Delayed-writes           | 0   |
| Delayed-errors           | 0   |
| Flush-commands          | 1   |
| Handler-delete           | 27  |
| Handler-read-first      | 1534 |
| Handler-read-key         | 608840 |
| Handler-read-next       | 652228 |
| Handler-read-prev       | 164  |
| Handler-read-rnd        | 14143 |
| Handler-read-rnd-next   | 1133372 |
| Handler-update           | 90   |
| Handler-write            | 131624 |
| Key-blocks-used         | 6682 |
| Key-read-requests       | 2745899 |
| Key-reads                | 6026 |
| Key-write-requests      | 63925 |
| Key-writes               | 63790 |
| Max_used_connections    | 20   |
| Not-flushed-key-blocks  | 0    |
| Not-flushed-delayed-rows | 0    |
| Open-tables             | 64   |
| Open-files               | 128  |
| Open-streams             | 0    |
| Opened-tables           | 517  |
| Questions                | 118245 |
| Select-full-join        | 0    |
| Select-full-range-join  | 0    |
| Select-range            | 2300 |
| Select-range-check      | 0    |
| Select-scan              | 642  |
| Slave-running           | OFF  |
```


Slave-open-temp-tables	0	
Slow-launch-threads	0	
Slow-queries	8	
Sort-merge-passes	0	
Sort-range	3582	
Sort-rows	116287	
Sort-scan	806	
Table-locks-immediate	82957	
Table-locks-waited	2	
Threads-cached	0	
Threads-created	793	
Threads-connected	1	
Threads-running	1	
Uptime	1662790	
+-----+-----+		

La lista de variables e información de estado aumenta con cada nuevo lanzamiento. Probablemente su versión utilice muchas más, razón por la que debe consultar la documentación para comprobar cuáles son los elementos adicionales exactos. Mas adelante veremos una explicación detallada, en la tabla 13.2.

La mayor parte de las distribuciones MySQL incorporan cuatro archivos de configuración de muestra:

- **my-huge.cnf:** Se utiliza en sistemas con más de 1 GB de memoria, asignada esencialmente a MySQL.
- **my-large.cnf:** Se utiliza en sistemas con al menos 512 MB de memoria, asignada esencialmente a MySQL.
- **my-medium.cnf:** Se utiliza en sistemas con al menos 32 MB de memoria asignada en su totalidad a MySQL o con al menos 128 MB en un equipo que se utilice para distintos propósitos (como por ejemplo un servidor dual Web/bases de datos).
- **my-small.cnf:** Se utiliza en sistemas con menos de 64 MB de memoria en los que MySQL no puede utilizar muchos de estos recursos.

Estos archivos se encuentran normalmente en */usr/share/doc/packages/MySQL/* (instalación RPM), */usr/local/mysql-max-4.x.x-platform-operating-system-extra/support-is* (instalación binaria Unix) o *C:\mysql* (Windows).

ADVERTENCIA: En Windows, la extensión **.cnf** puede entrar en conflicto con FrontPage y NetMeeting.

Como punto de partida, le sugerimos que sustituya su archivo **my.cnf** (o **my.ini**) por una de estas configuraciones y que seleccione la que más se adecue a sus necesidades.

La selección de la configuración adecuada para su sistema es parte del camino **hacia** la optimización, pero para conseguir resultados óptimos es necesario afinar con precisión la configuración del sistema y de los aspectos específicos del uso del mismo. En los siguientes apartados analizaremos algunas de las variables.

Optimización de table-cache

La variable `table_cache` es una de las de mayor utilidad. Cada vez que MySQL accede a una **tabla**, si hay espacio en la **caché**, la tabla se ubica en dicho espacio. El acceso a la tabla en **memoria** es más rápido que el acceso desde disco. Para determinar **si** necesita aumentar el valor de su variable `table_cache` debe examinar el valor de `open_tables` en periodos de **máximo tráfico** (uno de los valores de estado extendidos que vimos con las variables `SHOW STATUS` o `mysqladmin`). Si `open_tables` tiene el mismo valor que `table_cache` y el valor de `opened_tables` (otro valor de estado extendido) se **incrementa**, tendrá que aumentar `table_cache` si dispone de suficiente **memoria**.

NOTA: El número de **tablas** abiertas puede **ser** mayor que el **número de tablas de** su base de datos. **MySQL** permite subprocesos múltiples y permite la ejecución de **varias** consultas al mismo tiempo, y cada una de éstas puede **abrir** una tabla.

Veamos las siguientes situaciones, todas durante periodos de máximo tráfico.

- **Caso 1.** Este caso corresponde a un servidor **activo** pero no especialmente ocupado:

```
table-cache - 512
open-tables - 103
opened-tables - 1723
uptime - 4021421 (medido en segundos)
```

Parece que `table_cache` se ha configurado con un valor demasiado alto. El servidor lleva **funcionando** un tiempo (si acabara de empezar, no sabríamos si se ha alcanzado `table_cache` demasiado pronto o si es demasiado pronto para empezar a incrementar `opened_tables`). El **número** de tablas abiertas es razonablemente bajo y está muy por debajo de lo que debería, si consideramos este caso en el **contexto** de un **periodo** de máximo tráfico.

- **Caso 2.** Este caso se ha **tomado** de un servidor de desarrollo:

```
table-cache - 64
open-tables - 64
opened-tables - 431
uptime - 1662790 (medido en segundos)
```

Aunque el valor de `open_tables` es el máximo, el número de `open_tables` es considerablemente bajo, si consideramos que el servidor lleva un tiempo funcionando. No obtendríamos ninguna ventaja si aumentáramos `table_cache`.

- **Caso 3.** Se corresponde a un servidor activo con un rendimiento inferior al esperado:

```
table-cache - 64
open-tables - 64
opened-tables - 22423
uptime - 19538
```

En este ejemplo, `table_cache` se ha configurado con un valor demasiado bajo. La variable `open_tables` tiene el valor **máximo** y el número de `opened_tables` es elevado, aunque `uptime` es menor de seis horas. Si su sistema tiene memoria de reserva disponible, debería aumentar `table_cache`.

ADVERTENCIA: No debe configurar `table_cache` a ciegas con un valor elevado. Si no necesita un valor elevado, utilice uno que sea más razonable. Si el que configura es demasiado elevado, puede que se quede sin descriptores de archivos y, por tanto, puede que el rendimiento no sea el correcto o que se rechacen las conexiones.

Optimización de `key_buffer_size`

`key_buffer_size` afecta al tamaño de los bufer de índice que, a su vez, afectan a la velocidad de procesamiento de índices. Cuanto mayor sea el valor, mayor cantidad de índices podrá almacenar MySQL en memoria, a la que se accede con mayor velocidad que a un disco. Le sugerimos que la configure entre un cuarto y la mitad de la memoria disponible en su servidor (si se trata de un servidor dedicado a MySQL). Puede hacerse una idea de cómo ajustar `key_buffer_size` si lo compara con los valores de estado `key_read_requests` y `key_reads`.

La proporción entre `key_reads` y `key_read` debe ser la más baja posible, siendo 1:00 el límite superior aceptable (es mejor 1:1000; 1:10 no es aceptable). El valor `key_reads` indica el número de veces que hay que leer la clave desde disco, que es lo que se evita al configurar el búfer de claves con el mayor valor posible.

En los siguientes ejemplos se examinan dos posibilidades.

- **Caso 1.** Una situación favorable:

```
key-buffer-size - 402649088 (384M)
key-read-requests - 597579931
key-reads - 56188
```

- Caso 2. Una situación alarmante:

```
key-buffer-size - 16777216 (16M)
key-read-requests - 597579931
key-reads - 53832731
```

El Caso 1 refleja una situación favorable. La proporción es superior a 1: **10000** pero, el Caso 2 es alarmante, ya que presenta una preocupante proporción de 1:11. Como solución, podría incrementar `key_buffer_size` todo lo que le permita la memoria. Necesitará actualizar su hardware si no dispone de suficiente memoria para ello. La proporción entre `key_writes` y `key_writes_requests` también resulta de gran utilidad. Suele equivaler a 1 si normalmente inserta o actualiza los registros de uno en uno pero, si añade o actualiza grandes volúmenes de datos, tendrá que reducir este valor. El uso de instrucciones `INSERT DELAYED` también permite reducir esta proporción.

Control de un elevado número de conexiones

Un problema muy habitual, pero de fácil solución, se produce cuando los sistemas se colapsan con el error `Too many connections`. Cuando el número de `threads_connected` supera a menudo el número de `max_connections`, es necesario realizar un cambio. Si las consultas se procesan correctamente, la solución es muy sencilla: basta con aumentar el valor de `max_connections`. La mayoría de las aplicaciones deberían utilizar conexiones permanentes en lugar de conexiones ordinarias (por ejemplo, en PHP, se puede utilizar la función `pconnect()` en lugar de la función `connect()`). Las conexiones permanentes permanecen abiertas incluso después de que finalice la ejecución de la consulta lo que, en servidores muy activos, significa que la siguiente consulta no necesita ningún recurso para conectarse de nuevo. La utilización de un elevado número de conexiones permanentes pero sin usar consume menos recursos que si conectamos, desconectamos y volvemos a conectar.

NOTA: Las conexiones permanentes se ven afectadas por los parámetros `KeepAlive` del servidor Web Apache.

El siguiente ejemplo examina un servidor Web con una carga considerable que utiliza conexiones permanentes:

```
max_connections - 250
max_used_connections - 210
threads_connected - 202
threads_running - 1
```

Puede parecerle que, en este caso, MySQL malgasta recursos pero en realidad el valor 202 de `threads_connected` es permanente, en función del número de instancias del servidor Web y apenas consume recursos. De hecho sólo se

ejecuta un subproceso, razón por la que la base de datos no trabaja en exceso. Si `threads_connected` se acerca al valor de `max_connections` sin problemas aparentes, puede incrementar `max_connections` para evitar sobrepasar el límite de conexiones. El valor `max_connections_used` le indica si las conexiones se acercan al límite máximo. Si este valor se aproxima al límite o si equivale a `max_connections`, debe realizar dicho aumento.

Personalmente, considero que las conexiones permanentes son más indicadas aunque hay informes de que, como la sobrecarga de conexiones MySQL es mucho más ligera que la de otras bases de datos (como Oracle, en las que es necesario utilizar conexiones permanentes en la mayoría de los casos), apenas hay diferencia o que incluso puede influir negativamente en el rendimiento. Le sugerimos que pruebe el rendimiento de sus propios sistemas.

NOTA: Cuando realice las pruebas, asegúrese de que lo hace bajo carga. Existen algunos documentos en la Web con todo tipo de comparaciones equivocadas entre conexiones permanentes y no permanentes.

En un sistema como el del caso anterior, un valor `threads_running` creciente indica que la base de datos no procesa la carga. Si examina la lista de procesos puede identificar que consultas causan el bloqueo. A continuación incluimos parte del resultado de un servidor de bases de datos antes de que se colapsara. El número de `threads_connected` seguía aumentando hasta que el servidor no pudo procesar más y se colapsó. La salida `processlist` nos ayudó a identificar las consultas problemáticas:

```
% mysqladmin processlist;
Id      User      Host      Db      Command  Time  State  Info
6464    mysql     webserv2  news    Sleep    590
6482    mysql     webserv2  news    Sleep    158
6486    mysql     webserv2  news    Sleep    842
7549    mysql     webserv2  news    Sleep    185
8126    mysql     webserv2  news    Sleep    349
9938    mysql     webserv2  news    Sleep    320
1696    mysql     webserv2  news    Sleep    100
4143    mysql     webserv2  news    Sleep    98
5071    mysql     webserv2  news    Sleep    843
5135    mysql     webserv2  news    Sleep    155
92707   mysql     zubat...  news    Sleep    530
93014   mysql     zubat...  news    Query    13      Locked select
                                     s_id from arts
                                     where a-id =
                                     'E232625'
93060   mysql     zubat...  news    Sleep    190
93096   mysql     zubat...  news    Query    171     Copying
                                     to tmp
                                     table select
                                     arts.a_id,
```

						arts.headline1, nartsect.se_id, arts.mdate, arts.set-
93153	mysql	zubat...	news	Sleep	207	
93161	mysql	zubat...	news	Query	30	Locked SELECT DISTINCT arts.a-id FROM arts,keywordmap WHEREarts.s_id in (1) AND arts. Locked select arts.name, arts.headline1, arts.se_id, n_blurb.blurb, slook.name as sectna
93165	mysql	zubat...	news	Query	36	Locked select arts.name, arts.headline1, arts.se_id, n_blurb.blurb, slook.name as sectna
93204	mysql	zubat...	news	Query	31	Locked select arts.name, arts-headline1, arts.se_id, n_blurb.blurb, slook.name as sectna
93205	mysql	zubat...	news	Query	156	Copying to tmp table select distinct arts.a_id, arts-headline1, nartsect.se_id, arts.mdate, arts.set-
93210	mysql	zubat...	news	Query	50	Locked select arts-a-id, arts.headline1, nartfpg.se_id, nartfpg.se_id, arts.mdate, nartfpg.p
93217	mysql	zubat...	news	Query	38	Locked select arts.name, arts.headline1, arts.se_id, n_blurb.blurb, slook.name as sectna
93222	mysql	zubat...	news	Query	8	Locked select arts.name,

						arts.headline1, arts.se_id, n_blurb.blurb, slook2.name as sectn
93226	mysql	zubat...	news	Query	39	Locked select arts-name, arts.headline1, arts.se_id, n_blurb.blurb, slook.name as sectna
93237	mysql	zubat..	news	Query	33	Locked select arts.a_id, arts.headline1, nartfpg.se_id, nartfpg.se_id, arts.mdate, nartfpg.p
93244	mysql	zubat...	news	Query	99	Copying to tmp table select distinct arts.a_id, arts.headline1, nartsect.se_id, arts.mdate, arts.set-
93247	mysql	zubat...	news	Query	64	Locked select s_id from arts where a_id='32C436'
93252	mysql	zubat...	news	Query	120	Copying to tmp table select distinct arts.a_id, arts.headline1, nartsect.se_id, arts.mdate, arts.set-
93254	mysql	zubat...	news	Sleep	47	
93256	mysql	zubat...	news	Sleep	171	
93257	mysql	zubat...	news	Query	176	Copying to tmp table select distinct arts.a_id, arts.headline1, nartsect.se_id, arts.mdate, arts.set-
93261	mysql	zubat...	news	Sleep	349	
93262	mysql	zubat...	news	Sleep	1	
93263	mysql	zubat...	news	Query	153	Copying to tmp

93267	mysql	zubat...	news	Query	27	table select distinct arts.a-id, arts.headline1, nartsect.se_id, arts.mdate, arts.set-
93276	mysql	zubat...	news	Query	29	Locked select arts.name, arts-headline1, arts.se_id, n_blurb.blurb, slook.name as sectna
93278	mysql	zubat...	news	Query	183	Copying to tmp table select distinct arts.a_id, arts.headline1, nartsect.se_id, arts.mdate, arts.set
93280	mysql	zubat...	news	Sleep	36	
93285	mysql	zubat...	news	Sleep	10	
93284	mysql	zubat...	news	Query	49	Locked select arts-name, arts.headline1, arts.se_id, n_blurb.blurb, slook.name as sectna

De los dos servidores Web que aparecen en la lista, `webserv2` se comporta de forma normal (todos sus subprocesos se han completado y las conexiones estan latentes) pero en `zubat` las consultas se acumulan.

Hay demasiadas consultas y esta es sólo una pequeña parte de la lista completa. Sin embargo, debería examinar una consulta como la siguiente:

```
select distinct arts.a_id, arts.headline1, nartsect.se_id,
arts.mdate, arts.set ...
```

Fijese en que el estado de todas estas consultas es `Copying to tmp table` y que en las otras era `Locked`. En este caso, el problema radica en que el programador modifico la consulta para que dejara de utilizar un índice. En capitulos anteriores encontrara mas información al respecto.

El examen rutinario de la salida `processlist` puede ayudarle a identificar consultas lentas antes de que provoquen algún error de importancia, como puede ser el colapso del servidor. El valor `show_queries` es otro de los que debe examinar. Si aumenta constantemente, probablemente haya algún problema. Un sistema bien ajustado debería tener el menor número de consultas lentas posible. Algunas uniones complejas serán inevitablemente lentas, pero es muy probable que la presencia de consultas lentas se deba a una mala optimización.

Optimización de las variables `delayed-queue-size` y `back-log`

Como vimos en un capítulo anterior, `INSERT DELAYED` libera al cliente pero no procesa la consulta de forma inmediata si hay algo más en la misma. MySQL espera a que haya un hueco para poder procesar las inserciones. Y aquí entra en escena la variable `delayed_queue_size`. Si la configura con su valor predeterminado, 1.000, significa que después de 1.000 instrucciones retrasadas en cola, el cliente no se liberará y tendrá que esperar. No es aconsejable un número tan alto de consultas en cola, pero si su sistema realiza un elevado número de inserciones al mismo tiempo y los clientes tienen que esperar aunque utilice `INSERT DELAYED`, debería incrementar la variable `delayed_queue_size`.

Otra variable que contribuye a la gestión de pequeños incrementos de actividad es `back_log`. Si un sistema recibe un elevado número de solicitudes de conexión en un breve periodo de tiempo, MySQL contará las que no haya procesado como parte de la cola de conexiones pendientes. Cuando se alcanza el límite `back_log`, se rechazan todas las consultas que están en cola. Si su sistema recibe grandes cantidades de solicitudes de conexión en pequeñas ráfagas y aprecia que se produce un rechazo por esta razón, debería incrementar el valor `back_log`. Si su sistema está simplemente ocupado y las solicitudes llegan en un flujo constante, el incremento de este valor no servirá para nada. Le proporciona espacio a su servidor para que pueda procesar pequeñas ráfagas pero no sirve en un sistema sobrecargado.

Optimización de la variable `sort-buffer`

Ya describimos la variable `sort_buffer` al analizar la forma de acelerar las operaciones de `mysamchk` en un capítulo anterior, pero puede resultar de utilidad para precisar algunas operaciones cotidianas. Si suele realizar muchas operaciones de ordenación (y, por ejemplo, utiliza `ORDER BY` en tablas de gran tamaño), puede que le interese modificar `sort_buffer`. Cada subproceso que realiza una ordenación asigna un búfer de tamaño `sort_buffer_size`. El archivo de configuración `my-huge.cnf` (en sistemas con al menos 1 GB de memoria) utiliza como valor predeterminado de `sort_buffer` 256 M para `mysamchk` y 2 M para `mysqld`. Aunque quiera que la cifra de `mysqld` pueda

procesar ordenaciones de gran tamaño, si tiene varias conexiones simultáneas que ejecutan cláusulas ORDER BY, puede que se produzcan problemas de memoria ya que a cada una se le asigna una variable `sort_buffer`.

Configuración de tablas InnoDB

Para que las tablas InnoDB se ejecuten correctamente, es imprescindible configurar correctamente las variables, incluso más que con las tablas MyISAM. La más importante es `innodb_data_file_path`, que especifica el espacio disponible para las tablas (datos e índices). Especifica uno o varios archivos de datos y también les asigna un tamaño. Es aconsejable que el último archivo de datos sea autoextensible (solamente el último archivo de datos puede hacerlo). Por esta razón, en lugar de quedarnos sin espacio cuando se utiliza todo el espacio disponible, el archivo de datos autoextensible aumentará (de 8 en 8 MB) para almacenar los datos adicionales. Por ejemplo:

```
innodb_data_file_path=/disk1/ibdata1:900M;/disk2/  
ibdata2:50M:autoextend
```

En este caso, los dos archivos de datos se encuentran en distintos discos (`disk1` y `disk2`). En primer lugar, los datos se ubican en `ibdata1`, hasta que se alcanza el límite de 900 MB y, tras ello, se ubican en `ibdata2`. Una vez alcanzado el límite de 50 MB, `ibdata2` aumenta automáticamente en fragmentos de 8 MB. Si un disco se llena físicamente, tendrá que añadir otro archivo de datos en otro disco, lo que requiere configuración manual. Para ello, observe el tamaño físico del archivo de datos final y redondeelo hasta el siguiente megabyte. Configure específicamente este archivo de datos y añada la nueva definición de archivo de datos. Por ejemplo, si el `disk2` especificado anteriormente se llena con `ibdata2` a 109 MB, utilizaremos una definición similar a la siguiente:

```
innodb_data_file_path=/disk1/ibdata1:900M;/disk2/ibdata2:109M;/disk3/  
ibdata3:500M:autoextend
```

Tendrá que reiniciar el servidor para aplicar los cambios.

Presentación de las opciones mysqld

En la tabla 13.1 se incluye la descripción de las opciones de `mysqld`.

Tabla 13.1. Opciones de `mysqld`

Opción	Descripción
<code>-ansi</code>	MySQL no solo tiene numerosas extensiones sino también numerosas diferencias con el comporta-

Opción	Descripción
	miento del estandar ANSI (estandar SQL, segun el American National Standards Institute). Si se configura esta opcion, MySQL se ejecutara en modo ANSI (cuyos cambios describiremos posteriormente en este capítulo).
-b, -basedir=ruta	La ruta al directorio base o directorio de instalacion de MySQL. El resto de rutas suelen ser relativas a esta.
-big-tables	Permite grandes conjuntos de resultados ya que guarda todos los conjuntos temporales en un archivo cuando no hay suficiente memoria.
-bind-address=IP	La dirección de Protocolo de Internet (IP) o nombre de anfitrión al que se vincula MySQL.
-character-sets-dir=ruta	Directorio en el que se guardan los conjuntos de caracteres.
-chroot=ruta	Por motivos de seguridad, con esta opcion puede iniciar MySQL en un entorno chroot. De esta forma MySQL se ejecuta en un subconjunto de los directorios y oculta la estructura de directorios completa. Sin embargo, esto limita el uso de LOAD DATA INFILE y SELECT...INTO OUTFILE.
-core-file	Esta opcion crea un volcado de memoria en caso de que mysqld desaparezca inesperadamente. Algunos sistemas requieren que se especifique un -core-file-size. Otros no crean este volcado si se utiliza la opcion -user.
-h, -datadir=ruta	Ruta al directorio de datos.
-debug[...]=	Si MySQL se configura con -with-debug, se puede utilizar esta opcion para generar un archivo de seguimiento de lo que hace mysqld.
-default-character-set=conjunto de caracteres	Determina el conjunto de caracteres predeterminado (de forma predeterminada es latin1).
-default-table-type=tipo	Define el tipo de tabla predeterminado (que se crea si no se especifica el tipo de tabla en la instruccion CREATE). De forma predeterminada, sera MyISAM.
-delay-key-write -for-all-tables	Con esta opcion, MySQL no vacia los bufer clave cuando escribe en cualquier tabla MyISAM.
-des-key-file=nombre_de_archivo	Las claves predeterminadas se leen desde este archivo. Lo utilizan las funciones DES_ENCRYPT() y DES_DECRYPT().

Opción	Descripción
<code>-enable-external-locking</code>	Esta opción habilita el bloqueo del sistema. No debe utilizarse en sistemas en los que el demonio bloqueado no funciona de forma completa (esto se aplica a Linux aunque puede que en las nuevas versiones ya no se utilice).
<code>-enable-named-pipe</code>	En Windows NT/2000/XP, esta opción habilita la compatibilidad de canalizaciones con nombre.
<code>-T, -exit-info</code>	Una máscara de bits de distintos indicadores utilizada en depuración. No es aconsejable que la utilice, a menos que sepa lo que está haciendo.
<code>-flush</code>	Esta opción garantiza que MySQL vacía todos los cambios realizados en el disco después de cada instrucción SQL. Normalmente es el sistema operativo el que lo procesa. No necesita esta opción a menos que experimente algún problema.
<code>-, -help</code>	Muestra una lista de ayuda y sale.
<code>-init-file=archivo</code>	Indica a MySQL que ejecute las instrucciones SQL contenidas en este archivo cuando se inicie.
<code>-L, -language=...</code>	Esta opción determina el idioma que se utilizará en los mensajes de error de cliente. Puede ser el idioma o la ruta al archivo de idioma.
<code>-l, -log[=archivo]</code>	Las conexiones y las consultas se registrarán en el archivo especificado.
<code>-log-isam[=archivo]</code>	Esta opción registra en el archivo especificado todos los cambios realizados en los archivos MyISAM o ISAM (solamente se utiliza al depurar este tipo de tablas).
<code>-log-slow-queries [archivo]</code>	Registra todas las consultas que tardan más que el valor de la variable <code>long_query_time</code> (en segundos) en ejecutarse en el registro de consultas lentas.
<code>-log-update [archivo]</code>	Registra todas las actualizaciones en el registro de actualizaciones especificado. En su lugar, utilice <code>-log-bin</code> .
<code>-log-bin[=archivo]</code>	Registra todas las actualizaciones en el registro de actualizaciones binarias especificado.
<code>-log-long-format</code>	Registra más información. Si se utiliza el registro de consultas lentas (<code>-log-slow-queries</code>), también se registran todas las consultas que no utilicen un índice.

Opción	Descripción
<code>-low-priority-updates</code>	Si se utiliza esta opción, todas las inserciones, actualizaciones y eliminaciones tendrán una prioridad inferior a las selecciones . Cuando no quiera aplicar esta opción a todas las consultas, puede utilizar <code>SET OPTION SQL_LOW_PRIORITY_UPDATES=1</code> para aplicarla a un subproceso específico o <code>LOW_PRIORITY...</code> para aplicarla a una consulta <code>INSERT, UPDATE</code> o <code>DELETE</code> concreta.
<code>-memlock</code>	Bloquea <code>mysqld</code> en memoria . Esta opción solo está disponible si su sistema admite la función <code>mlockall()</code> (como es el caso de Solaris). Normalmente la utilizará si su sistema operativo tiene problemas y <code>mysqld</code> transfiere a disco. En el valor de la variable <code>locked_in_memory</code> puede comprobar si se ha utilizado la opción <code>-memlock</code> .
<code>-myisam-recover</code> [=opción [,opción...]]	Las opciones disponibles son <code>DEFAULT, BACKUP, FORCE, QUICK</code> o <code>"</code> . Si se configura con cualquiera de estas menos con " , cuando se inicia MySQL comprueba las tablas para ver si se han marcado como colapsadas o no se han cerrado correctamente . De ser así , realizará una comprobación en la tabla e intentará reparar las tablas dañadas . Si se utiliza la opción <code>BACKUP</code> , MySQL crea una copia de seguridad del archivo de datos <code>.MYD</code> si realiza algún cambio durante la reparación (y le asigna la extensión <code>.BAK</code>). La opción <code>FORCE</code> impone la reparación incluso si se pierden datos y la opción <code>QUICK</code> no revisa las filas si no hay bloques de eliminación en los datos . Todos los errores se anotan en el registro de errores, para que pueda ver lo que ha sucedido. Al configurar de forma conjunta las opciones <code>BACKUP</code> y <code>FORCE</code> , MySQL puede recuperarse automáticamente de muchos problemas (en este caso, con ayuda de la copia de seguridad, en caso de que algo salga mal). <code>DEFAULT</code> equivale a no utilizar ninguna opción.
<code>-pid-file=ruta</code>	Especifica la ruta al archivo de <code>Id.</code> del proceso.
<code>-P, -port=...</code>	El número de puerto que utiliza MySQL para comunicarse con conexiones TCP/IP .
<code>-o, -old-protocol</code>	Indica que MySQL utiliza el antiguo protocolo 3.20 para ser compatible con clientes de la misma antigüedad .

Opción	Descripción
<code>-one-thread</code>	Especifica que MySQL solamente utiliza un subproceso. Solamente debe utilizar esta opción para depurar.
<code>-O, -set-variable var=opción</code>	Define una variable que le permite optimizarla. La lista completa de variables se incluye en la tabla 13.2.
<code>-safe-mode</code>	Ignora algunas fases de optimización. Implica el uso de la opción <code>-skip-delay-key-write</code> .
<code>-safe-show-database</code>	Solamente se utiliza en las primeras versiones de MySQL 4. Si se configura, los usuarios que no tengan ningún privilegio relacionado con la base de datos, no verán enumerada dicha base de datos cuando ejecuten una instrucción <code>SHOW DATABASES</code> (el privilegio <code>SHOW DATABASES</code> elimina la necesidad de esta opción).
<code>-safe-user-create</code>	Esta opción refuerza la seguridad ya que permite a los usuarios crear nuevos usuarios (por medio de la instrucción <code>GRANT</code>) a menos que tengan el privilegio <code>INSERT</code> en la tabla <code>mysql.user</code> o en una de las columnas de esta tabla.
<code>-skip-concurrent-insert</code>	Anula las inserciones simultáneas (donde las selecciones y las inserciones se pueden realizar al mismo tiempo en tablas optimizadas). Solamente se utiliza para depuración.
<code>-skip-delay-key-write</code>	Hace que MySQL ignore la opción <code>delay_key_write</code> en todas las tablas.
<code>-skip-grant-tables</code>	Esta opción inicia MySQL sin las tablas de privilegio (todo el mundo tiene acceso completo). No debe utilizarla a menos que sea necesario (por ejemplo, cuando como raíz, haya olvidado la contraseña). Cuando termine, ejecute <code>mysqladmin flush-privileges</code> o <code>mysqladmin reload</code> para volver a utilizar las tablas de privilegios.
<code>-skip-host-cache</code>	Los nombres de anfitrión se suelen almacenar en cache, pero podemos obligar a MySQL a que consulte el servidor DNS cada vez que se conecte. Puede reducir las velocidades de conexión.
<code>-skip-external-locking</code>	Deshabilita el bloqueo del sistema, lo que tiene importantes consecuencias para <code>mysiamchk</code> , como se indica en un capítulo anterior.

Opción	Descripción
<code>-skip-name-resolve</code>	MySQL no resuelve nombres de anfitrión, por lo que todos los valores de las columnas <code>Host</code> de las tablas de privilegios deben ser una dirección IP concreta (o <code>localhost</code>). No resuelve nombres de servidor. Esta opción puede mejorar las velocidades de conexión si tiene muchos anfitriones o DNS bajos.
<code>-skip-networking</code>	Esta opción hace que MySQL solamente permita conexiones locales. No escuchara ninguna conexión TCP/IP. Puede ser una buena medida de seguridad.
<code>-skip-new</code>	MySQL utiliza ISAM como tipo de tabla predeterminado y no utiliza ninguna de las opciones que aparecieron como novedad en la versión 3.23. También implica el uso de <code>-skip-delay-key-write</code> . Esta opción ya no se necesita, a menos que cambie su comportamiento.
<code>-skip-symlink</code>	Esta opción garantiza que nadie puede modificar o cambiar el nombre de los archivos a los que apunta un archivo de enlace simbólico del directorio de datos. Debería utilizarla si no usa enlaces simbólicos como medida de seguridad para evitar que nadie borre o cambie el nombre de un archivo que no se encuentre en el directorio de datos <code>mysqld</code> .
<code>-skip-safemalloc</code>	Esta opción aumenta el rendimiento ya que evita la comprobación de sobrescrituras cada vez que se asigna y se libera memoria (estas comprobaciones se realizan cuando MySQL se configura con <code>-with-debug=full</code>).
<code>-skip-show-database</code>	Si se configura, la instrucción <code>SHOW DATABASES</code> no devuelve resultados a menos que el cliente tenga el privilegio <code>PROCESS</code> (el privilegio <code>SHOW DATABASES</code> , presentado en las primeras versiones de MySQL 4, elimina la necesidad de esta opción).
<code>-skip-stack-trace</code>	No utiliza rastreos de pila (lo que resulta muy útil para ejecutar <code>mysqld</code> bajo un depurador). Algunos sistemas requieren esta opción para obtener un archivo de volcado de memoria.
<code>-skip-thread-priority</code>	Desactiva el uso de prioridades de subprocesos para obtener un tiempo de respuesta más rápido.
<code>-socket=ruta</code>	La ruta del archivo de socket utilizado en conexiones locales (en lugar del predeterminado, normalmente <code>/tmp/mysql.sock</code>).

Opción	Descripción
<code>-sql-mode=opción</code> [, opción [, opción...]]	Estas opciones permiten definir las diferencias entre el estandar ANSI y MySQL. Se trata de REAL_AS_FLOAT, PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE, SERIALIZE, ONLY_FULL_GROUP_BY o "" (para restablecer los valores). El uso de todas ellas equivale a utilizar la opción <code>-ansi</code> . Mas adelante describiremos cada una de las diferencias.
<code>-temp-pool</code>	Solamente se necesita esta opción cuando un sistema operativo experimenta pérdidas de memoria al crear grandes cantidades de nuevos archivos con distintos nombres (como ocurría en algunas versiones de Linux). Por el contrario, MySQL utiliza un pequeño conjunto de nombres para archivos temporales.
<code>-transaction-isolation= { READ-UNCOMMITTED READ-COMMITTED REPEATABLE-READ SERIALIZABLE }</code>	Determina el nivel de aislamiento de transacciones predeterminado. Encontrara mas información en capítulos anteriores .
<code>-t, -tmpdir=ruta</code>	El directorio utilizado para almacenar archivos y tablas temporales. Resulta muy util para distinguirlo de su espacio temporal habitual en caso de que este sea demasiado escaso como para almacenar tablas temporales.
<code>-u, -user= [nombre_de_usuario { id_de_usuario]</code>	Proporciona un nombre de usuario para ejecutar MySQL. Cuando inicie mysqld como raíz, es necesario utilizar esta opción.
<code>-V, -version</code>	Muestra información sobre la versión.
<code>-W, -warnings</code>	Se muestran advertencias en el archivo de errores.

Descripción de las variables mysqld

Al ejecutar `mysqladmin variables` desde la línea de comandos o `SHOW VARIABLES` desde la línea de comandos `mysql`, aparece una extensa lista de variables. A **menudo**, están relacionadas con una opción que puede definir en el archivo de configuración. En la tabla 13.2 se incluye la **descripción** de las distintas variables que aparecen. En función de la configuración y la versión de su sistema, puede que no tenga todas estas opciones o, lo mas probable, como MySQL **añade** continuamente opciones nuevas, es que aparezcan muchas mas.

Tabla 13.2. Variables `mysqld`

Variable	Descripción
<code>ansi_mode</code>	MySQL no solo tiene distintas extensiones sino también numerosas diferencias con respecto al comportamiento ANSI estandar. Si se configura esta opción, MySQL se ejecutara en modo ANSI (mas adelante analizaremos los cambios que se producen)
<code>back_log</code>	El numero de solicitudes de conexión en cola que MySQL puede tener en espera antes de rechazar alguna de ellas. Es similar al tamaño de la cola de escucha de conexiones TCP/IP entrantes, que también limita el sistema operativo. Se aplicara el valor <code>back_log</code> mas bajo y el límite del sistema operativo. Consulte la documentación del sistema operativo (por ejemplo <code>manlisten</code> en Unix) si necesita mas informacibn.
<code>basedir</code>	La ruta al directorio base o al directorio de instalacion de MySQL.
<code>bdb_cache_size</code>	Tamaño de los datos e indices de cache asignados al bufer para tablas BDB. Si su sistema no utiliza tablas BDB, use la opción <code>-skip-bdb</code> para no gastar memoria.
<code>bdb_lock_detect</code>	El detector de bloqueo Berkeley, que puede ser DEFAULT, OLDEST, RANDOM O YOUNGEST.
<code>bdb_log_buffer_size</code>	El tamaño del bufer para registros BDB. Si su sistema no utiliza tablas BDB, utilice la opción <code>-skip-bdb</code> para no gastar memoria.
<code>bdb_home</code>	Directorio base de las tablas BDB, que equivaldra a <code>-datadir</code> .
<code>bdb_max_lock</code>	Numero maximo de bloqueos que se pueden aplicar a una tabla BDB. Puede aumentarlo si sus transacciones son extensas o sus consultas tienen que examinar gran cantidad de filas. Errores como <code>Lock table is out of available locks</code> o <code>Got error 12 from ...</code> indican que debe aumentar el valor. El valor predeterminado es 10000.
<code>bdb-no-recover</code>	Si se configura, MySQL no inicia BDB en modo de recuperación. Normalmente tendra que configurarla si hay algun fallo en los registros BDB que impida un inicio correcto.

Variable	Descripción
<code>bdb-no-sync</code>	Si se configura, MySQL no podrá vaciar los registros de forma sincronizada.
<code>Bdb_log_dir</code>	El directorio que contiene los registros BDB.
<code>Bdb_shared_data</code>	Si se configura, MySQL inicia BDB en modo multiproceso, lo que significa que no se utiliza <code>DB_PRIVATE</code> .
<code>bdb_tmpdir</code>	La ubicación del directorio de archivos temporales BDB.
<code>binlog_cache_size</code>	Tamaño de la cache para transacciones que se escriben en el registro binario. Si las transacciones son extensas y ocupan más de la cache predeterminada de 32 KB, debe aumentar este valor.
<code>character_set</code>	El conjunto de caracteres que se utiliza en caso de no especificar otro en concreto (normalmente latin1).
<code>character-sets</code>	Lista completa de los conjuntos de caracteres admitidos. Si va a compilar MySQL desde código fuente y sabe que nunca los va a utilizar, puede compilar MySQL para que no admita los conjuntos de caracteres adicionales.
<code>concurrent_inserts</code>	Si se activa (estado predeterminado), puede realizar inserciones en tablas MyISAM al mismo tiempo que ejecuta consultas, lo que supone un aumento del rendimiento (siempre que la tabla no contenga huecos de registros borrados anteriormente. Lo puede garantizar si optimiza regularmente las tabla~) Las opciones <code>-safe 0 -skip-lo</code> anulan.
<code>connect_timeout</code>	Tiempo en segundos que MySQL espera a los paquetes antes de rechazarlos con <code>Bad handshake</code> . El valor predeterminado es de 5 segundos. Nos ayuda a prevenir ataques de negación de servicio cuando se realizan gran cantidad de conexiones incorrectas para evitar la conexión de usuarios legítimos.
<code>datadir</code>	Directorio en el que se almacenan los datos.
<code>delay_key_write</code>	Si se activa (estado predeterminado), MySQL no vacía el bufer clave de una tabla en todas las actualizaciones de índice de tablas con la opción <code>DELAY_KEY_WRITE</code> . Por el contrario, solamente se vacía cuando se cierra la tabla. Esto aumenta la

Variable	Descripción
	velocidad a la que se escriben las claves pero también el riesgo de fallos, por lo que debe revisar las tablas regularmente. Puede especificar la opción <code>DELAY_KEY_WRITE</code> como predeterminada en todas las tablas si utiliza la opción <code>-delay-key-write-for-all-tables</code> . Las nuevas opciones <code>-safe</code> o <code>-skip</code> anulan la anterior.
<code>delayed_insert_limit</code>	Después de insertar filas <code>delayed_insert_limitrows</code> , MySQL comprueba si hay alguna instrucción SELECT pendiente y las procesa antes de continuar con las instrucciones <code>INSERT DELAYED</code> restantes. El valor predeterminado es de 100 filas.
<code>delayed_insert_timeout</code>	Determina el tiempo que debe esperar un subproceso INSERT DELAYED a las instrucciones INSERT antes de finalizar.
<code>delayed_queue_size</code>	Numero de filas asignadas a la cola <code>INSERT DELAYED</code> . Si se alcanza este limite , los clientes que ejecuten <code>INSERT DELAYED</code> tendrán que esperar hasta que haya espacio.
<code>flush</code>	Si se configura, MySQL vacía todos los cambios en disco después de cada instrucción SQL . Normalmente el sistema operativo se encarga de esta operación . El valor predeterminado es <code>OFF</code> ya que no es necesario utilizar esta opción a menos que haya algun problema.
<code>flush_time</code>	Tiempo en segundos entre eliminaciones automáticas (las tablas se cierran y se sincronizan en disco). Se suele configurar a menos que ejecute un sistema con pocos recursos o con Windows 95/98/Me .
<code>ft_min_word_len</code>	Longitud minima de palabras que se incluyen en un índice FULLTEXT . Después de cambiar este valor, tendrá que volver a generar todos los índices <code>FULLTEXT</code> . El valor predeterminado es 4 .
<code>ft_max_word_len</code>	Longitud maxima de palabras que se incluyen en un índice FULLTEXT . Después de cambiar este valor, tendrá que volver a generar todos los índices <code>FULLTEXT</code> . El valor predeterminado es 254 .
<code>ft_max_word_len_sort</code>	Las palabras con esta longitud o inferior se añaden al índice FULLTEXT con la recreación rapida del índice cuando este se vuelve a generar. Las pala-

Variable	Descripción
	<p>bras de mayor longitud se añaden al índice de la forma lenta. Es poco probable que tenga que modificar el valor predeterminado (20) a menos que las palabras de su índice tengan una longitud media inusual. Si se define un valor demasiado alto, el proceso sera mas lento ya que los archivos temporales seran mayores y se añadirán menos claves en los bloques de ordenacion. Si el valor es demasiado bajo, se añadirán demasiadas palabras de forma lenta.</p>
ft_boolean_syntax	<p>La lista de operadores admitidos por la instrucción MATCH ... AGAINST(... IN BOOLEAN MODE) (+ -><()~*:""&).</p>
have_innodb	<p>Se configura con el valor YES si MySQL es compatible con tablas InnoDB o con el valor DISABLE si se utiliza la opcion -skip-bdb.</p>
have_bdb	<p>Se configura con el valor YES si MySQL es compatible con tablas BDB o con el valor DISABLE si se utiliza la opcion -skip-bdb.</p>
have_raid	<p>Se configura con el valor YES si MySQL es compatible con la opcion RAID (matriz redundante de discos economicos).</p>
have_openssl	<p>Se configura con el valor YES si MySQL es compatible con cifrado SSL entre el cliente y el servidor.</p>
init_file	<p>El nombre del archivo que contiene instrucciones SQL que se ejecutarán cuando se inicie el servidor (de manera predeterminada, no se configura ningun archivo).</p>
innodb_data_home_dir	<p>Directorio principal de los archivos de datos InnoDB que se utiliza como parte comun de la ruta. Si no se menciona, sera el mismo que datadir. Al configurar esta variable con una cadena vacia puede utilizar rutas de archivo absolutas en innodb_data_file_path.</p>
innodb_data_file_path	<p>Rutas a archivos de datos individuales y a sus respectivos tamaños. La parte innodb_data_home_dir de la ruta se añade a esta para formar la ruta completa. Los tamaños de archivo se especifican en megabytes (M) o en gigabytes (G). Pueden superar los 4 GB en sistemas operativos que admiten archivos de gran tamaño y la suma de los tamaños debe alcanzar al menos los 10 MB.</p>

Variable	Descripción
<code>innodb_mirrored_log_groups</code>	Especifica el numero de copias identicas de grupos de registro que hay que almacenar de la base de datos. El valor actual debe ser 1.
<code>innodb_log_group_home_dir</code>	Especifica la ruta del directorio de archivos de registro InnoDB.
<code>innodb_log_files_in_group</code>	Especifica el numero de archivos de registro en el grupo de registro. El valor sugerido es 3 (los registros se escriben en rotación).
<code>innodb_log_file_size</code>	Especifica el tamaño en megabytes de cada uno de los archivos de registro de un grupo de registros. Los valores sugeridos son de 1 MB para <code>1/innodb_log_files_in_group</code> del <code>innodb_buffer_pool_size</code> . Un valor mayor ahorra procesos de entrada/salida (I/O) de disco ya que hay una menor actividad de eliminación, pero reduce la recuperación en caso de colapso. El tamaño total de los archivos de registro no debe superar los 4 GB en equipos de 32 bits.
<code>innodb_log_buffer_size</code>	Especifica el tamaño del búfer utilizado para escribir registros. El valor sugerido es de 8 MB. Cuanto mayor sea el bufer, se realizaran menos entradas y salidas de disco, ya que no es necesario escribir las transacciones en disco hasta que se ejecutan.
<code>innodb_flush_log_at_trx_commit</code>	Si se configura, los registros se vacian en disco cuando se ejecuta la transacción (y por lo tanto son permanentes y pueden sobrevivir a cualquier colapso). Normalmente debe configurar esta variable con el valor <code>ON</code> si las transacciones son importantes. Tambien puede configurarla como <code>OFF</code> si el rendimiento es lo mas importante y quiere reducir la entrada y la salida de disco en favor de la seguridad.
<code>innodb_log_arch_dir</code>	Especifica el directorio en el que se alojan los registros. Debe equivaler a <code>innodb_log_group_home_dir</code> ya que el almacenaje de registros no se utiliza actualmente.
<code>innodb_log_archive</code>	Si se configura, se archivarán los archivos de registro InnoDB. Actualmente, MySQL se recupera por medio de sus propios archivos de registro, por lo que debe configurar esta variable con el valor <code>OFF</code> .

Variable	Descripción
<code>innodb_buffer_pool-size</code>	Especifica el tamaño en bytes del bufer de memoria utilizado para almacenar en cache índices y datos de tablas. Cuanto mayor sea, el rendimiento será mejor ya que se necesita una menor entrada/salida de disco. Le sugerimos un 80 por ciento de memoria en servidores dedicados de bases de datos ya que cualquier valor mayor provocaría la paginación del sistema operativo.
<code>innodb_additional_mem_pool-size</code>	Especifica el tamaño en bytes de una agrupación de memoria utilizada para almacenar información sobre las estructuras de datos internas. Puede utilizar 2 MB, pero si tiene gran cantidad de tablas, asegúrese de asignar suficiente memoria ; en caso contrario, MySQL utilizará la del sistema operativo (si esto ocurre y aumenta el valor, verá las advertencias en el registro de errores).
<code>innodb file_io_threads</code>	El número de subprocesos de entrada/salida de archivos en InnoDB . El valor sugerido es 4, pero Windows puede beneficiarse de un parámetro mayor.
<code>innodb_lock_wait_timeout</code>	Tiempo en segundos que espera una transacción InnoDB a un bloqueo antes de invertirlo. InnoDB detecta inmediatamente puntos muertos en sus propias tablas de bloqueo, pero si provienen del exterior (por ejemplo de una instrucción LOCK TABLES), pueden producirse dichos puntos muertos, en cuyo caso utilizaremos este valor.
<code>innodb_flush_method</code>	Método de vaciado. El predeterminado es <code>fdatasync</code> y el alternativo es <code>o_DYSYNC</code> . Normalmente <code>fdatasync</code> es más rápido aunque en algunas versiones de Linux y Unix ha resultado ser más lento.
<code>interactive_timeout</code>	Tiempo en segundos que el servidor espera a cualquier actividad en una conexión interactiva (la que utiliza la opción CLIENT_INTERACTIVE al conectarse) antes de cerrarla. La opción <code>wait-timeout</code> se aplica a conexiones convencionales. El valor predeterminado es 2800 .
<code>join_buffer_size</code>	Tamaño en bytes del bufer utilizado para uniones completas (uniones en las que no se utilizan índices). El bufer se asigna a cada una de las uniones . Al aumentar este valor, se incrementa la velocidad de las uniones pero la mejor técnica para acelerar una unión consiste en añadir los índices apropiados.

Variable	Descripción
<code>key_buffer_size</code>	Tamaño en bytes del bufer utilizado con bloques de índices. Lo describiremos en un apartado posterior.
<code>language</code>	Ubicación del archivo de idioma utilizado en los mensajes de error.
<code>large_file-support</code>	ON si MySQL se ha compilado con compatibilidad con archivos de gran tamaño. ON es el valor predeterminado.
<code>locked_in_memory</code>	ON si MySQL se ha bloqueado en memoria (es decir, se ha iniciado mysqld con la opción <code>-memlock</code>). El valor predeterminado es OFF. Utilizaremos ON si el sistema operativo tiene problemas y mysqld se intercambia a disco.
<code>log</code>	ON si se activa el registro de todas las consultas.
<code>log_update</code>	ON si se activa el registro de todas las actualizaciones (debería utilizar el registro binario para esto).
<code>log_bin</code>	ON si se activa el registro binario de actualizaciones.
<code>log_slave_updates</code>	ON si se registran las actualizaciones que provienen del esclavo.
<code>long-query-time</code>	Tiempo en segundos que define una consulta lenta. Las consultas que tarden más de este valor harán que se incremente el contador <code>slowqueries</code> y se registrarán en el archivo de registro de consultas lentas si se ha activado el registro de las mismas.
<code>lower-case-table_names</code>	Se configura con el valor 1 si los nombres de las tablas se almacenan en minúsculas y no discriminan entre mayúsculas y minúsculas. El valor predeterminado es 0.
<code>max_allowed_packet</code>	Tamaño máximo permitido, expresado en bytes, de un paquete de datos. El bufer de mensajes se inicializa con el tamaño especificado por <code>next_buffer_length</code> pero puede superar este tamaño. Si utiliza extensas columnas BLOB o TEXT, configure esta variable con el valor de la de mayor tamaño.
<code>max_binlog_cache-size</code>	La mayor cantidad de memoria, en bytes, que puede utilizar una transacción multinstrucción sin iniciar un error de tipo "la transacción multinstrucción requiere más de los bytes <code>max_binlog_cache-size</code> de almacenamiento".

Variable	Descripción
<code>max_binlog_size</code>	Cuando el registro binario supera este tamaño, los registros se alternan y se crea uno nuevo.
<code>max_connections</code>	Numero maximo permitido de conexiones.
<code>max_connect_errors</code>	Número maximo de intentos de conexión por parte de un servidor antes de que se bloquee para recibir otras conexiones. Este limite intenta reducir la posibilidad de ataques de negación de servicio. Los servidores se pueden desbloquear por medio de <code>FLUSH HOSTS</code> .
<code>max_delayed_threads</code>	Número maximo de subprocesos que pueden procesar instrucciones <code>INSERT DELAYED</code> . Una vez alcanzado, las instrucciones <code>INSERT</code> posteriores seran inserciones normales y no utilizaran el atributo <code>DELAYED</code> .
<code>max_heap_table_size</code>	Tamaño maximo, en bytes, que pueden tener las tablas <code>HEAP</code> .
<code>max_join_size</code>	Las uniones que MySQL determine que devuelven un numero de filas superior a este limite devolveran un error. De esta forma se evita que los usuarios ejecuten de forma accidental (o a proposito) extensas consultas que devuelven millones de filas y que utilicen gran cantidad de recursos.
<code>max_sort_length</code>	Al ordenar campos <code>BLOB</code> o <code>TEXT</code> solamente se admite este maximo de bytes. Por ejemplo, si se configura como 1024, solamente se utilizaran los 1024 primeros caracteres para ordenar los campos.
<code>max_user_connections</code>	Determina el numero maximo de conexiones activas que puede tener un usuario. El valor predeterminado, 0, indica que no hay limite (excepto para <code>max-connections</code>).
<code>max_tmp_tables</code>	Numero maximo de tablas temporales que un cliente puede tener abiertas al mismo tiempo (en la actualidad no se utiliza esta variable; consulte su documentación para ver si ha cambiado).
<code>max_write_lock_count</code>	Si se produce esta cantidad de bloqueos de escritura consecutivos, MySQL permitira la ejecucion de un determinado numero de bloqueos de lectura.
<code>myisam_bulk_insert_tree_size</code>	Cuando MySQL inserta grandes volúmenes de datos (por ejemplo, <code>LOAD DATA INFILE...</code>), utiliza una cache en forma de arbol para acelerar el pro-

Variable	Descripción
<code>myisam_recover_options</code>	<p>ceso. Se trata del tamaño máximo, en bytes, de la cache de cada subproceso. El valor predeterminado es 8 MB; al configurarlo con el valor 0, la opción se desactiva. La cache solamente se utiliza cuando se añade a una tabla que no esta vacía.</p> <p>Las opciones disponibles son DEFAULT, BACKUP, FORCE, QUICK y OFF. Si se ejecuta con cualquiera de ellas menos con OFF, al iniciar MySQL se comprueban las tablas para comprobar si estan marcadas como colapsadas o no se han cerrado correctamente. De ser así, se revisa la tabla y se intentan reparar las tablas dañadas. Si se utiliza la opción BACKUP, MySQL creará una copia de seguridad del archivo de datos .MYD en caso de que se realice algún cambio durante el proceso de reparación (le asigna la extensión .BAK). La opción FORCE impone la reparación incluso aunque se puedan perder datos y la opción QUICK no comprueba las filas si no hay bloques de eliminación en los datos. Todos los errores se anotan en el registro de errores, para poder ver lo que ha sucedido. La configuración conjunta de las opciones BACKUP y FORCE permite que MySQL se recupere automáticamente de muchos problemas (y puede recurrir a la copia de seguridad en caso de problemas más graves).</p>
<code>myisam-sort-buffer_size</code>	<p>Tamaño en bytes del bufer asignado al ordenar o reparar un índice.</p>
<code>myisam_max_extra_sort_file_size</code>	<p>Cuando MySQL crea un índice, resta el tamaño de cache de claves del tamaño de la tabla temporal que utilizaría con la creación de índices rápidos. Si la diferencia es mayor que esta cantidad (especificada en megabytes), MySQL utiliza el método de cache de claves.</p>
<code>myisam_max_sort_file_size</code>	<p>Tamaño máximo (en megabytes) del archivo temporal que genera MySQL cuando crea o repara índices. Si se supera este tamaño, MySQL utiliza el método más lento de cache de claves para crear o reparar el índice.</p>
<code>net_buffer_length</code>	<p>Tamaño en bytes con el que se configura el bufer de comunicación entre consultas. Para ahorrar memoria en sistemas con una escasez de la misma, configure esta variable con la longitud prevista de</p>

Variable	Descripción
	las instrucciones SQL enviadas por los clientes. Si la instrucción supera esta longitud, se incrementa automáticamente hasta el tamaño de <code>max_allowed_packet</code> .
<code>net_read_timeout</code>	Tiempo expresado en segundos que MySQL espera datos de una conexión antes de cancelar la lectura. Si no se espera ningún dato, se utiliza <code>net_write_timeout</code> . <code>slave_net_timeout</code> se utiliza en la conexión principal-esclavo.
<code>net_retry_count</code>	Numero de intentos de lectura de un puerto de comunicaciones antes de cancelar.
<code>net_write_timeout</code>	Numero de segundos que se espera a que se escriba un bloque en una conexión antes de cancelar.
<code>open_files_limit</code>	MySQL utiliza este valor para reservar descriptores de archivos. Puede aumentarlo si se produce un error de tipo "demasiados archivos abiertos". Normalmente se configura con el valor 0, en cuyo caso MySQL utiliza el mayor de <code>max_connections*5</code> o <code>max_connections + table_cache*2</code> .
<code>pid_file</code>	Ruta al archivo pid.
<code>port</code>	Numero de puerto que utiliza MySQL para escuchar conexiones TCP/IP.
<code>protocol_version</code>	Version del protocolo que utiliza MySQL.
<code>record_buffer</code>	MySQL asigna un bufer de este tamaño (en bytes) a cada subproceso que realiza un examen secuencial (en el que los registros se leen por orden, uno detras de otro). Si realiza gran cantidad de estos exámenes, tendra que incrementar el valor.
<code>record_rnd_buffer</code>	Un bufer de este tamaño se asigna cuando se leen filas en orden no secuencial (por ejemplo, despues de ordenarlas). Las filas que se leen de esta forma evitan busquedas de disco. Si no se configura, sera igual que <code>record_buffer</code> .
<code>query_buffer_size</code>	El tamaño inicial, en bytes, asignado al bufer de consultas. Deberia resultar suficiente para la mayoría de las consultas; en caso contrario, aumentelo.
<code>query_cache_limit</code>	Límite, en bytes, de la cache de consultas. Los resultados superiores a este valor no se almacenan en cache. El valor predeterminado es de 1 MB.

Variable	Descripción
<code>query_cache_size</code>	El tamaño en bytes asignado a la cache de consultas (en la que se almacenan los resultados de consultas anteriores). 0 indica que la cache se ha deshabilitado.
<code>query_cache_startup_type</code>	Puede ser 0, 1 o 2. 0 (desactivado) indica que MySQL no almacena en cache ni recupera resultados. 1 (activado) significa que MySQL almacena en cache todos los resultados a menos que sean <code>SQL_NO_CACHE</code> . 2 (solicitud) significa que solamente se almacenan en cache las consultas con <code>SQL_CACHE</code> .
<code>safe_show_database</code>	Solamente se utiliza en las primeras versiones de MySQL 4. Si se configura, los usuarios sin privilegios de la base de datos no verán dicha base de datos enumerada cuando ejecuten una instrucción <code>SHOW DATABASES</code> (el privilegio <code>SHOW DATABASES</code> elimina esta necesidad).
<code>server_id</code>	El Id. del servidor. En operaciones de duplicación, es muy importante para identificar al servidor.
<code>skip_external_locking</code>	Desactiva el bloqueo del sistema si esta definido como <code>ON</code> . Tiene importantes consecuencias para <code>myisamchk</code> (en capítulos anteriores encontrará más información al respecto).
<code>skip_networking</code>	Es <code>ON</code> si MySQL solamente permite conexiones locales.
<code>skip_show_database</code>	Si se configura, la instrucción <code>SHOW DATABASES</code> no devuelve resultados a menos que el cliente tenga el privilegio <code>PROCESS</code> (el privilegio <code>SHOW DATABASES</code> , que apareció en las primeras versiones de MySQL 4, elimina esta necesidad).
<code>slave_net_timeout</code>	Tiempo en segundos que espera MySQL a recibir más datos de una conexión principal/esclavo antes de cancelar la lectura.
<code>slow_launch_time</code>	Tiempo en segundos antes de que el inicio de un subproceso incremente el contador <code>show_launch_threads</code> .
<code>socket</code>	Ruta al socket de Unix utilizado por el servidor.
<code>sort_buffer</code>	Tamaño en bytes asignado al bufer que utilizan las ordenaciones. Encontrará más información al principio del capítulo.
<code>table_cache</code>	Número de tablas abiertas para todos los subprocesos. Encontrará más información en un apartado anterior.

Variable	Descripción
<code>table_type</code>	Tipo de tabla predeterminado (normalmente MyISAM).
<code>thread_cache_size</code>	Numero de subprocesos que se guardan en cache para su reutilizacion. Los nuevos subprocesos se obtienen de la cache en caso de que haya alguno disponible, mientras que los subprocesos cliente se ubican en la cache durante la desconexion siempre que haya espacio disponible. Si tiene una gran cantidad de nuevas conexiones, puede aumentar este valor para mejorar el rendimiento. Los sistemas con una buena implementación de subprocesos no suelen aprovechar esta ventaja . Puede ver su eficacia si la compara con las variables de estado <code>Connections</code> y <code>Threads_created</code> .
<code>thread_concurrency</code>	En sistemas Solaris , MySQL utiliza este valor para determinar si debe invocar la funcion <code>thr_setconcurrency()</code> , que ayuda al sistema de subprocesos ya que conoce el numero de subprocesos que deben ejecutarse simultáneamente .
<code>thread_stack</code>	Tamaño en bytes de la pila de cada subproceso. El comportamiento del análisis comparativo <code>crashme</code> depende de este valor.
<code>timezone</code>	Zona horaria del servidor
<code>tmp_table_size</code>	Tamaño maximo de una tabla temporal en memoria . Si supera este valor, se convierte, automaticamente, en una tabla MyISAM en disco. Si realiza gran cantidad de consultas que generan extensas tablas temporales (como ocurre con clausulas <code>GROUP BY</code> complejas) y dispone de memoria suficiente, es recomendable que aumente este valor.
<code>tmpdir</code>	Directorio utilizado para almacenar archivos y tablas temporales.
<code>version</code>	Numero de version del servidor
<code>wait_timeout</code>	Tiempo en segundos que espera MySQL a la actividad de una conexión antes de cerrarla. La variable <code>interactive_timeout</code> se aplica a conexiones interactivas.

Análisis de todas las variables de estado

Las variables de estado se restablecen cada vez que se reinicia el servidor. Nos permiten controlar el comportamiento del servidor y tambien identificar posibles

cuellos de botella, problemas y las mejoras que podemos realizar. En la tabla 13.3 se incluye una exhaustiva lista de estas variables.

Tabla 13.3. Valores de estado MySQL

Valor	Descripción
Aborted_clients	Indica el numero de conexiones canceladas debido a que, por alguna razon, el cliente no las cerro correctamente. Esto puede suceder si el cliente no invoca la funcion <code>mysql_close()</code> antes de salir, si se han excedido los límites <code>wait_timeout</code> o <code>interactive_timeout</code> o si el cliente la ha cerrado durante una transferencia.
Aborted_connects	Indica el numero de intentos de conexión fallidos al servidor MySQL. Puede deberse a que el cliente haya intentado conectarse con la contraseña equivocada, que no tenga permiso para conectarse, que tarde mas de los segundos <code>connect_timeout</code> en obtener un paquete de conexión o que el paquete no contenga la información correcta.
Bytes_received	El numero de bytes que se han recibido de todos los clientes.
Bytes_sent	Numero de bytes que se han enviado a todos los clientes.
Com_[instrucción]	Existe una de estas variables para cada tipo de instruccion. El valor indica el numero de veces que se ha ejecutado esta instruccion.
Connections	Numero de intentos de conexión al servidor MySQL.
Created_tmp_disk_tables	Numero de tablas temporales implicitas en disco que se crearon durante la ejecución de instrucciones.
Created_tmp_tables	Numero de tablas temporales implicitas en memoria que se crearon durante la ejecución de instrucciones.
Created_tmp_files	Numero de archivos temporales creados por <code>mysqld</code> .
Delayed_insert_threads	Numero de subprocesos de inserción retrasados actualmente en uso.
Delayed_writes	Numero de registros escritos por una instruccion <code>INSERT DELAYED</code> .
Delayed_errors	Numero de registros escritos por una instruccion <code>INSERT DELAYED</code> cuando se produce un error. El

Valor	Descripción
	error mas habitual es la presencia de claves duplicadas.
Flush_commands	Numero de instrucciones FLUSH ejecutadas.
Handler_commit	Numero de comandos COMMIT internos.
Handler_delete	Numero de veces que se ha eliminado una fila de una tabla.
Handler_read_first	Numero de veces que se ha leído la primera entrada de un índice, lo que normalmente indica un examen completo del índice (por ejemplo, si se indexa indexed_col, la instrucción SELECT indexed_col from nombre_de_tabla genera un examen completo del índice).
Handler_read_key	Numero de solicitudes en las que se utiliza un índice al leer una fila. Es aconsejable aumentar este valor de forma rapida, ya que indica que sus consultas utilizan indices.
Handler_read_next	Numero de solicitudes para leer la siguiente fila de un índice. Este valor aumenta si realiza un examen completo del índice o si consulta un índice en funcion de una constante de rango.
Handler_read_prev	Numero de solicitudes para leer la fila anterior de un índice. Se puede utilizar con una instrucción de tipo SELECT fieldlist ORDER BY fields DES.
Handler_read_rnd	Numero de solicitudes de lectura de una fila en funcion de una posición fija. Las consultas que requieren que se ordenen los resultados incrementaran este contador.
Handler_read_rnd_next	Numero de solicitudes para leer la siguiente fila del archivo de datos. No es aconsejable utilizar un valor alto, lo que significaria que las consultas no utilizan los indices y tienen que leer desde el archivo de datos.
Handler_rollback	Número de comandos ROLLBACK internos.
Handler_update	Número de solicitudes para actualizar un registro en una tabla.
Handler_write	Numero de solicitudes para insertar un registro en una tabla.
Key_blocks_used	Numero de bloques utilizados en la cache de claves.

Valor	Descripción
Key_read_requests	Número de solicitudes que leen un bloque de claves desde la cache de claves. La proporción Key_reads:Key_read_requests no debe ser mayor de 1:100 (por ejemplo, no es aconsejable 1:10).
Key_reads	Número de lecturas físicas que leen un bloque de claves desde disco. La proporción Key_reads:Key_read_requests no debe ser mayor de 1:100 (por ejemplo, no es aconsejable 1:10).
Key_write_requests	Numero de solicitudes que hace que se escriba un bloque de claves en cache.
Key_writes	Numero de veces en que se ha escrito físicamente un bloque de claves en disco.
Max_used_connections	Número máximo de conexiones que se utilizan en un momento dado. Encontrara mas informacion en apartados anteriores.
Not_flushed_key= blocks	Numero de bloques de claves que se han modificado en la cache de claves pero que todavia no se han vaciado en disco.
Not_flushed_delayed_rows	Numero de registros en colas INSERT DELAY en espera de ser escritos.
Open_tables	Numero de tablas actualmente abiertas. Encontrara mas informacion al respecto en apartados anteriores.
Open_files	Numero de archivos actualmente abiertos.
Open_streams	Numero de flujos abiertos. Se utiliza principalmente con propositos de registro.
Opened_tables	Numero de tablas abiertas. Encontrara mas informacion al respecto en apartados anteriores.
Qcache_queries_in_cache	Numero de consultas en cache.
Qcache_inserts	Numero de consultas añadidas a la cache.
Qcache_hits	Numero de ocasiones en las que se ha accedido a la cache de consultas.
Qcache_not_cached	Numero de consultas no almacenadas en cache (debido a su excesivo tamaño o por QUERY_CACHE_TYPE).
Qcache_free_memory	Cantidad de memoria disponible para la cache.

Valor	Descripción
<code>Qcache_total_blocks</code>	Numero total de bloques de la cache de consultas.
<code>Qcache_free_blocks</code>	Numero de bloques de memoria libres en la cache de consultas.
<code>Questions</code>	Numero total de consultas iniciadas.
<code>Rpl_status</code>	Estado de duplicación segura (solamente se utiliza en las últimas versiones de MySQL 4).
<code>Select_full_join</code>	Numero de uniones que se han realizado sin utilizar indices. No es aconsejable un valor demasiado alto.
<code>Select_full_range_join</code>	Numero de uniones realizadas a traves de una busqueda de rangos en la tabla de referencias.
<code>Select_range</code>	Numero de uniones realizadas por medio de rangos en la primera tabla (puede utilizar un valor alto).
<code>Select_range_check</code>	Número de uniones realizadas sin indices que buscaron indices despues de cada fila.
<code>Select_scan</code>	Numero de uniones ejecutadas que realizaron un examen completo de la primera tabla.
<code>Slave_open_temp_tables</code>	Número de tablas actualmente abiertas procesadas por un esclavo.
<code>Slave_running</code>	ON u OFF. Si se define como ON , el servidor es un esclavo conectado al principal.
<code>Slow_launch_threads</code>	Numero de subprocesos que tardaron mas del valor de <code>show_launch_time</code> en crearse.
<code>Slow_queries</code>	Número de consultas que tardaron mas del valor de <code>long_query_time</code> .
<code>Sort_merge_passes</code>	Numero de combinaciones ejecutadas durante una ordenacion. Si este valor aumenta excesivamente, debe incrementar el <code>sort_buffer</code> .
<code>Sort_range</code>	Numero de ordenaciones realizadas con rangos.
<code>Sort_rows</code>	Numero de filas ordenadas.
<code>Sort_scan</code>	Numero de ordenaciones realizadas al examinar la tabla.
<code>ssl_[variables]</code>	Contenidos de distintas variables utilizadas por SSL. No se implementa en las primeras versiones de MySQL 4.
<code>Table_locks_immediate</code>	Numero de ocasiones en que se realizo inmediatamente un bloqueo de tablas.

Valor	Descripción
Table_locks_waited	Numero de ocasiones en que no se realizo inmediatamente un bloqueo de tablas. Un valor elevado es síntoma de problemas de rendimiento. Para optimizarlo, tendra que mejorar sus consultas e indices, utilizar otro tipo de tablas, dividir sus tablas o recurrir a la duplicacion.
Threads_cached	Numero de subprocesos en la cache de subprocesos.
Threads_connected	Numero de conexiones actualmente abiertas.
Threads_created	Numero de subprocesos creados para procesar conexiones.
Threads_running	Numero de subprocesos activos (no latentes).
Uptime	Tiempo, en segundos, que lleva el servidor en funcionamiento.

Cambio de valores de variables con el servidor en funcionamiento

Hasta la version 4.0.3 de MySQL, era necesario reiniciar el servidor para poder modificar los valores de las variables. En la actualidad, puede utilizar la instruccion SET para realizar cualquier cambio sin tener que apagar el servidor.

Puede utilizar la instruccion SET de dos formas. Con el método predeterminado, el cambio que realice solamente afecta a SESSION lo que significa que, al volver a conectarse (y en conexiones posteriores), la variable se configurara con el parametro especificado en el archivo de configuracion. Si especifica la palabra clave GLOBAL, todas las nuevas conexiones utilizarán este valor. Sin embargo, al reiniciar el servidor, siempre utilizara los valores definidos en el archivo de configuracion, por lo que tambien tendra que realizar los cambios en esta parte. Para definir una variable con la opción GLOBAL, necesita el permiso SUPER, cuya sintaxis es la siguiente:

```
SET [GLOBAL | SESSION] sql_variable=expresión, [[GLOBAL |
SESSION]?
variable_sql=expresión...]
```

El siguiente ejemplo:

```
mysql> SET SESSION max_sort_length=2048;
```

es identico a:

```
mysql> SET max_sort_length=2048;
```

Existe una **sintaxis** alternativa que se utiliza, por motivos de compatibilidad, con otros sistemas de gestión de bases de datos (DBMS) y que **emplea** la construcción @@, como se indica a continuación:

```
SET @@{global local}. Variable_sql=expresión, [@@{global! local}.variable_sql=expresión]
```

Para repetir el ejemplo anterior en esta sintaxis, debe utilizar lo siguiente:

```
mysql> SET @@local.max_sort_length=2048;
```

SESSION y LOCAL son sinónimos.

Si después de experimentar con la nueva variable decide recuperar el valor anterior, no es necesario **buscarlo** en el archivo de configuración.

Puede **utilizar la palabra clave** DEFAULT para restablecer un valor GLOBAL con el valor del archivo de configuración o un valor SESSION por el valor GLOBAL.

Por ejemplo:

```
mysql> SET SESSION max_sort_length=DEFAULT;
```

Y

```
mysql> SET GLOBAL max_sort_length=DEFAULT;
```

En la tabla 13.4 se describen las variables que se configuran de forma no estándar.

Tabla 13.4. Variables no estándar

Sintaxis	Descripción
AUTOCOMMIT= 0 1	Cuando se configura (1), MySQL ejecuta instrucciones COMMIT de forma automática a menos que las envuelva en instrucciones BEGIN y COMMIT. También confirma todas las transacciones abiertas cuando se configura AUTOCOMMIT.
BIG_TABLES = 0 1	Cuando se configura (1), todas las tablas temporales se almacenan en disco en lugar de en memoria. Esto hace que las tablas sean más lentas pero evita que nos quedemos sin memoria. El valor predeterminado es 0.
INSERT_ID = #	Define el valor AUTO_INCREMENT (para que la siguiente instrucción INSERT que utilice un campo AUTO_INSERT emplee este valor).
LAST_INSERT_ID = #	Determina el valor devuelto por la siguiente instrucción LAST_INSERT_ID().

Sintaxis	Descripción
LOW_PRIORITY_ UPDATES = 0 1	Cuando se configura (1), todas las instrucciones de actualizacibn (INSERT, UPDATE, DELETE, LOCK TABLE WRITE) esperan a que no haya lecturas pendientes (SELECT, LOCK TABLE READ) en la tabla a la que quieren acceder.
MAX_JOIN_SIZE = valor DEFAULT	Al configurar un tamaño de filas maximo, puede evitar que MySQL ejecute consultas que no utilicen correctamente los indices o que puedan reducir la velocidad del servidor cuando se ejecutan en periodos de trafico maximo. Al configurar esta variable con cualquier valor que no sea DEFAULT se restablece SQL_BIG_SELECTS. Si se configura SQL_BIG_SELECTS, se ignora MAX_JOIN_SIZE. Si la consulta ya se encuentra en caché, MySQL ignora este límite y devuelve los resultados.
QUERY_CACHE_TYPE = OFF ON DEMAND	Define el parametro de cache de consultas para el subproceso.
QUERY_CACHE_TYPE = 0 1 2	Define el parametro de cache de consultas para el subproceso.
SQL_AUTO_IS_NULL = 0 1	Si se configura (1, el predeterminado), la ultima fila insertada para un AUTO_INCREMENT se puede localizar con WHERE columna_autoincremento IS NULL. MS Access y otros programas ODBC lo utilizan.
SQL_BIG_SELECTS = 0 1	Si se configura (1, valor predeterminado), MySQL admite consultas de gran tamaño. Si no se configura (0), MySQL no admitira consultas en las que tenga que examinar filas de mas de max_join_size. Resulta muy útil para evitar la ejecución de consultas accidentales o maliciosas que pueden colapsar el servidor.
SQL_BUFFER_RESULT = 0 1	Si se configura (1), MySQL añade los resultados de las consultas en una tabla temporal (en algunos casos, aumenta el rendimiento al liberar previamente los bloqueos de tabla).
SQL_LOG_OFF = 0 1	Si se configura (1), MySQL no registra el cliente (no es el registro de actualizacibn). Es necesario el permiso SUPER.
SQL_LOG_UPDATE = 0 1	Si no se configura (0), MySQL no utiliza el registro de actualizaciones en el cliente. Es necesario el permiso SUPER.

Sintaxis	Descripción
SQL_QUOTE_SHOW_CREATE = 0 1	Si se configura (1, valor predeterminado), MySQL añade comillas a los nombres de tablas y columnas .
SQL_SAFE_UPDATES = 0 1	Si se configura (1), MySQL no ejecuta instrucciones UPDATE o DELETE que no utilicen un índice o una cláusula LIMIT , que contribuye a evitar accidentes.
SQL_SELECT_LIMIT = valor DEFAULT	Define el numero maximo de registros (de forma predeterminada es ilimitado) que se pueden devolver con una instrucción SELECT . LIMIT tiene preferencia sobre esto.
TIMESTAMP = valor_marca_de_tiempo DEFAULT	Define el tiempo del cliente . Puede utilizarlo para obtener la marca de tiempo original cuando utilice el registro de actualización para restablecer filas . El valor_marca_de_tiempo es una marca de tiempo de la época de Unix .

Mejoras en el hardware para acelerar el servidor

El hardware es uno de los elementos que puede mejorar (si dispone de los recursos economicos) en un servidor de bases de datos con un rendimiento pobre. Como **regla** general, en primer lugar debe adquirir la mayor **cantidad de memoria** posible, tras **ello** el disco mas rapido que encuentre y, por ultimo, la CPU de mayor velocidad. Lo mas indicado es realizar una prueba comparativa del **rendimiento** del sistema, con ayuda de las distintas herramientas disponibles en el mismo para comprobar si se trata de la CPU, de la **memoria**, de la velocidad del disco o de una **combinación** que **forma** el **cuello de botella**. De esta **forma** podra hacerse una idea de las variaciones que debe aplicar y los elementos que debe actualizar. La ejecucion del paquete de **análisis comparativo** (que describiremos mas adelante), tambien contribuye a mostrar el rendimiento de distintos tipos de tareas.

Memoria

La **memoria** es el elemento mas importante ya que **permite modificar** las variables mysqld. Con una gran **cantidad de memoria** puede crear caches de claves y tablas de gran **tamaño**. La mayor **cantidad de memoria** posible **permite** a MySQL utilizar, de **forma** mas rapida, la **memoria** en lugar de discos y, cuanto mas rapida

sea la **memoria**, mas rapido sera el acceso de MySQL a los datos almacenados. Por si **solas**, las grandes cantidades de **memoria** no son tan utiles como la modificacion de variables `mysqld` para que utilice **memoria** adicional, por lo que no basta con esperar que la **memoria** se encargue de **todo** por nosotros.

Discos

En ultima instancia, MySQL tendra que obtener datos del disco, razon por la que la velocidad de los discos es tan importante. El tiempo de busqueda del disco es importante porque determina la velocidad a la que se **mueve** el disco fisico para obtener los datos que necesita, razon por la que debe seleccionar el disco con mejor velocidad de busqueda.

Al mismo tiempo, los discos SCSI suelen ser mas rapidos que los discos IDE, por lo que son mas aconsejables.

Una de las mejoras que puede realizar es dividir los datos en distintos discos (el sistema operativo divide los datos en partes y los distribuye equitativamente por los distintos discos) asi como utilizar enlaces simbolicos (un enlace desde el directorio de datos hasta otro disco). Las tablas **InnoDB** tienen un mecanismo para dividir datos entre varios discos, **pero** las tablas **MyISAM** no (están **formadas** por **archivos** individuales), por lo que la division u otras **formas** de RAID pueden resultar de gran utilidad. En capitulos posteriores lo describiremos con mayor detalle.

CPU

Cuanto mas rapido sea el procesador, mas rapidos seran los calculos que se realicen y con mayor velocidad se **podrán** devolver los resultados al cliente. Ademas de la velocidad del procesador, tambien son importantes la velocidad del bus y el **tamaño** de la cache. El **análisis** de los procesadores disponibles se escapa a los objetivos de este libro y probablemente cuando se publique, la **información** estara desfasada, por lo que es aconsejable que investigue con atencion las prestaciones de su procesador para comprobar su rendimiento en distintos **análisis comparativos**.

Uso de análisis comparativos

La distribuciones de MySQL incorporan un paquete de **análisis comparativo** denominado `run-all-tests`. Puede utilizarlo para **probar** los distintos DBMS y analizar su rendimiento.

Para utilizarlo, necesita Perl, el modulo **DBI Perl** y el modulo **DBD** del DBMS que quiera **probar**.

En la tabla 13.5 se **recogen** las opciones de `run-all-tests`.

Tabla 13.5. Opciones de run-all-tests

Opción	Descripción
-comments	Añade un comentario al resultado del análisis comparativo.
-cmp=servidor [,servidor...]	Ejecuta la prueba con límites de los servidores especificados. Al ejecutar todos los servidores con el mismo -cmp, los resultados de la prueba se pueden comparar entre los distintos servidores SQL.
-create-options=#	Especifica un argumento adicional en todas las instrucciones CREATE. Por ejemplo, para crear todas las tablas como tablas BDB, utilizaríamos -create-options=TYPE=BDB.
-database	Especifica la base de datos en la que se crean las tablas de prueba. La opción predeterminada es la base de datos de prueba.
-debug	Muestra información de depuración. Normalmente la utilizará al depurar una prueba.
-dir	Indica si se almacenan o no los resultados de la prueba. El valor predeterminado es el resultado predeterminado.
-fast	Permite el uso de comandos SQL ANSI no estándar para que la prueba sea más rápida.
-fast-insert	Utiliza inserciones rápidas siempre que sea posible, lo que incluye listas de valores múltiples, como INSERT INTO nombre_de_tabla VALUES (valores), (valores) o simplemente INSERT INTO nombre_de_tabla VALUES (valores) en lugar de INSERT INTO nombre_de_tabla (campos) VALUES (valores).
-field-count	Especifica la cantidad de campos que tendrá la tabla de prueba. Normalmente se utiliza al depurar una prueba.
-force	Continúa la prueba incluso si se produce un error. Elimina tablas antes de crear otras nuevas. Solamente se utiliza al depurar una prueba.
-groups	Indica la cantidad de grupos diferentes que habrá en la prueba. Solamente se utiliza al depurar.
-lock-tables	Permite el uso de bloqueos de tablas para aumentar la velocidad.
-log	Guarda los resultados en el directorio -dir.

Opción	Descripción
-loop-count (Predeterminado)	Indica cuantas veces se ejecuta el bucle de prueba. Solamente se utiliza al depurar.
-help	Muestra una lista de opciones.
-host='nombre de servidor'	Especifica el equipo en el que se encuentra el servidor de bases de datos. El valor predeterminado es localhost.
-machine="nombre del equipo/sistema operativo"	Nombre del equipo/sistema operativo que se añade al nombre de archivo de los resultados del análisis comparativo. El valor predeterminado es el nombre y la version del sistema operativo.
-odbc	Utiliza el controlador ODBC DBI para conectarse a la base de datos.
-password='contraseña'	Indica la contraseña de usuario con la que se conecta la prueba.
-socket='socket'	Indica el socket al que hay que conectarse (si se admiten sockets).
-regions	Indica cómo se prueban los niveles AND . Solamente se utiliza al depurar una prueba.
-old-headers	Obtiene los encabezados de análisis comparativo antiguos del archivo RUN- antiguo.
-server='nombre del servidor'	Especifica el DBMS en que se realiza la prueba. Puede ser Access, Adabas, AdabasD , Empress, Oracle, Informix, DB2, mSQL , MS-SQL, MySQL , Pg, Solid y Sybase. El predeterminado es SQL.
-silent	No muestra información sobre el servidor cuando se inicia la prueba.
-skip-delete	Indica que no se eliminen las tablas de prueba creadas. Solamente se utiliza al depurar.
-skip-test=prueba1 [,prueba2,...]	Excluye las pruebas especificadas al ejecutar el análisis comparativo.
-small-test	Aumenta la velocidad de las pruebas al utilizar límites mas reducidos.
-small-tables	Utiliza menos filas para realizar las pruebas. Debe utilizar esta opción si la base de datos no puede procesar tablas de gran tamaño por alguna razón (por ejemplo, con pequeñas particiones).
-suffix	Añade un sufijo al nombre de la base de datos en el nombre de archivo de resultado del análisis com-

Opción	Descripción
	parativo. Se utiliza cuando queremos realizar varias pruebas sin reemplazar los resultados . Al utilizar la opción <code>-fast</code> , automáticamente el sufijo es <code>_fast</code> .
<code>-random</code>	Genera valores iniciales aleatorios para la secuencia de ejecución de pruebas , lo que se puede utilizar para imitar condiciones reales.
<code>-threads=#</code>	Indica el número de subprocesos utilizados en análisis comparativos multiusuario. El valor predeterminado es 15.
<code>-tcpip</code>	Utiliza TCP/IP para conectarse al servidor. Esto permite que la prueba realice varias conexiones consecutivas ya que la pila TCP/IP se puede llenar.
<code>-time-limit</code>	Indica un límite de tiempo , en segundos, para el bucle de prueba antes de que esta finalice así como el resultado estimado. El valor predeterminado es de 600 segundos.
<code>-use-old-results</code>	Utiliza los resultados antiguos del directorio <code>-dir</code> en lugar de realizar realmente las pruebas.
<code>-user='nombre_de_usuario'</code>	Especifica el usuario bajo el que conectarse.
<code>-verbose</code>	Muestra información adicional. Solamente se utiliza al depurar la prueba.
<code>-optimization='comentarios'</code>	Añade comentarios sobre optimizaciones realizadas antes de la prueba.
<code>-hw='comentarios'</code>	Añade comentarios sobre el hardware utilizado en esta prueba.

Para ejecutar `run-all-tests`, cambie al directorio `sql-bench` desde el directorio base.

A continuación mostramos un ejemplo del resultado del análisis comparativo:

```
% cd sql-bench
% perl run-all-tests --small-test --password='g00r002b'
Benchmark DBD suite: 2.14
Date of test:      2002-07-21 21:35:42
Running tests on:  Linux 2.2.5-15 i686
Arguments:        --small-test
Comments:
Limits from:
```


Server version: MySQL 4.0.1 alpha max log
 Optimization: None
 Hardware:

ATIS: Total time: 19 wallclock secs (5.23 usr 0.96 sys +
 0.00 cusr
 0.00 csys = 0.00 CPU)
 alter-table: Total time: 2 wallclock secs (0.12 usr 0.03 sys
 + 0.00
 cusr 0.00 csys = 0.00 CPU)
 big-tables: Total time: 1 wallclock secs (0.43 usr 0.10 sys
 + 0.00
 cusr 0.00 csys = 0.00 CPU)
 connect: Total time: 8 wallclock secs (2.90 usr 0.66 sys +
 0.00
 cusr 0.00 csys = 0.00 CPU)
 create: Total time: 0 wallclock secs (0.15 usr 0.01 sys +
 0.00
 cusr 0.00 csys = 0.00 CPU)
 insert: Total time: 31 wallclock secs (8.47 usr 1.43 sys +
 0.00
 cusr 0.00 csys = 0.00 CPU)
 select: Total time: 55 wallclock secs (17.76 usr 1.71 sys +
 0.00
 cusr 0.00 csys = 0.00 CPU)
 transactions: Test skipped because the database doesn't support
 transactions
 wisconsin: Total time: 42 wallclock secs (9.55 usr 1.84 sys +
 0.00
 cusr 0.00 csys = 0.00 CPU)

All 9 test executed successfully

Totals per operation:

Operation	seconds	usr	sys	cpu	tests
alter-table-add	1.00	0.07	0.00	0.00	92
alter-table-drop	0.00	0.03	0.00	0.00	46
connect		0.00	0.22	0.02	0.00
100					
connect+select_1_row	1.00	0.27	0.04	0.00	100
connect+select_simple	1.00	0.27	0.04	0.00	100
count		1.00	0.13	0.00	0.00
100					
count-distinct	1.00	0.13	0.02	0.00	100
count-distinct-2	1.00	0.16	0.02	0.00	100
count-distinct-big	1.00	0.12	0.04	0.00	30
count-distinct-group	1.00	0.17	0.00	0.00	100
count_distinct-group-on-key	1.00	0.13	0.01	0.00	100
count-distinct-group-on-key-parts	1.00	0.16	0.01	0.00	100
count-distinct-key-prefix	1.00	0.12	0.01	0.00	100
count-group-on-key-parts	1.00	0.09	0.00	0.00	100
count-on-key	20.00	6.11	0.60	0.00	5100
create+drop	0.00	0.01	0.00	0.00	10

create_MANY_tables	0.00	0.01	0.00	0.00	10
create-index	1.00	0.00	0.00	0.00	8
create_key+drop	0.00	0.12	0.01	0.00	100
create-table	1.00	0.03	0.00	0.00	31
delete-all-many-keys	1.00	0.08	0.00	0.00	1
delete-big	0.00	0.01	0.00	0.00	1
delete-big-many-keys	1.00	0.07	0.00	0.00	128
delete_key	0.00	0.06	0.01	0.00	100
delete-range	1.00	0.01	0.00	0.00	12
drop-index	0.00	0.00	0.00	0.00	8
drop-table	0.00	0.01	0.00	0.00	28
drop-table-when-MANY-tables	0.00	0.00	0.00	0.00	10
insert		57.00	13.90	2.51	0.00
44768					
insert-duplicates	1.00	0.29	0.04	0.00	1000
insert-key	0.00	0.04	0.01	0.00	100
insert-many-fields	0.00	0.12	0.00	0.00	200
insert_select_1_key	0.00	0.00	0.00	0.00	1
insert_select_2_keys	0.00	0.00	0.00	0.00	1
min_max		1.00	0.06	0.01	0.00
60					
min_max_on_key	17.00	8.12	0.64	0.00	7300
multiple_value_insert	0.00	0.03	0.00	0.00	1000
order-by-big	1.00	0.30	0.10	0.00	10
order-by-big-key	1.00	0.29	0.14	0.00	10
order-by-big-key2	1.00	0.28	0.12	0.00	10
order-by-big-key-desc	0.00	0.35	0.08	0.00	10
order-by-big-key-diff	0.00	0.35	0.05	0.00	10
order-by-big-key-prefix		1.00	0.31	0.09	0.00
10					
order_by_key2_diff	0.00	0.01	0.00	0.00	10
order-by-key-prefix	0.00	0.02	0.00	0.00	10
order-by-range	0.00	0.03	0.00	0.00	10
outer_join	1.00	0.01	0.00	0.00	10
outer_join_found	1.00	0.01	0.01	0.00	10
outer-join-not-found	1.00	0.03	0.01	0.00	10
outer-join-on-key	0.00	0.01	0.00	0.00	10
select-1-row	1.00	0.27	0.06	0.00	1000
select-1-row-cache	1.00	0.18	0.07	0.00	1000
select-2-rows	1.00	0.43	0.05	0.00	1000
select-big	0.00	0.31	0.10	0.00	17
select-big-str	1.00	0.55	0.22	0.00	100
select-cache	4.00	0.95	0.18	0.00	1000
select-cache2	4.00	1.28	0.11	0.00	1000
select_column+column	1.00	0.35	0.06	0.00	1000
select_diff_key	0.00	0.02	0.00	0.00	10
select-distinct	1.00	0.30	0.06	0.00	80
select-group	4.00	0.61	0.09	0.00	391
select_group_when_MANY_tables	0.00	0.00	0.00	0.00	10
select-join	1.00	0.06	0.03	0.00	10
select-key	0.00	0.02	0.01	0.00	20
select_key2	0.00	0.02	0.00	0.00	20
select_key2_return_key	1.00	0.12	0.00	0.00	20

<code>select_key2_return_prim</code>	0.00	0.00	0.00	0.00	20
<code>select-key-prefix</code>	0.00	0.05	0.00	0.00	20
<code>select-key-prefix-join</code>	2.00	0.62	0.16	0.00	10
<code>select-key-return-key</code>	0.00	0.02	0.00	0.00	20
<code>select-many-fields</code>	1.00	0.31	0.10	0.00	200
<code>select-range</code>	2.00	0.23	0.05	0.00	41
<code>select-range-key2</code>	1.00	0.43	0.02	0.00	505
<code>select-range-prefix</code>	1.00	0.42	0.05	0.00	505
<code>select-simple</code>	0.00	0.21	0.04	0.00	1000
<code>select-simple-cache</code>	0.00	0.14	0.05	0.00	1000
<code>select-simple-join</code>	0.00	0.13	0.05	0.00	50
<code>update-big</code>	1.00	0.01	0.00	0.00	10
<code>update-of-key</code>	1.00	0.20	0.03	0.00	500
<code>update-of-key-big</code>	0.00	0.02	0.01	0.00	13
<code>update-of-primary-key-many-keys</code>	0.00	0.12	0.02	0.00	256
<code>update-with-key</code>	4.00	1.03	0.12	0.00	3000
<code>update-with-key-prefix</code>	1.00	0.58	0.01	0.00	1000
<code>wisc_benchmark</code>	2.00	0.70	0.16	0.00	34
TOTALS	156.00	43.84	6.55	0.00	76237

El paquete de análisis comparativo resulta muy útil para comparar distintas plataformas.

MySQL incorpora un conjunto de resultados, pero están datados y no tienen una utilidad especial.

Le sugerimos que repita la prueba personalmente para que se ajuste a su situación en concreto.

También es importante analizar comparativamente sus propias aplicaciones (bajo la mayor carga posible) antes de utilizarlas. Una aplicación que puede ayudarle a añadir carga a su servidor es `super-smack`, que puede descargar en el sitio de MySQL.

Otra secuencia de comandos de gran utilidad distribuida con MySQL es `crash-me`, que verifica la funcionalidad de una aplicación específica y prueba la fiabilidad del servidor bajo carga (véase la tabla 13.6).

Obtiene su nombre de los resultados generados cuando una instalación no supera la prueba.

También es portable y puede probar varias plataformas de bases de datos con propósitos de comparación.

Como resultado de su comportamiento, nunca debe ejecutarse en un entorno activo. No sólo puede colapsar el servidor de bases de datos sino que también consume grandes cantidades de memoria, lo que puede afectar a otros programas que se ejecuten en el servidor.

No obstante debe saber que lo ha desarrollado MySQL, por lo que resalta las ventajas e inconvenientes de MySQL por motivos de comparación. Por ejemplo, los desencadenadores y procedimientos, que MySQL no implementa actualmente, parecen, al ver el resultado de `crash-me`, tan importantes como otras funciones estándar de MySQL, como el uso de `||` por `OR` en lugar de la concatenación de cadenas.

Tabla 13.6. Opciones crash-me

Opción	Descripción
<code>-help, -Information</code>	Muestra una lista de opciones de ayuda.
<code>-batch-mode</code>	Realiza la prueba sin solicitar entradas y se desactiva si encuentra algún error.
<code>-comment='algún comentario'</code>	Añade los comentarios especificados al archivo de límites <code>crash-me</code> .
<code>-check-server</code>	Realiza una nueva conexión al servidor cada vez que comprueba si este sigue en funcionamiento. Puede resultar muy útil si una consulta anterior provoca la devolución de datos erróneos.
<code>-database='base de datos'</code>	Crea las tablas de prueba en esta base de datos. El valor predeterminado es <code>test</code> .
<code>-dir='nombre_de_directorio'</code>	Guarda los resultados en este directorio.
<code>-debug</code>	Muestra gran cantidad de resultados para ayudar en la depuración en caso de que se produzca un problema.
<code>-fix-limit-file</code>	Aplica un nuevo formato al archivo de límites <code>crash-me</code> . No vuelve a ejecutar <code>crash-me</code> .
<code>-force</code>	Inicia la prueba inmediatamente, sin advertirlo y sin esperar ninguna entrada. Puede utilizar esta opción para automatizar la prueba.
<code>-log-all-queries</code>	Muestra todas las consultas ejecutadas. Se utiliza principalmente al depurar <code>crash-me</code> .
<code>-log-queries-to-file='nombre_de_archivo'</code>	Registra consultas completas en el archivo especificado.
<code>-host='nombre_de_anfitrión'</code>	Realiza la prueba en el equipo especificado. El valor predeterminado es <code>localhost</code> .
<code>-password='contraseña'</code>	Especifica la contraseña del usuario actual.
<code>-restart</code>	Guarda los estados durante cada una de las pruebas, lo que permite, en caso de fallo, poder continuar desde el punto en que se detuvo sin necesidad de reiniciar con las mismas opciones.
<code>-server='nombre_de_servidor'</code>	Especifica el servidor en el que se realiza la prueba. Se incluyen Access, Adabas, AdabasD, Empress, Oracle, Informix, DB2, Mimer, mSQL,

Opción	Descripción
	MS-SQL, MySQL, Pg, Solid o Sybase. El predeterminado es MySQL. No se informara de los nombres de otros servidores.
<code>-user='nombre_de_usuario'</code>	Especifica el nombre de usuario con el que se realiza la conexión.
<code>-start-cmd='comando para reiniciar el servidor'</code>	Se utilizará el comando especificado para reiniciar el servidor de bases de datos en caso de que se caiga (la disponibilidad de esta opción lo dice todo).
<code>-sleep='tiempo en segundos'</code>	Especifica el tiempo de espera, en segundos, antes de reiniciar el servidor. El valor predeterminado es de 10 segundos.

A continuación incluimos un ejemplo de `crash-me`:

```
% perl crash-me -password='g00r002b'
Running crash-me 1.57 on 'MySQL 4.0.1 alpha max log'
```

```
I hope you didn't have anything important running on this
server....
Reading old values from cache: /usr/local/mysql-max-4.0.1-
alpha-pc-
linux-gnu-i686/sql-bench/limits/mysql.cfg
```

NOTE: You should be familiar with 'crash-me -help' before continuing!

```
This test should not crash MySQL if it was distributed together
with
the running MySQL version.
If this is the case you can probably continue without having to
worry
about destroying something.
```

```
Some of the tests you are about to execute may require a lot of
memory. Your tests WILL adversely affect system performance.
It's
not uncommon that either this crash-me test program, or the
actual
database back-end, will DIE with an out-of-memory error. So
might
any other program on your system if it requests more memory at
the
wrong time.
```

```
Note also that while crash-me tries to find limits for the
database server
it will make a lot of queries that can't be categorized as
'normal'.
```

It's not unlikely that crash-me finds some limit bug in your server so
if you run this test you have to be prepared that your server may die during it!

We, the creators of this utility, are not responsible in any way if your database server unexpectedly crashes while this program tries to find the limitations of your server. By accepting the following question with 'yes', you agree to the above!

You have been warned!

Start test (yes/no) ?
Tables without primary key: yes
SELECT without FROM: yes
Select constants: yes
Select table-name.*: yes
Allows ' and " as string markers: yes
Double ' as ' in strings: yes
Multiple line strings: yes
" as identifier quote (ANSI SQL): error
' as identifier quote: yes
[] as identifier quote: no
Column alias: yes
Table alias: yes
Functions: yes
Group functions: yes
Group functions with distinct: yes
Group by: yes
Group by position: yes
Group by alias: yes
Group on unused column: yes
Order by: yes
Order by position: yes
Order by function: yes
Order by on unused column: yes
Order by DESC is remembered: no
Compute: no
INSERT with Value lists: yes
INSERT with set syntax: yes
allows end ';': yes
LIMIT number of rows: with LIMIT
SELECT with LIMIT #,#: yes
Alter table add column: yes
Alter table add many columns: yes
Alter table change column: yes
Alter table modify column: yes
Alter table alter column default: yes
Alter table drop column: yes
Alter table rename table: yes

rename table: yes
truncate: yes
Alter table add constraint: yes
Alter table drop constraint: no
Alter table add unique: yes
Alter table drop unique: with drop key
Alter table add primary key: with constraint
Alter table add foreign key: yes
Alter table drop foreign key: with drop foreign key
Alter table drop primary key: drop primary key
Case insensitive compare: yes
Ignore end space in compare: yes
Group on column with null values: yes
Having: yes
Having with group function: yes
Order by alias: yes
Having on alias: yes
binary numbers (0b1001): no
hex numbers (0x41): yes
binary strings (b'0110'): no
hex strings (x'lace'): no
Value of logical operation (1=1): 1
Simultaneous connections (installation default): 101
query size: 1048574

Supported sql types

Type character(1 arg): yes
Type char(1 arg): yes
Type char varying(1 arg): yes
Type character varying(1 arg): yes
Type boolean: no
Type varchar(1 arg) yes
Type integer: yes
Type int: yes
Type smallint: yes
Type numeric(2 arg): yes
Type decimal(2 arg): yes
Type dec(2 arg): yes
Type bit: yes
Type bit(1 arg): yes
Type bit varying(1 arg): no
Type float: yes
Type float(1 arg): yes
Type real: yes
Type double precision: yes
Type date: yes
Type time: yes
Type timestamp: yes
Type interval year: no
Type interval year to month: no
Type interval month: no
Type interval day: no
Type interval day to hour: no

Type *interval day to minute*: no
Type *interval day to second*: no
Type *interval hour*: no
Type *interval hour to minute*: no
Type *interval hour to second*: no
Type *interval minute*: no
Type *interval minute to second*: no
Type *interval second*: no
Type *national character varying(1 arg)*: yes
Type *national character(1 arg)*: yes
Type *nchar(1 arg)*: yes
Type *national char varying(1 arg)*: yes
Type *nchar varying(1 arg)*: yes
Type *national character varying(1 arg)*: yes
Type *timestamp with time zone*: no

Supported odbc types

Type *binary(1 arg)*: yes
Type *varbinary(1 arg)*: yes
Type *tinyint*: yes
Type *bigint*: yes
Type *datetime*: yes

Supported extra types

Type *blob*: yes
Type *byte*: no
Type *long varbinary*: yes
Type *image*: no
Type *text*: yes
Type *text(1 arg)*: no
Type *mediumtext*: yes
Type *long varchar(1 arg)*: no
Type *varchar2(1 arg)*: no
Type *mediumint*: yes
Type *middleint*: yes
Type *int unsigned*: yes
Type *int1*: yes
Type *int2*: yes
Type *int3*: yes
Type *int4*: yes
Type *int8*: yes
Type *uint*: no
Type *money*: no
Type *smallmoney*: no
Type *float4*: yes
Type *float8*: yes
Type *smallfloat*: no
Type *float(2 arg)*: yes
Type *double*: yes
Type *enum(1 arg)*: yes
Type *set(1 arg)*: yes
Type *int(1 arg) zerofill*: yes
Type *serial*: no

Type char(1 arg) binary: yes
Type int not null auto-increment: yes
Type abstime: no
Type year: yes
Type datetime: yes
Type smalldatetime: no
Type timespan: no
Type reftime: no
Type int not null identity: no
Type box: no
Type bool: yes
Type circle: no
Type polygon: no
Type point: no
Type line: no
Type lseg: no
Type path: no
Type interval: no
Type serial: no
Type inet: no
Type cidr: no
Type macaddr: no
Type varchar2(1 arg): no
Type nvarchar2(1 arg): no
Type number(2 arg): no
Type number(1 arg): no
Type number: no
Type long: no
Type raw(1 arg): no
Type long raw: no
Type rowid: no
Type mlslabel: no
Type clob: no
Type nclob: no
Type bfile: no
Remembers end space in char(): no
Remembers end space in varchar(): no
Supports 0000-00-00 dates: yes
Supports 0001-01-01 dates: yes
Supports 9999-12-31 dates: yes
Supports 'infinity dates: error
Type for row id: auto-increment
Automatic row id: rowid

Supported sql functions

Supported odbc functions

Supported extra functions

Supported where functions

Supported sql group functions

Group function AVG: yes
Group function COUNT (*): yes
Group function COUNT column name: yes
Group function COUNT(DISTINCT expr): yes
Group function MAX on numbers: yes
Group function MAX on strings: yes
Group function MIN on numbers: yes
Group function MIN on strings: yes
Group function SUM: yes
Group function ANY: no
Group function EVERY: no
Group function SOME: no

Supported extra group functions

Group function BIT-AND: yes
Group function BIT-OR: yes
Group function COUNT(DISTINCT expr,expr,...) : yes
Group function STD: yes
Group function STDDEV: yes
Group function VARIANCE: no

mixing of integer and float in expression: yes
No need to cast from integer to float: yes
Is 1+NULL = NULL: yes
Is concat('a',NULL) = NULL: yes
LIKE on numbers: yes
column LIKE column: yes
update of column= -column: yes
String functions on date columns: yes
char are space filled: no
DELETE FROM table1,table2...: no
Update with sub select: no
Calculate 1-1: yes
ANSI SQL simple joins: yes
max text or blob size: 1048543 (cache)
constant string size in where: 1048539 (cache)
constant string size in SELECT: 1048565 (cache)
return string size from function: 1047552 (cache)
simple expressions: 1837 (cache)
big expressions: 10 (cache)
stacked expressions: 1837 (cache)
tables in join: 63 (cache)
primary key in create table: yes
unique in create table: yes
unique null in create: yes
default value for column: yes
default value function for column: no
temporary tables: yes
create table from select: yes
index in create table: yes
null in index: yes
null in unique index: yes
null combination in unique index: yes

null in unique index: yes
index on column part (extension): yes
different namespace for index: yes
case independent table names: no
drop table if exists: yes
create table if not exists: yes
inner join: yes
left outer join: yes
natural left outer join: yes
left outer join using: yes
left outer join odbc style: yes
right outer join: yes
full outer join: no
cross join (same as from a,b): yes
natural join: yes
union: no
union all: no
intersect: no
intersect all: no
except: no
except all: no
except: no
except all: no
minus: no
natural join (incompatible lists): yes
union (incompatible lists): no
union all (incompatible lists): no
intersect (incompatible lists): no
intersect all (incompatible lists): no
except (incompatible lists): no
except all (incompatible lists): no
except (incompatible lists): no
except all (incompatible lists): no
minus (incompatible lists): no
subqueries: no
insert INTO ... SELECT ...: yes
atomic updates: no
views: no
foreign key syntax: yes
foreign keys: no
Create SCHEMA: no
Column constraints: no
Table constraints: no
Named constraints: no
NULL constraint (SyBase style): yes
Triggers (ANSI SQL): no
PSM procedures (ANSI SQL): no
PSM modules (ANSI SQL): no
PSM functions (ANSI SQL): no
Domains (ANSI SQL): no
many tables to drop table: yes
drop table with cascade/restrict: yes
- as comment (ANSI): yes

```

// as comment: no
# as comment: yes
/* */ as comment: yes
insert empty string: yes
Having with alias: yes
table name length: 64 (cache)
column name length: 64 (cache)
select alias name length: +512 (cache)
table alias name length: +512 (cache)
index name length: 64 (cache)
max char() size: 255 (cache)
max varchar() size: 255 (cache)
max text or blob size: 1048543 (cache)
Columns in table: 3398 (cache)
unique indexes: 32 (cache)
index parts: 16 (cache)
max index part length: 255 (cache)
index varchar part length: 255 (cache)
indexes: 32
index length: 500 (cache)
max table row length (without blobs): 65534 (cache)
table row length with nulls (without blobs): 65502 (cache)
number of columns in order by: +64 (cache)
number of columns in group by: +64 (cache)
crash-me safe: yes
reconnected 0 times

```

Ejecucion de MySQL en modo ANSI

La ejecución de MySQL en modo ANSI hace que se comporte de forma mas estandar de lo habitual, lo que facilita la tarea de cambiar a otra base de datos en una fase posterior.

Si se inicia MySQL con la opción `-ansi`, se aplicaran las siguientes diferencias al comportamiento de MySQL:

- El simbolo `| |` no significa OR; por el contrario, se aplica a concatenaciones de cadenas. Es la opción `sql-mode mysql_d PIPES_AS_CONCAT`.
- La presencia de espacios delante de nombres de funciones no genera errores y, en consecuencia, todos los nombres de funciones se convierten en palabras reservadas. Es la opción `sql-mode mysql_d IGNORE_SPACE`.
- REAL es sinónimo de FLOAT no de DOUBLE. Es la opción `sql-mode mysql_d REAL_AS_FLOAT`.
- El nivel de aislamiento de transacciones predeterminado se configura como SERIALIZABLE. Es la opción `sql-mode mysql_d SERIALIZE`.
- El caracter de comillas dobles (") sera un identificador, no un caracter de cadenas. Es la opción `sql-mode mysql_d ANSI_QUOTES`.

Uso de distintos lenguajes en MySQL

Es evidente que los datos que aiiada a MySQL pueden estar en cualquier idioma pero hay muchos usuarios en el mundo que no hablan ingles y que utilizan MySQL. MySQL AB, la empresa actualmente responsable de MySQL esta ubicada en Suecia y la mayoría de los principales programadores son escandinavos. Por esta razon, no debe sorprenderle que las distribuciones de MySQL sean compatibles con otros idiomas. A continuación le mostramos la lista de los idiomas admitidos, que probablemente se amplíe en el futuro: checo, danes, holandes, ingles (predeterminado), estonio, francés, alemán, griego, hungaro, italiano, coreano, noruego, polaco, portugues, rumano, ruso, esloveno, espiail y sueco.

Como mostrar mensajes de error en otro idioma

El inicio de MySQL para que muestre mensajes de error en uno de estos idiomas es tan sencillo como utilizar la opción `-language` o `-L`. Para hacerlo en el archivo de configuración, basta con aiiadir la siguiente linea:

```
language=french
```

Tambien puede editar personalmente los mensajes de error (para que su base de datos tenga un toque mas personal) o contribuir con su propio conjunto de mensajes en otro idioma para compartirlo con la comunidad de MySQL. Para cambiar los mensajes de error, basta con editar el archivo `errmsg.txt` en el correspondiente directorio de idioma (normalmente `/share/nombre_del_idioma` en el directorio base de MySQL), ejecutar la utilidad `cmp_error` y reiniciar el servidor. Por ejemplo:

```
% cp errmsg.txt errmsg.bak
% vi errmsg.txt
```

En este caso hemos modificado el siguiente mensaje de error:

```
"No Database Selected",
```

para convertirlo en:

```
"Haven't you forgotten something - No Database Selected",
```

y tras ello lo guardamos como:

```
"errmsg.txt" 229 lines, 12060 characters written
% comp_err errmsg.txt errmsg.sys
Found 226 messages in language file errmsg.sys
```

Seguidamente reiniciamos el servidor para aplicar los nuevos mensajes de error:

```
% mysqladmin shutdown
% /etc/rc.d/init.d/mysql start
% mysql -uroot -pg00r002b
mysql> SELECT • FROM a;
```

ERROR 1046: Haven't you forgotten something - No Database Selected

Tendra que repetir los cambios si actualiza a una nueva version de MySQL.

Utilización de un conjunto de caracteres diferente

De forma predeterminada, MySQL utiliza el conjunto de caracteres Latin1 (ISO-8859-1). El conjunto de caracteres determina que caracteres se van a utilizar, asi como la ordenacion de las consultas. Para cambiar el conjunto de caracteres debe modificar el valor de la opcion `-default-character-set` al iniciar el servidor. Entre los conjuntos de caracteres actualmente disponibles se encuentran los siguientes:

latin1	dos	estonia
big5	german1	hungarian
czech	hp8	koi8_ukr
euc_kr	koi8_ru	win1251ukr
gb2312	latin2	greek
gbk	swe7	win1250
latin1_de	usa7	croat
sjis	cp1251	cp1257
tis620	danish	latin5
ujis	hebrew	
dec8	win1251	

Puede comprobar que conjuntos de caracteres estan disponibles en su distribucion si consulta el valor de la variable `character_set`.

Al cambiar un conjunto de caracteres, tendrá que volver a generar sus indices para garantizar que se ordenan en funcion de las reglas del nuevo conjunto de caracteres.

De forma predeterminada, MySQL se compila con `-with-extra-charsets=complex`, para que el resto de conjuntos de caracteres esten disponibles en caso de que los necesite. Si tiene pensado compilar personalmente MySQL y sabe que nunca va a necesitar otro conjunto de caracteres, puede utilizar la opcion `-with-extra-charsets=none`.

Como añadir un conjunto de caracteres propio

Tambien puede añadir su propio conjunto de caracteres. Si se trata de un conjunto de caracteres simple y no necesita compatibilidad de caracteres multibyte

o rutinas de comparación de cadenas para ordenar, la operación resulta sencilla. Resulta más compleja si requiere estas opciones adicionales. Para añadir el conjunto de caracteres, siga los pasos descritos a continuación:

1. **Añada** el nuevo conjunto de caracteres al archivo `sql/share/charsets/Index` y asignele un Id. **exclusivo**. La ruta puede diferir en algunas distribuciones **pero** siempre será el archivo `Index`. En este ejemplo, denominamos **martian** al nuevo conjunto de caracteres y le asignamos el Id. **31**:

```
# sql/share/charsets/Index
#
# Este archivo enumera todos los conjuntos de caracteres
# disponibles.

big5                1
czech               2
dec8                3
dos                 4
german1             5
hp8                 6
koi8_ru             7
latin1              8
latin2              9
swe7                10
usa7                11
ujis                12
sjis                13
cp1251              14
danish              15
hebrew              16
# El conjunto de caracteres win1251 se ha quedado obsoleto. En
# su lugar utilice cp1251.
win1251             17
tis620              18
euc_kr              19
estonia             20
hungarian           21
koi8_ukr           22
win1251ukr         23
gb2312              24
greek               25
win1250             26
croat               27
gbk                 28
cp1257              29
latin5              30
martian             31
```

2. Cree el archivo `.conf` y añádalo al directorio, por ejemplo, `sql/share/charsets/martian.conf`. Utilice uno de los archivos `.conf` existentes como punto de partida.

En el archivo `.conf`, las líneas que empiezan con el símbolo `#` son comentarios, las palabras se separan con espacios en blanco y todas las palabras deben estar en formato hexadecimal. Hay cuatro matrices que, en orden, son `ctype` (contiene 257 elementos), `to_lower` y `to_upper` (cada una con 256 elementos), y `sort_order` (también con 256 elementos). A continuación le mostramos un ejemplo de archivo `.conf` (se trata del archivo `latin1.conf` estándar):

```
# Archivo de configuración del conjunto de caracteres latin1

# Matriz ctype (debe tener 257 elementos)
00
20 20 20 20 20 20 20 20 20 28 28 28 28 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
48 10 10 10 10 10 10 10 10 10 10 10 10 10 10
84 84 84 84 84 84 84 84 84 84 10 10 10 10 10
10 81 81 81 81 81 81 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 01 10 10 10 10
10 82 82 82 82 82 82 02 02 02 02 02 02 02 02
02 02 02 02 02 02 02 02 02 02 02 10 10 10 20
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
48 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 10 01 01 01 01 01 01 02
02 02 02 02 02 02 02 02 02 02 02 02 02 02 02
02 02 02 02 02 02 02 10 02 02 02 02 02 02 02

# Matriz to-lower (debe tener 256 elementos)
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
70 71 72 73 74 75 76 77 78 79 7A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
F0 F1 F2 F3 F4 F5 F6 D7 F8 F9 FA FB FC FD FE DF
EO E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF

# Matriz to-upper (debe tener 256 elementos)
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
```


30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	F7	D8	D9	DA	DB	DC	DD	DE	FF

Matriz `sort_order` (debe tener 256 elementos)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
41	41	41	41	5C	5B	5C	43	45	45	45	45	49	49	49	49
44	4E	4F	4F	4F	4F	5D	D7	D8	55	55	55	59	59	DE	DF
41	41	41	41	5C	5B	5C	43	45	45	45	45	49	49	49	49
44	4E	4F	4F	4F	4F	5D	F7	D8	55	55	55	59	59	DE	FF

La matriz `c_type` contiene valores de bits, con un elemento para cada caracter. Las matrices `to_lower` y `to_upper` simplemente almacenan los caracteres en minúsculas y mayúsculas correspondientes a cada uno de los miembros del conjunto de caracteres.

Por ejemplo, `to_lower['A']` contiene `a`, mientras que `to_upper['z']` contiene `Z`.

La matriz `sort_order` indica la forma de ordenar los caracteres (normalmente se corresponde con `to_upper`, en cuyo caso el orden no distinguirá entre mayúsculas y minúsculas). Todas las matrices se indexan por el valor de los caracteres a excepción de `c_type` que se indexa por el valor del caracter + 1 (un viejo legado).

3. Añada el nuevo conjunto de caracteres (`martian.conf`) a las listas `CHARSETS_AVAILABLE` y `COMPILED_CHARSETS` del archivo `configure.in`.
4. Vuelva a configurar y compilar MySQL, y pruebe el nuevo conjunto de caracteres.

Si se siente con animos de emprender la inclusion de un nuevo **conjunto** de caracteres complejo, este proceso requiere algunos pasos adicionales. Consulte la documentacion de MySQL para ver lo que necesita (asi como la documentacion de los conjuntos de caracteres complejos ya existentes: czech, gbk, sjis y tis160).

Resumen

Para aprender a obtener el maximo rendimiento de nuestro servidor de bases de datos, es muy importante comprender las distintas opciones disponibles para retocarlo. Para ver como se ha configurado un servidor existente, debe utilizar la **instrucción** `SHOW VARIABLES` así como `SHOW STATUS` para ver como se ha procesado. El resultado de estas dos instrucciones puede **revelar** muchos **problemas** ocultos, incluyendo consultas no optimizadas, una pobre **utilización** de la **memoria** disponible o la necesidad de una **actualización**.

MySQL cuenta con cuatro archivos de configuracion que pueden ayudarle a obtener un mejor rendimiento del servidor. Solamente tiene que escoger, entre `my-huge.cnf`, `my-large.cnf`, `my-medium.cnf` o `my-small.cnf`, el que mas se aproxime a las necesidades de su servidor.

Dos de las variables mas sencillas de modificar, y de las mas importantes, son `table_cache` (el numero de tablas que MySQL puede tener abiertas) y `key_buffer_size` (**número** de indices que puede tener en **memoria**, **minimizando** el **acceso a disco**).

Las bases de datos InnoDB **tienen** sus **propias** peculiaridades y funcionan de **forma** diferente a las tablas MyISAM, en las que **cada** tabla esta relacionada con archivos especificos. La configuracion de InnoDB requiere una cuidadosa **plani-**ficacion ya que el espacio de disco se asigna de **forma** anticipada.

El hardware tambien constituye una **forma** sencilla de mejorar el rendimiento de un servidor, principalmente la **memoria**, la CPU o los discos.

MySQL incluye un paquete de **análisis comparativo** que se puede utilizar para contrastar el rendimiento de distintas plataformas, incluso de otras bases de **da-**tos.

MySQL se ha desarrollado en Escandinavia e incluye compatibilidad con otros **idiomas** además del ingles. Resulta muy sencillo mostrar mensajes de error en otros **idiomas** o aiiadir un **conjunto** de caracteres.

14

Seguridad de bases de datos

La seguridad no es un **elemento** adicional opcional. El control de fugas de seguridad una vez establecidos todos los elementos resulta mucho mas **complicado** que la **protección correcta** de los datos desde el **principio**. Y, como **administrador** de bases de datos (DBA), debe confiar en sus usuarios, **pero** si borran accidentalmente una tabla cuya presencia desconocian, sera el administrador el que reciba las culpas. La mayor **parte** de este capitulo se centra en las **formas** de gestionar los usuarios y en garantizar que solamente realizan las acciones **necesarias**. En este capitulo veremos los siguientes aspectos:

- Seguridad **al** conectarse
- **Modificación** y asignacion de contraseñas
- Gestion de usuarios y permisos
- Tablas de permisos **MySQL**
- GRANT y REVOKE
- Privilegios peligrosos
- Seguridad de aplicaciones y del sistema
- Aspectos de seguridad relacionados con `LOAD DATA LOCAL`

Seguridad al conectarse

Cuando nos conectamos, resulta poco seguro hacerlo la siguiente **forma**:

```
% mysql -username -ppassword
```

A lo largo del capítulo utilizaremos este formato por motivos de conveniencia, para que la contraseña sea visible en los ejemplos. Sin embargo, un usuario preocupado por la seguridad no debería conectarse de esta **forma** por **las** siguientes razones:

- Cualquiera puede **observar** por encima de nuestro hombro y ver la contraseña plasmada en texto.
- La contraseña puede ser visible en el historial (por ejemplo, en Unix, cualquiera que tenga acceso a la terminal de otro usuario puede desplazarse por los comandos más recientes y ver la contraseña).
- Programas que ven el estado del sistema (como ps de Unix) pueden ver la contraseña plasmada en texto.

Por el contrario, debe conectarse e introducir la contraseña cuando así se lo soliciten:

```
% mysql -uroot -p  
Enter password:
```

Si necesita almacenar la contraseña en un archivo, asegúrese de que la protege correctamente.

Por ejemplo, si la contraseña se almacena en el archivo `~/.cnf` en el directorio principal del usuario en un servidor, este archivo no puede ser leído por nadie más.

Evidentemente, el usuario raíz del sistema puede leer este archivo. Asegúrese de que el usuario raíz del sistema no es necesariamente el usuario raíz de **MySQL**. Del mismo modo, las aplicaciones suelen utilizar un archivo de **configuración** para almacenar la contraseña de la base de datos. No olvide proteger también este archivo.

ADVERTENCIA: Nunca debe almacenar un archivo de configuración que contiene una contraseña de base de datos para una aplicación Web, a cualquier otra contraseña, en el árbol Web.

Por último, no utilice la variable de entorno `MYSQL_PWD` para almacenar su contraseña. Tampoco debe especificarla en la línea de comandos. Las variables de entorno no son seguras.

Gestion de usuarios y permisos

MySQL dispone de un sistema de permisos bien diseiinado, flexible y facil de gestionar. Los permisos **permiten** o **prohiben** que determinados usuarios o **equi-**pos anfitrión se conecten **al** servidor de bases de datos y que realicen **determina-**das operaciones en las bases de datos, tablas o incluso en columnas especificas de las tablas.

Veamos algunos posibles **casos**:

- Un sitio Web de noticias incluye un servidor de bases de datos, un servidor Web y una Intranet en la que **los** empleados actualizan las noticias. Las conexiones desde el servidor Web solamente **tendrán permiso** para ejecutar consultas SELECT en la base de datos y las conexiones desde la Intranet permitiran **consultas** UPDATE e INSERT por **parte** de los empleados.
- Un sistema de transacciones financieras tiene una base de datos que **contiene** un registro de registros y una base de datos con balances de clientes. Se **permiten** actualizaciones en la base de datos de balances de clientes **pero** no en la de registros.
- Un sistema de reservas tiene usuarios convencionales que solamente **pueden** añadir registros en una determinada tabla y un administrador que **puede** actualizar dicha tabla.

La base de datos mysql

Cuando se **instala** MySQL, la base de datos `mysql` es una de las que se crea automaticamente.

Un **profundo** conocimiento de las tablas de esta base de datos resulta fundamental para poder administrar de **forma** eficaz la seguridad en el sistema (tabla 14.1). Hay seis tablas en la base de datos `mysql` que **afectan** el acceso al sistema:

```
mysql> USE mysql;
Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_mysql |
+-----+
| columns_priv    |
| db              |
| un.             |
| host            |
| tables_priv     |
| user            |
+-----+
6 rows in set (0.00 sec)
```

Tabla 14.1. Las tablas MySQL

Tabla	Descripción
user	Enumera los usuarios y los equipos y contraseñas asociadas que pueden acceder al servidor, así como los permisos globales que tienen. Es aconsejable desactivar todos los permisos globales y, en su lugar, permitir el acceso individual en una de las otras tablas.
db	Enumera las bases de datos a las que pueden acceder los usuarios. Los permisos otorgados se aplican a todas las tablas de la base de datos.
Host	Junto con la tabla db, permite un acceso más controlado en función de un determinado equipo.
Tables_priv	Enumera el acceso a tablas concretas. Los permisos otorgados se aplican a todas las columnas de la tabla.
Columns_priv	Enumera el acceso a tablas específicas.
un.	Todavía no se utiliza.

Campos de las tablas

A continuación veremos las tablas de la base de datos `mysql`. Las tablas de su distribución pueden ser ligeramente distintas.

```
mysql> SHOW COLUMNS FROM user;
```

Field	Type	Null	Key	Default	Extra
Host	varchar(60) binary		PRI		
User	varchar(16) binary		PRI		
Password	varchar(16) binary				
Select_priv	enum('N','Y')			N	
Insert_priv	enum('N','Y')			N	
Update_priv	enum('N','Y')			N	
Delete_priv	enum('N','Y')			N	
Create_priv	enum('N','Y')			N	
Drop_priv	enum('N','Y')			N	
Reload_priv	enum('N','Y')			N	
Shutdown_priv	enum('N','Y')			N	
Process_priv	enum('N','Y')			N	
File_priv	enum('N','Y')			N	
Grant_priv	enum('N','Y')			N	

Field	Type	Null	Key	Default	Extra
References_priv	enum('N','Y')			N	
Index_priv	enum('N','Y')			N	
Alter_priv	enum('N','Y')			N	
Show_db_priv	enum('N','Y')			N	
Super_priv	enum('N','Y')			N	
Create_tmp_table_priv	enum('N','Y')			N	
Lock_tables_priv	enum('N','Y')			N	
Execute_priv	enum('N','Y')			N	
Repl_slave_priv	enum('N','Y')			N	
Repl_client_priv	enum('N','Y')			N	
ssl_type	enum('','ANY', 'X509', 'SPECIFIED')				
ssl_cipher	blob				
x509-issuer	blob				
x509-subject	blob				
max_questions	int(11) unsigned				
max_updates	int(11) unsigned				
max_connections	int(11) unsigned				

31 rows in set (0.00 sec)

mysql> SHOW COLUMNS FROM db;

Field	Type	Null	Key	Default	Extra
Host	char(60) binary		PRI		
Db	char(64) binary		PRI		
User	char(16) binary		PRI		
Select_priv	enum('N','Y')			N	
Insert_priv	enum('N','Y')			N	
Update_priv	enum('N','Y')			N	
Delete_priv	enum('N','Y')			N	
Create_priv	enum('N','Y')			N	
Drop_priv	enum('N','Y')			N	
Grant_priv	enum('N','Y')			N	
References_priv	enum('N','Y')			N	
Index_priv	enum('N','Y')			N	
Alter_priv	enum('N','Y')			N	

13 rows in set (0.01 sec)

mysql> SHOW COLUMNS FROM host;

Field	Type	Null	Key	Default	Extra
Host	char(60) binary		PRI		
Db	char(64) binary		PRI		
Select_priv	enum('N','Y')			N	
Insert_priv	enum('N','Y')			N	

Field	Type	Null	Key	Default	Extra
Update_priv	enum('N','Y')			N	
Delete_priv	enum('N','Y')			N	
Create_priv	enum('N','Y')			N	
Drop_priv	enum('N','Y')			N	
Grant_priv	enum('N','Y')			N	
References_priv	enum('N','Y')			N	
Index_priv	enum('N','Y')			N	
Alter_priv	enum('N','Y')			N	

12 rows in set (0.01 sec)

mysql> SHOW COLUMNS FROM tables_priv;

Field	Type	Null	Key	Default	Extra
Host	char(60) binary		PRI		
Db	char(64) binary		PRI		
User	char(16) binary		PRI		
Table_name	char(60) binary		PRI		
Grantor	char(77)		MUL		
Timestamp	timestamp(14)	YES		NULL	
Table_priv	set('Select','Insert', 'Update','Delete', 'Create','Drop', 'Grant','References', 'Index','Alter')				
Column_priv	set('Select','Insert', 'Update','References')				

8 rows in set (0.01 sec)

mysql> SHOW COLUMNS FROM columns_priv;

Field	Type	Null	Key	Default	Extra
Host	char(60) binary		PRI		
Db	char(64) binary		PRI		
User	char(16) binary		PRI		
Table_name	char(64) binary		PRI		
Column_name	char(64) binary		PRI		
Timestamp	timestamp(14)	YES		NULL	
Column_priv	set('Select','Insert', 'Update','References')				

7 rows in set (0.01 sec)

mysql> SHOW COLUMNS FROM func;

Field	Type	Null	Key	Default	Extra
name	char(64) binary		PRI		
ret	tinyint(1)			0	
dl	char(128)				
type	enum('function',			function	


```

+-----+-----+-----+-----+
      'aggregate ')
4 rows in set (0.01 sec)

```

En la tabla 14.2 se describen los distintos privilegios.

Tabla 14.2. Significado de las columnas

Columna	Descripción
Host	Equipo anfitrión desde el que se conecta el usuario.
User	Nombre de usuario proporcionado para la conexión (la opción -u).
Password	Contraseña con la que se conecta el usuario (la opción -p).
Db	Base de datos a la que el usuario intenta realizar la operación .
Select_priv	Permiso para devolver datos desde una tabla (una instrucción SELECT). Los resultados SELECT que se pueden calcular sin necesidad de acceder a una tabla devuelven un resultado aunque el usuario no tenga privilegios SELECT .
Insert_priv	Permiso para añadir nuevos registros a la tabla (una instrucción INSERT).
Update_priv	Permiso para modificar datos en una tabla (una instrucción UPDATE).
Delete_priv	Permiso para eliminar registros de una tabla (una instrucción DELETE).
Create_priv	Permiso para crear bases de datos y tablas.
Drop_priv	Permiso para borrar bases de datos o tablas.
Reload_priv	Permiso para volver a cargar la base de datos (una instrucción FLUSH o una recarga, actualización o eliminación emitida desde mysqladmin).
Shutdown_priv	Permiso para cerrar el servidor.
Process_priv	Permiso para ver los procesos MySQL actuales o para eliminar procesos MySQL (para instrucciones SHOW PROCESSLIST o KILL SQL).
File_priv	Permiso para leer y escribir archivos en el servidor (para instrucciones LOAD DATA FILE o SELECT INTO OUTFILE). Todos los archivos que el usuario MySQL puede leer son legibles.

Columna	Descripción
Grant_priv	Permiso para conceder privilegios disponibles en el usuario a otros usuarios.
References_priv	Actualmente no se utiliza en MySQL.
Index_priv	Permiso para crear, modificar o eliminar índices.
Alter_priv	Permiso para cambiar la estructura de una tabla (una instrucción ALTER).
Show_db_priv	Permiso para ver todas las bases de datos.
Super_priv	Permiso para conectarse, incluso si se alcanza el máximo de conexiones, y ejecutar los comandos CHANGE MASTER, KILL en subprocesos, depuración mysqladmin, PURGE MASTER LOGS y SET GLOBAL.
Create_tmp_table_priv	Permiso para crear una tabla temporal (CREATE TEMPORARY TABLE).
Lock_tables_priv	Permiso para bloquear una tabla para la que el usuario tiene permiso SELECT.
Execute_priv	Permiso para ejecutar procedimientos almacenados (previsto para MySQL 5).
Repl_client_priv	Permiso para solicitar la duplicación de esclavos y principales.
Repl_slave_priv	Permiso para duplicar (consultar un capítulo anterior).
ssl_type	Solamente se concede el permiso de conexión si se utiliza SSL.
ssl_cipher	Solamente se concede el permiso de conexión si hay un cifrado específico.
x509_issuer	Solamente se concede el permiso de conexión si el certificado se emite por un emisor específico.
x509_subject	Solamente se concede el permiso de conexión si el certificado contiene un asunto concreto.
max_questions	Número máximo de consultas por hora que puede ejecutar el usuario.
max_updates	Número máximo de actualizaciones por hora que puede ejecutar el usuario.
max_connections	Máximo de conexiones por hora que puede realizar el usuario.

Como examina MySQL permisos para conceder el acceso

Cuando un usuario intenta conectarse, MySQL examina en primer lugar la tabla de usuarios para confirmar que se enumera el usuario **concreto**, el anfitrión y la contraseña. En caso contrario, se le niega el acceso **al** usuario. Cuando el usuario intenta conectarse directamente a una base de datos, la tabla db se **examinara** si el usuario pasa las comprobaciones restantes. Si el usuario no tiene **permiso** para conectarse a la base de datos, el acceso no se concede.

Cuando un usuario conectado intenta realizar una operación administrativa (por **ejemplo**, `mysqladmin shutdown`), MySQL examina la **columna** relacionada con la operación de la tabla del usuario. Si se concede el **permiso** para la operación solicitada, la operación sigue adelante. En caso contrario, se cancela.

Cuando un usuario conectado intenta realizar una operación relacionada con la base de datos (SELECT, UPDATE, etc.), MySQL examina el campo **relacionado** de la tabla del usuario. Si se concede el **permiso** para la operación solicitada (SELECT, UPDATE, etc.), la operación se **permite**. En caso contrario, MySQL pasa **al siguiente paso**. **A continuación**, se examina la tabla db. MySQL busca la base de datos en la que el usuario realiza la operación. Si no existe, se deniega el **permiso** y se cancela la operación. Si la base de datos existe y coinciden el **anfitrión** y el usuario, se examina el campo relacionado con la operación. Si se concede el **permiso** para la operación solicitada, la operación será satisfactoria. Si el **permiso** no se concede, MySQL prosigue con el siguiente **paso**. Si existe la **combinación** base de datos y usuario, y el campo anfitrión está en blanco, MySQL examina la tabla para ver si el anfitrión puede realizar la operación solicitada. Si el anfitrión y la base de datos se encuentran en la tabla del anfitrión, el campo relacionado **tanto** en la tabla del anfitrión **como** en la tabla db determina el éxito de la operación. Si se concede el **permiso** en **ambas** tablas, la operación continúa. En caso contrario, MySQL pasa **al siguiente paso**.

MySQL examina la tabla `tables_priv` y toma en **consideración** la tabla en la que se realiza la operación. Si no existe la **combinación** anfitrión, db, usuario y tabla, la operación se cancela. Si existe, se examina el campo relacionado. Si el **permiso** no se concede, MySQL pasa **al siguiente paso**. Si se concede el **permiso**, la operación sigue adelante.

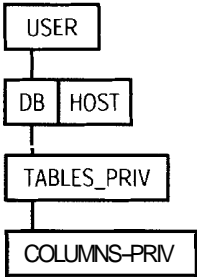


Figura 14.1. Precedencia de las tablas de permiso MySQL

Por ultimo, MySQL examina las tablas `columns_priv` y utiliza las `columns` de la tabla utilizada en la operacion. Si se concede el `permiso` relacionado con la operacion solicitada, la operacion sigue adelante. En caso contrario, la operacion se **cancela**.

El orden de precedencia de las tablas de `permiso` MySQL se muestra en la figura 14.1.

Como completar las tablas de permiso

Las tablas de `permiso` se completan con determinados valores predeterminados:

```
mysql> SELECT * FROM user;
```

Host	User	Password	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv	Drop_priv	Reload_priv	Shutdown_priv	Process_priv	File_priv	Grant_priv	References_priv	Index_priv	Alter_priv	Show_db_priv	Super_priv	Create_tmp_table_priv	Lock_tables_priv	Execute_priv	Repl_slave_priv	Repl_client_priv	ssl_type	ssl_cipher	x509_issuer	x509_subject	max_questions	max_updates	max_connections
localhost	root		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
test.testhost.co.za	root		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

```

| Y          | Y          | Y          | Y          | Y
| Y
| Y          | Y          | Y          | Y          | Y
| Y          | Y          | Y          | Y          | Y
| Y          | Y          | Y          | Y          | Y
|          |          |          |          |
0 |
0 |          0 |
| localhost          |          |
| N
| N          | N          | N          | N          | N
| N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
|          |          |          |          |
0 |
0 |          0 |
| test.testhost.co.za |          |
| N
| N          | N          | N          | N          | N
| N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
|          |          |          |          |
0 |          0 |          0 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
4 rows in set (0.05 sec)
mysql> SELECT * FROM db;
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
| Host | Db      | User | Select-priv | Insert_priv |
Update_priv |
Delete_priv | Create_priv | Drop_priv | Grant_priv |
References_priv |
Index_priv | Alter_priv |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
| %    | test   |      | Y           | Y           | Y
| Y
| Y          | Y          | N          | Y           | Y           | Y
|

```

```

Y          |
| %      | test\__% |      | Y          | Y          | Y
| Y
| Y          | Y          | N          | Y          | Y
|
Y          |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+
2 rows in set (0.01 sec)
mysql> SELECT * FROM host;
Empty set (0.00 sec)
mysql> SELECT • FROM tablespriv;
Empty set (0.00 sec)
mysql> SELECT * FROM columns-priv;
Empty set (0.00 sec)

```

Apreciara que los parametros predeterminados no son seguros. Cualquiera puede conectarse desde el servidor local como usuario raiz y disponer de control absoluto. Un usuario anonimo (el que no proporciona ningun nombre de usuario) puede conectarse desde el servidor local a la base de datos de prueba predeterminada y a cualquier base de datos cuyo nombre comience por `test`.

NOTA: En Unix, MySQL utiliza el nombre de usuario de conexión Unix en caso de no indicar ningún nombre de usuario. Esto significa que cualquiera que se conecte como raiz puede acceder a MySQL sin necesidad de especificar un nombre de usuario y dispondrá de todos los permisos.

Una de las primeras tareas que se deben realizar en una nueva instalacion es definir nuevos permisos y, al menos, una nueva contrasefia para el usuario raiz.

Manipulación directa de las tablas de permisos

Hay dos formas de definir permisos: por medio de las instrucciones GRANT y REVOKE de MySQL o cambiando directamente los valores en las tablas. La forma mas sencilla y mas aconsejable consiste en utilizar las instrucciones GRANT y REVOKE, pero es importante que entienda como afectan las tablas a los permisos. Por el momento, veremos la forma de cambiar permisos mediante la modificacion de los valores de las tablas con las instrucciones SQL básicas INSERT, UPDATE y DELETE. Veremos el otro método en un apartado posterior. Para añadir una contraseiia para el usuario raiz, debe escribir lo siguiente:

```

mysql> UPDATE user SET password=PASSWORD('g00r002b') WHERE
user='root';
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2 Changed: 2 Warnings: 0

```

Fíjese en el uso de la funcion PASSWORD(). Es necesario utilizarla al actualizar directamente las tablas.

Codifica la contraseiia para que no se pueda leer con tan sólo mirar a los contenidos de las tablas. Por ejemplo:

```
mysql> SELECT host,user,password FROM user;
+-----+-----+-----+
| host          | user | password          |
+-----+-----+-----+
| localhost    | root | 43b591f759a842a9 |
| test.testhost.co.za | root | 43b591f759a842a9 |
| localhost    |     |                   |
| test.testhost.co.za |     |                   |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

ADVERTENCIA: Debe prestar especial atención a la hora de modificar directamente los permisos. Al descuidar la cláusula WHERE, todas las contraseñas cambian y, de forma instantánea, ningún usuario podrá conectarse

Los cambios realizados en los permisos no tiene un efecto inmediato cuando se realizan directamente en las tablas MySQL. MySQL tiene que volver a leer las tablas concedidas. Para ello, puede emitir FLUSH PRIVILEGES, mysqladmin flush-privileges o mysqladmin reload.

```
mysql> INSERT INTO user (Host,User,Password) VALUES
('localhost',
 'administrator', PASSWORD('admin_pwd'));
Query OK, 1 row affected (0.09 sec)
```

Antes de que se eliminen los permisos, estos datos no son efectivos. Puede concctarse como usuario administrador sin necesidad de contraseiia:

```
% mysql -uadministrator;
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6 to server version: 4.0.1-alpha-
max-log
```

Antes de volver a cargar la base de datos, se acepta la conexión como administrador ya que, al no encontrar el nombre específico, la conexión es la misma que para un usuario anonimo, no se requiere contraseiia. Puede verlo si se fija en el tercer y cuarto registro de la tabla de usuario. Despues de la eliminación, el usuario administrador ya no podrá conectarse sin una contraseiia.

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
mysql> exit
Bye
% mysql -uadministrator;
ERROR 1045: Access denied for user: 'administrator@localhost'
(Using password: NO)
```

No es aconsejable utilizar el usuario raíz para algo que no sea la **administración**. Las conexiones diarias **deben** realizarse a través de usuarios con permisos desarrollados específicamente para las **tareas** que **realice** el usuario. En este sistema de **ventas**, añadiremos dos usuarios, un administrador y un usuario **convencional**. El administrador tendrá **permiso** para realizar cualquier **operación** y el usuario **convencional** tendrá determinadas limitaciones. Para añadir el administrador, basta con añadir un registro a la tabla de usuario y asignarle un **conjunto completo** de permisos.

Sin embargo, esto implica que el administrador de este sistema de **ventas** tendrá total acceso a cualquier otra base de datos que se desarrolle en el sistema. Siempre es aconsejable limitar permisos a nivel de usuario y, tras **ello**, activarlos en un nivel inferior. Para **ello**, añadiremos un registro al usuario y a la tabla de la base de datos.

Utilizaremos una **instrucción INSERT** sin especificar campos (para que **resulte más sencillo**) con el ejemplo de la tabla db, en **caso** de que siga los ejemplos propuestos. Asegurese de que en su **distribución** los campos coinciden con los campos de las tablas, ya que pueden haber cambiado:

```
mysql> INSERT INTO user (host,user,password)
VALUES ('localhost','administrator',PASSWORD('l3tm31n'));
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO db
VALUES ('localhost','firstdb','administrator','y','y','y','y','n','n',
'n','n','n','n');
Query OK, 1 row affected (0.01 sec)
```

Ahora, las tablas contienen la siguiente **información**:

```
mysql> SELECT * FROM user;
+-----+-----+-----+-----+-----+-----+
| Host          | User          | Password          |
|
| Select_priv  | Insert_priv  | Update_priv  | Delete_priv  |
| Create_priv  |
| Drop_priv   | Reload_priv  | Shutdown_priv | Process_priv |
| File_priv   |
| Grant_priv  | References_priv | Index_priv  | Alter_priv  |
| Show_db_priv |
| Super_priv  | Create_tmp_table_priv | Lock_tables_priv |
| Execute_priv |
| Repl_slave_priv | Repl_client_priv | ssl_type  | ssl_cipher  |
| x509_issuer  | x509_subject  | max_questions | max_updates |
| max_connections |
+-----+-----+-----+-----+-----+-----+

```



```

+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| localhost                | root                |
| Y                        | Y                  | Y                  | Y                  | Y
| Y                        | Y                  | Y                  | Y                  | Y
| Y                        | Y                  | Y                  | Y                  | Y
| Y                        | Y                  | Y                  | Y                  | Y
| Y                        | Y                  | Y                  | Y                  | Y
|                          |                    |                    |                    |
0 |                          |                    |                    |                    |
0 |                          |                    |                    |                    |
| test.testhost.co.za     | root                |
| Y                        | Y                  | Y                  | Y                  | Y
| Y                        | Y                  | Y                  | Y                  | Y
| Y                        | Y                  | Y                  | Y                  | Y
| Y                        | Y                  | Y                  | Y                  | Y
| Y                        | Y                  | Y                  | Y                  | Y
|                          |                    |                    |                    |
0 |                          |                    |                    |                    |
0 |                          |                    |                    |                    |
| localhost                |                      |
| N                        | N                  | N                  | N                  | N
| N                        | N                  | N                  | N                  | N
| N                        | N                  | N                  | N                  | N
| N                        | N                  | N                  | N                  | N
| N                        | N                  | N                  | N                  | N
|                          |                    |                    |                    |
0 |                          |                    |                    |                    |
0 |                          |                    |                    |                    |
| test.testhost.co.za     |                      |
| N                        | N                  | N                  | N                  | N
| N                        | N                  | N                  | N                  | N
| N                        | N                  | N                  | N                  | N
| N                        | N                  | N                  | N                  | N
| N                        | N                  | N                  | N                  | N
|                          |                    |                    |                    |
0 |                          |                    |                    |                    |
0 |                          |                    |                    |                    |
| localhost                | administrator       |
26981a09472b4835 | N
| N                        | N                  | N                  | N                  | N
| N                        | N                  | N                  | N                  | N
| N                        | N                  | N                  | N                  | N
| N                        | N                  | N                  | N                  | N
| N                        | N                  | N                  | N                  | N

```

```

|          |          |          |
0 |
0 |          0 |

+-----+-----+-----+-----+
|          |          |          |          |
+-----+-----+-----+-----+
|          |          |          |          |
+-----+-----+-----+-----+
|          |          |          |          |
+-----+-----+-----+-----+
|          |          |          |          |
+-----+-----+-----+-----+
5 rows in set (0.05 sec)

mysql> SELECT * FROM db;;
+-----+-----+-----+-----+
|          |          |          |          |
+-----+-----+-----+-----+
| Host      | Db      | User      | Select_priv |
Insert_priv |
Update_priv | Delete_priv | Create_priv | Drop_priv |
Grant_priv |
References_priv | Index_priv | Alter_priv |
+-----+-----+-----+-----+
|          |          |          |          |
+-----+-----+-----+-----+
| %        | test    |          | Y          | Y
| Y
| Y        | Y       | Y       | N          | Y
| Y        | Y       |          |          |
| %        | test\-% |          | Y          | Y
| Y        | Y       | Y       | Y          | N
| Y
| Y        | Y       |          |          |
| localhost | firstdb | administrator | Y          | Y
| Y
| Y        | N       | N       | N          | N
| N        | N       |          |          |
+-----+-----+-----+-----+
|          |          |          |          |
+-----+-----+-----+-----+
3 rows in set (0.01 sec)

```

El administrador puede conectarse a la base de datos con la contraseña pero solamente puede manipular los datos de la base de datos firstdb. No olvide eliminar las tablas antes de aplicar estos permisos:

```

% mysqladmin reload -u root -p
Enter password:
% mysql mysql;
ERROR 1045: Access denied for user: 'root@localhost' (Using
password: NO)

```

Si no se ha conectado como raíz, obtendrá un error que indica que el usuario anónimo no tiene **permiso**:

```
% mysql mysql;  
ERROR 1044: Access denied for user: '@localhost' to database  
'mysql'
```

Uso de GRANT y REVOKE para manipular las tablas de permisos

En lugar de actualizar directamente las tablas y tener que eliminar la base de datos, una **técnica** mas sencilla consiste en utilizar las instrucciones GRANT y REVOKE para gestionar los permisos. La **sintaxis** básica de GRANT es la siguiente:

```
GRANT privilege ON table-or-database-name TO user_name@hostname  
IDENTIFIED BY 'password'
```

Para añadir un usuario **convencional** a este sistema de ventas, podría utilizar lo siguiente:

```
mysql> GRANT SELECT ON sales.* TO regular-user@localhost  
IDENTIFIED BY '13tm37n_2';  
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT * FROM user;
```

Host	User	Password	Select-priv	Insert-priv	Update-priv	Delete-priv	Create-priv	Drop-priv	Reload-priv	Shutdown-priv	Process-priv	File-priv	Grant-priv	References-priv	Index-priv	Alter-priv	Show_db_priv	Super-priv	Create_tmp_table_priv	Lock-tables-priv	Execute-priv	Repl_slave_priv	Repl_client_priv	ssl_type	ssl_cipher	x509_issuer	x509_subject	max_questions	max_updates	max_connections	
localhost	root		Y																												
			Y																												
			Y	Y																											
			Y																												

```

| Y          | Y          | Y          | Y          | Y
| Y          | Y          | Y          | Y          | Y
| Y          | Y          | Y          | Y          | Y
|           |           |           |           |
0 |
0 |           0 |
| test.testhost.co.za      | root          |
| Y
| Y          | Y          | Y          | Y          | Y
| Y
| Y          | Y          | Y          | Y          | Y
| Y          | Y          | Y          | Y          | Y
| Y          | Y          | Y          | Y          | Y
|           |           |           |           |
0 |
0 |           0 |
| localhost                |                |
| N
| N          | N          | N          | N          | N
| N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
|           |           |           |           |
0 |
0 |           0 |
| test.testhost.co.za      |                |
| N
| N          | N          | N          | N          | N
| N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
|           |           |           |           |
0 |
0 |           0 |
| localhost                | administrator  |
26981a09472b4835 | N
| N          | N          | N          | N          | N
| N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
|           |           |           |           |
0 |
0 |           0 |
| localhost                | regular-user   | 1bfcf83b2eb5e59
| N
| N          | N          | N          | N          | N
| N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N

```

```

|          |          |          |          |
0 |
0 |          0 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
6 rows in set (0.05 sec)
mysql> SELECT * FROM db;
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| Host      | Db      | User      | Select-priv |
Insert-priv |
Update-priv | Delete-priv | Create-priv | Drop-priv |
Grant-priv |
References-priv | Index-priv | Alter-priv |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| %        | test   |          | Y          | Y
| Y
| Y        | Y      | Y        | N          | Y
| Y        | Y      |          |           |
| %        | test\-% |          | Y          | Y
| Y
| Y        | Y      | Y        | N          | Y
| Y        | Y      |          |           |
| localhost | firstdb | administrator | Y          | Y
| Y
| Y        | N      | N        | N          | N
| N        | N      |          |           |
| localhost | sales   | regular-user | Y          | N
| N        | N      | N        | N          | N
N
| N        | N      |          |           |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
4 rows in set (0.01 sec)

```

La contraseíia se codifica automáticamente cuando se genera con **GRANT** , por lo que no es necesario utilizar la función **PASSWORD()** para codificarla. De hecho, si lo hace, la codificara dos veces. Puede cambiar la contraseíia si vuelve a emitir los mismos permisos con una nueva contraseíia.

También puede denegar permisos de la misma forma en que los concede:

```

mysql> REVOKE SELECT ON sales.* FROM regular-user@localhost;
Query OK, 0 rows affected (0.01 sec)
mysql> SELECT * FROM user;

```

```

+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| Host                | User                | Password
|
| Select-priv | Insert-priv | Update-priv | Delete-priv |
| Create-priv |
| Drop-priv | Reload-priv | Shutdown-priv | Process-priv |
| File-priv |
| Grant-priv | References-priv | Index-priv | Alter-priv |
| Show_db_priv |
| Super-priv | Create_tmp_table_priv | Lock tables-priv |
| Execute-priv |
| Repl_slave_priv | Repl_client_priv | ssl_type | ssl_cipher
| x509_issuer | x509_subject | max_questions | max_updates |
| max_connections |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| localhost          | root                |
| Y                  |                      |
| Y                  | Y                    | Y                    | Y
| Y                  | Y                    | Y                    | Y
| Y                  | Y                    | Y                    | Y
| Y                  | Y                    | Y                    | Y
| Y                  | Y                    | Y                    | Y
|                    |                      |                      |
0 |                    |                      |
0 |                    | 0 |                  |
| test.testhost.co.za | root                |
| Y                  |                      |
| Y                  | Y                    | Y                    | Y
| Y                  | Y                    | Y                    | Y
| Y                  | Y                    | Y                    | Y
| Y                  | Y                    | Y                    | Y
|                    |                      |                      |
0 |                    |                      |
0 |                    | 0 |                  |
| localhost          |                      |
| N                  |                      |
| N                  | N                    | N                    | N
| N                  | N                    | N                    | N
| N                  | N                    | N                    | N

```

```

| N          | N          | N          | N
|           |           |           |
0 |
0 |           0 |
| test.testhost.co.za |           |
| N
| N          | N          | N          | N          | N
| N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
|           |           |           |           |
0 |
0 |           0 |
| localhost | administrator |
26981a09472b4835 | N
| N          | N          | N          | N          | N
| N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
|           |           |           |           |
0 |
0 |           0 |
| localhost | regular-user | 1bfcf83b2eb5e59
| N
| N          | N          | N          | N          | N
| N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
|           |           |           |           |
0 |
0 |           0 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
6 rows in set (0.05 sec)
mysql> SELECT * FROM db;
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| Host      | Db      | User      | Select-priv |
Insert-priv |
Update-priv | Delete-priv | Create-priv | Drop-priv |
Grant-priv  |
References-priv | Index-priv | Alter-priv  |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

```

+-----+-----+
| %      | test  |          | Y      | Y
| Y
| Y      | Y     | Y        | N      | Y
| Y      | Y     |          |
| %      | test\-% |         | Y      | Y
| Y      | Y     | Y        | Y      | N
| Y
| Y      | Y     |          |
| localhost | firstdb | administrator | Y      | Y
| Y
| Y      | N     | N        | N      | N
| N      | N     |          |
+-----+-----+
+-----+-----+
+-----+-----+

```

3 rows in set (0.00 sec)

Apreciara que se ha eliminado **todo** rastro del usuario de la tabla db pero que este sigue existiendo en la tabla de usuarios. No se puede eliminar de la tabla sin borrarla directamente. Un usuario sin permisos (denominado **permiso** USAGE) puede conectarse **al servidor**, acceder a determinada **información** y ver las bases de datos existentes, como sucedia en las **primeras** versiones de MySQL 4. Por ejemplo:

```

mysql> exit
Bye
% mysql -uregular_user -pl3tm37n_2
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 21 to server version: 4.0.1-alpha-
max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

```

```

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| firstdb  |
| mysql    |
| test     |
+-----+
3 rows in set (0.00 sec)

```

Para eliminar el rastro del usuario, borrelo directamente de la tabla de usuarios (mientras esta conectado como raiz):

```

mysql> exit
Bye
% mysql mysql -uroot -pg00r002b
Welcome to the MySQL monitor.  Commands end with ; or \g.

```


Your MySQL connection id is 22 to server version: 4.0.1-alpha-max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

```
mysql> DELETE FROM user WHERE user='regular_user';
```

```
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT * FROM user;
```

```
+-----+-----+-----+-----+
| Host | User | Password |
+-----+-----+-----+-----+
| localhost | root | | | |
| Y | Y | Y | Y | Y |
| Y | Y | Y | Y | Y |
| Y | Y | Y | Y | Y |
| Y | Y | Y | Y | Y |
| Y | Y | Y | Y | Y |
| 0 | 0 |
| test.testhost.co.za | root |
| Y | Y | Y | Y | Y |
| Y | Y | Y | Y | Y |
| Y | Y | Y | Y | Y |
| Y | Y | Y | Y | Y |
| 0 |
```

```

0 |          0 |
| localhost          |
| N
| N          | N          | N          | N          | N
| N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
|          |          |          |          |
0 |
0 |          0 |
| test.testhost.co.za |
| N
| N          | N          | N          | N          | N
| N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
|          |          |          |          |
0 |
0 |          0 |
| localhost          | administrator |
26981a09472b4835 | N
| N          | N          | N          | N          | N
| N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
| N          | N          | N          | N          | N
|          |          |          |          |
0 |
0 |          0 |
+-----+-----+-----+-----+
| +-----+-----+-----+-----+
| +-----+-----+-----+-----+
| +-----+-----+-----+-----+
| +-----+-----+-----+-----+
| +-----+-----+-----+-----+
| +-----+-----+-----+-----+
5 rows in set (0.05 sec)

```

En la tabla 14.3 se incluye la descripción de todos los privilegios disponibles.

Tabla 14.3. Privilegios

Privilegio	Descripción
ALL	Concede todos los permisos basicos.
ALL PRIVILEGES	Igual que ALL.
ALTER	Permiso para cambiar la estructura de una tabla (una instrucción ALTER), excluyendo los indices.

Privilegio	Descripción
CREATE	Permiso para crear bases de datos o tablas, pero no índices.
CREATE TEMPORARY TABLES	Permiso para crear una tabla temporal (instrucción CREATE TEMPORARY TABLE).
DELETE	Permiso para eliminar registros de una tabla (una instrucción DELETE).
DROP	Permiso para borrar bases de datos o tablas, pero no índices.
EXECUTE	Permiso para ejecutar procedimientos almacenados (previsto para MySQL 5).
FILE	Permiso para leer y escribir archivos en el servidor (para instrucciones LOAD DATA INFILE o SELECT INTO OUTFILE). Todos los archivos que el usuario puede leer son legibles.
GRANT	Permiso para conceder a un usuario permisos propiedad de otro usuario.
INDEX	Permiso para crear, modificar o borrar índices.
INSERT	Permiso para añadir nuevos registros a la tabla (una instrucción INSERT).
LOCK TABLES	Permiso para bloquear una tabla para la que el usuario tiene permiso SELECT.
PROCESS	Permiso para ver los procesos MySQL actuales o para eliminarlos (para instrucciones SHOW PROCESSLISTS o KILL SQL).
REFERENCES	Actualmente no se utiliza en MySQL.
RELOAD	Permiso para volver a cargar la base de datos (una instrucción FLUSH o una recarga, actualización o eliminación generada desde mysqladmin).
REPLICATION CLIENT	Permiso para preguntar sobre la duplicación de esclavos y maestros.
REPLICATION SLAVE	Permiso para duplicar desde el servidor (los esclavos lo necesitan para duplicar).
SHOW DATABASES	Permiso para ver todas las bases de datos.
SELECT	Permiso para devolver datos de una tabla (una instrucción SELECT).
SHUTDOWN	Permiso para desactivar el servidor.
SUPER	Permiso para conectarse, incluso si se alcanza el máximo de conexiones, y ejecutar los comandos

Privilegio	Descripción
	CHANGE MASTER, KILL en subprocesos, depuración mysqld, PURGE MASTER LOGS y SET GLOBAL.
UPDATE	Permiso para modificar datos en una tabla (una instrucción UPDATE).
USAGE	Permiso para conectarse al servidor y ejecutar instrucciones disponibles para todos (en las primeras versiones de MySQL 4, incluía SHOW DATABASES).

El ejemplo anterior concedía permisos para todas las tablas de la base de datos de ventas. Lo puede modificar si cambia el nombre de la base de datos y de las tablas que concede (tabla 14.4).

Tabla 14.4. Nombre de la base de datos y de tablas

Nombre	Descripción
••	Todas las tablas de todas las bases de datos
•	Todas las tablas de la base de datos actual
nombredelabase-dedatos.*	Todas las tablas de la base de datos nombre delabasededatos
nombredelabasededatos.nombredelatabla	La tabla nombredelatabla de la base de datos nombredelabasededatos.

Por ejemplo:

```
mysql> GRANT SELECT ON *.* TO regular_user@localhost IDENTIFIED
BY '13tm37n_2';
Query OK, 0 rows affected (0.00 sec)
```

Como el permiso se concede en todas las bases de datos, no se necesita una entrada en la tabla de la base de datos; basta con la tabla de usuarios, con el campo selectpriv configurado, con el valor Y:

```
mysql> SELECT * FROM user;
+-----+-----+-----+-----+
| Host          | User          | Password          |
+-----+-----+-----+-----+
|                |                |                |
+-----+-----+-----+-----+
|                |                |                |
+-----+-----+-----+-----+
| Host          | User          | Password          |
+-----+-----+-----+-----+
| Select-priv   |                |                |
+-----+-----+-----+-----+
| Insert-priv   | Update-priv   | Delete-priv       | Create-priv       |
+-----+-----+-----+-----+
| Drop-priv     |                |                |                |
+-----+-----+-----+-----+
```

```

Reload-priv | Shutdown-priv | Process-priv | File-priv |
Grant-priv |
References-priv | Index-priv | Alter-priv | ssl_type |
ssl_cipher |
x509_issuer | x509_subject |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
| localhost          | root          | 43b591f759a842a9 | Y
| Y                  | Y             | Y                 | Y
| Y                  |               |                   |
| Y                  | Y             | Y                 | Y
| Y                  | Y             | NONE              |
|
|
| test.testhost.co.za | root          | 43b591f759a842a9 | Y
| Y                  | Y             | Y                 | Y
| Y                  | Y             | Y                 | Y
| Y                  | Y             | NONE              |
|
|
| localhost          |               |                   | N
| N                  | N             | N                 | N
| N                  |               |                   |
| N                  | N             | N                 | N
| N                  | N             | NONE              |
|
|
| test.testhost.co.za |               |                   | N
| N                  | N             | N                 | N
| N                  |               |                   |
| N                  | N             | N                 | N
| N                  | N             | NONE              |
|
|
| localhost          | administrator | 74126e0c6742d7e9 | N
| N                  | N             | N                 | N
| N                  | N             | N                 | N
| N                  | N             | NONE              |
|
|
| localhost          | regular-user  | 1bfcf83b2eb5e591 | Y
| N                  | N             | N                 | N
| N                  | N             | N                 | N
| N                  | N             | NONE              |
|
|
+-----+-----+-----+-----+

```

```

+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

```

Uso de SET para definir contraseñas de usuario

Otra forma de cambiar contraseñas es por medio de la instrucción SET PASSWORD. **Cualquier** usuario que no sea anónimo puede cambiar su propia contraseña de esta forma (otra de las razones por la que conviene prestar especial atención a la asignación de usuarios. En ocasiones ocurre que un usuario cambia su contraseña y deniega el acceso a otros, ya que no sabía que compartían este nombre de usuario).

Puede definir una contraseña para el usuario que haya utilizado para conectarse de esta forma:

```

mysql> SET PASSWORD=PASSWORD('g00r002b2');
Query OK, 0 rows affected (0.00 sec)

```

Un usuario con acceso a la tabla de usuarios en la base de datos mysql también puede definir contraseñas para otros usuarios si lo especifica:

```

mysql> SET PASSWORD FOR root=PASSWORD('g00r002b');
Query OK, 0 rows affected (0.00 sec)

```

No olvide utilizar la función PASSWORD() para codificar la contraseña. Si no lo hace, la contraseña se almacenará en la tabla de usuarios pero, como al realizarse la conexión la contraseña se codifica automáticamente antes de compararla en la tabla de usuarios, no podrá conectarse (si lo intenta, tendrá que consultar uno de los apartados posteriores para poder continuar):

```

mysql> SET PASSWORD FOR root='g00r002b';
Query OK, 0 rows affected (0.00 sec)

```

Tras ello, después de salir, no podrá volver a conectarse como raíz:

```

% mysql -uroot -pg00r002b2
ERROR 1045: Access denied for user: 'root@localhost' (Using
password: YES)

```

Uso de mysqladmin para definir contraseñas de usuario

Al utilizar mysqladmin, al igual que con la instrucción GRANT, no debería utilizar la función PASSWORD():

```

% mysqladmin -uroot -pg00r002b password g00r002b

```

Permisos comodín

No es necesario introducir 1.001 servidores si esta es la cantidad de servidores a la que quiere conceder acceso. MySQL acepta comodines en la tabla de servidoro-

res. Por ejemplo, el siguiente código permite al usuario conectarse desde un servidor cuyo nombre termina por `marsoburst.co.za`:

```
mysql> GRANT SELECT ON sales.* TO
regular_user@"%.marsobust.co.za"
IDENTIFIED BY '13tm37n';
Query OK, 0 rows affected (0.03 sec)
mysql> SELECT * FROM user WHERE host LIKE '%mars%';
+-----+-----+-----+-----+
| Host                                     | User           | Password      | | |
|---|---|---|---|---|
|                                           |               |               |
|                                           |               |               |
|                                           |               |               |
|                                           |               |               |
|                                           |               |               |
| Host                                     | User           | Password      |
|-----|-----|-----|-----|
| %marsobust.co.za                       | regular-user   |               |
| 1bfcf83b2eb5e591 | N              |               | N              |
| N                                         |               |               | N              |
| N                                         |               |               | N              |
| N                                         |               |               | N              |
| N                                         |               |               | N              |
|                                           |               |               |               |
| 0 |                                           |               |               |               |
| 0 |                                           |               |               |               |
+-----+-----+-----+-----+
1 row in set (0.05 sec)
```

Las comillas de la **instrucción GRANT** permiten la **utilización** de comodines o de cualquier otro caracter especial. Tambien puede añadir comodines **directamente** a las tablas MySQL.

Que hacer si no puede conectarse o no tiene permisos

No es imposible. Puede que en un descuido utilice DELETE donde no debe o que daie los **archivos** que contienen las tablas de permisos y no pueda conectarse, ni siquiera **como** raiz. No se preocupe, hay una **solución**.

En primer lugar, detenga MySQL. Como usuario raiz en Unix, si **ejecuta** MySQL fuera de /INIT.d, tendra la posibilidad de ejecutar lo siguiente:

```
% /etc/rc.d/init.d/mysql stop
Killing mysqld with pid 5091
Wait for mysqld to exit\c
.\c
.\c
.\c
.\c
020612 01:14:41 mysqld ended

done
```

En **caso** contrario, tendra que cancelar los **procesos concretos** relacionados con MySQL:

```
% ps -ax |grep mysql
5195 pts/0 S 0:00 sh /usr/local/mysql/bin/mysqld_safe
-datadir=/usr/lo
5230 pts/0 S 0:00 /usr/local/mysql-max-4.0.1-alpha-pc-
linux-gnu-i686/bi
5232 pts/0 S 0:00 /usr/local/mysql-max-4.0.1-alpha-pc-
linux-gnu-i686/bi
5233 pts/0 S 0:00 /usr/local/mysql-max-4.0.1-alpha-pc-
linux-gnu-i686/bi
5234 pts/0 S 0:00 /usr/local/mysql-max-4.0.1-alpha-pc-
linux-gnu-i686/bi
5235 pts/0 S 0:00 /usr/local/mysql-max-4.0.1-alpha-pc-
linux-gnu-i686/bi
5236 pts/0 S 0:00 /usr/local/mysql-max-4.0.1-alpha-pc-
linux-gnu-i686/bi
5237 pts/0 S 0:00 /usr/local/mysql-max-4.0.1-alpha-pc-
linux-gnu-i686/bi
5238 pts/0 S 0:00 /usr/local/mysql-max-4.0.1-alpha-pc-
linux-gnu-i686/bi
5239 pts/0 S 0:00 /usr/local/mysql-max-4.0.1-alpha-pc-
linux-gnu-i686/bi
5240 pts/0 S 0:00 /usr/local/mysql-max-4.0.1-alpha-pc-
linux-gnu-i686/bi
```



```

5241 pts/0      S          0:00 /usr/local/mysql-max-4.0.1-alpha-pc-
linux-gnu-i686/bi
% kill 5195 5230 5232 5233 5234 5235 5236 5237 5238 5239 5240
5241
mysqld ended

```

En caso de que no **funcione**, puede utilizar `kill-9` (seguido por el Id. del proceso) para cancelarlo.

En Windows, basta con recurrir al **Administrador de tareas** para cerrar MySQL. Tras **ello**, reinicie MySQL sin las tablas de permisos (para ignorar todas las restricciones):

```
% mysqld_safe --skip-grant-tables
```

Seguidamente, por medio de GRANT o con `mysqladmin`, debería poder añadir una contraseña raíz, directamente en las tablas:

```
% mysqladmin -u root password 'g00r002b'
```

No olvide detener el servidor y reiniciarlo sin `--skip-grant-tables` para activar su contraseña raíz.

Que hacer si la tabla de usuarios se daña

En ocasiones, la tabla de usuarios se daña por lo que no se puede cambiar la contraseña con `mysqladmin`. Esto me sucedió después de un **fallo** general y puede sucederle a cualquiera que manipule directamente **los archivos**. Si quiere seguir este ejemplo, puede imitar la pérdida de la tabla de usuarios si le cambia el nombre y, tras **ello**, vacía las tablas (en caso contrario, la original se guardara en cache):

```

% mv user.MYD user-bak.olddata
% mysql -uroot -pg00r002b;
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6 to server version: 4.0.1-alpha-
max-log

```

```
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

```

mysql>FLUSH TABLES;
mysql> SELECT * FROM user;
ERROR 1016: Can't open file: 'user.MYD'. (errno: 145)

```

Si este es el caso, tendrá la posibilidad de iniciar MySQL sin las tablas de concesión de permisos y, tras **ello**, **intentar** eliminar la tabla:

```

mysql> DROP TABLE user;
Query OK, 0 rows affected (0.01 sec)

CREATE TABLE user (
  Host varchar(60) binary NOT NULL default '',
  User varchar(16) binary NOT NULL default '',

```

```

Password varchar(16) binary NOT NULL default '',
Select_priv enum('N','Y') NOT NULL default 'N',
Insert_priv enum('N','Y') NOT NULL default 'N',
Update_priv enum('N','Y') NOT NULL default 'N',
Delete_priv enum('N','Y') NOT NULL default 'N',
Create_priv enum('N','Y') NOT NULL default 'N',
Drop_priv enum('N','Y') NOT NULL default 'N',
Reload_priv enum('N','Y') NOT NULL default 'N',
Shutdown_priv enum('N','Y') NOT NULL default 'N',
Process_priv enum('N','Y') NOT NULL default 'N',
File_priv enum('N','Y') NOT NULL default 'N',
Grant_priv enum('N','Y') NOT NULL default 'N',
References_priv enum('N','Y') NOT NULL default 'N',
Index_priv enum('N','Y') NOT NULL default 'N',
Alter_priv enum('N','Y') NOT NULL default 'N',
Show_db_priv enum('N','Y') NOT NULL default 'N',
Super_priv enum('N','Y') NOT NULL default 'N',
Create_tmp_table_priv enum('N','Y') NOT NULL default 'N',
Lock_tables_priv enum('N','Y') NOT NULL default 'N',
Execute_priv enum('N','Y') NOT NULL default 'N',
Repl_slave_priv enum('N','Y') NOT NULL default 'N',
Repl_client_priv enum('N','Y') NOT NULL default 'N',
ssl_type enum('', 'ANY', 'X509', 'SPECIFIED') NOT NULL default '',
ssl_cipher blob NOT NULL,
x509_issuer blob NOT NULL,
x509_subject blob NOT NULL,
max_questions int(11) unsigned NOT NULL default '0',
max_updates int(11) unsigned NOT NULL default '0',
max_connections int(11) unsigned NOT NULL default '0',
PRIMARY KEY (Host,User)
) TYPE=MyISAM COMMENT='Users and global privileges';
Query OK, 0 rows affected (0.00 sec)

```

Sin embargo, esto probablemente no le de permiso:

```

mysql> GRANT SELECT ON sales.* TO regular_user@localhost
IDENTIFIED BY '13tm37n';
ERROR 1047: Unknown command
mysql> exit
Bye
[root@test mysql]# mysqladmin -uroot password 'g00r002b'
mysqladmin: unable to change password; error: 'You must have
privileges
to update tables in the mysql database to be able to change
passwords
for others'

```

Tendra que añadir algunos valores a la tabla. En este caso, añadirá los valores predeterminados. Asegurese de que coinciden con las columnas de la tabla de usuarios que ha creado, en caso de que sea distinta:

```
[root@test mysql]# mysql mysql
```

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 5 to server version: 4.0.1-alpha-max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

```
mysql>INSERT INTO user VALUES ('localhost', 'root', '', 'Y',
'Y', 'Y',
'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
'Y', 'Y',
'Y', 'Y', 'Y', 'Y', 'Y', '', '', '', '', 0, 0, 0);
Query OK, 1 row affected (0.01 sec)
mysql>INSERT INTO user VALUES ('%', 'root', '', 'Y', 'Y', 'Y',
'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
'Y', 'Y',
'Y', 'Y', 'Y', 'Y', '', '', '', '', 0, 0, 0);
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO user VALUES ('localhost', '', '', 'N', 'N',
'N',
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
'N', 'N',
'N', 'N', 'N', 'N', '', '', '', '', 0, 0, 0);
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO user VALUES ('localhost', '', '', 'N', 'N',
'N',
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
'N', 'N',
'N', 'N', 'N', 'N', '', '', '', '', 0, 0, 0);
Query OK, 1 row affected (0.00 sec)
```

Sera necesario volver a cargar (o vaciar) las tablas de privilegios para activar los permisos y, tras ello, empezar a ejecutar comandos:

```
mysql> exit
Bye
[root@test mysql]# mysqladmin reload
[root@test mysql]# mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 7 to server version: 4.0.1-alpha-max-log
```

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

```
mysql> GRANT SELECT ON sales.* TO regular-user@localhost
IDENTIFIED BY '13tm37n';
Query OK, 0 rows affected (0.00 sec)
```

Otras opciones de GRANT

De forma predeterminada, MySQL no permite que un usuario pase sus permisos a otro usuario. Y, por motivos de control, le sugiero que no permita que sus usuarios lo hagan. Probablemente porque, para empezar, no es una buena idea y no es aconsejable que otro usuario lo reemplace. Pero si tiene que hacerlo obliga-

toriamente, por ejemplo en caso de que haya varios usuarios en los que confía, existe una solución. La opción WITH GRANT OPTION permite que un usuario ceda cualquier permiso que posea a otro usuario. A continuación veremos un ejemplo práctico que utiliza dos bases de datos: ventas y clientes. El administrador crea un regular-user2 con permiso para realizar consultas SELECT en la base de datos de ventas y, tras ello, concede la opción GRANT al primer regular-user, que tiene permiso para realizar consultas SELECT en la tabla de clientes y que, a su vez, recibe los mismos derechos que regular-used:

```
mysql> GRANT SELECT ON sales.* TO regular_user2@localhost
IDENTIFIED BY '13tm37n';
Query OK, 0 rows affected (0.01 sec)
mysql> GRANT SELECT ON customer.* TO regular_user@localhost
IDENTIFIED
  BY '13tm37n' WITH GRANT OPTION;
Query OK, 0 rows affected (0.00 sec)
mysql> exit
Bye
% mysql -u regular-user2 -p13tm37n
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 4.0.1-alpha-
max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> GRANT SELECT ON customer.* TO regular-user@localhost
IDENTIFIED
  BY '13tm37n' WITH GRANT OPTION;
ERROR 1044: Access denied for user: 'regular_user2@localhost'
to database 'customer'
```

Regular_user2 no puede ceder nada a otro usuario:

```
mysql> exit
Bye
[root@test /root]# mysql -u regular-user -p13tm37n
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 4.0.1-alpha-
max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer
mysql> GRANT SELECT ON customer.* TO regular_user@localhost
IDENTIFIED
  BY '13tm37n' WITH GRANT OPTION;
Query OK, 0 rows affected (0.00 sec)
```

Sin embargo, regular-user puede ceder permisos a regular_user2. Existen otras opciones de gran utilidad para evitar la posibilidad de que un usuario se adueñe de conexiones y que limitan las consultas, actualizaciones o conexiones a un determinado número por hora. Las tres opciones son las siguientes:

MAX-QUERIES-PER-HOUR n

```
MAX-UPDATES-PER-HOUR n
MAX-CONNECTIONS-PER-HOUR n
```

Sin estas tres opciones, la única limitación a las actividades del usuario es la variable `max_user_connections`. Pero es global, por lo que no podrá limitar a un tipo de usuario de una actividad concreta.

Utilidad de la limitación de usuarios

La limitación de consultas intensivas de bases de datos, como las búsquedas que emplean gran cantidad de uniones de tablas de gran tamaño, puede resultar de utilidad. La aplicación podría conectarse como un usuario diferente con esta limitación. Sigue siendo el mejor lugar para aplicar la limitación pero en algunos casos también resulta útil una limitación por horas, por ejemplo cuando existe la posibilidad de ataques de negación de servicio o cuando un usuario pueda utilizar varias conexiones y provocar un rendimiento negativo del servidor de bases de datos.

Tipos de limitación de usuarios

Normalmente las conexiones tienen un impacto mínimo en la base de datos, pero existe la posibilidad de que un usuario se apropie de `max_user_connections`. La configuración de `MAX_CONNECTIONS_PER_HOUR` es la mejor solución. En ocasiones puede que no le preocupe el número de conexiones pero si un usuario que realice múltiples consultas al mismo tiempo o que realice consultas extrañas. Puede configurar `MAX_QUERIES_PER_HOUR` para evitar que los usuarios realicen consultas innecesarias y malgasten recursos al realizar demasiadas consultas de gran tamaño en un breve periodo de tiempo.

Las actualizaciones tienen un mayor impacto en el rendimiento que las selecciones y un impacto más agresivo cuando se utiliza el bloqueo de tablas (como el tipo de tabla MyISAM predeterminado). Puede utilizar `MAX_UPDATES_PER_HOUR` para limitar actualizaciones por motivos de rendimiento o cuando los usuarios no deban realizar muchas actualizaciones.

TRUCO: No pasa nada por ser un poco paranoico. Los usuarios pueden abusar de la base de datos, intencionadamente o no. Si no ve la razón de otorgar determinados permisos, no lo haga. Siempre resulta más sencillo conceder un permiso que revocarlo después de ser concedido. He visto muchos sistemas de gran tamaño en los que las medidas de seguridad consistían únicamente en un usuario y una contraseña. Cuando esto se compromete, es prácticamente imposible añadir ninguna limitación.

Como ejemplo de limitaciones de usuarios, puede limitar las conexiones de `regular-user2` a dos por hora:

```
mysql> GRANT SELECT ON sales.* TO regular_user2@localhost
IDENTIFIED BY '13tm37n' WITH MAX_CONNECTIONS-PER-HOUR 2;
Query OK, 0 rows affected (0.00 sec)
```

Si `regular_user2` excede el numero de conexiones, obtendra el siguiente error:

```
ERROR 1226: User 'regular-user2' has exceeded the
'max_connections'
resource (current value: 2)
```

Del mismo modo, si se asigna `MAX_QUERIES_PER_HOUR` y se excede su valor, el mensaje de error sera el siguiente:

```
ERROR 1226: User 'regular-user2' has exceeded the
'max_questions'
resource (current value: 4)
```

TRUCO : Utilice las limitaciones de forma razonable. Si un usuario debe realizar solamente una consulta por hora, tenga en cuenta que puede que haya introducido una consulta incorrecta y que tenga que volver a realizar otra nueva.

Estrategia para gestionar usuarios de forma segura

Cuanto mas complejas sean sus necesidades, mas compleja tendra que ser su estrategia. En sitios Web sencillos basta con dos usuarios: un administrador que puede actualizar datos y un usuario de la aplicacion Web que solamente puede realizar selecciones en determinadas tablas, por ejemplo. El principio general dicta que solamente se deben conceder al usuario los permisos que necesita y ninguno mas. Si posteriormente necesita permisos adicionales, la concesion de los mismos sera muy sencilla. Pero la revocación de los mismos es otro asunto.

Los usuarios de MySQL suelen ser de tres tipos. Hay usuarios individuales (por ejemplo Anique o Channette), aplicaciones (por ejemplo un sistema de salarios o un sitio Web de noticias) y funciones (por ejemplo, la actualización de los salarios o de las noticias). Sin embargo, se pueden mezclar de distintas formas. Por ejemplo, puede que Anique actualice tanto los salarios como las noticias, utilizando ambas aplicaciones y con ambas funciones. El DBA debe decidir si concede a Anique y a Channette sus propias contraseñas, si asigna contraseñas a los sistemas de salarios y noticias o si crea un usuario en funcion de si se actualizan o no los salarios y las noticias.

Si opta por usuarios como individuos y asigna a Anique su propia contraseña, solamente tendra que recordar una conexión a la base de datos. Pero tras ello, necesita acceso para actualizar tanto la base de datos de salarios como la de noticias. Si Anique, o la aplicacion que utilice, comete un error, puede que afecte a datos con los que no deberia estar trabajando. Por ejemplo, si las bases de datos de salarios y de noticias tienen una tabla `days-data`, y como la base de datos de noticias aumenta continuamente hasta que se archiva y los datos sobre salarios se eliminan manualmente una vez procesados, puede que Anique elimine la tabla de noticias aunque su intención sea eliminar la de salarios.

Si opta por usuarios como aplicaciones, **habrá** resuelto algunos de estos problemas. Sin embargo, parece que el usuario tendrá que recordar dos contraseñas. Al mismo tiempo, no podrá saber que usuario ha realizado cambios en la base de datos. Sin embargo, hay una **solución**. Como la seguridad es necesaria, es muy probable que el individuo tenga que conectarse a la aplicación (lo que le **permite** realizar el seguimiento de **los** cambios que **realice** en la base de datos) y, tras **ello**, la aplicación se conectará a la base de datos. El usuario puede tener el mismo nombre de usuario y contraseña en **ambas** aplicaciones, **pero** nunca podrá borrar noticias cuando se conecte como la aplicación de salarios (ya que **no le ha concedido permiso** para actualizar la base de datos de noticias).

Por su **parte**, las aplicaciones suelen tener varias funciones, con numerosos niveles de usuario. Es probable que cualquiera pueda ver **los** detalles de **los** salarios, **pero sólo** el administrador puede actualizarlos. Al concederle a la aplicación **permiso** para actualizar datos, potencialmente se **permite** a un usuario convencional **hacer** lo mismo. También debe tener en cuenta el proceso de desarrollo: un programador en quien **confía** desarrolla el componente de administración de salarios de la aplicación y **un** equipo de programadores de **rango** inferior diseñan el componente para ver los salarios. Al emitir la misma contraseña para la **aplicación**, estos últimos podrían actualizar **los** datos aunque no debieran y probablemente no se les permitiera hacerlo.

En este **caso**, puede asignar nombres de usuario basados en una **combinación** de funciones y aplicación (administración de salarios, revisión de salarios, **administrador** de noticias, revisión de noticias).

A **continuación** le mostramos algunos principios que debe tener en cuenta:

- Nunca conceda a un usuario la contraseña raíz. Siempre deberían conectarse con otro nombre de usuario y otra contraseña.
- Intente conceder el **menor** número de permisos posible (pero sea razonable; siempre encontrará algún **sadico** que se regocija **al** conceder permisos **consulta** a **consulta**, como por **ejemplo** los que **permiten** que el usuario lea la **columna** de apellidos y después les obligan a implorar por otro **permiso** para leer la tabla de nombres). Los permisos **globales** asignados en la tabla de usuarios **deben** ser siempre **N** y, tras **ello**, acceder a bases de datos concretas concedidas en la tabla db.
- Para datos clave, es posible realizar el seguimiento de **los** cambios realizados por un individuo. Por lo general, la gente interactúa con la base de datos a través de una aplicación. La carga de la gestión de accesos en el nivel de individuos suele recaer en la aplicación.

Como evitar la concesión de privilegios peligrosos

Aunque siempre debe emitir los mínimos permisos necesarios, existen algunos privilegios que son especialmente peligrosos, en los que el riesgo de seguridad

puede superar **al** factor de conveniencia. Recuerde que nunca debe conceder acceso de caracter global.

Los siguientes privilegios pueden resultar una amenaza para la seguridad:

- Cualquier privilegio en la base de datos **mysql**. Un usuario con malas intenciones puede conseguir el acceso incluso despues de ver las contraseñas codificadas.
- ALTER. Un usuario puede modificar las tablas de privilegios, como por ejemplo cambiar su nombre, lo que las inutilizara.
- DROP. Si un usuario utiliza DROP en la base de datos `mysql`, desapareceran las limitaciones de permisos.
- FILE. Los usuarios con el privilegio FILE pueden acceder a cualquier archivo que **todo** el mundo pueda leer. Tambien pueden crear un archivo que tenga los privilegios del usuario MySQL.
- GRANT. Permite que un usuario pueda otorgar sus privilegios a otros, que pueden no ser tan fiables como el usuario original.
- PROCESS. Las consultas en ejecucion pueden ser vistas en texto **sencillo**, lo que incluye a cualquiera que cambie o defina contraseñas.
- SHUTDOWN. Es improbable que un DBA conceda este privilegio de buenas a primeras. No **hace** falta decir que los usuarios con el privilegio SHUTDOWN pueden cerrar el servidor y denegar todos los accesos **al** mismo.

Conexiones SSL

De forma predeterminada, la **conexión** entre el cliente y el servidor no esta codificada. En la **mayoría** de las arquitecturas de red no supondria riesgo alguno ya que las conexiones entre el cliente y el servidor de bases de datos no son publicas. Pero hay **casos** en los que es necesario transmitir los datos por **líneas públicas** y una **conexión** sin codificar **permite**, potencialmente, que cualquiera pueda ver los datos que se transmiten.

MySQL se puede configurar para que admita conexiones SSL, aunque esto afecte **al** rendimiento. Para **ello**, siga los pasos descritos a **continuación**:

1. **Instale** la biblioteca openssl, que encontrara en `www.openssl.org/`.
2. Configure MySQL con la **opción** `-with-vio -with-openssl`.

Si tiene que comprobar si una instalacion existente de MySQL es compatible con SSL (o si su instalacion ha **funcionado** correctamente), compruebe si la variable `have_openssl` se ha configurado **como** YES:


```
mysql> SHOW VARIABLES LIKE '%ssl';
+-----+-----+
| Variable-name | Value |
+-----+-----+
| have_openssl | YES   |
+-----+-----+
1 row in set (0.00 sec)
```

Una vez **admitido** SSL, puede utilizarlo con distintas opciones de concesion de permisos (tabla 14.5).

Tabla 14.5. Opciones de concesion de permisos en SSL

Opción	Descripción
REQUIRE SSL	El cliente debe conectarse con codificación SSL.
REQUIRE X509	El cliente necesita un certificado valido para conectarse.
REQUIRE ISSUER cert_issuer	El cliente necesita un certificado valido emitido por cert_issuer para conectarse.
REQUIRE SUBJECT cert_subject	El cliente necesita un certificado valido con el asunto cert_subject.
REQUIRE CIPHER cipher	El cliente debe utilizar el cifrado especificado .

REQUIRE SSL es la **menos** restrictiva de las opciones SSL. Se acepta codificación SSL de cualquier tipo. Le resultara muy util cuando no quiera enviar texto sencillo **pero** baste con una simple **codificación** de la conexion:

```
mysql> GRANT ALL PRIVILEGES ON securedb.* TO root@localhost
IDENTIFIED
  BY "g00r002b" REQUIRE SSL;
Query OK, 0 rows affected (0.01 sec)
```

REQUIRE X509 es identica, **pero** es mas restrictiva ya que el certificado debe ser valido:

```
mysql> GRANT ALL PRIVILEGES ON securedb.* TO root@localhost
IDENTIFIED
  BY "g00r002b" REQUIRE X509;
Query OK, 0 rows affected (0.01 sec)
```

REQUIRE ISSUER y REQUIRE SUBJECT son mas seguras ya que el **certificado** debe provenir de un **emisor concreto** o **contener un asunto** determinado:

```
mysql> GRANT ALL PRIVILEGES ON securedb.* TO root@localhost
IDENTIFIED
```

```

BY "g00r002b" REQUIRE ISSUER "C=ZA, ST=Western Cape, L=Cape
Town,
O=Mars Inc CN=Lilian Nomvete/Email=lilian@marsorbust.co.za";
Query OK, 0 rows affected (0.01 sec)
mysql> GRANT ALL PRIVILEGES ON securedb.* TO root@localhost
IDENTIFIED
BY "g00r002b" REQUIRE SUBJECT "C=ZA, ST=Western Cape, L=Cape
Town,
O=Mars Inc CN=Benedict Mhlala/
Email=benedict@marsorbust.co.za";
Query OK, 0 rows affected (0.01 sec)

```

REQUIRE CIPHER le permite asegurarse de que no se utilizan algoritmos SSL debiles y especificar un cifrado concreto:

```

mysql> GRANT ALL PRIVILEGES ON secure*. * TO root@localhost
IDENTIFIED
BY "g00r002b" REQUIRE CIPHER "EDH-RSA-DES-CBC3-SHA";
Query OK, 0 rows affected (0.01 sec)

```

Puede especificar alguna o todas de las opciones anteriores al mismo tiempo (AND es opcional):

```

mysql> GRANT ALL PRIVILEGES ON securedb.* TO root@localhost
IDENTIFIED
BY "g00r002b" REQUIRE ISSUER "C=ZA, ST=Western Cape, L=Cape
Town,
O=Mars Inc CN=Lilian Nomvete/Email=lilian@marsorbust.co.za"
AND
SUBJECT "C=ZA, ST=Western Cape, L=Cape Town, O=Mars Inc
CN=Benedict
Mhlala/Email=benedict@marsorbust.co.za" AND CIPHER "EDH-RSA-
DES-CBC3-SHA";
Query OK, 0 rows affected (0.01 sec)

```

Seguridad de aplicaciones

La mayoría de las lagunas de seguridad se deben a aplicaciones de mala calidad. Existen algunos aspectos que debe evitar:

- Nunca se fie de los datos de los usuarios. Debe verificar en todo momento los datos introducidos por un usuario.
- La inclusion de comillas en un formulario de un sitio Web es una forma de intrusion habitual. Por ejemplo, una aplicacion toma un nombre de usuario y una contraseiia, y ejecuta una consulta como `SELECT * FROM contrasefias WHERE nombre_de_usuario='$nombre_de_usuario' AND contraseña='$contrasefia'`. Las aplicaciones sin un diseiio cuidado permitiran que `$password` contenga algo como

`aaa';DELETE FROM passwords;` . Al analizar la consulta, MySQL cree que la comilla simple que aparece detras de `aaa` es el final de la consulta y por eso pasa a la siguiente. La mayoría de los lenguajes disponen de sencillas funciones para evitarlo e ignorar todas las comillas de la cadena, como por ejemplo `mysql_real_escape_string()` en C o `addslashes()` en PHP.

- Compruebe el tamaño de los datos. Un calculo complejo puede funcionar correctamente con un numero de un solo digito, pero un numero de 250 digitos pasado por un usuario puede provocar el colapso de la aplicacion.
- Elimine todos los caracteres especiales de las cadenas pasadas a MySQL.
- Utilice comillas en los numeros asi como en las cadenas.

Seguridad del sistema

De forma predeterminada, MySQL se ejecuta como su propio usuario en Unix. El usuario y el grupo son `mysql`. Nunca debe permitir el acceso al sistema como usuario `mysql`; debe reservarlo unicamente para la propia base de datos. MySQL tambien crea un directorio independiente en el que almacena los archivos de datos. A este directorio solamente puede acceder el usuario `mysql`. Los parametros predeterminados se han seleccionado por una razon; siga estos principios:

- Separe el directorio de datos.
- Proteja el directorio de datos (nadie debe poder leer y mucho menos escribir en el directorio de datos MySQL).
- Ejecute MySQL como su propio usuario.

Problemas de seguridad relacionados con LOAD DATA LOCAL

La recuperación desde un equipo cliente puede ser necesaria pero implica un riesgo de seguridad. Cualquiera puede utilizar `LOAD DATA LOCAL` para leer todos los archivos a los que tenga acceso el usuario con el que se conectan. Pueden hacerlo mediante la creación de una tabla y leyendo desde la misma una vez cargados los datos. Si se conectan con el mismo usuario que el servidor Web y tienen el acceso apropiado para ejecutar consultas, puede resultar peligroso. De forma predeterminada, MySQL permite `LOAD DATA LOCAL`. Para evitarlo y no permitir `all LOAD DATA LOCAL`, inicie el servidor MySQL con la opcion `-local-infile=0`. Tambien puede compilar MySQL sin la opcion `-enable-local-infile`.

Resumen

El mecanismo de gestión de usuarios de MySQL es muy potente, flexible y, a menudo, no se utiliza correctamente. La base de datos `mysql` contiene **las distintas** tablas de permisos que **permiten** el control del acceso en **función** del usuario, el anfitrión o la **acción** que se lleve a cabo. Puede actualizar directamente estas tablas con instrucciones SQL (en cuyo **caso**, la **eliminación** de las mismas activará los cambios) o con ayuda de **las** instrucciones GRANT y REVOKE.

La **primera** tarea de una nueva instalación debe ser la **emisión** de una contraseña raíz. Hasta entonces, cualquiera puede conectarse como raíz y tener total acceso a todos **los** elementos. Para **ello**, puede utilizar `mysqladmin`, la **instrucción** SET o GRANT.

MySQL **admite** conexiones SSL para **mejorar** la seguridad. No aparecen instaladas de forma predeterminada ya que afectan **al** rendimiento.

También hemos descrito algunos principios generales de **protección** de datos:

- Nunca conceda una contraseña raíz a un usuario. Los usuarios siempre **deben** conectarse con otro nombre de usuario.
- Nunca permita el acceso a la tabla de usuarios, ni siquiera para **leerla**. Con tan **sólo** ver la contraseña codificada, un usuario puede conseguir un acceso total.
- Intente conceder el **menor** número de permisos posible. Esto significa que la tabla de usuarios contiene N en todas las columnas.
- En datos de gran importancia, puede realizar el seguimiento de **los** cambios que efectúa cualquier individuo. Por lo general, la gente interactúa con una base de datos a través de una aplicación. La carga de la gestión de accesos en el nivel de individuos suele recaer en la aplicación.
- Asegurese de que no se puede conectar como usuario raíz desde ningún **servidor** sin una contraseña.
- No debe almacenar las contraseñas en texto sencillo y **no deben** ser **palabras** del diccionario.
- Revise periódicamente **los** privilegios de **los** usuarios y asegurese de que nadie ha concedido a nadie privilegios innecesarios.

15

Instalacion de MySQL

Si es un usuario novel, puede que necesite instalar su propia copia de MySQL para tener algo con lo que practicar. O puede que hasta el momento esta tarea la hayan realizado administradores de servicios y solamente sienta curiosidad. Sea cual sea la razon, si utiliza MySQL a menudo, es muy probable que tenga que instalarlo en un momento u otro. Si le interesa acometer esta tarea cuanto antes, este capitulo la facilitara el proceso para que, una vez finalizado, se pregunte a que se debia tanto misterio.

En este capitulo abordaremos los siguientes temas:

- Instalacion de una distribución fuente o binaria
- Instalacion en Windows
- Instalacion en Unix
- Instalacion de distribuciones fuente y binarias
- Compilacion optima de MySQL
- Instalacion de varios servidores en el mismo equipo
- Uso de `mysql_multi` para gestionar varios servidores
- Actualización de la version 3.23 de SQL a la version 4

Instalacion de una distribucion fuente o binaria

MySQL es un desarrollo de codigo abierto, lo que significa que el codigo fuente esta disponible gratuitamente para **todo** el mundo que lo quiera. Los sistemas operativos como Linux y FreeBSD tambien son de codigo abierto, **pero** Windows es un software propietario, lo que significa que el codigo fuente es propiedad y esta bajo el control de Microsoft. Como el codigo fuente de MySQL esta disponible, hay dos **formas** de instalar MySQL:

- Instalacion **binaria**, lo que significa que se utiliza una distribucion ya **compilada** por programadores de MySQL (o de terceros).
- Instalacion fuente, lo que significa que debe compilar e instalar personalmente el codigo fuente MySQL.

Normalmente la instalacion **binaria** de MySQL resulta mas sencilla y rapida **pero** la decision depende de distintos factores, asi como de su facilidad para compilar software. Los usuarios de Windows apenas **tienen** que hacerlo, **pero** los usuarios de FreeBSD, por ejemplo, lo **tendrán** que **hacer a menudo**. Existen diversas razones para decantarse por una instalacion:

- El sistema en el que quiere instalar no tiene una distribucion **binaria**. En el **momento** de **creación** del libro, habia distribuciones **binarias** disponibles para Linux, FreeBSD, Windows, **Solaris**, **MacOS X**, HP-UX, AIX, SCO, SGI Iris, **Dec OSF** y **BSDi**, aunque no todos tenian distribuciones para la ultima version de MySQL.
- Puede que le interese optimizar MySQL con un compilador diferente o por medio de opciones de compilacion distintas.
- Puede que necesite algo que la distribucion **binaria** no ofrezca, como conjuntos de caracteres adicionales, la **solución** de algun **fallo** o una configuracion diferente.

En las tablas 15.1 y 15.2 se describen los directorios de una instalacion **binaria** y una instalacion fuente respectivamente.

Tabla 15.1. Directorios de una instalacion binaria

Directorio	Descripción
bin	Almacena los ejecutables binarios, incluyendo el tan importante <code>msyqld</code> , así como todas las utilidades como <code>mysqladmin</code> , <code>mysqlcheck</code> y <code>mysqldump</code> .

Directorio	Descripción
<code>data</code>	Las bases de datos, así como los archivos de registro.
<code>include</code>	Archivos de encabezado C.
<code>lib</code>	Las bibliotecas compiladas.
<code>scripts</code>	Contiene la secuencia de comandos <code>mysql_install_db</code> .
<code>share/mysql</code>	Un directorio para cada idioma con archivos que incluyen los mensajes de error para ese idioma en concreto.
<code>sql-bench</code>	Resultados y utilidades de análisis comparativo.

Tabla 15.2. Directorios de una instalación fuente

Directorio	Descripción
<code>bin</code>	Almacena los ejecutables binarios, incluyendo el tan importante <code>mysqld</code> , así como todas las utilidades como <code>mysqladmin</code> , <code>mysqlcheck</code> y <code>mysqldump</code> .
<code>include</code>	Archivos de encabezado C.
<code>info</code>	Archivos de documentación en formato Info.
<code>lib</code>	Las bibliotecas compiladas.
<code>libexec</code>	En una instalación fuente predeterminada, el servidor <code>mysqld</code> se almacena aquí, no en el directorio <code>bin</code> .
<code>share/mysql</code>	Un directorio para cada idioma con archivos que incluyen los mensajes de error para ese idioma en concreto.
<code>sql-bench</code>	Resultados y utilidades de análisis comparativo.
<code>var</code>	Las bases de datos y los archivos de registro.

Instalación de MySQL en Windows

Para instalar MySQL en un equipo bajo Windows, necesita lo siguiente:

- Un sistema operativo de la familia Windows, es decir, actualmente, Windows 95/98/Me o Windows NT/2000/XP.
- Una copia de los ejecutables de MySQL o el código fuente MySQL (si quiere compilar personalmente MySQL).

- Un programa para descomprimir el archivo de distribución
- Espacio suficiente para MySQL en su sistema.
- Compatibilidad con TCP/IP. Si su equipo puede conectarse a Internet, ya lo tendrá. En caso contrario, instalelo (es un protocolo de red).
- Si quiere conectar MySQL a través de ODBC, por ejemplo para conectar Microsoft Access a MySQL, necesitará el controlador MyODBC.

Instalación de una distribución binaria en Windows

Para realizar la instalación en Windows NT/2000/XP, debe asegurarse de que se ha conectado como usuario con privilegios de administrador. Para actualizar desde una versión anterior de MySQL, tendrá que detener el servidor. Si se ejecuta como servicio, deténgalo con lo siguiente:

```
C:\> NET STOP MySQL
```

O también puede utilizar msyqladmin:

```
C:\mysql\bin> mysqladmin -u root -pg00r002b shutdown
```

Si quiere cambiar el ejecutable que está utilizando (por ejemplo, `mysqld-max-nt` por `mysqld-opt`) en Windows NT/2000/XP, tendrá que eliminar el servicio:

```
C:\mysql\bin> mysqld-max-nt -remove
```

Tras ello, siga los pasos descritos a continuación:

1. Descomprima el archivo de distribución comprimido en un directorio temporal.
2. Ejecute el ejecutable `setup.exe` para iniciar la instalación. De forma predeterminada, MySQL se instala en `C:\mysql` aunque muchos usuarios de Windows prefieren ubicarlo en una posición como `C:\Archivos de programa\MySQL`. Si cambia la ubicación, tendrá que especificarla en el archivo de configuración (normalmente `my.ini`):

```
basedir=D:/ruta de instalación/  
datadir=D:/ruta de datos/
```

3. MySQL incluye una serie de archivos ejecutables. Debe seleccionar uno en función de sus necesidades (tabla 15.3).

Tabla 15.3. Archivos ejecutables

Archivo	Descripción
<code>mysqld</code>	Un binario que admite depuración, la comprobación automática de la asignación de memoria, ta-

Archivo	Descripción
	blas transaccionales (InnoDB y BDB) y enlaces simbolicos.
mysqld-opt	Un binario optimizado sin compatibilidad con tablas transaccionales (InnoDB o BDB).
mysqld-nt	Un binario optimizado que admite (para su uso con NT/2000/XP). Se puede ejecutar en Windows 95/98/Me pero no se crearan canalizaciones con nombre ya que estos sistemas operativos no las admiten .
mysqld-max	Un binario optimizado que admite tablas transaccionales (InnoDB y BDB) y enlaces simbolicos.
mysqld-max-nt	Un binario optimizado que admite tablas transaccionales (InnoDB y BDB) que son canalizaciones con nombre cuando se ejecutan en NT/2000/XP, y enlaces simbolicos.

Instalacion de MySQL como servicio en Windows NT/2000/XP

Si realmente quiere ejecutar MySQL en Windows, seguramente quiera hacerlo como servicio. Esto **permite** que el proceso se inicie automaticamente (la opcion mas aconsejable para un servidor de bases de datos) **al** iniciarse Windows y que se cierre automaticamente cuando se cierre Windows. Los servicios se ejecutan desde Windows y no se ven afectados por usuarios que se conectan y se **desconectan**. Con NT/2000/XP, instale MySQL como servicio de esta forma:

```
C:\> c:\mysql\bin\mysqld-max-nt -install
```

Si no quiere que MySQL se inicie automaticamente **pero** quiere que lo haga como servicio, ejecute el mismo comando con la opcion manual:

```
C:\mysql\bin> mysqld-max-nt -install-manual
```

Tras ello, podra iniciar el servicio de esta forma:

```
C:\> net start mysql
The MySql service is starting.
The MySql service was started successfully.
```

Y detenerlo con la habitual `mysqladmin shutdown` o con lo siguiente:

```
C:\> net stop mysql
The MySql service is stopping.....
The MySql service was stopped successfully
```

Para **eliminarlo** como servicio, debe ejecutar lo siguiente:

```
C:\> c:\mysql\bin\mysqld-max-nt -remove
```

También puede utilizar el panel de control de **servicios** y pulsar **Iniciar** o **Detener**.

Debe saber que Windows NT tiene un problema al cerrar MySQL automáticamente ya que no espera lo suficiente para que se cierre antes de **apagarlo** (por lo no se produce un cierre limpio y aumentan los riesgos de dañarlos).

Para solucionar este problema, abra el Editor de registros `\winnt\system32\regedt32.exe` y defina un nuevo valor en milisegundos para `WaitToKillServiceTimeout` en `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control` del **árbol de registros**.

En capítulos anteriores **encontrará más información sobre la forma de iniciar MySQL**. En la tabla 15.1 se incluye una breve **descripción de los contenidos de los directorios** que se acaban de crear.

Instalacion de MySQL en Unix

Para instalar MySQL en un equipo Unix, necesitara lo siguiente:

- Un sistema operativo de la familia Unix (Linux, **FreeBSD**, etc).
- Una copia de los ejecutables de MySQL o el código fuente MySQL (si quiere compilar personalmente MySQL).
- **gunzip** (gzip o zcat) y tar para extraer y descomprimir el **archivo de distribución** (le recomendamos las versiones GNU).
- Espacio suficiente para MySQL en su sistema.
- make y un compilador C++ (como **gcc**) si tiene **pensado** compilar el código fuente.

Instalacion de una distribucion binaria (tar) en Unix

Para instalar MySQL en Unix desde una distribución **binaria**, siga los pasos descritos a **continuación**:

1. Cambie a usuario raíz. Es muy probable que tenga que actuar como raíz para ejecutar los siguientes comandos:

```
% su -  
Password:
```

2. **Añada** el usuario MySQL y el grupo MySQL con los que se ejecutará MySQL. Nunca debe ejecutar MySQL como usuario raíz. Si lo desea, puede asignar otro nombre al usuario y al grupo:

```
% groupadd mysql  
% useradd -g mysql mysql
```

NOTA: Puede que los comandos sean ligeramente distintos en su versión de Unix (por ejemplo, `adduser` y `addgroup`).

3. Cambie al directorio en el que quiera ubicar MySQL. De forma predeterminada, MySQL espera ubicarse en `/usr/local` si se trata de una instalación binaria, pero puede indicar cualquier otro. Si cambia la ubicación, tendrá que aplicar los cambios a la configuración y a algunas de las utilidades distribuidas con MySQL para que apunten a la nueva ubicación:

```
% cd /usr/local
```

4. Extraiga el archivo:

```
% gunzip -c /home/ mysql-max-4.x.x-platform-os-extra.tar.gz |  
tar -xf -
```

El nombre de archivo que verá dependerá de la distribución que utilice. En este caso vemos `mysql-max-4.0.2-alpha-pc-linux-gnu-i686`. Asegúrese de que tiene la versión correcta para su sistema.

ADVERTENCIA: Se sabe que la versión Sun de tar produce problemas, por lo que en su lugar debe utilizar la versión GNU.

Una vez completados estos pasos, se creará un nuevo directorio en función del nombre de la distribución que instale, como se indica a continuación:

```
% ls -l my*  
total 1  
drwxr-xr-x 13 mysql users 1024  
Jul 1 14:15 mysql-max-4.x.x-platform-os-extra
```

Es un nombre poco práctico para su utilización diaria, por lo que debe crear un enlace simulado `mysql` que apunte al nuevo directorio de forma que `/usr/local/mysql/` sea la ruta a MySQL:

```
% ln -s mysql-max-4.x.x-platform-os-extra mysql  
% ls -l my*  
lrwxrwxrwx 1 root root 40  
Jul 27 23:07 mysql -> mysql-max-4.x.x-platform-os-extra
```

El directorio recién instalado contiene lo siguiente:

```
% cd mysql  
% ls -l  
total 4862  
-rw-r--r-- 1 mysql users 19106 Jul 1 14:06 COPYING  
-rw-r--r-- 1 mysql users 28003 Jul 1 14:06 COPYING.LIB  
-rw-r--r-- 1 mysql users 122323 Jul 1 13:16 ChangeLog
```

```

-rw-r--r-- 1 mysql users 6808 Jul 1 14:06 INSTALL-
BINARY
-rw-r--r-- 1 mysql users 1937 Jul 1 13:16 README
drwxr-xr-x 2 mysql users 1024 Jul 1 14:15 bin
-rwxr-xr-x 1 mysql users 773 Jul 1 14:15 configure
drwxr-x- 4 mysql users 1024 Jul 1 14:15 data
drwxr-xr-x 2 mysql users 1024 Jul 1 14:15 include
drwxr-xr-x 2 mysql users 1024 Jul 1 14:15 lib
drwxr-xr-x 2 mysql users 1024 Jul 1 14:15 man
-rw-r--r-- 1 mysql users 2508431 Jul 1 14:06 manual.html
-rw-r--r-- 1 mysql users 2159032 Jul 1 14:06 manual.txt
-rw-r--r-- 1 mysql users 91601 Jul 1 14:06
manual-toc.html
drwxr-xr-x 6 mysql users 1024 Jul 1 14:15 mysql-
test
drwxr-xr-x 2 mysql users 1024 Jul 1 14:15 scripts
drwxr-xr-x 3 mysql users 1024 Jul 1 14:15 share
drwxr-xr-x 7 mysql users 1024 Jul 1 14:15 sql-bench
drwxr-xr-x 2 mysql users 1024 Jul 1 14:15 support-
files
drwxr-xr-x 2 mysql users 1024 Jul 1 14:15 tests

```

Ya se ha instalado MySQL. Para instalar las tablas de permisos (como vimos en un capítulo anterior), ejecute la secuencia de comandos `mysql_install_db`:

```

% scripts/mysql_install_db
Preparing db table
Preparing host table
Preparing user table
Preparing func table
Preparing tables-priv table
Preparing columns-priv table
Installing all prepared tables
020701 23:19:07 ./bin/mysqld: Shutdown Complete

```

Tras ello, cambie el propietario para garantizar que MySQL y el directorio de datos están bajo el control del recién creado usuario MySQL, en caso de que no sea así:

```

% chown -R root /usr/local/mysql
% chgrp -R mysql /usr/local/mysql
% chown -R mysql /usr/local/mysql/data

```

Seguidamente, MySQL está preparado para ejecutarse con `mysqld_safe`:

```

% /usr/local/mysql/bin/mysqld_safe -user=mysql &

```

TRUCO: No olvide asignar una contraseña raíz con `mysqladmin` al terminar la instalación; en caso contrario, MySQL estará disponible para todo el mundo en el sistema. Al mismo tiempo, debería definir el archivo de configuración justo después del proceso de instalación.

En la tabla 15.1 encontrara un repaso de los contenidos de los nuevos directorios que se acaban de crear.

Instalacion de una distribución binaria (rprn) en Unix

Red Hat Linux tambien le permite instalar MySQL desde un archivo RPM. En la tabla 15.4 se incluye la lista completa de archivos RPM disponibles (los números de version reflejan la version que este utilizando)

Tabla 15.4. Archivos RPM

Archivo	Descripción
MySQL-4.x.x-platform-os-extra.rpm	El software servidor MySQL, necesario a menos que simplemente se conecte a un servidor existente.
MySQL-client-4.x.x-platform-os-extra.rpm	El software cliente de MySQL, necesario para conectarse a un servidor MySQL.
MySQL-bench-4.x.x-platform-os-extra.rpm	Distintas pruebas y análisis comparativos de MySQL. Se necesitan los archivos Perl y <code>mysql.mysql</code> rpm.
MySQL-devel-4.x.x-platform-os-extra.rpm	Distintas bibliotecas y archivos necesarios para compilar otros clientes MySQL.
MySQL-shared-4.x.x-platform-os-extra.rpm	Bibliotecas compartidas cliente MySQL
MySQL-embedded-4.x.x-platform-os-extra.rpm	El servidor MySQL incrustado.
MySQL-4.x.x-platform-os-extra.src.rpm	El código fuente de los archivos rpm anteriores. No es necesario si realiza una instalación binaria.
MySQL-Max-4.x.x-platform-os-extra	El rpm Max de MySQL (compatible con tablas InnoDB, etc.).

Para instalar los archivos rpm, basta con ejecutar la utilidad rpm con cada uno de los rpm que quiera instalar. Normalmente lo minimo que debe instalar es el cliente y el servidor:

```
% rpm -i MySQL-4.x.x-platform-os-extra.rpm MySQL-client-4.x.x-platform-os-extra.rpm
```

La instalacion a traves de rpm genera una estructura ligeramente distinta que la que se obtiene por medio de una instalacion binaria convencional. Los datos se

almacenan en el directorio `/var/lib/mysql` y, al añadir entradas al directorio `/etc/rc.d/`, así como al crear una secuencia de comandos `/etc/rc.d/INIT.d/mysql`, MySQL se configura para que empiece automáticamente después del inicio. Debe prestar especial atención si realiza una instalación **sobre** otra anterior de esta forma, ya que tendrá que rehacer todos los cambios que haya efectuado.

Tras ello, MySQL se puede ejecutar con `mysqld_safe`:

```
% /usr/local/mysql/bin/mysqld_safe -user=mysql &
```

No olvide asignar una contraseña raíz y revisar su **archivo** de configuración.

Instalación desde código fuente en Unix

Es poco probable que quiera compilar MySQL desde código fuente para producción a menos que tenga experiencia a este respecto. Pero puede que haya alguien que disfrute del **desafío**, por lo que en este **apartado** describiremos detalladamente el proceso.

TRUCO: Es muy importante que consulte la última documentación de MySQL ya que la siguiente información se actualiza con gran rapidez y aparecen nuevas distribuciones para sustituir a las anteriores.

Para realizar una instalación desde una distribución de código fuente, necesita lo siguiente:

- `gunzip` (`gzip` o `zcat`). La versión GNU funciona.
- `tar` (`cl tar` GNU funciona, pero el `tar` de **Solaris** ha causado problemas en el **pasado**).
- `make` (le recomendamos utilizar el `make` de GNU, los de **Solaris** y **FreeBSD** causan problemas).
- `gcc` o `pgcc` (u otro compilador C++ ANSI). Actualmente le recomendamos la versión 2.95.2, aunque las distintas distribuciones se compilan con diferentes compiladores. Le sugerimos que consulte la **documentación** MySQL más reciente para ver que compilador corresponde a su sistema operativo y si se conoce algún problema con otros compiladores.

Para instalar desde código fuente, siga **los** pasos descritos a **continuación**:

- I. Para empezar, igual que con las instalaciones binarias, cambie a usuario raíz y añada el usuario y el grupo MySQL:

```
% su -  
Password:  
% groupadd mysql  
% useradd -g mysql mysql
```

2. Cambie al directorio en el que quiera almacenar los archivos (por ejemplo, `/usr/local/src` o el que resulte apropiado).

3. Descomprima los archivos:

```
% gunzip -c /tmp/mysql-4.x.x-platform-os-extra.tar.gz | tar
-xf -
```

4. Tras ello, se habrá creado un nuevo directorio. Cambie a este directorio, desde el que debe configurar y generar MySQL:

```
% cd mysql-4.x.x-extra
```

5. Ejecute la secuencia de comandos `configure`, que viene incluida en la distribución y le permite configurar distintas opciones de instalación. Hay un gran número de opciones disponibles. Algunas de las más útiles las describiremos más adelante, otras las mencionaremos en un apartado posterior.

6. De forma predeterminada, MySQL compilado desde código fuente se instala en `/usr/local` y los archivos de datos y de registro se almacenan en `/usr/local/var`. Para cambiar estas ubicaciones, utilice la opción `prefix`, como se muestra en el ejemplo:

```
% ./configure --prefix=/usr/local/mysql
```

De esta forma se cambia el prefijo de toda la instalación por `/usr/local/mysql`. Por otra parte, también puede cambiar la ubicación del directorio de datos por `/usr/local/mysql/data`, mientras conserva el resto de la instalación, como se muestra a continuación:

```
% ./configure --prefix=/usr/local \
    --localstatedir=/usr/local/mysql/data
```

Si lo prefiere, puede realizar alguna de estas acciones:

- Si no quiere compilar el servidor sino solamente los programas cliente para conectarse a un servidor existente, utilice la opción `--without-server`:

```
% ./configure --without-server
```

- Para utilizar `libmysqld.a`, la biblioteca MySQL incrustada, necesita la opción `--with-embedded-server`:

```
% ./configure --with-embedded-server
```

- Para cambiar la ubicación predeterminada del archivo de socket (normalmente `/tmp`), utilice `configure` de esta forma (el nombre de ruta debe ser absoluto):

```
% ./configure --with-unix-socket-path=/usr/local/sockets/
mysql.sock
```

- Para obtener el conjunto completo de opciones disponibles, ejecute lo siguiente:

```
% ./configure --help
```

7. Una vez completadas estas operaciones (que pueden llevar su tiempo, en función de su configuración), tendrá que generar los binarios con el comando `make`:

```
% make
```

8. Tras ello, debe instalarlos:

```
% make install
```

9. Seguidamente, prosiga como si hubiera instalado un binario: cree las tablas de permisos y cambie la propiedad de los archivos. Los siguientes ejemplos asumen que se ha decidido por el prefijo `/usr/local/mysql`:

```
% cd /usr/local/mysql
% scripts/mysql_install_db
Preparing db table
Preparing host table
Preparing user table
Preparing func table
Preparing tables-priv table
Preparing columns-priv table
Installing all prepared tables
010726 19:40:05 ./bin/mysqld: Shutdown Complete
% chown -R root /usr/local/mysql
% chgrp -R mysql /usr/local/mysql
% chown -R mysql /usr/local/mysql/data
```

10. Tras ello, ya puede ejecutar MySQL con `mysqld-safe`:

```
% /usr/local/mysql/bin/mysqld_safe --user=mysql &
```

No olvide asignar una contraseña raíz y comprobar su archivo de configuración.

Compilación óptima de MySQL

Las distribuciones estándar de MySQL se aproximan bastante a una compilación óptima, pero si quiere obtener el mayor rendimiento posible, tendrá que realizar algunas mejoras. También es muy fácil realizar lo contrario y frenar el proceso por lo que debe prestar especial atención al utilizar los siguientes consejos:

- Puede definir indicadores o el nombre del compilador utilizado por el compilador, como por ejemplo:

```
% CFLAGS=-O3
```



```
% CXX=gcc
% CXXFLAGS=-O3
% CC=gcc
% export CC CFLAGS CXX CXXFLAGS
```

- **Vincule de forma estática, no dinámica (es decir, utilice la opción `-static`). Utiliza más espacio de disco pero se ejecuta a más velocidad (13 por ciento en Linux de acuerdo a las mediciones MySQL).**
- Los **binarios MySQL** se suelen compilar con **gcc** porque **pgcc (gcc Pentium)** causa problemas en procesadores que no son Intel. La **compilación** con **pgcc** si su procesador es **de la familia Pentium de Intel** puede tener alguna ventaja (1 por ciento **según pruebas MySQL** y con una mejora de hasta un 10 por ciento). Del mismo modo, **en un servidor Sun, el compilador SunPro C++ es aproximadamente un 5 por ciento más rápido que gcc.**
- **Optimice hasta el máximo nivel posible (-O3 con gcc).**
- **Compile sin depurar (la opción `-without-debug`). Se ejecuta entre un 20 y un 35 por ciento más rápido que si utiliza la opción `-with-debug-full` y aproximadamente un 11 por ciento más rápido que si utiliza `-with-debug`.**
- **Las distribuciones de MySQL son compatibles con todos los conjuntos de caracteres. Utilice la opción `-with-extra-charsets=none` si emplea el conjunto de caracteres predeterminado ISO-8859-1 (Latin1). Utilice la opción `-with-charset=xxx` para compilar únicamente los conjuntos de caracteres que quiera utilizar.**
- **Si ejecuta Linux en equipos x86, la compilación sin punteros de marco (`-fomit-frame-pointer` o `-fomit-frame-pointer-ffixed-ebp`) genera una mejora de entre un 1 y un 4 por ciento.**

Instalacion de varios servidores en el mismo equipo

Existen diversas razones para ejecutar varios servidores MySQL en el mismo equipo. No obtendra una mejora del rendimiento y es muy probable que no quiera permitir que las distintas versiones accedan a los mismos datos. Tambien, es probable que lo haga para probar una nueva version de MySQL sin eliminar la instalacion anterior. Si tiene pensado ejecutar varias versiones de MySQL en el mismo equipo, debe asegurarse de que no intentan utilizar el mismo archivo de socket o que escuchan por el mismo puerto TCP/IP. También deben contar con su propio archivo `pid`. Los valores predeterminados son el puerto 3306 y `/tmp/mysql.sock` en la mayoría de los sistemas.

Una forma muy aconsejable de gestionarlo es por medio de la utilidad `mysqld-multi`, que veremos mas adelante. Puede cambiar el puerto predeterminado y los parametros TCP/IP en el archivo de configuracion siempre que se trate de un archivo de configuracion distinto que el del otro servidor. Por ejemplo:

```
socket=/tmp/mysql2.sock
port=3307
```

Por medio de la opción `-socket`, los clientes pueden conectarse a servidores que se ejecuten en un socket diferente:

```
% mysql -socket=/tmp/mysql2.sock -uroot -pg00r002b
```

Tambien puede especificar el servidor al que hay que conectarse si indica el archivo de configuracion que debe utilizar el cliente. Por ejemplo:

```
% mysql -defaults-file=/usr/local/mysql2/etc/my.cnf -uroot -
pg00r002b
```

Si compila personalmente MySQL, configure el segundo servidor con otro numero de puerto, otra ruta de socket y otro directorio de instalacion. Por ejemplo:

```
% ./configure --with-tcp-port=3307 \
               --with-unix-socket-path=/tmp/mysql2.sock \
               --prefix=/usr/local/mysql2
```

ADVERTENCIA: Nunca debe haber mas de un servidor que controle los mismos datos, lo que constituye un evidente **riesgo de daños**. **Tampoco** debe escribir **los** mismos **archivos** de registro.

Las distribuciones MySQL incorporan una utilidad denominada `mysqld-multi`, una herramienta muy util para gestionar varios servidores MySQL (que se ejecuten en distintos puertos y sockets). Para utilizar `mysqld-multi`, debe configurar su archivo de configuracion con una seccion `mysqld-multi`, asi como con secciones para cada servidor MySQL que ejecute. Por ejemplo:

```
[mysqld-multi]
mysqld      = /usr/local/bin/mysqld_safe
mysqladmin = /usr/local/bin/mysqladmin
user        = root
password    = g00r002b
[mysqld1]
socket      = /tmp/mysql.sock
port        = 3306
pid-file     = /usr/local/mysql/var/hostname.pid
datadir     = /usr/local/mysql/var
language    = /usr/local/share/mysql/english
user        = hartmann
```

```

[mysqld2]
socket      - /tmp/mysql.sock2
port        - 3307pid-file  = /usr/local/mysql/var2/
hostname.pid
datadir     - /usr/local/mysql/var2
language    - /usr/local/share/mysql/french
user        - yves
[mysqld3]
socket      - /tmp/mysql.sock3
port        = 3308
pid-file    = /usr/local/mysql/var3/hostname.pid
datadir     - /usr/local/mysql/var3
language    - /usr/local/share/mysql/german
user        - cleo
[mysqld4]
socket      - /tmp/mysql.sock4
port        - 3309
pid-file    - /usr/local/mysql/var4/hostname.pid
datadir     - /usr/local/mysql/var4
language    - /usr/local/share/mysql/english
user        = caledon

```

La sintaxis de `mysqld-multi` es la siguiente:

```

mysqld-multi [opción/opciones] {start|stop|report}
[grupo_número, grupo_número2...]

```

Tomando la configuración del archivo que acabamos de ver, en el siguiente ejemplo `mysqld-multi` informa del estado del servidor y, tras ello, se utiliza para cerrarlo:

```

% mysqld-multi -user=root -password=g00r002b report 1
Reporting MySQL servers
MySQL server from group: mysqld1 is running
% mysqld-multi -user=root -password=g00r002b stop 1
% 020729 04:20:50 mysqld ended
% mysqld_multi -user=root -password=g00r002b report 1
Reporting MySQL servers
MySQL server from group: mysqld1 is not running

```

En la tabla 15.5 se describen las opciones de `mysqld-multi`.

Tabla 15.5. Opciones de `mysqld_multi`

Opción	Descripción
<code>-config-file=...</code>	Define un archivo de configuración distinto para los grupos (no afecta al grupo <code>[mysqld-multi]</code>).
<code>-example</code>	Proporciona un archivo de configuración de muestra.
<code>-help</code>	Muestra la ayuda y sale.

Opción	Descripción
<code>-log=..</code>	Especifica el archivo de registro, tomando la ruta y el nombre completo del archivo. Si este archivo ya existe, los registros se adjuntan al final del archivo.
<code>-mysqladmin=...</code>	Ruta completa del binario <code>mysqladmin</code> , utilizado para cerrar el servidor.
<code>-mysqld=...</code>	Ruta completa y nombre de la biblioteca <code>mysqldadmin</code> que se va a utilizar o, mas a menudo, el binario <code>mysqld-safe</code> . Las opciones se pasan a <code>mysqld</code> . Tendra que cambiar <code>mysqld-safe</code> o asegurarse de que se encuentra en su variable de entorno <code>PATH</code> .
<code>-no-log</code>	Se escribe en la salida estandar en lugar de en un archivo de registro. La opción predeterminada es el archivo de registro.
<code>-password=...</code>	La contraseña del usuario de <code>mysqladmin</code> .
<code>-tcp-ip</code>	Hace que <code>mysqld_multi</code> se conecte a los servidores MySQL a través de TCP/IP en lugar de utilizar un socket Unix. De forma predeterminada, la conexión se realiza por medio de un socket en Unix.
<code>-user=...</code>	El usuario de <code>mysqladmin</code> . Asegurese de que este usuario tiene los privilegios adecuados para realizar lo que necesite (<code>shutdown-priv</code>).
<code>-versión</code>	Muestra el numero de version y sale.

Como evitar problemas de instalacion comunes

Cuando todo sale mal, puede resultar extremadamente frustrante, sobre todo si todavia no ha empezado. Es muy complicado, especialmente para los usuarios sin experiencia, saber que hacer cuando se enfrentan a un extraño mensaje de error (si tienen la suerte de obtener uno).

En el siguiente apartado veremos algunos de los problemas de instalacion mas comunes.

Problemas al iniciar `mysqld`

Los problemas al iniciar `mysqld` se hacen evidentes al intentar instalar sin exito las tablas de permisos. Existen diversas razones para estos problemas, como vera en la siguiente lista, y el examen del registro de errores es la mejor forma de saber cuál es el problema.

- Puede que tenga un problema en su archivo de **configuración** (my. **cnf** o my. **ini**). Revise atentamente la **sintaxis** o utilice el archivo de configuración estandar incluido en su **distribución** para ver si puede indicar MySQL.
- Otro error muy habitual es este:

```
Can't start server: Bind on unix socket...
```

o el siguiente:

```
Can't start server: Bind on TCP/IP port: Address already in use
```

Este error se produce cuanto intenta instalar una segunda copia de MySQL en el mismo puerto o socket de una instalacion existente. Asegurese de especificar un puerto o socket distinto antes de empezar.

- Los problemas relacionados con los permisos tambien son habituales. Compruebe que ha seguido los pasos descritos en la **sección** de instalacion para que **al menos sean correctos los directorios** MySQL. Si utiliza sockets, tendra que verificar si tiene **permiso** para escribir tambien el archivo de socket (normalmente en /tmp).
- Otro de los problemas habituales se produce con las bibliotecas. Por ejemplo, si utiliza Linux y ha instalado bibliotecas compartidas, asegurese de que la ubicacion de las mismas se enumera en su archivo `/etc/ld.so.conf`. Por ejemplo, si tiene:

```
/usr/local/lib/mysql/libmysqlclient.so
```

Asegurese de que `/etc/ld.so.conf` contiene:

```
/usr/local/lib/mysql
```

Y ejecute `ldconfig`.

- Al iniciar MySQL, si hay tablas **BDB**, puede encontrar un problema como el siguiente:

```
020814 19:18:02 bdb: warning: ./bdb/news.db: No such file
or directory
020824 19:18:02 Can't init databases
```

Esto significa que **BDB** tiene problemas para recuperar un archivo de registro existente. Puede iniciar MySQL con la **opción** `-bdb-no-recover` o desplazar los archivos de registro.

Problemas de compilacion

Si tiene problemas al realizar la compilacion y debe realizarla una segunda ocasion, tendra que asegurarse de que **configure** se ejecuta desde cero; en caso contrario, utilizara **información** de su instancia anterior, almacenada en el

archivo `config.cache`. Tendrá que eliminar este archivo **cada vez** que realice la configuración. Del mismo modo, los **archivos** de objetos antiguos pueden seguir presentes y, para garantizar una nueva compilación limpia, tendrá que eliminarlos. Ejecute lo siguiente:

```
% rm config.cache
% make clean
```

También puede ejecutar `distclean` si lo tiene.

Puede que su compilador este obsoleto. En la actualidad, MySQL sugiere la **utilización** de `gcc 2.95.2` o `egcs 1.0.3a`, pero es muy probable que haya cambiado, por lo que le aconsejamos que consulte la última **documentación**. Otros problemas pueden aparecer como resultado de una versión de `make` incompatible. Actualmente, MySQL recomienda `make GNU`, versión 3.75 o superior.

Si obtiene un error al compilar `sql_yacc.cc`, puede que no tenga suficiente espacio en el disco. En algunos **casos**, la **compilación** de este archivo utiliza demasiados recursos (incluso cuando parece que hay multitud de ellos **disponibles**). El error puede ser uno de los siguientes:

```
Internal compiler error: program cclplus got fatal signal 11
Out of virtual memory
Virtual memory exhausted
```

Al ejecutar `configure` con la opción `-low-memory` se suele solucionar este problema:

```
% ./configure --with-low-memory
```

Si tiene problemas con las bibliotecas relacionadas, como `g++`, `libg++` o `libstdc++` (puede que no estén disponibles), intente configurar `gcc` para que sea su compilador C++, como se indica a **continuación**:

```
% CXX="gcc -O3" ./configure
```

La vinculación estática, **además** de ser más indicada, también puede resolver problemas relacionados con referencias sin definición.

Problemas de Windows

Si pulsa dos veces sobre `setup.exe` y el proceso se inicia pero nunca termina, puede que haya algo que interfiera con MySQL. Intente uno de estos procedimientos:

- Cierre todas las aplicaciones de Windows, incluyendo los servicios y las de la bandeja del sistema.
- Por otro parte, intente la instalación en **modo seguro** (pulse **F8** al iniciar y seleccione esta opción en el menú).

- En el peor de los casos tendrá que reinstalar Windows e instalar primero MySQL, antes de ningún otro programa. Es poco probable que se produzca este problema en equipos de producción dedicados como servidores de bases de datos MySQL. Suelen aparecer en equipos multifunción en los que se ejecutan todo tipo de aplicaciones.

Actualización de MySQL 3.x a MySQL 4

MySQL ha sufrido importantes mejoras de desarrollo de la versión 3.23.xx a la versión 4 y existen numerosas diferencias a las que debe prestar atención a la hora de realizar la actualización:

- El script `mysqld_safe` sustituya `safe_mysqld`.
- Existe gran cantidad de nuevos privilegios en la tabla de usuarios (en la base de datos `mysql`). MySQL proporciona una secuencia de comandos para añadir estos nuevos permisos a la vez que conserva los existentes. Se denomina `mysql_fix_privilege_tables` y adopta los privilegios `REPLICATION SLAVE` y `REPLICATION CLIENT` del antiguo privilegio `FILE`, y los privilegios `SUPER` y `EXECUTE` del antiguo `PROCESS`. Sin ejecutar esta secuencia de comandos, todos los usuarios tendrán privilegios `SHOW DATABASES`, `CREATE TEMPORARY TABLES` y `LOCK TABLES`.
- Los atributos de `length` y `max length` (en la estructura `MYSQL_FIELD`) son ahora `unsigned long` en lugar de `unsigned int`.
- La antigua opción `-safe-show-database` ha quedado obsoleta (ya no hace nada, puesto que se ha reemplazado por el privilegio `SHOW DATABASES` en la tabla de usuarios).
- Se ha cambiado el nombre de una serie de variables:

```
mysam-bulk-insert-tree-size por bulk-insert-buffer-size
query-cache-startup-type por query-cache-type
record-buffer por read-buffer-size
record-rnd-buffer por read_rnd_buffer_size
sort-buffer por sort-buffer-size
warnings por log-warnings
```

- Algunas opciones de inicio de `mysqld` tienen un nuevo nombre:


```
-skip-locking por -skip-external-locking
-enable-locking por -external-locking
```
- El tamaño de los parámetros de inicio `mysam_max_extra_sort_file` y `mysam_amx_extra_sort` se proporciona ahora en bytes, no en megabytes.

- Algunas opciones de inicio han quedado obsoletas (por el momento siguen funcionando):

```
record-buffer
sort-buffer
warnings
```

- El bloqueo externo aparece ahora desactivado de forma predeterminada.
- Las siguientes variables SQL tienen un nuevo nombre (los nombres antiguos siguen funcionando, pero han quedado obsoletos):

```
SQL_BIG_TABLES por BIG-TABLES
SQL_LOW_PRIORITY_UPDATES por LOW-PRIORITY-UPDATES
SQL_MAX_JOIN_SIZE por MAX-JOIN-SIZE
SQL_QUERY_CACHE_TYPE por QUERY-CACHE-TYPE
```

- SIGNED es un palabra reservada.
- Las columnas de tipo DOUBLE y FLOAT ya no ignoran el indicador UNSIGNED.
- Las columnas BIGINT ahora almacenan enteros con mayor eficacia que las cadenas.
- El comportamiento predeterminado de la función STRCMP () distingue ahora entre mayúsculas y minúsculas ya que utiliza el conjunto de caracteres predeterminado cuando realiza comparaciones.
- TRUNCATE TABLE es más rápido que DELETE FROM nombre_de_la_tabla ya que no devuelve el número de filas eliminadas.
- Las funciones LOCATE () e INSTR () distinguen ahora entre mayúsculas y minúsculas si uno de los argumentos es una cadena binaria.
- Ahora, cuando se le pasa una cadena a la función HEX (), todos los caracteres se convierten en dos dígitos hexadecimales.
- La instrucción SHOW INDEX tiene dos columnas adicionales: Null e Index_type.
- Ya no se pueden ejecutar las instrucciones TRUNCATE TABLE o DROP DATABASE cuando hay un bloqueo activo (ya sea de LOCK TABLES o de una transacción). En su lugar se devuelve un error.
- La cláusula ORDER BY nombre_de_columna DESC ordenará primero los valores NULL en todos los casos, mientras que antes lo hacía sin coherencia alguna.
- Los resultados de todas las operaciones de bits (<<, >>, |, &, -) son ahora sin firma, al igual que el resultado de la resta entre dos enteros, cuando uno de ellos no tenga firma (esto se puede desactivar si se inicia MySQL con la opción -sql-mode=NO_UNSIGNED_SUBTRACTION).

- Si quiere utilizar la instrucción `MATCH...AGAINST (...IN BOOLEAN MODE)` tendrá que volver a generar sus tablas con `ALTER TABLE nombre_de_tabla TYPE=MyISAM`. Esto se aplica incluso si las tablas ya son `ISAM`.
- Es necesario especificar una cláusula `IGNORE` cuando se utilice una instrucción de tipo `INSERT INTO...SELECT` o, en caso contrario, `MySQL` se detendrá y posiblemente realice una inversión.
- La función `RAND(seed)` ahora devuelve una serie de números aleatorios diferentes que antes (para distinguir `RAND(valor_inicial)` de `RAND(valor_inicial+1)`).
- El formato de `SHOW OPEN CHANGES` ha cambiado.
- Ya no se admiten las antiguas funciones del API `C mysql_drop_db()`, `mysql_create_db()` y `mysql_connect()`. Puede compilar `MySQL` con la opción `CFLAGS=-DUSE_OLD_FUNCTIONS`, pero debe actualizar sus clientes para que utilicen el API de la versión 4.0.
- Si utiliza el módulo `DBD::mysql` de Perl, tendrá que utilizar una versión más reciente que 1.2218, ya que las versiones anteriores utilizaban la antigua función `drop-db()`.
- En lugar de utilizar `SET SQL_SLAVE_SKIP_COUNTER=#`, tendrá que emplear `SET GLOBAL SQL_SLAVE_SKIP_COUNTER=#`.
- Los clientes multiprocesamiento deben utilizar las funciones `mysql_thread_init()` y `mysql_thread_end()`.

Resumen

La instalación de `MySQL` no es complicada. Si opta por una distribución binaria, la más recomendable en la mayoría de los casos, basta con apuntar y pulsar en Windows, y utilizar algunos sencillos comandos en Unix. También hay razones válidas para optar por una distribución de código fuente, si `MySQL` todavía no está disponible en formato binario para su plataforma o si necesita obtener el mayor rendimiento de `MySQL` compilándolo de forma óptima.

Sea cual fuere la razón, con la correcta manipulación de las opciones de compilación, podrá obtener una instalación rápida pero estable de `MySQL`.

Las bases de datos suelen aumentar hasta alcanzar tamaños considerables y una excesiva **cantidad** de datos en una unidad significa que esta y el controlador de la misma se convierten en un preocupante **cuello** de botella. El uso de **múltiples** discos y controladores (por medio de RAID) es uno de los métodos para solucionar este problema y la **mayoría** de las modalidades de RAID disponen de opciones de redundancia adicional.

Los enlaces simbólicos **permiten** dividir bases de datos o tablas entre varios discos sin recurrir a RAID. Al crear un enlace de una unidad a otra, se **libera** el **cuello** de botella de la unidad más congestionada.

En este capítulo nos centraremos en los siguientes aspectos:

- Uso de RAID
- Uso de enlaces simbólicos

Significado de RAID

RAID significa *matriz redundante de discos económicos*, no *matriz redundante de discos independientes*. Es sorprendente la rapidez con la que una fuente

erronea se duplica en Internet, aunque el termino *independiente* no es del todo equivocado. El termino original proviene de un informe escrito en 1987 por los investigadores David Patterson, Garth Gibson y Randy Kantz. Mejora el rendimiento y la tolerancia. Lo contrario se suele denominar SLED.

Existen distintos tipos de RAID, que describiremos en los siguientes apartados. Todos ellos se basan en tres conceptos basicos: *duplicado* (repetición de escritura en otra unidad), *division* (dispersion de datos por distintas unidades) y *paridad* (examen de bits para evitar y recuperarse de errores).

RAID 0

RAID 0 (en ocasiones denominado *division*, ya que realmente no es un tipo de RAID) divide los datos en bloques y los distribuye por distintas unidades. El tamaño de los bloques es el mismo en todas las unidades, pero se pueden definir tamaños de bloque distintos en funcion de las circunstancias. Esto permite aumentar el rendimiento, ya que uno de los principales cuellos de botella desplaza la cabeza de la unidad. RAID 0 aumenta las probabilidades de solicitudes simultáneas de datos en distintas unidades, lo que significa que la lectura se produce al mismo tiempo sin tener que esperar a que termine una, se vuelva a situar la cabeza y, tras ello, se lea el segundo grupo de datos.

Por lo general, cuantas mas unidades haya, mejor sera el rendimiento. El rendimiento mejora incluso mas si cada unidad tiene su propio controlador, aunque no es imprescindible. Basta con asegurarse de que los controladores pueden procesar la carga si esta es responsable de varias unidades. Sin embargo, RAID 0 no permite la tolerancia a fallos. De hecho, aumenta las posibilidades de que se produzcan fallos, ya que hay mas de una unidad que puede fallar y cancelar la disponibilidad de los datos. No es aconsejable utilizar RAID 0 en entornos en los que la disponibilidad de los datos sea fundamental. Necesitara al menos dos unidades para implementar RAID 0. Realmente no es un tipo de RAID y en ocasiones se le denomina, en tono humoristico, AID, ya que no contiene ninguna forma de redundancia (la R de RAID).

En la figura 16.1, que muestra la implementación de RAID 0 en tres dispositivos, el primer bloque de datos se escribe en el dispositivo A, el segundo en el B y el tercero en el C. El siguiente bloque se escribira de nuevo en el primer dispositivo, y asi sucesivamente. El primer y tercer bloque de datos se pueden leer al mismo tiempo, ya que existen en distintas unidades.

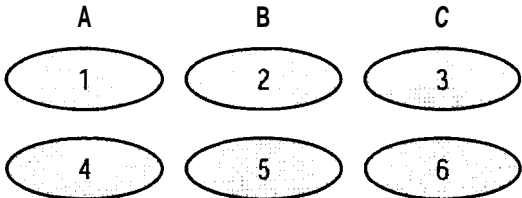


Figura 16.1. RAID 0

RAID 1

RAID 1 (también denominado *duplicación*) se utiliza cuando lo que se escribe en una unidad se repite en otra. Esto **mejora** la tolerancia a fallos ya que existe una copia de seguridad actualizada en caso de que falle una unidad. Los fallos de unidad son los fallos de hardware más habituales y RAID 1 nos protege de este tipo de fallos. El rendimiento de escritura es escaso, ya que hay varias escrituras simultáneas, aunque las lecturas son ligeramente más **rápidas**, ya que se puede acceder a varias unidades. Necesitará **al menos** dos unidades para implementar RAID 1.

En la figura 16.2 vemos la implementación de RAID 1 en dos unidades; cada una contiene una copia idéntica.

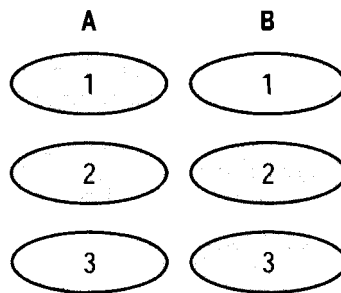


Figura 16.2. RAID 1

RAID 2 y RAID 3

RAID 2 es una versión que apenas se utiliza que **emplea** los códigos de conexión de error Hamming (que utiliza 3 bits de una **palabra** de 7 bits para comprobar y corregir los errores) y es particularmente útil en unidades que no realizan ningún tipo de **protección** contra errores (como las unidades SCSI).

RAID 3 es igual que RAID 0 a **excepción** de que también define una unidad dedicada para errores de conexión, lo que proporciona un cierto nivel de **tolerancia** a fallos. Los datos se dividen en el nivel de bits entre las distintas unidades. Otra unidad almacena los datos de **paridad**. La **paridad** se determina durante la escritura y se comprueba durante la lectura. La **información** de paridad **permite** la **recuperación** en caso de que falle una unidad. Necesitará **al menos** tres unidades para implementar RAID 3. Normalmente requiere la implementación de hardware para que resulte de utilidad. Las lecturas y escrituras de pequeño tamaño son **rápidas**, pero los bloques de datos de gran tamaño requieren que los datos se lean desde todas las unidades, lo que significa que el rendimiento es tan lento como en una sola unidad.

En la figura 16.3 podemos ver cómo se utiliza RAID 3 para dividir los datos entre dos unidades (A y B) con una tercera que se utiliza para almacenar la **paridad**.

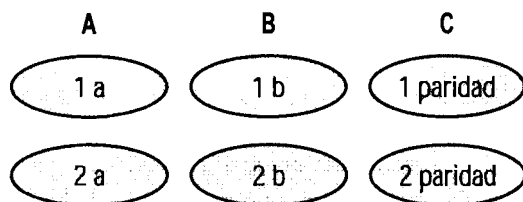


Figura 16.3. RAID 3

RAID 4

RAID 4 es similar a RAID 3 a excepción de que la división se realiza en el nivel de bloques, no en el de bytes. Las lecturas **menores** del tamaño de un bloque son mas **rápidas** (y generalmente aumenta la velocidad cuando se añade una nueva unidad). Al igual que ocurre con RAID 3, se necesitan **al menos** tres unidades y los datos de **paridad** siempre **permiten** la **recuperación** en caso de que falle una unidad. La unidad de **paridad** puede convertirse en un **cuello de botella**, pero RAID 5 puede solucionar este problema.

En la figura 16.4 vemos como se utiliza RAID 4 para dividir los datos entre tres unidades (A, B y C) con una cuarta (D) que se utiliza para almacenar la **paridad**.

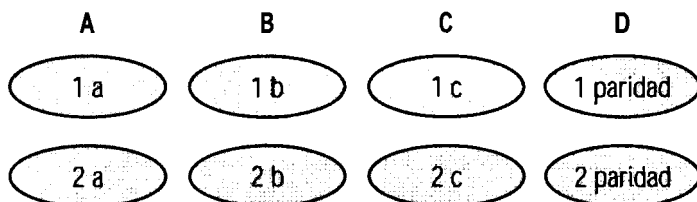


Figura 16.4. RAID 4

RAID 5

RAID 5 también **permite** la división, así como la de datos de **corrección** de errores, lo que mejora **tanto** el rendimiento como la tolerancia a fallos. Es similar a RAID 4, a excepción de que **los** datos de **paridad** se almacenan en cada una de las unidades. Las escrituras son mas **rápidas** que en RAID 4 (no hay **cuello de botella** de unidades) pero las lecturas son mas **lentas**, ya que la **información** de **paridad** ocupa espacio en cada unidad y debe ignorarse. RAID 5 es mas recomendable para servidores de bases de datos ya que aumenta la redundancia y el rendimiento.

Al menos se necesitan tres unidades para implementar RAID 5.

En la figura 16.5 podemos ver que RAID 5 se utiliza para dividir datos en tres unidades (A, B y C) y que cada una contiene datos de **paridad**.

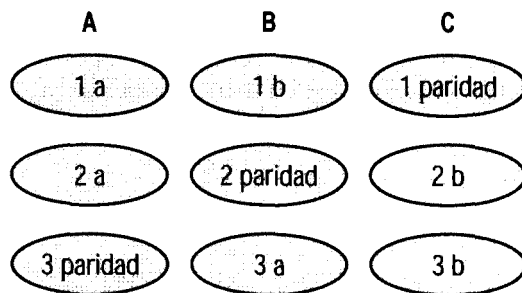


Figura 16.5. RAID 5

RAID 10

RAID 10 es una combinación de RAID 1 y RAID 0 (duplicación y división). Proporciona todas las ventajas de rendimiento de la división, así como la tolerancia a errores de la duplicación. Combina lo mejor de ambos, pero el coste es elevado. Requiere al menos cuatro unidades para su implementación.

En la figura 16.6 vemos la implementación de RAID 10 en cuatro unidades (A, B, C y D). A y B duplican los datos (los bloques de datos del uno al cuatro aparecen en ambas unidades) y las unidades C y D se encargan de acelerar el rendimiento mediante la división de los datos. RAID 10 es aconsejable para servidores de bases de datos, ya que proporciona el mayor nivel de mejora de rendimiento y tolerancia a fallos aunque supone un importante coste (por el número de unidades).

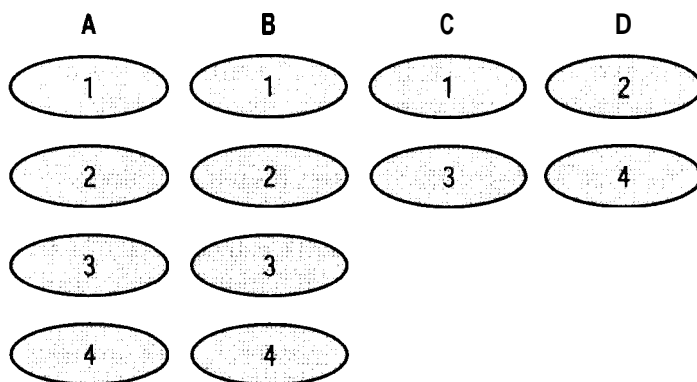


Figura 16.6. RAID 10

RAID 0+1

A menudo se confunde RAID 0+1 con RAID 10. Mientras que RAID 10 es una matriz dividida de unidades cuyos segmentos se duplican, RAID 0+1 es una matriz duplicada de unidades, cuyos segmentos se dividen. Generalmente, RAID

0+1 se selecciona cuando el rendimiento es una prioridad mayor que la fiabilidad y RAID 10 cuando la fiabilidad es mas importante que el rendimiento. RAID 0+1 es tambien **caro** y requiere **al menos** cuatro unidades para su implementacion. En la figura 16.7 se muestra como se implementa RAID 0+1 entre cuatro unidades (A, B, C y D).

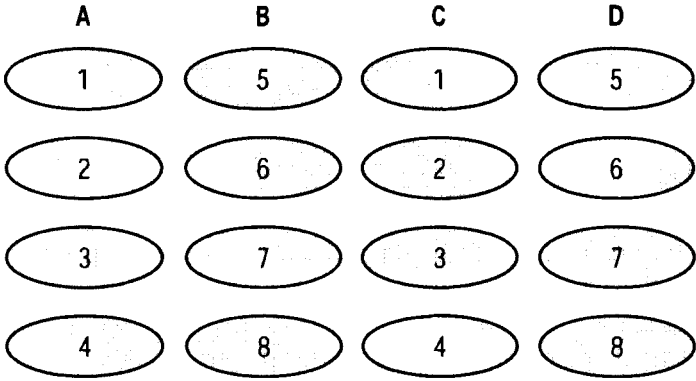


Figura 16.7. RAID 0+1

Otros tipos de RAID

Los anteriores tipos de RAID no son todos los que **existen**; hay otros, aunque algunos practicamente no se utilizan.

Puede implementar RAID por medio de hardware o de software (RAID de hardware y RAID de software) o mediante una **combinación** de ambos. La opcion `with-raid-option` de MySQL es una **forma limitada** de RAID de software (actualmente RAID 0). Su principal finalidad consiste en superar la **limitación del tamaño** de los archivos. Las versiones posteriores ampliaran su utilidad. La duplicacion, aunque no es estrictamente RAID, es otra opcion de software similar a RAID (RAID 1). RAID de hardware resulta mas sencillo de utilizar ya que una vez en funcionamiento, el dispositivo de hardware presenta multiples unidades como una sola ante el software y se encarga de toda la redundancia y la division, para que MySQL continúe como de costumbre. RAID de software utiliza el software (por ejemplo, `vinum`, una herramienta que se ejecuta en **FreeBSD** y que implementa RAID 0, RAID 1 y RAID 5), lo que en consecuencia tiene un efecto beneficioso en la CPU. Si tiene una CPU con ciclos libres, el RAID de software puede ser indicado.

Uso de enlaces simbolicos

Resulta muy sencillo utilizar enlaces simbolicos para mejorar el rendimiento de su base de datos y reducir la **latencia** del disco. La idea es que en lugar de almacenar todos los datos e indices en un disco, se crea un enlace simbolico desde

ese disco a otro, en el que realmente se almacenan los datos. Actualmente, sólo se pueden vincular simbólicamente las tablas y bases de datos MyISAM. Las primeras versiones de MySQL tenían problemas para vincular tablas simbólicamente (volvían a su ubicación original cuando se realizaban determinadas operaciones), pero en la versión 4 se han solucionado muchos de estos problemas.

Vinculación simbólica de bases de datos

Para crear un enlace simbólico para una base de datos MyISAM, siga los pasos descritos a continuación:

1. Cree el directorio de base de datos en la nueva ubicación.
2. Asegurese de que los permisos y la propiedad son correctos (700 y `mysql.mysql`).
3. Cree un enlace simbólico en el directorio de datos que apunte a la nueva ubicación.

A continuación, probaremos la creación de una nueva base de datos, `s_db`, que es la que vincularemos simbólicamente. Imaginemos que tiene un directorio, `disk2`, que es el disco secundario en el que quiere ubicar la base de datos. Cree los directorios en los que va a almacenar los datos (`/disk2/mysql/data/s_db`) y cambie los permisos y la propiedad, primero en un sistema Unix, como se indica a continuación:

```
% cd /disk2
% mkdir mysql
% mkdir mysql/data
% mkdir mysql/data/s_db
% chown mysql /disk2/mysql/data/s_db/
% chgrp mysql /disk2/mysql/data/s_db/
% chmod 700 /disk2/mysql/data/s_db/
```

Tras ello, de nuevo en el directorio de datos, cree el enlace simbólico:

```
% cd /usr/local/mysql/data
% ln -s /disk2/mysql/data/s_db s_db
```

Una vez creado, se creará su base de datos (recuerde que las bases de datos MyISAM son simples subdirectorios en el directorio de datos). Puede confirmarlo si se conecta a MySQL:

```
% /usr/local/mysql/bin/mysql -uroot -pg00r002b
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6202 to server version: 4.0.2-
alpha-max-log
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> SHOW DATABASES LIKE 's_db';
```



```

+-----+
| Database (s_db) |
+-----+
| s_db           |
+-----+
1 row in set (0.00 sec)

```

Seguidamente, para ver que esos datos se ubican en el segundo disco, cree y complete una pequeña tabla, como se indica a continuación:

```

mysql> USE s_db
Database changed
mysql> CREATE TABLE s1( f1 INT);
Query OK, 0 rows affected (0.23 sec)
mysql> INSERT INTO s1 VALUES(1);
Query OK, 1 row affected (0.03 sec)
Check that the new data has been created on the secondary disk:
mysql> exit
Bye
% ls -l /disk2/mysql/data/s_db/
total 14
-rw-rw- 1 mysql  mysql      5 Jul  8 02:26 s1.MYD
-rw-rw- 1 mysql  mysql    1024 Jul  8 02:26 s1.MYI
-rw-rw- 1 mysql  mysql    8550 Jul  8 02:25 s1.frm

```

Para crear un enlace simbólico para una base de datos en un sistema Windows, existen algunas diferencias. Los permisos son más sencillos y para crear un enlace simbólico, en lugar de utilizar `ln -s`, basta con crear un archivo de texto con la extensión `.sym`. En primer lugar, cree un archivo con el nombre `s2_db.sym` en su directorio de datos con el siguiente texto:

```
D:\s_db
```

Como esta operación se ha realizado en el interior de su directorio de datos, aparecerá en el mismo nivel que el resto de bases de datos **MyISAM** (en este caso, `firstdb`, `mysql` y `test`) y cuando se conecte a MySQL, lo verá como una base de datos existente:

```

C:\mysql\bin>dir ..\data\

Directory of C:\mysql\data
09/03/2002  08:58p      <DIR>
09/03/2002  08:58p      <DIR>
09/03/2002  08:31p      <DIR>
09/03/2002  08:22p      <DIR>
09/03/2002  08:23p      4,342 mysql.err
09/03/2002  08:57p      7 s2_db.sym
09/03/2002  08:22p      <DIR>
                2 File(s)          4,349 bytes

...
C:\mysql\bin>mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.

```

Your MySQL connection id is 8 to server version: 4.0.3-beta-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

```
mysql> SHOW DATABASES;
```

```
+-----+
| Database |
+-----+
| firstdb  |
| mysql    |
| s2_db    |
| test     |
+-----+
```

4 rows in set (0.00 sec)

```
mysql> USE s2_db
```

Database changed

```
mysql> CREATE TABLE s1(f1 INT);
```

Query OK, 0 rows affected (0.23 sec)

```
mysql> INSERT INTO s1 VALUES(1);
```

Query OK, 1 row affected (0.03 sec)

```
C:\mysql\bin>dir d:\s2_db
```

```
...
Directory of d:\s2_db
09/03/2002  09:38p        <DIR>
09/03/2002  09:38p        <DIR>
                                ..
09/03/2002  09:38p                8,550 s1.frm
09/03/2002  09:38p                5 s1.MYD
09/03/2002  09:38p            1,024 s1.MYI
                                3 File(s)          9,579 bytes
```

...

NOTA: Si el código anterior no funciona, puede que esté ejecutando una versión anterior de MySQL (en cuyo caso tendrá que añadir `-use-symbolic-links` a una línea de su archivo de configuración `my.ini`). También es posible que su versión de MySQL no se haya compilado con `-DUSE_SYMDIR`, en cuyo caso, los enlaces simbólicos no funcionarán. Normalmente, los servidores `mysql-max` y `mynsql-max-nt` se compilan con esta opción; sin embargo, debería consultar la última documentación.

Vinculación simbólica de tablas

El uso de tablas simbólicas en tablas individuales no es recomendable ya que hay algunas funciones que no funcionan correctamente con estas tablas (pero compruebe la última documentación ya que es probable que cambie pronto).

Las funciones que todavía no funcionan con tablas vinculadas simbólicamente son las siguientes:

- `BACKUP TABLE` y `RESTORE TABLE` (los enlaces simbólicos se perderán).

- mysqldump no almacena información sobre enlaces simbólicos en el volcado.
- ALTER TABLE (ignora las opciones INDEX/DATA DIRECTORY="ruta" CREATE TABLE).

Para vincular simbólicamente una tabla al crearla, es necesario utilizar la opción INDEX o DATA DIRECTORY PATH. DATA DIRECTORY crea un enlace simbólico para el archivo .MYD e INDEX DIRECTORY ubica el archivo .MY. En el siguiente ejemplo se ubica el archivo de datos de una nueva tabla en la base de datos `firstdb` en el directorio que creamos anteriormente:

```
mysql> USE firstdb;
Database changed
mysql> CREATE TABLE s_table (a int) DATA DIRECTORY =
  '/disk2/mysql/data/s_db';
Query OK, 0 rows affected (0.20 sec)
mysql> INSERT INTO s_table VALUES(1);
Query OK, 1 row affected (0.01 sec)
```

Puede comprobar que el archivo `.frm` (que contiene la estructura) se encuentra en el directorio de datos habitual y que el archivo de datos `.MYD` se encuentra en la nueva ubicación:

```
% cd /usr/local/mysql/data/firstdb/
% ls -l /disk2/mysql/data/s_db/
total 16
-rw-rw-- 1 mysql mysql    5 Jul  8 02:26 s1.MYD
-rw-rw-- 1 mysql mysql  1024 Jul  8 02:26 s1.MYI
-rw-rw-- 1 mysql mysql  8550 Jul  8 02:25 s1.frm
-rw-rw-- 1 mysql mysql    5 Jul  8 05:35 s_table.MYD
% ls -l s*
lrwxrwx-x 1 mysql mysql   34 Jul  8 05:34 s_table.MYD ->
  /disk2/mysql/data/s_db/s_table.MYD
-rw-rw-- 1 mysql mysql  1024 Jul  8 05:35 s_table.MYI
-rw-rw-- 1 mysql mysql  8548 Jul  8 05:34 s_table.frm
```

MySQL ha creado un enlace simbólico para la tabla. Podría haber creado explícitamente este enlace (prestando la misma atención a los permisos como al crear el enlace simbólico de la base de datos).

Los archivos de datos y de índices se pueden crear en ubicaciones distintas a las de este ejemplo, siempre que exista el directorio `/disk3/mysql/data/indexes/`:

```
mysql> CREATE TABLE s2_table (a int) DATA DIRECTORY =
  '/disk2/mysql/data/s_db' INDEX DIRECTORY =
  '/disk3/mysql/data/indexes';
Query OK, 0 rows affected (0.04 sec)
```

Vea los archivos en sus nuevas ubicaciones:

```
% ls -l /disk3/mysql/data/indexes/
```

```

total 2
-rw-rw-- 1 mysql mysql 1024 Jul  8 06:01 s2_table.MYI
.  ls -l /disk2/mysql/data/s_db/
...
-rw-rw-- 1 mysql mysql  0 Jul  8 06:01 s2_table.MYD
...
% ls -l /usr/local/mysql/data/firstdb/
...
lrwxrwx-x 1 mysql mysql   35 Jul  8 06:01 s2_table.MYD ->
  /disk2/mysql/data/s_db/s2_table.MYD
lrwxrwx-x 1 mysql mysql   38 Jul  8 06:01 s2_table.MYI ->
  /disk3/mysql/data/indexes/s2_table.MYI
-rw-rw-- 1 mysql mysql 8548 Jul  8 06:01 s2_table.frm
...

```

NOTA: Las opciones **INDEX DIRECTORY** y **DATA DIRECTORY** no funcionan al ejecutar **MySQL** en Windows (aunque debe consultar la última documentación).

Resumen

RAID es un método que consiste en utilizar varios discos para almacenar datos. RAID 0 (división) reparte bloques de datos entre varios discos. Aumenta el rendimiento pero no tiene capacidad de redundancia. RAID 1 (duplicación) reduce la velocidad de escritura y puede aumentar la de lectura, pero se utiliza principalmente para evitar fallos de disco. RAID 2, 3, 4 y 5 utilizan la división así como distintos tipos de paridad para tolerancia a fallos. RAID 10 y RAID 0+1 combinan la duplicación y la división (aunque las implementan de forma diferente: RAID 10 se centra más en la fiabilidad y RAID 0+1 en la velocidad).

Los enlaces simbólicos nos permiten ubicar bases de datos o tablas MyISAM en una ubicación diferente a la del directorio de datos convencional, normalmente en una unidad distinta.

Apendices

A

Guía de referencia de la sintaxis de MySQL

En este apéndice se incluyen las instrucciones y la sintaxis SQL utilizadas en la versión 4.0 de MySQL.

Para versiones posteriores es aconsejable consultar la documentación correspondiente a su distribución o visitar el sitio de MySQL (www.mysql.com).

La convención utilizada a lo largo de los apéndices es la siguiente:

- Los corchetes ([]) indican un elemento opcional. Por ejemplo:

```
SELECT expresion [FROM nombre_de_tabla [WHERE cláusula_where]]
```

indica que la expresión es obligatoria (por ejemplo `SELECT 42/10`) y que la cláusula `WHERE` es opcional pero solamente puede existir si existe la cláusula opcional `FROM nombre_de_tabla` (podríamos tener `SELECT * FROM t1`, pero no `SELECT * WHERE f1 > 10`, ya que entonces faltaría la cláusula `nombre_de_tabla`).

- Una barra vertical (|) separa las distintas alternativas.

Por ejemplo:

```
CREATE [UNIQUE | FULLTEXT] INDEX
```

indica que `UNIQUE` y `FULLTEXT` son opciones distintas.

- Las llaves ({ }) indican que es necesario seleccionar una de las opciones. Por ejemplo:

```
CREATE TABLE ... [TYPE = {BDB | HEAP | ISAM | InnoDB | MERGE |
MRG_MYISAM | MYISAM }]
```

Si se especifica la **cláusula opcional** TYPE, será necesario especificar una de las siguientes opciones: BDB, HEAP, ISAM, InnoDB, MERGE, MRG_MYISAM o MYISAM.

- Tres puntos (...) indican que la **opción** se puede repetir. Por ejemplo:

```
SELECT expresion,...
```

indica que la expresión se puede repetir (separada por una coma), **como** se indica a **continuación**: SELECT f1, f2, f3.

ALTER

La **sintaxis** de ALTER es la siguiente:

```
ALTER [IGNORE] TABLE nombre_de_tabla especificación_alter [,
especificación-alter- ...]
```

La **sintaxis** de especificación_alter puede ser una de las siguientes:

```
ADD [COLUMN] definition-create [FIRST | AFTER nombre-de-campo ]
ADD [COLUMN] (definition-create, definition-create,...)
ADD INDEX [nombre-de-índice] (nombre_de_campo_de_índice,...)
ADD PRIMARY KEY (nombre_de_campo_de_índice,...)
ADD UNIQUE [nombre_de_índice] (nombre-de-índice,...)
ADD FULLTEXT [nombre_de_índice] (nombre-de-índice,...)
ADD [CONSTRAINT simbolo] FOREIGN KEY nombre_de_índice
(nombre-de-índice,...) [referencia_definición]
ALTER [COLUMN] nombre-de-campo {SET DEFAULT literal | DROP
DEFAULT}
CHANGE [COLUMN] antiguo_nombre_de_campo definition-create [FIRST |
AFTER nombre-de-campo]
MODIFY [COLUMN] definition-create [FIRST | AFTER nombre_de_campo]
DROP [COLUMN] nombre-de-campo
DROP PRIMARY KEY
DROP INDEX nombre-de-índice
DISABLE KEYS
ENABLE KEYS
RENAME [TO] nuevo_nombre_de_tabla
ORDER BY nombre-de-campo
opciones_de_tabla
```

ALTER TABLE le **permite** cambiar la **estructura** de una tabla existente. Puede añadir **columnas** (ADD), cambiar definiciones y nombres de **columnas**

(CHANGE), modificar definiciones de **columnas** sin cambiar el nombre (MODIFY), eliminar **columnas** o índices (DROP), cambiar el nombre de las tablas (RENAME), ordenar datos (ORDER) así **como** activar (ENABLE) y desactivar (DISABLE) índices.

Una extensión MySQL que no sea ANSI es **significa** que ALTER TABLE puede contener varios componentes (CHANGE, AND, etc.) en una instrucción.

Necesitará **permiso** ALTER, INSERT y CREATE en la tabla para **utilizar** ALTER TABLE.

IGNORE (extensión MySQL no ANSI) **hace** que MySQL elimine registros que puedan generar una clave principal o **única** duplicada. Normalmente MySQL **cancela** y ALTER **falla**.

FIRST y ADD...AFTER le **permiten** especificar donde se debe añadir un **campo** a la definición.

ANALYZE TABLE

```
ANALYZE TABLE nombre_de_tabla [,nombre_de_tabla...]
```

En tablas MyISAM y BDB, analiza y almacena la **distribución** de claves de las tablas especificadas. Bloquea las tablas con un bloqueo de lectura durante la duración de la operación.

BACKUP TABLE

```
BACKUP TABLE nombre_de_tabla [,nombre_de_tabla...] TO  
'nombre_de_ruta'
```

En tablas MyISAM, copia **los archivos** de datos y de definición de datos en el directorio de copias de seguridad.

BEGIN

```
BEGIN
```

La instrucción BEGIN **inicia** una transacción o un **conjunto** de instrucciones. La transacción permanece abierta hasta la siguiente instrucción COMMIT o ROLLBACK.

CHECK TABLE

```
CHECK TABLE nombre_tbl [,nombre_tbl...] [opción [opción...]]
```


La opción puede ser una de las siguientes:

```
CHANGED
EXTENDED
FAST
MEDIUM
QUICK
```

Comprueba la presencia de **errores** en una tabla **MyISAM** o **BDB** y, en tablas **MyISAM**, actualiza las estadísticas del **índice**. La opción **QUICK** no **analiza** las filas para comprobar los enlaces. La opción **FAST** solamente comprueba las tablas que no se cerraron correctamente. La opción **CHANGED** es **la misma** que **FAST** pero **también** comprueba las tablas que han **cambiado** desde la última comprobación. La opción **MEDIUM** verifica que los enlaces eliminados son correctos y la opción **EXTENDED** realiza una búsqueda completa de todas las claves de todas las filas.

COMMIT

```
COMMIT
```

La **instrucción COMMIT** **finaliza** una **instrucción** o un **conjunto de instrucciones** y **vuelca los resultados** en disco.

CREATE

La **sintaxis** de **CREATE** puede ser una de las siguientes:

```
CREATE DATABASE [IF NOT EXISTS] nombre_de_base_de_datos
CREATE [UNIQUE|FULLTEXT] INDEX nombre_de_indice ON nombre_de_tabla
(nombre_de_campo [(longitud)],...)
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nombre_de_tabla [(crear _
definicion,...)]
[opciones_de_tabla] [instruction-select]
```

La **sintaxis** de **crear _ definición** puede ser una de las siguientes:

```
tipo nombre_de_campo [NOT NULL | NULL] [DEFAULT
valor_predeterminado]
[AUTO-INCREMENT] [PRIMARY KEY] [definición_de_referencia]
PRIMARY KEY (nombre_de_campo_de_indice,...)
KEY [nombre_de_indice] (nombre_de_campo_de_indice,...)
INDEX [nombre_de_indice] (nombre_de_campo_de_indice,...)
UNIQUE [INDEX] [nombre_de_indice] (nombre_de_campo_de_indice,...)
FULLTEXT [INDEX] [nombre_de_indice]
(nombre_de_campo_de_indice,...)
[CONSTRAINT simbolo] FOREIGN KEY [nombre_de_indice]
(nombre_de_campo_de_indice,...)
```

[definición_de_referencia]
CHECK (expr)

La sintaxis de type puede ser una de las siguientes:

TINYINT[(longitud)] [UNSIGNED] [ZEROFILL]
SMALLINT[(longitud)] [UNSIGNED] [ZEROFILL]
MEDIUMINT[(longitud)] [UNSIGNED] [ZEROFILL]
INT[(longitud)] [UNSIGNED] [ZEROFILL]
INTEGER[(longitud)] [UNSIGNED] [ZEROFILL]
BIGINT[(longitud)] [UNSIGNED] [ZEROFILL]
REAL[(longitud,decimales)] [UNSIGNED] [ZEROFILL]
DOUBLE[(longitud,decimales)] [UNSIGNED] [ZEROFILL]
FLOAT[(longitud,decimales)] [UNSIGNED] [ZEROFILL]
DECIMAL(longitud,decimales) [UNSIGNED] [ZEROFILL]
NUMERIC(longitud,decimales) [UNSIGNED] [ZEROFILL]
CHAR(longitud) [BINARY]
VARCHAR(longitud) [BINARY]
DATE
TIME
TIMESTAMP
DATETIME
TINYBLOB
BLOB
MEDIUMBLOB
LONGBLOB
TINYTEXT
TEXT
MEDIUMTEXT
LONGTEXT
ENUM(valor1,valor2,valor3,...)
SET(valor1,valor2,valor3,...)

La sintaxis de nombre_de_campo_de_índice es la siguiente:

nombre_de_campo [(longitud)]

La sintaxis de definición_de_referencia es la siguiente:

REFERENCES nombre_de_tabla [(nombre_de_campo_de_índice,...)]
[MATCH FULL
| MATCH PARTIAL] [ON DELETE opción_de_referencia] [ON UPDATE
opción_de_referencia]

La sintaxis de opción_de_referencia es la siguiente:

RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

La sintaxis de opciones_de_tabla puede ser una de las siguientes:

TYPE = (BDB | HEAP | ISAM | InnoDB | MERGE | MRG_MYISAM | MYISAM)
AUTO-INCREMENT = #
AVG_ROW_LENGTH = #
CHECKSUM = {0 | 1}

```

COMMENT = «cadena"
MAX-ROWS = #
MIN_ROWS = #
PACK-KEYS = {0 | 1 | DEFAULT}
PASSWORD = «cadena"
DELAY-KEY-WRITE = {0 | 1}
ROW-FORMAT= { predeterminado | dinámico | fijo | comprimido }
RAID-TYPE= (1 | STRIPED | RAID0 ) RAID-CHUNKS=# RAID-CHUNKSIZE=#
UNION = (nombre_de_tabla, [nombre_de_tabla...])
INSERT-METHOD= (NO | FIRST | LAST )
DATA DIRECTORY="ruta_absoluta_a_directorio"
INDEX DIRECTORY="ruta_absoluta_a_directorio"

```

La sintaxis de `instrucción select` puede ser la siguiente:

```
[IGNORE | REPLACE] SELECT ... (instrucción select)
```

La instrucción **CREATE** crea una base de datos, una tabla o un **índice**.

Las tablas **TEMPORARY** existen siempre que la conexión este activa. Para ello necesitara el permiso **CREATE TEMPORARY TABLES**.

De forma predeterminada, las definiciones de campos **adoptan** el valor **NULL**. Los campos numericos **adoptan** el valor **0** (excepto con **AUTO_INCREMENT**) y los campos de cadena, una cadena vacia (excepto los campos **ENUM**, que toman de forma predeterminada la primera opcion). De forma predeterminada, los campos de fecha y hora completan el campo con ceros.

Los campos **AUTO_INCREMENT** empiezan a contar desde **1** de forma predeterminada y se **incrementan** en uno cada vez que se añade un nuevo registro.

KEY e **INDEX** son sinonimos en este contexto.

PRIMARY KEY especifica que el **índice** no puede contener duplicados y el campo (o combinación de campos), debe especificarse como **NOT NULL**.

UNIQUE especifica que el **índice** no puede contener duplicados.

La opcion **RAID_TYPE** contribuye a que los sistemas operativos que no pueden admitir archivos de gran tamaño, superen esta limitación. La opción **STRIPED** es la unica que se utiliza actualmente. En tablas **MyISAM**, esto crea **subdirectorios** dentro del directorio de bases de datos, cada uno con una parte del **archivo**. Los primeros **1024*RAID_CHUNKSIZE** bytes se **incluyen** en la **primera** parte, los siguientes **1024*RAID_CHUNKSIZE** en la **siguiente** y asi sucesivamente.

Las opciones **DATA DIRECTORY="directorio"** e **INDEX DIRECTORY="directorio"** especifican **rutas** absolutas a la ubicacion en la que se almacenan los datos o los indices.

La opcion **PACK_KEYS=1** agrupa campos numericos en el **índice** de tablas **MyISAM** (y, de forma predeterminada, cadenas).

Solamente resulta util si tiene indices con gran cantidad de numeros duplicados.

Puede utilizar **AVG_ROW_LENGTH** para que **MySQL** se haga una idea de la longitud media de las **filas de la tabla**. Solamente resulta **útil** si la tabla es de gran tamaño y los registros son de un tamaño variable.

CHECKSUM se puede definir como 1 en tablas MyISAM si quiere realizar una comprobación de todas las tablas, lo que facilita la **reparación** de la tabla en caso de que este dañada aunque también reduce su velocidad.

COMMENT es un comentario de hasta 60 caracteres.

MAX_ROWS y MIN_ROWS especifican el número máximo y **mínimo** de filas que se **van a almacenar** en la tabla.

PASSWORD codifica el **archivo de definición** de datos (.frm) con una **contraseña**.

DELAY_KEY_WRITE hace que MySQL espere hasta que se cierre una tabla MyISAM **antes** de actualizar el **índice**, lo que aumenta la velocidad de **operaciones** INSERT y UPDATE.

ROW_FORMAT especifica si una tabla MyISAM debe ser FIXED o DYNAMIC.

DELETE

La sintaxis de DELETE puede ser una de las siguientes:

```
DELETE [LOW-PRIORITY | QUICK] FROM nombre-de-tabla [WHERE
  clausula-where] [ORDER BY ...] [LIMIT filas]
DELETE [LOW-PRIORITY | QUICK] nombre-de-tabla [.*]
  [, nombre-de-tabla [.*] ...] FROM referencias de tablas [WHERE
  clausula-where]
DELETE [LOW-PRIORITY | QUICK] FROM tabla[.*], [tabla[.*]
  ...] USING referencias de tabla [WHERE clausula-where]
```

La **instrucción** DELETE borra registros de la tabla (o tablas) que cumplen la **clausula_where** (o todos los registros si no existe esta clausula).

La **palabra** clave LOW PRIORITY hace que DELETE espere hasta que no haya ningún cliente leyendo la tabla antes de procesarla.

La **palabra** clave QUICK hace que MySQL no combine hojas de **índice** durante DELETE, lo que en ocasiones **resulta** más rápido.

LIMIT determina el número máximo de registros que se pueden eliminar.

La clausula ORDER BY hace que MySQL borre registros en un determinado orden (lo que **resulta** muy útil con una clausula LIMIT).

DESC

DESC equivale a DESCRIBE

DESCRIBE

```
DESCRIBE nombre-de-tabla {nombre_de_campo | comodin}
```

DESCRIBE devuelve la **definición** de la tabla y los campos especificados (igual que SHOW COLUMNS FROM nombre_de_tabla).

Se puede utilizar un comodín como **parte del nombre** de archivo y puede ser un signo % que equivale a un número de caracteres o un guion bajo (_), que equivale a un carácter.

DO

La sintaxis de DO es la siguiente:

```
DO expresion, [expresion, ...]
```

DO tiene el **mismo** efecto que SELECT, a **excepción** de que no devuelve resultados (lo que lo **hace** ligeramente más rápido).

DROP

La sintaxis de DROP es la siguiente:

```
DROP DATABASE [IF EXISTS] nombre_de_base_de_datos
DROP TABLE [IF EXISTS] nombre_de_tabla [, nombre-de-tabla, ...]
[RESTRICT | CASCADE]
DROP INDEX nombre_de_indice ON nombre_de_tabla
```

DROP DATABASE elimina la base de datos y todas sus tablas.

DROP TABLE elimina la tabla especificada.

DROP INDEX elimina el **índice** especificado.

MySQL devuelve un error si la base de datos no existe, a **menos** que se utilice la **cláusula IF EXISTS**.

DROP TABLE confirma, automáticamente, **las** transacciones activas.

RESTRICT y CASCADE no se implementan actualmente.

EXPLAIN

```
EXPLAIN nombre_de_tabla
EXPLAIN consults-select
```

consulta_select es la misma que la que especificamos en la descripción de SELECT.

El uso de EXPLAIN con un nombre de tabla equivale a DESCRIBE nombre_de_tabla. El uso de EXPLAIN con una consulta proporciona **información sobre cómo** se ejecutará la consulta, lo que resulta muy **útil** para optimizar la consulta y aprovechar **al máximo** los índices asociados.

FLUSH

```
FLUSH opción_de_vaciado [, opción_de_vaciado] ...
```

La opción FLUSH puede ser una de las siguientes:

```
DES_KEY_FILE
HOSTS
LOGS
QUERY CACHE
PRIVILEGES
STATUS
TABLES
[TABLE | TABLES] nombre_de_tabla [, nombre_de_tabla...]
TABLES WITH READ LOCK
USER-RESOURCES
```

Al vaciar DES_KEY_FIELDS se vuelven a cargar las claves DES. Con la opción HOSTS se vacía la caché del servidor (que, por ejemplo, se utiliza después de cambiar direcciones IP). Al vaciar los registros (LOGS), se cierran y se vuelven a abrir los archivos de registro y se incrementa el registro binario. Al vaciar QUERY CACHE se defragmenta la cache de consultas. Al vaciar los PRIVILEGES se vuelven a cargar las tablas de permisos desde la base de datos mysql. Al vaciar STATUS se restablecen las variables de estado. Al vaciar TABLES sucede lo mismo que al vaciar QUERY CACHE, pero también se cierran todas las tablas abiertas. Solamente se pueden especificar determinadas tablas para ser vaciadas. Puede añadir un READ LOCK a las tablas, que resulta muy útil para bloquear un grupo de tablas por motivos de creación de copias de seguridad. Al vaciar USER_RESOURCES se restablecen los recursos de usuario (utilizados para limitar consultas, conexiones y actualizaciones por hora).

GRANT

```
GRANT tipo_de_privilegio [(lista_de_campos)] [, tipo_de_privilegio
[(lista_de_campos)]
... ] ON {nombre_de_tabla | * | *.* | nombre_de_base_de_datos.*}
TO nombre_de_usuario
[IDENTIFIED BY [PASSWORD] 'contrasefia'] [, nombre_de_usuario
[IDENTIFIED BY
'contrasefia']...] [REQUIRE NONE | [{SSL| X509}] [CIPHER cifrado
[AND]]
[ISSUER emisor [AND]] [SUBJECT asunto]] [WITH [GRANT OPTION |
MAX_QUERIES_PER_HOUR # | MAX-UPDATES-PER-HOUR # |
MAX_CONNECTIONS_PER_HOUR #]]
```

GRANT otorga a un usuario un determinado tipo de permiso. En la tabla A.1 se incluye la descripción de los privilegios disponibles.

Tabla A.1. Privilegios

Privilegio	Descripción
ALL	Concede todos los permisos básicos.
ALL PRIVILEGES	Igual que ALL.
ALTER	Permiso para cambiar la estructura de una tabla (una instrucción ALTER) a excepción de los índices.
CREATE	Permiso para crear bases de datos y tablas, a excepción de índices.
CREATE TEMPORARY TABLES	Permiso para crear una tabla temporal.
DELETE	Permiso para eliminar registros de una tabla (una instrucción DELETE).
DROP	Permiso para eliminar bases de datos o tablas, a excepción de los índices.
EXECUTE	Permiso para ejecutar procedimientos almacenados (previsto para MySQL 5).
FILE	Permiso para leer y escribir archivos en el servidor (para instrucciones LOAD DATA INFILE o SELECT INTO OUTFILE). Todos los archivos que el usuario mysql pueda leer son legibles.
INDEX	Permiso para crear, modificar o borrar índices.
INSERT	Permiso para añadir nuevos registros a la tabla (una instrucción INSERT).
LOCK TABLES	Permiso para bloquear una tabla para la que el usuario tiene permiso SELECT.
PROCESS	Permiso para ver los procesos MySQL actuales o para eliminar procesos MySQL (para instrucciones SHOW PROCESSLIST o KILL SQL).
REFERENCES	No se utiliza actualmente en MySQL. Se ofrece para compatibilidad con SQL ANSI (se aplica al uso de claves secundarias).
RELOAD	Permiso para volver a cargar la base de datos (una instrucción FLUSH o una recarga, actualización o vaciado emitido desde mysqladmin).
REPLICATION CLIENT	Permiso para preguntar sobre esclavos y principales de duplicación.
SHOW DATABASES	Permiso para ver todas las bases de datos.

Privilegio	Descripción
SELECT	Permiso para devolver datos de una tabla (una instrucción SELECT).
SHUTDOWN	Permiso para cerrar el servidor.
SUPER	Permiso para conectarse, incluso si se alcanza el número máximo de conexiones, y ejecutar comandos CHANGE MASTER, KILL para subprocesos, depuración de mysqladmin, PURGE MASTER LOGS y SET GLOBAL.
UPDATE	Permiso para modificar datos en una tabla (una instrucción UPDATE).
USAGE	Permiso para conectarse al servidor y ejecutar instrucciones disponibles para todos (en las primeras versiones de MySQL 4, incluía SHOW DATABASES).

INSERT

La sintaxis de INSERT puede ser una de las siguientes:

```

INSERT [LOW-PRIORITY | DELAYED] [IGNORE] [INTO] nombre-de-tabla
    [(nombre_de_campo,...)] VALUES ((expresion |
    DEFAULT),...), (...),...
INSERT [LOW-PRIORITY | DELAYED] [IGNORE] [INTO] nombre-de-tabla
    [(nombre_de_campo,...)] SELECT ...
INSERT [LOW-PRIORITY | DELAYED] [IGNORE] [INTO] nombre-de-tabla
    SET nombre_de_campo =(expresion | DEFAULT), ...
INSERT [LOW-PRIORITY] [IGNORE] [INTO] nombre-de-tabla [(lista de
campos)] SELECT ...

```

INSERT añade nuevas filas a una tabla. Sin la lista inicial de campos, se asume que los campos están en el mismo orden que en la definición, y que debe haber un valor para cada uno de ellos.

Todas las columnas que no se definen explícitamente se configuran con su valor predeterminado.

La palabra clave LOW PRIORITY hace que INSERT espere a que no haya clientes leyendo la tabla antes de procesarla.

Con la palabra clave DELAYED, MySQL libera el cliente pero espera para realizar la inserción.

IGNORE hace que MySQL ignore las inserciones que resultan en la duplicación de claves principales o exclusivas, en lugar de cancelarlas.

INSERT...SELECT le permite insertar en una tabla desde filas existentes de una o varias tablas.

JOIN

MySQL acepta cualquiera de las siguientes **sintaxis** para JOIN:

```
nombre-de-tabla, nombre-de-tabla
nombre-de-tabla [CROSS] JOIN nombre-de-tabla
nombre-de-tabla INNER JOIN condicion nombre-de-tabla
nombre-de-tabla STRAIGHT-JOIN nombre-de-tabla
nombre_de_tabla LEFT [OUTER] JOIN condicion nombre-de-tabla
nombre-de-tabla LEFT [OUTER] JOIN nombre-de-tabla
nombre_de_tabla NATURAL [LEFT [OUTER]] JOIN nombre-de-tabla
nombre_de_tabla LEFT OUTER JOIN nombre-de-tabla ON
expresion_condicional
nombre_de_tabla RIGHT [OUTER] JOIN condicibn nombre-de-tabla
nombre-de-tabla RIGHT [OUTER] JOIN nombre-de-tabla
nombre-de-tabla NATURAL [RIGHT [OUTER]] JOIN nombre-de-tabla
```

La tabla puede ser simplemente `nombre_de tabla`, utilizar un alias (con AS) o especificar o ignorar índices (con USE/IGNORE).

La **sintaxis** de condición es la siguiente:

```
ON expresión_condicional [ USING (nombres_de_campos)
```

`expresión_condicional` es lo mismo que lo que puede incluir una **cláusula** WHERE.

KILL

```
KILL id-subproceso
```

Elimina el subproceso especificado. Puede utilizar **SHOW PROCESSLIST** para identificar los Id. de los subprocesos. Se necesita el **privilegio SUPER** para eliminar subprocesos que no **sean** propiedad de la **conexión** actual.

LOAD DATA INFILE

La **sintaxis** de LOAD DATA INFILE es la siguiente:

```
LOAD DATA [LOW-PRIORITY | CONCURRENT] [LOCAL] INFILE
'archivo.txt' [REPLACE | IGNORE] INTO TABLE nombre-de-tabla
[FIELDS [TERMINATED BY '\t'] [[OPTIONALLY] ENCLOSED BY
'] [ESCAPED BY '\\'] ] [LINES TERMINATED BY '\n']
[IGNORE numero LINES] [(nombre_de_campo,...)]
```

LOAD DATA lee datos de un **archivo** de texto y los **añade** a una tabla. Es una **forma** mas rapida de **añadir** grandes volumenes de datos que por medio de INSERT.

La **palabra** clave LOCAL indica que el archivo se encuentra en el equipo cliente; en **caso** contrario, se asume que se encuentra en el servidor de bases de datos. LOCAL no funciona si el servidor se ha iniciado con la opción `-local-infile=0` o si el cliente no ha podido admitirla.

Los **archivos** del servidor **deben** ser legibles para todos o encontrarse en el directorio de bases de datos. También necesitará el **permiso** FILE para **utilizar** LOAD DATA en un archivo del servidor.

En el servidor, se **supone** que el archivo se encuentra en el directorio de bases de datos de la base de datos actual si no se indica ninguna ruta. Si la ruta es relativa, se asume que proviene del directorio de datos. También se pueden utilizar **rutas** absolutas.

La **palabra** clave LOW PRIORITY hace que LOAD DATA espere hasta que no haya ningún cliente leyendo la tabla antes de procesarla.

La **palabra** clave CONCURRENT **permite** que otros subprocesos puedan acceder a una tabla MyISAM al mismo tiempo que se **ejecuta** LOAD DATA (lo que reducirá la velocidad de LOAD DATA).

La **palabra** clave REPLACE hace que MySQL **elimine** y sustituya un registro existente si tiene la misma clave principal o exclusiva que el registro que se va a añadir. IGNORE hace que MySQL continúe con el siguiente registro.

Si se especifica una **cláusula** FIELDS, **al menos** se necesita una de las siguientes opciones: TERMINATED BY, [OPTIONALLY] ENCLOSED BY y ESCAPED BY. Si no se especifica ninguna **cláusula** FIELDS, se asume que las predeterminadas serán FIELDS TERMINATED BY '\t' ENCLOSED BY '' ESCAPED BY '\\'. Estas **cláusulas** especifican el carácter al final de un campo (de **forma** predeterminada, una tabulación), que **rodean** al campo (de **forma** predeterminada nada) y el carácter de salida (de **forma** predeterminada, la barra invertida). Debe prestar especial atención cuando utilice **rutas** de Windows para salir de la ruta correctamente.

Sin una cláusula LINES, se asume que la predeterminada será LINES TERMINATED BY '\n'. Especifica el carácter al final de un registro (de **forma** predeterminada, una nueva **línea**). La opción IGNORE **numero** LINES ignora una serie de **líneas** en la parte superior del archivo (lo que resulta muy útil cuando el archivo contiene un encabezado).

LOAD DATA INFILE es el **complemento** de SELECT...INTO INFILE.

LOCK TABLES

```
LOCK TABLES nombre_de_tabla [AS alias] {READ | [READ LOCAL] |  
[LOW-PRIORITY]  
WRITE) [, nombre_de_tabla {READ | [LOW-PRIORITY] WRITE) ...}
```

LOCK TABLES **incluye** un bloqueo en las tablas especificadas. El bloqueo puede ser READ (el resto de conexiones no pueden escribir, **sólo** leer), READ

LOCAL (igual que READ a excepción de que se permite que escriban otras conexiones no conflictivas) o WRITE (que bloquea la lectura y la escritura desde otras conexiones).

Si el bloqueo WRITE es LOW PRIORITY, los bloqueos READ se añaden antes.

Normalmente los bloqueos WRITE tienen una mayor prioridad.

OPTIMIZE

```
OPTIMIZE TABLE nombre-de-tabla [, nombre-de-tabla]...
```

En tablas MyISAM, ordena el índice, actualiza las estadísticas y desfragmenta el archivo de datos.

En tablas BDB, es igual que ANALYZE TABLE.

Bloquea la tabla durante la duración de la operación (que puede llevar su tiempo).

RENAME

La sintaxis de RENAME es la siguiente:

```
RENAME TABLE nombre-de-tabla TO nuevo_nombre_de_tabla  
[, nombre_de_tabla2 TO nuevo_nombre_de_tabla2, ...]
```

RENAME le permite asignar un nuevo nombre a una tabla o conjunto de tablas.

También puede cambiar una tabla a una nueva base de datos si especifica nombre_de_base_de_datos.nombre_de_tabla, siempre que la base de datos se encuentre en el mismo disco.

Necesita los permisos ALTER y DROP en la tabla antigua, y los permisos CREATE e INSERT en la nueva.

REPAIR TABLE

```
REPAIR TABLE nombre-de-tabla [, nombre_de_tabla...] [EXTENDED]  
[QUICK] [USE-FRM]
```

Repara una tabla MyISAM dañada. Con la opción QUICK, solamente se repara el árbol de índices.

Con EXTENDED, el índice se vuelve a crear fila a fila. Con USER_FRM, el índice se repara en función del archivo de datos (para cuando falte el índice o este totalmente dañado).

REPLACE

La **sintaxis** de REPLACE puede ser una de las siguientes:

```
REPLACE [LOW-PRIORITY | DELAYED] [INTO] nombre-de-tabla
  [(nombre-de-campo,...)] VALUES (expresion,...), (...),...
REPLACE [LOW-PRIORITY | DELAYED] [INTO] nombre-de-tabla
  [(nombre-de-campo,...)] SELECT ...
REPLACE [LOW-PRIORITY | DELAYED] [INTO] nombre-de-tabla SET
  nombre-de-campo =expresión, nombre-de-campo =expresión, ...
```

REPLACE es exactamente **igual** que INSERT, a **excepción** de que cuando MySQL encuentra un registro con una clave principal o exclusiva que ya existe, la elimina y la reemplaza.

RESET

```
RESET opción_reset [,option-reset]...
```

opción_reset puede ser una de las siguientes:

```
MASTER
QUERY CACHE
SLAVE
```

RESET MASTER **elimina** todos los registros binarios y vacía el **índice** de registros binarios. RESET SLAVE restablece la **posición** de un esclavo en la **duplicación** de un principal. RESET QUERY CACHE vacía la cache de **consultas**.

RESTORE TABLE

```
RESTORE TABLE nombre-de-tabla [,nombre_de_tabla...] FROM 'ruta'
```

Recupera una tabla de la que se ha creado una copia de seguridad con BACKUP TABLE.

No sobrescribe las tablas existentes.

REVOKE

```
REVOKE tipo_de_privilegio [(lista_de_campos)] [,tipo_de_privilegio
  [(lista_de_campos)]
  ...] ON {nombre-de-tabla | * | *.* | nombre_de_base_de_datos.*}
FROM nombre_de_usuario
[, nombre_de_usuario ...]
```

Elimina los privilegios concedidos previamente a los usuarios especificados. `tipo_de_privilegio` puede ser cualquiera de los privilegios enumerados para GRANT.

ROLLBACK

ROLLBACK

La instrucción ROLLBACK elimina una transacción o conjunto de instrucciones, y deshace todas las instrucciones de esa transacción.

SELECT

La sintaxis de SELECT es la siguiente:

```
SELECT [STRAIGHT-JOIN] [SQL_SMALL_RESULT] [SQL_BIG_RESULT]
      [SQL_BUFFER_RESULT] [SQL_CACHE | SQL_NO_CACHE]
      [SQL_CALC_FOUND_ROWS] [HIGH-PRIORITY] [DISTINCT |
      DISTINCTROW | ALL] expresion, ... [INTO {OUTFILE |
      DUMPFILe} 'nombre_de_archivo' opciones_de_exportación]
[FROM nombres_de_tabla
[WHERE cláusula_where] [GROUP BY {entero-sin-firma |
      nombre_de_campo | formula) [ASC | DESC], ... [HAVING
      definición_where] [ORDER BY (entero-sin-firma |
      nombre_de_campo | formula) [ASC | DESC], ...] [LIMIT
      [desplazamiento,] filas] [PROCEDURE nombre_procedimiento] [FOR
      UPDATE | LOCK IN SHARE MODE]]]
```

Las instrucciones SELECT devuelven datos de tablas. expresión suele ser una lista de campos (con una función en caso de que sea necesario) pero también puede ser un cálculo o función que no tiene nada que ver con los campos. Por ejemplo:

```
SELECT VERSION();
```

o, como se indica a continuación:

```
SELECT 42/10;
```

Los campos se pueden especificar como `nombre_de_campo`, `nombre_de_tabla.nombre_de_campo`, o `nombre_de_base_de_datos.nombre_de_tabla.nombre_de_campo`. Las formas más extensas son necesarias en caso de ambigüedad.

A la expresión también se le puede asignar un alias con la palabra clave AS. Por ejemplo:

```
SELECT 22/7 AS about-pi
```

La expresion puede utilizarse en cualquier punto de la instrucción (pero no en la clausula WHERE, que normalmente se determina en primer lugar). La clausula `nombres_de_tabla` es una lista de las tablas que se utilizan en la consulta, separadas por comas. Tambien puede utilizar un alias. Por ejemplo:

```
SELECT watts FROM wind-water-solar-power AS n;
```

Tambien puede controlar el uso de indices de MySQL si no le convence la opcion de MySQL (que puede ver si utiliza EXPLAIN) y utilizar las clausulas USE INDEX e IGNORE INDEX despues del nombre de la tabla. La sintaxis es la siguiente:

```
nombre_de_tabla [[AS] alias] [USE INDEX (lista de indices)]  
[IGNORE INDEX (lista de indices)]
```

La clausula ORDER BY ordena los resultados devueltos en orden ascendente (opcion predeterminada o al utilizar la palabra clave ASC) o descendente (DESC). No es necesario que utilice elementos devueltos explicitamente en la expresion. Por ejemplo:

```
SELECT team-name FROM results ORDER BY points DESC
```

La clausula WHERE esta formada por condiciones (que pueden contener funciones) que debe cumplir una fila para poder ser devuelta:

```
SELECT team-name FROM results WHERE points > 10
```

GROUP BY agrupa filas de resultados, que resultan muy útiles cuando se emplea una funcion agregada. Existen dos extensiones MySQL no ANSI que puede utilizar con GROUP BY: ASC y DESC. Tambien puede utilizar campos en la expresion que no se mencionen en las cláusulas GROUP BY. Por ejemplo:

```
SELECT team-name, team-address, SUM(points) FROM teams GROUP BY  
team-name DESC
```

La clausula HAVING tambien es una condición, pero se implementa en ultimo lugar para que pueda aplicarla a los elementos que agrupa. Por ejemplo:

```
SELECT team-name, SUM(points) FROM teams GROUP BY team-name HAVING  
SUM(points) > 20
```

No lo utilice como sustituto de la clausula WHERE, ya que reduce la velocidad de las consultas. DISTINCT y su sinonimo, DISTINCTROW, indica que la fila devuelta debe ser exclusiva. ALL (la opcion predeterminada), devuelve todas las filas, sean o no exclusivas.

HIGH_PRIORITY (extension MySQL no ANSI) otorga a SELECT una prioridad mayor que a cualquier actualización.

SQL_BIG_RESULT y SQL_SMALL_RESULT (extensiones MySQL no ANSI) ayudan al optimizador de MySQL y le indican el tamaño de los resultados antes

de que **inicie** el procesamiento. **Ambas** se utilizan con cláusulas **GROUP BY** y **DISTINCT**, y normalmente hacen que MySQL utilice una tabla temporal para obtener mayor velocidad.

SQL_BUFFER_RESULT (extension MySQL no ANSI) hace que MySQL **incluya el resultado** en una tabla temporal.

LIMIT adopta uno o dos argumentos para limitar el numero de filas devueltas. Si es un argumento, sera el numero maximo de filas que se devuelven; si son dos, el **primero** se corresponde **al** desplazamiento y el **segundo al** numero maximo de filas que se devuelven. Si el segundo argumento es **-1**, MySQL devolvera todas las filas desde el desplazamiento especificado hasta el final. Por ejemplo, para devolver desde la fila **2 hacia adelante**, utilice lo siguiente:

```
SELECT f1 FROM t1 LIMIT 1,-1
```

SQL_CALC_FOUND_ROWS hace que MySQL **calcule** el numero de filas que se **tendrían que haber devuelto** si no hubiera una **cláusula LIMIT**. Esta cifra se puede obtener con ayuda de la **función SELECT FOUND_ROWS()**.

SQL_CACHE hace que MySQL almacene el resultado **en la caché** de consultas y, **SQL_NO_CACHE**, que no lo haga. Se trata de dos extensiones **MySQL** no ANSI.

STRAIGHT JOIN (una extension MySQL no ANSI) hace que el optimizador combine las tablas en el **orden** en el que aparecen en la **cláusula FROM**, lo que puede aumentar la velocidad de las consultas si las tablas se **combinan** de una **forma** que no sea optima (utilice **EXPLAIN** para comprobarlo).

SELECT...INTO OUTFILE 'nombre_de_archivo' escribe los resultados en un nuevo archivo (que **todo el mundo puede leer**) en el servidor. Necesita el **permiso FILE** para utilizarlo. Es el **complemento** de **LOAD DATA INFILE** y utiliza las mismas opciones.

Al **utilizar INTO DUMPFILE**, MySQL escribe una **fila** en el archivo, sin **columnas** o terminaciones de linea y sin conversiones de escape.

Con tablas **InnoDB** y **BDB**, la cláusula **FOR UPDATE** escribe bloqueos en las filas.

SET

```
SET [GLOBAL | SESSION] nombre_de_variable =expresión, [[GLOBAL |  
SESSION |  
LOCAL ] nombre_de_variable =expresión...]
```

SET le **permite** definir valores de variables. **SESSION** (o **LOCAL**, un sinonimo) es el valor predeterminado y define el valor mientras dure la **conexión** actual. **GLOBAL** requiere el **privilegio SUPER** y define la variable para todas las nuevas conexiones hasta que se reinicie el servidor. Tendra que definirla en el archivo de **configuración** para que la **opción** permanezca activa una vez reiniciado

el servidor. Si utiliza `SHOW VARIABLES`, podrá ver la lista completa de variables. En la tabla A.2 se recogen las variables que no se definen de forma estandar.

Tabla A.2. Variables que no se definen de forma estandar

Sintaxis	Descripción
<code>AUTOCOMMIT= 0 1</code>	Cuando se configura (1), MySQL ejecuta las instrucciones automáticamente a menos que las envuelva en instrucciones <code>BEGIN</code> y <code>COMMIT</code> . También lo hace con las transacciones abiertas si define <code>AUTOCOMMIT</code> .
<code>BIG_TABLES = 0 1</code>	Cuando se configura (1), todas las tablas temporales se almacenan en disco y no en memoria. Esto reduce la velocidad de las tablas temporales pero evita el problema de quedarse sin memoria. El valor predeterminado es 0.
<code>INSERT_ID = #</code>	Define el valor <code>AUTO_INCREMENT</code> (para que la siguiente instrucción <code>INSERT</code> que utilice un campo <code>AUTO_INCREMENT</code> utilice este valor).
<code>LAST_INSERT_ID = #</code>	Define el valor devuelto por la siguiente función <code>LAST_INSERT_ID()</code> .
<code>LOW_PRIORITY_UPDATES = 0 1</code>	Cuando se configura (1), todas las instrucciones de actualización (<code>INSERT</code> , <code>UPDATE</code> , <code>DELETE</code> , <code>LOCK TABLE WRITE</code>) esperan a que no haya lecturas pendientes (<code>SELECT</code> , <code>LOCK</code> , <code>TABLE READ</code>) en la tabla a la que quieren acceder.
<code>MAX_JOIN_SIZE = valor DEFAULT</code>	Al determinar un tamaño de fila máximo, puede evitar que MySQL ejecute consultas que no empleen correctamente los índices o que puedan reducir la velocidad del servidor cuando se ejecuten en periodos de máximo tráfico. Si se configura con otro valor que no sea <code>DEFAULT</code> , se restablece <code>SQL_BIG_SELECTS</code> . Si se configura <code>SQL_BIG_SELECTS</code> , se ignora <code>MAX_JOIN_SIZE</code> . Si la consulta ya se ha almacenado en cache, MySQL ignorará este límite y devolverá los resultados.
<code>QUERY_CACHE_TYPE = OFF ON DEMAND</code>	Define el parametro de cache de consultas para el subproceso.
<code>QUERY_CACHE_TYPE = 0 1 2</code>	Define el parametro de cache de consultas para el subproceso.
<code>SQL_AUTO_IS_NULL = 0 1</code>	Si se configura (1, el predeterminado), la ultima fila insertada en un campo <code>AUTO_INCREMENT</code> se puede localizar con <code>WHERE columnas_de_</code>

autoincremento ISNULL. Lo utilizan programas como Microsoft Access y otros que se conectan a través de ODBC.

SQL_BIG_SELECTS = 0 1	Si se configura (1, el predeterminado), MySQL permite consultas de gran tamaño. Si no se configura (0), no permitira las consultas en las que tenga que examinar mas filas del valormax_join_size rows. Resulta muy util para evitar ejecutar consultas accidentales o peligrosas que puedan colapsar el servidor.
SQL_BUFFER_RESULT = 0 1	Si se configura (1), MySQL almacena los resultados de las consultas en tablas temporales (en algunos casos aumenta el rendimiento ya que primero libera los bloqueos de tabla).
SQL_LOG_OFF = 0 1	Si se configura (1), MySQL no registra el cliente (no se trata del registro de actualización). Se necesita el permiso SUPER.
SQL_LOG_UPDATE = 0 1	Si no se configura (0), MySQL no utilizara el registro de actualizaciones en el cliente. Se necesita el permiso SUPER.
SQL_QUOTE_SHOW_CREATE = 0 1	Si se configura (1, el predeterminado), MySQL añadira comillas a los nombres de tablas y columnas.
SQL_SAFE_UPDATES = 0 1	Si se configura (1), MySQL no ejecutara instrucciones UPDATE o DELETE que no utilicen un índice o una clausula LIMIT, lo que contribuye a evitar accidentes no deseados.
SQL_SELECT_LIMIT = valor DEFAULT	Determina el numero maximo de registros (de forma predeterminada, ilimitado) que se pueden devolver con una instrucción SELECT. LIMIT tiene preferencia sobre esta variable.
TIMESTAMP = valor_ marca_de_tiempo DEFAULT	Define el tiempo para el cliente. Se puede utilizar para obtener la marca de tiempo original cuando se emplea el registro de actualizaciones para restablecer filas. valor_marca_de_tiempo es una marca de tiempo de la epoca de Unix.

La antigua sintaxis de SET OPTION se ha eliminado, por lo que no debería utilizarla mas.

SET TRANSACTION

```
SET [GLOBAL | SESSION] TRANSACTION ISOLATION LEVEL { READ UNCOMMITTED
| READ COMMITTED | REPEATABLE READ | SERIALIZABLE }
```

Define el nivel de aislamiento de transacciones. De forma predeterminada, solamente se aplica a la siguiente transacción, a menos que se utilicen las palabras clave `SESSION` o `GLOBAL` (que determinan el nivel de todas las transacciones en la conexión actual o en todas las transacciones de las nuevas conexiones).

SHOW

La sintaxis de `SHOW` puede ser una de las siguientes:

```
SHOW DATABASES [LIKE expresion]
SHOW [OPEN] TABLES [FROM nombre_de_base_de_datos] [LIKE expresion]
SHOW [FULL] COLUMNS FROM nombre_de_tabla [FROM
nombre_de_base_de_datos] [LIKE expresion]
SHOW INDEX FROM nombre_de_tabla [FROM nombre_de_base_de_datos]
SHOW TABLE STATUS [FROM nombre_de_base_de_datos] [LIKE expresion]
SHOW STATUS [LIKE expresion]
SHOW VARIABLES [LIKE expresion]
SHOW LOGS
SHOW [FULL] PROCESSLIST
SHOW GRANTS FOR usuario
SHOW CREATE TABLE nombre_de_tabla
SHOW MASTER STATUS
SHOW MASTER LOGS
SHOW SLAVE STATUS
```

`SHOW` enumera las bases de datos, tablas o columnas, o proporciona información de estado sobre el servidor.

El comodín puede formar parte del nombre de la base de datos, tabla o archivo, y puede ser un símbolo `%`, que equivale a una serie de caracteres o un guion bajo (`_`), que equivale a un carácter.

TRUNCATE

```
TRUNCATE TABLE nombre-de-tabla
```

La instrucción `TRUNCATE` elimina todos los registros de una tabla. Es más rápida que la instrucción equivalente `DELETE` ya que utiliza `DROP` y `CREATE` en la tabla. No resulta segura para las transacciones (por lo que devolvería un error si hay alguna transacción o algún bloqueo activo).

UNION

```
SELECT ... UNION [ALL] SELECT ... [UNION SELECT ...]
```

UNION combina varios resultados en uno.

Sin la **palabra** clave ALL, las filas son exclusivas.

UNLOCK TABLES

```
UNLOCK TABLES
```

Libera todas las tablas de la **conexión** actual.

UPDATE

```
UPDATE [LOW-PRIORITY] [IGNORE] nombre_de_tabla SET  
nombre_de_campo1=expresión1 [, nombre_de_campo2=expresión2, ...]  
[WHERE cláusula_where] [LIMIT #]
```

La **instrucción** UPDATE **actualiza** los contenidos de las filas existentes en la base de datos.

La **cláusula** SET especifica que campos se **actualizan** y **cuáles serán** los nuevos valores.

La **cláusula** **where** proporciona las condiciones que debe cumplir la fila para poder **actualizarla**.

IGNORE hace que **MySQL** ignore las actualizaciones que **generan** una clave principal o exclusiva en lugar de cancelarlas.

La **palabra** clave **LOW PRIORITY** **hace** que la **actualización** espere hasta que no haya clientes leyendo la tabla antes de procesarla.

La expresión puede adoptar el valor actual de un campo; por ejemplo, para añadir 5 a las comisiones de todos los empleados, **podría** utilizar lo siguiente:

```
UPDATE employee SET commission=commission+5;
```

LIMIT determina el número máximo de registros que se va a actualizar

USE

```
USE nombre_de_base_de_datos
```

Cambia la base de datos actualmente activa por la base de datos especificada.

Funciones y operadores de MySQL

MySQL dispone de una gran cantidad de útiles operadores y funciones. Los operadores son uno de los elementos básicos de MySQL sin los que no podrá avanzar mucho. Por otra parte, existen muchas y complejas funciones, algunas de las cuales nunca utilizara, pero merece la pena repasar la lista completa ya que puede encontrar funciones que le sean de utilidad y otras que guarde en la recámara para utilizarlas más adelante.

Operadores lógicos

Los operadores lógicos, o booleanos, comprueban si algo es verdadero o falso. Devuelven 0 si la expresión es falsa y 1 si es verdadera. Los valores nulos se procesan de distintas formas, en función del operador. Normalmente devuelven un valor NULL.

AND, &&

```
valor1 AND valor1  
valor1 && valor2
```

Devuelven verdadero (1) si ambos valores son verdaderos

Por ejemplo:

```
mysql> SELECT 1 AND 0;
+-----+
| 1 AND 0 |
+-----+
|         0 |
+-----+
mysql> SELECT 1=1 && 2=2;
+-----+
| 1=1 && 2=2 |
+-----+
|           1 |
+-----+
```

OR, ||

```
valor1 OR valor2
valor1 || valor 2
```

Devuelven verdadero (1) si alguno de los dos valores, `valor1` o `valor2`, es verdadero.

Por ejemplo:

```
mysql> SELECT 1 OR 1;
+-----+
| 1 OR 1 |
+-----+
|         1 |
+-----+
mysql> SELECT 1=2 || 2=3;
+-----+
| 1=2 || 2=3 |
+-----+
|           0 |
+-----+
```

NOT, !

```
NOT valor1
! valor1
```

Devuelven lo contrario del `valor1`, que es verdadero si `valor1` es falso y falso si `valor2` es verdadero. Por ejemplo:

```
mysql> SELECT !1;
+-----+
| !1 |
+-----+
| 0 |
+-----+
```

```
mysql> SELECT NOT(1=2);
+-----+
| NOT(1=2) |
+-----+
|          1 |
+-----+
```

Operadores aritmeticos

Los operadores aritmeticos realizan calculos matematicos basicos. Si alguno de los valores es **nulo**, los resultados de toda la **operación** tambien lo seran. Por motivos del calculo, las **cadenas** se convierten en numeros. Algunas **cadenas** se convierten **al** numero equivalente (como las **cadenas** '1' y '33'); otras se convierten en 0 (como las **cadenas** 'uno' y 'abc').

+

`valor1 + valor2`

Suma dos valores.

Por ejemplo:

```
mysql> SELECT 1+3;
+----+
| 1+3 |
+----+
|    4 |
+----+
mysql> SELECT 15+"9";
+-----+
| 15+"9" |
+-----+
|      24 |
+-----+
```

-

`valor1 - valor2`

Resta **valor2** de **valor1**.

Por ejemplo:

```
mysql> SELECT 1-9;
+----+
| 1-9 |
+----+
|   -8 |
+----+
```

valor1 * valor2

Multiplca dos valores. Por ejemplo:

```
mysql> SELECT 12 * 10;
+-----+
| 12 * 10 |
+-----+
|      120 |
+-----+
```

/

valor1 / valor2

Divide valor1 por valor2. Por ejemplo:

```
mysql> SELECT 4/2;
+-----+
| 4/2 |
+-----+
| 2.00 |
+-----+
mysql> SELECT 10005.00000/10004.00000;
+-----+
| 10005.00000/10004.00000 |
+-----+
|                          1.0001000 |
+-----+
```

%

Devuelve el modulo (el resto que se obtiene despues de dividir valor1 por valor2). Por ejemplo:

```
mysql> SELECT 3%2;
+-----+
| 3%2 |
+-----+
|    1 |
+-----+
```

Operadores de comparacion

Los operadores de comparacion comparan valores y devuelven verdadero o falso (1 o 0) en función de los resultados. Si hay un valor nulo, en la mayoría de

los casos el operador devolvera el valor NULL. Se pueden comparar distintos tipos (cadenas, numeros, fechas, etc.) aunque si los tipos son diferentes tendra que prestar especial atencion. **MySQL convierte los tipos a su equivalente todo lo bien que puede.**

Si se trata de comparar **cadenas**, se comparan sin distinguir entre **mayúsculas** y **minúsculas**, a **menos** que **sean** BINARY. Por **ejemplo**, **A** es lo mismo que **a**, pero BINARY **A** no es lo mismo que BINARY **a**. En este **caso primero** vienen **las mayúsculas**, por lo que BINARY **A** es **menor** que BINARY **a**. Del mismo modo, la cadena **10** es **menor** que la cadena **2** porque, **al** ser una cadena, se compara de izquierda a derecha. La **primera** comprobacion es ver si **1** es **menor** que **2** y, como si lo es, la comprobacion se detiene en ese **punto** (lo mismo que az es anterior ab).

=

```
valor1 = valor2
```

Verdadero si **tanto** valor1 como valor2 son iguales. Si alguno es **nulo**, devolvera NULL.

Por **ejemplo**:

```
mysql> SELECT 1=2;
+----+
| 1=2 |
+----+
|    0 |
+----+
mysql> SELECT 'A' = 'a';
+-----+
| 'A' = 'a' |
+-----+
|          1 |
+-----+
mysql> SELECT BINARY 'a' = 'A';
+-----+
| BINARY 'a' = 'A' |
+-----+
|          0 |
+-----+
mysql> SELECT NULL=NULL;
+-----+
| NULL=NULL |
+-----+
|          NULL |
+-----+
```

!=, <>

```
valor1 <> valor2
valor1 != valor2
```


Verdadero si valor1 no es igual a valor2.

Por ejemplo:

```
mysql> SELECT 'a' != 'A';
+-----+
| 'a' != 'A' |
+-----+
|           0 |
+-----+
mysql> SELECT BINARY 'a' <> 'A';
+-----+
| BINARY 'a' <> 'A' |
+-----+
|           1 |
+-----+
```

>

valor1 > valor2

Verdadero si valor1 es mayor que valor2.

Por ejemplo:

```
mysql> SELECT 1>2;
+-----+
| 1>2 |
+-----+
|    0 |
+-----+
mysql> SELECT 'b'>'a';
+-----+
| 'b'>'a' |
+-----+
|          1 |
+-----+
```

<

valor1 < valor2

Verdadero si valor1 es menor que valor2.

Por ejemplo:

```
mysql> SELECT 'b' < 'd';
+-----+
| 'b' < 'd' |
+-----+
|           1 |
+-----+
mysql> SELECT '4' < '34';
+-----+
|           1 |
+-----+
```

```
| '4' < '34' |
+-----+
|           0 |
+-----+
```

>=

```
valor1 >= valor2
```

Verdadero si `valor1` es mayor o igual que `valor2`.

Por ejemplo:

```
mysql> SELECT 4 >= 4;
+-----+
| 4 >= 4 |
+-----+
|       1 |
+-----+
```

<=

```
valor1 <= valor2
```

Verdadero si `valor1` es **menor** o igual que `valor2`

Por ejemplo:

```
mysql> SELECT 4 <= 3;
+-----+
| 4 <= 3 |
+-----+
|       0 |
+-----+
```

<=>

```
valor1 <=> valor2
```

Verdadero si `valor1` es igual a `valor2`, incluyendo los valores nulos. Esto le **permite** creer que NULL es un valor real y, por lo **tanto**, obtener un resultado verdadero o falso (en lugar de NULL) cuando utilice NULL en una comparación con valores que no **sean** NULL. Por el contrario, **MySQL** se niega a dar una respuesta a la pregunta "¿4 es igual a NULL? En su lugar, indica que la expresión `4=NULL` tiene **como** resultado algo indeterminado (NULL).

Por ejemplo:

```
mysql> SELECT NULL<=>NULL;
+-----+
| NULL<=>NULL |
+-----+
```

```

|          1 |
+-----+
mysql> SELECT 4 <=> NULL;
+-----+
| 4 <=> NULL |
+-----+
|          0 |
+-----+

```

IS NULL

valor1 IS NULL

Verdadero si valor1 es nulo (no es falso).

Por ejemplo:

```

mysql> SELECT NULL IS NULL;
+-----+
| NULL IS NULL |
+-----+
|          1 |
+-----+
mysql> SELECT 0 IS NULL;
+-----+
| 0 IS NULL |
+-----+
|          0 |
+-----+

```

BETWEEN

valor1 BETWEEN valor2 AND valor3

Verdadero si valor1 esta incluido entre valor2 y valor3.

Por ejemplo:

```

mysql> SELECT 1 BETWEEN 0 AND 2;
+-----+
| 1 BETWEEN 0 AND 2 |
+-----+
|          1 |
+-----+
mysql> SELECT 'a' BETWEEN 'A' and 'B';
+-----+
| 'a' BETWEEN 'A' and 'B' |
+-----+
|          1 |
+-----+
mysql> SELECT BINARY 'a' BETWEEN 'A' AND 'C';
+-----+
| BINARY 'a' BETWEEN 'A' AND 'C' |
+-----+

```

```

+-----+
|                                             |
+-----+
0 |

```

LIKE

```
valor1 LIKE valor2
```

Verdadero si valor1 coincide con valor2 en un patron de coincidencia SQL. Un porcentaje (%) hace referencia a cualquier numero de caracteres y un guion bajo (_) equivale a un caracter.

Por ejemplo:

```

mysql> SELECT 'abc' LIKE 'ab_';
+-----+
| 'abc' LIKE 'ab_' |
+-----+
|                   | 1 |
+-----+
mysql> SELECT 'abc' LIKE '%c';
+-----+
| 'abc' LIKE '%c' |
+-----+
|                   | 1 |
+-----+

```

IN

```
valor1 IN (valor2 [valor3,...])
```

Verdadero si valor1 equivale a cualquier valor de la lista separada por comas.

Por ejemplo:

```

mysql> SELECT 'a' IN('b','c','aa');
+-----+
| 'a' IN('b','c','aa') |
+-----+
|                       | 0 |
+-----+
mysql> SELECT 'a' IN('A','B');
+-----+
| 'a' IN('A','B') |
+-----+
|                   | 1 |
+-----+

```

REGEXP, RLIKE

```
valor1 REGEXP valor2
```

valor1 RLIKE valor2

Verdadero si `valor1` coincide con `valor2` con una expresión regular. En la tabla **B.1** se enumeran los caracteres de expresiones regulares.

Tabla **B.1**. Caracteres de expresiones regulares

Carácter	Descripción
*	Coincide con ninguna o más instancias de la subexpresión que le precede.
+	Coincide con una o más instancias de la subexpresión que le precede.
?	Coincide con ninguna o más instancias de la subexpresión que le precede.
.	Coincide con todos los caracteres.
[xyz]	Coincide con x, y o z (alguno de los caracteres entre corchetes).
[A-Z]	Coincide con todas las letras mayúsculas.
[a-z]	Coincide con todas las letras minúsculas.
[0-9]	Coincide con todos los dígitos.
^	Ancla la coincidencia desde el principio.
\$	Ancla la coincidencia hasta el final.
	Separa las subexpresiones de la expresión regular.
{n,m}	La subexpresión debe aparecer al menos n veces, pero no más de m.
{n}	La subexpresión debe aparecer exactamente n veces.
{n, }	La subexpresión debe aparecer al menos n veces.
()	Agrupar caracteres en subexpresiones.

Por ejemplo:

```
mysql> SELECT 'pqwxyz' REGEXP 'xyz';
+-----+
| 'pqwxyz' REGEXP 'xyz' |
+-----+
|                          1 |
+-----+
mysql> SELECT 'xyz' REGEXP '^x';
+-----+
```

```

| 'xyz' REGEXP '^x' |
+-----+
|                   1 |
+-----+
mysql> SELECT 'abcdef' REGEXP 'g+' ;
+-----+
| 'abcdef' REGEXP 'g+' |
+-----+
|                   0 |
+-----+
mysql> SELECT 'abcdef' REGEXP 'g*';
+-----+
| 'abcdef' REGEXP 'g*' |
+-----+
|                   1 |
+-----+
mysql> SELECT 'ian' REGEXP 'iai*n';
+-----+
| 'ian' REGEXP 'iai*n' |
+-----+
|                   1 |
+-----+
mysql> SELECT 'aaaa' REGEXP 'a(3,)' ;
+-----+
| 'aaaa' REGEXP 'a(3,)' |
+-----+
|                   1 |
+-----+
mysql> SELECT 'aaaa' REGEXP '^aaa$';
+-----+
| 'aaaa' REGEXP '^aaa$' |
+-----+
|                   0 |
+-----+
mysql> SELECT 'abcabcabc' REGEXP 'abc{3}';
+-----+
| 'abcabcabc' REGEXP 'abc{3}' |
+-----+
|                   0 |
+-----+
mysql> SELECT 'abcabcabc' REGEXP '(abc)(3)';
+-----+
| 'abcabcabc' REGEXP '(abc){3}' |
+-----+
|                   1 |
+-----+
mysql> SELECT 'abcbbcccc' REGEXP '[abc](3)';
+-----+
| 'abcbbcccc' REGEXP '[abc](3)' |
+-----+
|                   1 |
+-----+
mysql> SELECT 'abcbbcccc' REGEXP '(a|b|c){3}';

```

```

+-----+
| 'abcbcccc' REGEXP '(a|b|c){3}' |
+-----+
|                                     | 1 |
+-----+

```

Operadores de bits

Los operadores de bits no se utilizan muy a menudo. Le permiten trabajar con valores de bits y realizar operaciones de bits en sus consultas.

&

```
valor1 & valor2
```

Realiza una operación AND en orden de bits. Convierte los valores a binarios y compara los bits. Solamente si ambos bits correspondientes son 1, el bit resultante también será 1.

Por ejemplo:

```

mysql> SELECT 2&1;
+----+
| 2&1 |
+----+
|    0 |
+----+
mysql> SELECT 3&1;
+----+
| 3&1 |
+----+
|    1 |
+----+

```

|

```
valor1 | valor2
```

Realiza una operación OR en orden de bits. Convierte los valores a binarios y compara los bits. Si alguno de los bits correspondientes es 1, el bit resultante también será 1.

Por ejemplo:

```

mysql> SELECT 2|1;
+----+
| 2|1 |
+----+
|    3 |
+----+

```

<<

```
valor1 << valor2
```

Convierte `valor1` a binario y desplaza los bits de `valor1` hacia la izquierda la cantidad de `valor2`.

Por ejemplo:

```
mysql> SELECT 2<<1;
+-----+
| 2<<1 |
+-----+
|    4 |
+-----+
```

>>

```
valor1 >> valor2
```

Convierte `valor1` en binario y desplaza sus bits hacia la derecha la cantidad de `valor2`.

Por ejemplo:

```
mysql> SELECT 2>>1;
+-----+
| 2>>1 |
+-----+
|    1 |
+-----+
```

Funciones de fecha y hora

Las funciones de fecha y hora se utilizan cuando trabajamos con tiempo, como al devolver la hora actual en un determinado formato o para ver cuantos dias quedan para una determinada fecha. En la mayoría de los casos, los valores de tipo `date` se almacenan como AAAA-MM-DD (por ejemplo 2002-12-25) y los valores de tipo `time` se almacenan como hh:mm:ss (por ejemplo, 11:23:43). También hay un tipo `datetime`, que se almacena como AAAA-MM-DD hh:mm:ss. La mayoría de las funciones que aceptan horas o fechas admiten el formato `datetime` e ignoran la parte que no necesitan. Del mismo modo, si no tiene muchos valores (cuando se le pide hh:mm:ss, sólo introduce la parte mm:ss), MySQL asumirá que el resto son ceros y realizará la operación correctamente. En lugar de dos puntos (:) y guiones (-), puede utilizar cualquier delimitador en las cadenas de fecha y hora, siempre que sea consistente.

Determinadas funciones utilizan un tipo de datos dado (por ejemplo, `DATE_ADD()`, que requiere un intervalo para realizar su cálculo).

A **continuación** mostramos los tipos de fecha y hora:

- SECOND
- MINUTE
- HOUR
- DAY
- MONTH
- YEAR
- MINUTE–SECOND: "**mm:ss**" (por ejemplo, "41:23")
- HOUR–MINUTE: "**hh:mm**" (por ejemplo, "12:23")
- DAY_HOUR: "DD hh" (por ejemplo, "11 09")
- YEAR–MONTH: "YYYY-MM" (por ejemplo, "2002-12")
- HOUR–SECOND: "**hh:mm:ss**" (por ejemplo, "11:24:36")
- DAY–MINUTE: "DD **hh:mm**" (por ejemplo, "09 11:31")
- DAY_SECONDS: "DD **hh:mm:ss**" (por ejemplo, "09 11:31:21")

Para realizar calculos de fechas, tambien puede utilizar los operadores habituales (+, -, etc.) en lugar de funciones de fecha. MySQL realiza correctamente las conversiones entre tipos. Cuando por ejemplo **añade** un mes **al** mes 12, MySQL **incrementa** el año y calcula correctamente los meses.

ADDDATE

`ADDDATE(fecha, INTERVAL tipo de expresión)`

Sinónimo de `DATE_ADD ()`.

CURDATE

`CURDATE ()`

Sinónimo de la funcion `CURRENT_DATE ()`.

CURRENT_DATE

`CURRENT – DATE ()`

Devuelve la fecha actual del sistema bien como la cadena **AAAA-MM-DD** o como el numero **AAAAMMDD**, en funcion del **contexto**. Por ejemplo:

```
mysql> SELECT CURRENT_DATE ( ) ;
```

```

+-----+
| CURRENT-DATE() |
+-----+
| 2002-09-10    |
+-----+
mysql> SELECT CURRENT_DATE()+1;
+-----+
| CURRENT-DATE()+1 |
+-----+
|                20020911 |
+-----+

```

CURRENT_TIME

`CURRENT-TIME()`

Devuelve la hora actual del sistema bien como la cadena **hh:mm:ss** o como el número **hhmmss**, en funcion del contexto. Por ejemplo:

```

mysql> SELECT CURRENT-TIME();
+-----+
| CURRENT-TIME() |
+-----+
| 23:53:15      |
+-----+
mysql> SELECT CURRENT-TIME() + 1;
+-----+
| CURRENT-TIME() + 1 |
+-----+
|                235434 |
+-----+

```

CURRENT_TIMESTAMP

`CURRENT-TIMESTAMP()`

Esta funcion equivale a la funcion `NOW()`.

CURTIME

`CURTIME()`

Sinónimo de la función `CURRENT_TIME()`

DATE-ADD

`DATE-ADD(fecha, INTERVAL tipo de expresion)`

Añade un determinado **periodo** de tiempo a la fecha especificada. Puede utilizar un valor negativo para la expresion, en cuyo **caso** se restara. El **tipo** debe ser

uno de los enumerados al principio de este apartado y la expresión debe coincidir con dicho tipo.

Por ejemplo:

```
mysql> SELECT DATE-ADD('2002-12-25',INTERVAL 1 MONTH);
+-----+
| DATE-ADD('2002-12-25',INTERVAL 1 MONTH) |
+-----+
| 2003-01-25                               |
+-----+
mysql> SELECT DATE_ADD('2002-12-25 13:00:00',INTERVAL -14
HOUR);
+-----+
| DATE_ADD('2002-12-25 13:00:00',INTERVAL -14 HOUR) |
+-----+
| 2002-12-24 23:00:00                               |
+-----+
```

DATE_FORMAT

DATE_FORMAT(*fecha*,*cadena_formato*)

Aplica un formato a la fecha especificada en función de la cadena de formato, que puede estar formada por los especificadores enumerados en la tabla B.2.

Tabla B.2. Especificadores de formato de fecha

Especificador	Descripción
%a	Abreviatura del nombre del día (Dom-Sab)
%b	Abreviatura del nombre del mes (Ene-Dic)
%c	Mes numerico (1-12)
%D	Día numerico del mes con sufijo en Inglés (1st, 2nd, etc.)
%d	Día numerico del mes con dos digitos, comprendido entre 00 y 31
%e	Día numerico del mes con uno o dos digitos, comprendido entre 0 y 31
%H	Hora comprendida entre 00 y 23
%h	Hora comprendida entre 01 y 12
%I	Minutos comprendidos entre 00 y 59
%I	Hora comprendida entre 01 y 12
%j	Día del año, comprendido entre 001-366

Especificador	Descripción
%k	Hora con uno o dos digitos, comprendida entre 0 y 23
%l	Hora con un digito, comprendida entre 1 y 12
%M	Nombre del mes, Enero-Diciembre
%m	Mes numerico, comprendido entre 01 y 12
%p	A.M. o P.M.
%r	Horario de 12 horas, hh:mm:ss A.M. o P.M.
%S	Segundos comprendidos entre 00 y 59
%s	Segundos comprendidos entre 00 y 59
%T	Horario de 24 horas, hh:mm:ss
%U	Semana comprendida entre 00 y 53, siendo el domingo el primer dia de la semana
%u	Semana comprendida entre 00 y 53, siendo el lunes el primer dia de la semana
%V	Semana comprendida entre 01 y 53, siendo el domingo el primer dia de la semana
%v	Semana comprendida entre 01 y 53, siendo el lunes el primer dia de la semana
%W	Nombre del dia de la semana, comprendido entre domingo y sabado
%w	Dia de la semana comprendido entre 0 para el domingo y 6 para el sabado
%X	Año numerico de cuatro digitos para la semana en la que el domingo es el primer dia
%x	Año numerico de cuatro digitos para la semana en la que el lunes es el primer dia
%Y	Año numerico de cuatro digitos
%y	Año numerico de dos digitos
%8	Signo de porcentaje con conversion de escape

Por ejemplo:

```
mysql> SELECT DATE-FORMAT('1999-03-02','%c %M');
+-----+
| DATE-FORMAT('1999-03-02','%c %M') |
+-----+
| 3 March                               |
```

DATE_SUB

DATE-SUB(*fecha*, INTERVAL *tipo de expresion*)

Resta un determinado **periodo** de tiempo de la fecha especificada. Puede **utili-**zar un valor negativo en la expresion, en cuyo **caso** se sumara. El **tipo** debe ser uno de los enumerados **al** principio de este **apartado** y la expresion debe coincidir con este tipo.

Por ejemplo:

```
mysql> SELECT DATE-SUB('2002-12-25 13:00:00',INTERVAL "14:13"
MINUTE-SECOND);
+-----+
| DATE_SUB('2002-12-25 13:00:00',INTERVAL "14:13"
MINUTE-SECOND) |
+-----+
| 2002-12-25 12:45:47
|
+-----+
```

DAYNAME

DAYNAME(*fecha*)

Devuelve el nombre del dia de la fecha especificada.

Por ejemplo:

```
mysql> SELECT DAYNAME('2000-12-25');
+-----+
| DAYNAME('2000-12-25') |
+-----+
| Monday                 |
+-----+
```

DAYOFMONTH

DAYOFMONTH(*fecha*)

Devuelve el dia del mes de la fecha proporcionada **como** un numero **compre-**ndido entre 1 y 31.

Por ejemplo:

```
mysql> SELECT DAYOFMONTH('2000-01-01');
+-----+
| DAYOFMONTH('2000-01-01') |
+-----+
|                            1 |
+-----+
```

DAYOFWEEK

`DAYOFWEEK(fecha)`

Devuelve el día de la semana de la fecha proporcionada como número comprendido entre 1, para el domingo, y 7 para el sábado, estándar ODBC.

Por ejemplo:

```
mysql> SELECT DAYOFWEEK('2000-01-01');
+-----+
| DAYOFWEEK('2000-01-01') |
+-----+
|                          7 |
+-----+
```

Utilice `WEEKDAY()` para devolver el índice de día comprendido entre 0 y 6, de lunes a domingo.

DAYOFYEAR

`DAYOFYEAR(fecha)`

Devuelve el día del año de la fecha proporcionada como un número comprendido entre 1 y 366.

Por ejemplo:

```
mysql> SELECT DAYOFYEAR('2000-12-25');
+-----+
| DAYOFYEAR('2000-12-25') |
+-----+
|                          360 |
+-----+
```

EXTRACT

`EXTRACT(tipo_de_fecha FROM fecha)`

Utiliza el tipo de fecha especificada para devolver la parte de la fecha. Puede consultar la lista de tipos de fechas que aparece antes del inicio de las funciones de fecha.

Por ejemplo:

```
mysql> SELECT EXTRACT(YEAR FROM '2002-02-03');
+-----+
| EXTRACT(YEAR FROM '2002-02-03') |
+-----+
|                          2002 |
+-----+
mysql> SELECT EXTRACT(MINUTE-SECOND FROM '2002-02-03
12:32:45');
```

```

+-----+
| EXTRACT(MINUTE_SECOND FROM '2002-02-03 12:32:45') |
+-----+
|                                     3245 |
+-----+

```

FROM_DAYS

`FROM_DAYS (numero)`

Convierte el numero especificado a una fecha basada en el numero de dias transcurridos desde el 1 de enero del año 0, y devuelve el resultado. No tiene en cuenta los dias perdidos en el cambio al calendario Gregoriano.

Por ejemplo:

```

mysql> SELECT FROM_DAYS(731574);
+-----+
| FROM_DAYS(731574) |
+-----+
| 2002-12-25      |
+-----+

```

FROM_UNIXTIME

`FROM_UNIXTIME(marca_de_tiempo_unix [, cadena_formato])`

Convierte la marca de tiempo especificada en una fecha y devuelve el resultado. A la fecha devuelta se le aplica un formato si se proporciona una cadena de formato. La cadena de formato puede ser cualquiera de las de la funcion `DATE_FORMAT()`. Por ejemplo:

```

mysql> SELECT FROM_UNIXTIME(100);
+-----+
| FROM_UNIXTIME(100) |
+-----+
| 1970-01-01 00:01:40 |
+-----+
mysql> SELECT FROM_UNIXTIME(1031621727,'%c %M');
+-----+
| FROM_UNIXTIME(1031621727,'%c %M') |
+-----+
| 9 September |
+-----+

```

HOUR

`HOUR(hora)`

Devuelve la hora de la hora especificada, comprendida entre 0 y 23.

Por ejemplo:

```
mysql> SELECT HOUR('06:59:03');
+-----+
| HOUR('06:59:03') |
+-----+
|                   6 |
+-----+
```

MINUTE

MINUTE(hora)

Devuelve los minutos de la hora especificada, comprendidos entre 0 y 59.

Por ejemplo:

```
mysql> SELECT MINUTE('00:01:03');
+-----+
| MINUTE('00:01:03') |
+-----+
|                   1 |
+-----+
```

MONTH

MONTH(fecha)

Devuelve el mes de la fecha especificada, comprendido entre 1 y 12.

Por ejemplo:

```
mysql> SELECT MONTH('2000-12-25');
+-----+
| MONTH('2000-12-25') |
+-----+
|                   12 |
+-----+
```

MONTHNAME

MONTHNAME(fecha)

Devuelve el nombre del mes de la fecha especificada.

Por ejemplo:

```
mysql> SELECT MONTHNAME('2000-12-25');
+-----+
| MONTHNAME('2000-12-25') |
+-----+
| December                 |
+-----+
```


NOW

NOW()

Devuelve la marca de hora actual (fecha y hora en formato AAAA-MM-DD hh:mm:ss), bien como cadena o como numero en funcion del contexto. La funcion devolvera el mismo resultado en varias llamadas a una misma consulta.

Por ejemplo:

```
mysql> SELECT NOW();
+-----+
| NOW()          |
+-----+
| 2002-09-10 00:58:06 |
+-----+
```

Equivale a las funciones CURRENT_TIMESTAMP() y SYSDATE()

PERIOD-ADD

PERIOD-ADD(periodo, meses)

Aiade los meses al periodo (especificado como AAMM o AAAAMM) y devuelve el resultado como AAAAMM.

Por ejemplo:

```
mysql> SELECT PERIOD-ADD(200205,3);
+-----+
| PERIOD-ADD(200205,3) |
+-----+
|          200208      |
+-----+
mysql> SELECT PERIOD-ADD(200205,-42);
+-----+
| PERIOD-ADD(200205,-42) |
+-----+
|          199811      |
+-----+
```

PERIOD_DIFF

PERIOD_DIFF(periodo1, periodo2)

Devuelve el numero de meses comprendidos entre periodo1 y periodo2 (que se especifican en el formato AAMM o AAAAMM).

Por ejemplo:

```
mysql> SELECT PERIOD_DIFF(200212,200001);
+-----+
| PERIOD_DIFF(200212,200001) |
+-----+
```

```

|                                     35 |
+-----+
mysql> SELECT PERIOD-DIFF(199903,199904);
+-----+
| PERIOD-DIFF(199903,199904) |
+-----+
|                               -1 |
+-----+

```

QUARTER

`QUARTER (fecha)`

Devuelve el trimestre de la fecha especificada, comprendido entre 1 y 4.
Por ejemplo:

```

mysql> SELECT QUARTER('2002-06-30');
+-----+
| QUARTER('2002-06-30') |
+-----+
|                               2 |
+-----+

```

SEC-TO-TIME

`SEC-TO-TIME (segundos)`

Convierte los segundos en hora y devuelve una cadena (hh:mm:ss) o un número (hhmmss), en función del contexto. **Por ejemplo:**

```

mysql> SELECT SEC-TO-TIME(1000);
+-----+
| SEC-TO-TIME(1000) |
+-----+
| 00:16:40          |
+-----+
mysql> SELECT SEC-TO-TIME(-10000);
+-----+
| SEC-TO-TIME(-10000) |
+-----+
| -02:46:40          |
+-----+

```

SECOND

`SECOND (hora)`

Devuelve los segundos de la hora especificada, comprendidos entre 0 y 58
Por ejemplo:

```
mysql> SELECT SECOND('00:01:03');
```

```

+-----+
| SECOND('00:01:03') |
+-----+
|                      3 |
+-----+

```

SUBDATE

SUBDATE(*fecha*, INTERVAL *tipo de expresión*)

Sinónimo de DATE_SUB().

SYSDATE

SYSDATE()

Sinónimo de la función NOW().

TIME_FORMAT

TIME_FORMAT(*hora*, *formato*)

Identico a DATE_FORMAT() a excepción de que solamente se puede utilizar el subconjunto de formatos relacionados con la hora (de lo contrario, devolverá NULL).

TIME_TO_SEC

TIME_TO_SEC(*hora*)

Convierte la hora en segundos y devuelve el resultado. Por ejemplo:

```

mysql> SELECT TIME_TO_SEC('00:01:03');
+-----+
| TIME_TO_SEC('00:01:03') |
+-----+
|                      63 |
+-----+

```

TO_DAYS

TO_DAYS(*fecha*)

Devuelve el número de días transcurridos desde el 1 de enero del año 0 en la fecha especificada. No tiene en cuenta los días perdidos debido al cambio al calendario Gregoriano. Por ejemplo:

```
mysql> SELECT TO_DAYS('2000-01-01');
```

```

+-----+
| TO_DAYS('2000-01-01') |
+-----+
|                730485 |
+-----+

```

UNIX_TIMESTAMP

UNIX_TIMESTAMP ([fecha])

Devuelve un entero sin firma que representa la marca de tiempo Unix (los segundos transcurridos desde medianoche del 1 de enero de 1970) bien de la hora del sistema (si se invoca sin un parametro) o bien de la fecha especificada.

Por ejemplo:

```

mysql> SELECT UNIX_TIMESTAMP();
+-----+
| UNIX_TIMESTAMP() |
+-----+
|        1031621727 |
+-----+
mysql> SELECT UNIX_TIMESTAMP('1970-01-01 00:01:40');
+-----+
| UNIX_TIMESTAMP('1970-01-01 00:01:40') |
+-----+
|                                     100 |
+-----+

```

WEEK

WEEK(date [,inicio_semana])

Devuelve la semana de un determinado aïo de la fecha especificada, comprendida entre 0 y 53. Se asume que la semana empieza el domingo, a menos que se defina como 1 el argumento opcional *inicio_semana*, en cuyo caso la semana empezara en lunes.

Tambien se puede definir explícitamente en 0 para que empiece en domingo. La funcion devolvera 0 en fechas anteriores al primer domingo (o lunes) del aïo.

Por ejemplo:

```

mysql> SELECT WEEK('2002-06-31');
+-----+
| WEEK('2002-06-31') |
+-----+
|                26 |
+-----+
mysql> SELECT WEEK('2002-06-31',1);
+-----+

```

```

| WEEK('2002-06-31',1) |
+-----+
|                          27 |
+-----+
mysql> SELECT WEEK('1998-12-31',1);
+-----+
| WEEK('1998-12-31',1) |
+-----+
|                          53 |
+-----+
mysql> SELECT WEEK('1998-01-01');
+-----+
| WEEK('1998-01-01') |
+-----+
|                          0 |
+-----+

```

Utilice la función `WEEKDAY()` para desplazar la semana por el año anterior si la fecha es anterior al primer **domingo** (o lunes) del año.

WEEKDAY

`WEEKDAY (fecha)`

Devuelve el día de la semana de la fecha proporcionada como un número comprendido entre 0 (lunes) y 6 (domingo).

Por ejemplo:

```

mysql> SELECT WEEKDAY('2000-01-01');
+-----+
| WEEKDAY('2000-01-01') |
+-----+
|                          5 |
+-----+

```

Utilice `DAYOFWEEK()` para devolver el índice de día según el estándar ODBC (1-7, domingo-sábado).

YEAR

`YEAR (fecha)`

Devuelve el año de la fecha especificada, comprendido entre 1000 y 9999.

Por ejemplo:

```

mysql> SELECT YEAR('2002-06-30');
+-----+
| YEAR('2002-06-30') |
+-----+
|                2002 |
+-----+

```

YEARWEEK

`YEARWEEK (fecha [, inicio_semana])`

Devuelve una **combinación** del aiiio y la semana de la fecha especificada. Se asume que la semana empieza el domingo, a **menos** que defina como 1 el **argumento opcional inicio_semana**, en cuyo **caso** se **supone** que empieza en lunes. **También** se puede **configurar** explícitamente como 0 para que **empiece** en **domingo**. El aiiio puede ser el aiiio anterior a la fecha en fechas anteriores **al primer domingo** (o lunes) del presente aiiio o del siguiente. Por ejemplo:

```
mysql> SELECT YEARWEEK('2002-12-25');
+-----+
| YEARWEEK('2002-12-25') |
+-----+
|                200251 |
+-----+
mysql> SELECT YEARWEEK('1998-12-31',1);
+-----+
| YEARWEEK('1998-12-31',1) |
+-----+
|                199853 |
+-----+
mysql> SELECT YEARWEEK('1998-01-01');
+-----+
| YEARWEEK('1998-01-01') |
+-----+
|                199752 |
+-----+
```

Utilice la función `WEEK()` para devolver la semana en un determinado aiiio.

Funciones de cadena

Las **funciones** de cadena suelen adoptar **argumentos** de cadena y devolver **resultados** de cadena. **Al** contrario de lo que **ocurre** en la **mayoría** de los lenguajes de **programación**, el primer carácter de la cadena se corresponde a la **posición** 1, no a 0.

ASCII

`ASCII (cadena)`

Devuelve el valor ASCII del primer carácter (el que se encuentra más a la izquierda), 0 si la cadena está vacía y NULL si la cadena es **nula**. Por ejemplo:

```
mysql> SELECT ASCII('a');
+-----+
| ASCII('a') |
+-----+
```

```

| _____ 97 |
+-----+
mysql> SELECT ASCII('aa');
+-----+
| ASCII('az') |
+-----+
| _____ 97 |
+-----+

```

Utilice `ORD()` para devolver el valor ASCII si el caracter es un caracter multibyte.

BIN

`BIN(numero)`

Devuelve el valor binario (una **representación** de cadena) del numero BIGINT especificado, 0 si el numero no se puede convertir (la funcion lo convertira siempre que pueda hacerlo desde la izquierda) y NULL si es nulo.

Por ejemplo:

```

mysql> SELECT BIN(15);
+-----+
| BIN(15) |
+-----+
| 1111    |
+-----+
mysql> SELECT BIN('8');
+-----+
| BIN('8') |
+-----+
| 1000     |
+-----+
mysql> SELECT BIN('2w');
+-----+
| BIN('2w') |
+-----+
| 10       |
+-----+
mysql> SELECT BIN('w2');
+-----+
| BIN('w2') |
+-----+
| 0        |
+-----+

```

Esta funcion equivale a `CONV(number, 10, 2)`.

BIT_LENGTH

`BIT-LENGTH(cadena)`

Devuelve la longitud de la cadena en bits.

Por ejemplo:

```
mysql> SELECT BIT-LENGTH('MySQL');
+-----+
| BIT_LENGTH('MySQL') |
+-----+
| 40                   |
+-----+
```

CHAR

CHAR(número1[, número2[, ...]])

Esta función devuelve los caracteres que se obtendrían si **cada** número fuera un entero convertido desde código ASCII, ignorando los valores nulos. Los decimales se redondean al valor entero **más** próximo. Por ejemplo:

```
mysql> SELECT CHAR(97,101,105,111,117);
+-----+
| CHAR(97,101,105,111,117) |
+-----+
| aeiou                    |
+-----+
mysql> SELECT CHAR(97.6,101,105,111,117);
+-----+
| CHAR(0.97,101,105,111,117) |
+-----+
| beiou                    |
+-----+
```

CHAR-LENGTH

Sinónimo de la función LENGTH(), a excepción de que los caracteres multibyte solamente se cuentan una vez.

CARÁCTER_LENGTH

Sinónimo de la función LENGTH(), a excepción de que los caracteres multibyte solamente se cuentan una vez.

CONCAT

CONCAT(cadena1[, cadena2[, ...]])

Concatena los argumentos de la cadena y devuelve la cadena resultante o NULL si algún argumento es NULL. Los argumentos que no son **cadena**s se convierten a **cadena**s.

Por ejemplo:

```
mysql> SELECT CONCAT('a','b');
+-----+
| CONCAT('a','b') |
+-----+
| ab              |
+-----+
mysql> SELECT CONCAT('a',12);
+-----+
| CONCAT('a',12) |
+-----+
| a12            |
+-----+
mysql> SELECT CONCAT(.3,'NULL');
+-----+
| CONCAT(.3,'NULL') |
+-----+
| 0.3NULL           |
+-----+
mysql> SELECT CONCAT(.3,NULL);
+-----+
| CONCAT(.3,NULL) |
+-----+
| NULL            |
+-----+
```

CONCAT-WS

CONCAT-WS(separador, cadena1[, cadena2[, ...]])

Similar a CONCAT a excepción de que el primer argumento es un separador situado entre cada una de las cadenas concatenadas. Ignora todas las cadenas nulas (a excepción del separador, en cuyo caso el resultado sera NULL). No es necesario que el separador sea una cadena.

Por ejemplo:

```
mysql> SELECT CONCAT-WS('-', 'a', 'b');
+-----+
| CONCAT_WS('-', 'a', 'b') |
+-----+
| a-b                       |
+-----+
mysql> SELECT CONCAT-WS(1, .3, .4);
+-----+
| CONCAT_WS(1, .3, .4) |
+-----+
| 0.310.4              |
+-----+
mysql> SELECT CONCAT-WS(NULL, 'a', 'b');
+-----+
| CONCAT_WS(NULL, 'a', 'b') |
```

```

+-----+
| NULL          |
+-----+
mysql> SELECT CONCAT_WS('-', 'a', NULL, 'c');
+-----+
| CONCAT_WS('-', 'a', NULL, 'c') |
+-----+
| a-c          |
+-----+

```

CONV

CONV(numero, de_base, a_base)

Convierte un número de una base a otra. Devuelve el número convertido **representado** como cadena. 0 si la conversión no se puede realizar (la **función** realizara la conversión siempre que pueda hacerlo desde la izquierda) y NULL si el número es **nulo**. Se **supone** que el número es un entero, **pero** se puede pasar como cadena. Se **supone** que no tiene **firma a menos** que la base sea un número negativo. Las bases pueden ser cualquier valor comprendido entre 2 y 36 (a_base puede ser negativo).

Por ejemplo:

```

mysql> SELECT CONV(10, 2, 10);
+-----+
| CONV(10, 2, 10) |
+-----+
| 2              |
+-----+
mysql> SELECT CONV('a', 16, 2);
+-----+
| CONV('a', 16, 2) |
+-----+
| 1010            |
+-----+
mysql> SELECT CONV('3f', 16, 10);
+-----+
| CONV('3f', 16, 10) |
+-----+
| 63              |
+-----+

mysql> SELECT CONV('z3', 16, 10);
+-----+
| CONV('z3', 16, 10) |
+-----+
| 0                |
+-----+
1 row in set (0.00 sec)
mysql> SELECT CONV('3z', 16, 10);
+-----+

```

```
| CONV('3z',16,10) |
+-----+| 3      |
+-----+
```

ELT

ELT(numero, cadena1 [,cadena2, ...])

Utiliza **numero** como **índice** para determinar que cadena devuelve; **1** devuelve la **primera** cadena, **2** la segunda y así sucesivamente. Devuelve **NULL** si no hay ninguna cadena que coincida.

Por ejemplo:

```
mysql> SELECT ELT(2,'one','two');
+-----+
| ELT(2,'one','two') |
+-----+
| two                |
+-----+
mysql> SELECT ELT(0,'one','two');
+-----+
| ELT(0,'one','two') |
+-----+
| NULL                |
+-----+
```

La función **FIELD()** es el complemento de **ELT()**.

EXPORT-SET

EXPORT_SET(número,on,off[,separador[,número_de_bits]])

Examina **numero** en binario y, por cada bit que se defina, devuelve **on**; por cada bit que no se defina, devuelve **off**. El **separador** predeterminado es una coma, **pero** puede especificar cualquier otro. Se utilizan **64 bits** **pero** puede cambiar el valor de **número-de-bits**.

Por ejemplo:

```
mysql> SELECT EXPORT-SET(2,1,0,' ',4);
+-----+
| EXPORT_SET(2,1,0,' ',4) |
+-----+
| 1 0 1 0 0              |
+-----+
mysql> SELECT EXPORT-SET(7,'ok','never',' : ',6);
+-----+
| EXPORT_SET(7,'ok','never',' : ',6) |
+-----+
| ok : ok : ok : never : never : never |
+-----+
```

FIELD

FIELD(cadena, cadena1 [, cadena2 , ...])

Devuelve el **índice** de cadena en la lista siguiente. Si cadena1 coincide, el **índice** sera 1.

Si se trata de cadena2, sera 2 y así sucesivamente. Devuelve 0 si no se encuentra la cadena.

Por ejemplo:

```
mysql> SELECT FIELD('b','a','b','c');
+-----+
| FIELD('b','a','b','c') |
+-----+
|                        2 |
+-----+
mysql> SELECT FIELD('a','aa','b','c');
+-----+
| FIELD('a','aa','b','c') |
+-----+
|                        0 |
+-----+
```

FIND_IN_SET

FIND_IN_SET(cadena, lista de cadenas)

Similar a FIELD() ya que devuelve un **índice** que coincide con la cadena, **pero** esta **función** busca bien una cadena separada por comas o el tipo SET. Devuelve 1 si la cadena coincide con la **primera** subcadena antes de la coma (el **elemento** del conjunto), 2 si coincide la segunda subcadena, y así sucesivamente. Devuelve 0 si no encuentra coincidencias.

Apreciara que coincide con subcadenas completas separadas por comas, no sólo con **partes** de la cadena.

Por ejemplo:

```
mysql> SELECT FIND_IN_SET('b','a,b,c');
+-----+
| FIND_IN_SET('b','a,b,c') |
+-----+
|                        2 |
+-----+
mysql> SELECT FIND_IN_SET('a','aa,bb,cc');
+-----+
| FIND_IN_SET('a','aa,bb,cc') |
+-----+
|                        0 |
+-----+
1 row in set (0.00 sec)
```

HEX

`HEX(cadena o numero)`

Devuelve el valor hexadecimal (una **representación** de cadena) del numero BIGINT especificado, 0 si el numero no se puede convertir (la **función** lo hara siempre que pueda realizar la conversion desde la izquierda) o NULL si es **nulo**.

Si el **argumento** es un numero, se convierte a hexadecimal (similar a la **función** `CONV(numero, 10, 16)`). Si es una cadena, cada caracter de la misma se convierte a su equivalente **numérico** en la tabla ASCII (por ejemplo, a = 97, b = 98, etc.) y, a su vez, cada uno de estos numeros se convierten a su equivalente hexadecimal.

Por ejemplo:

```
mysql> SELECT HEX(13);
```

```
+-----+
| HEX(13) |
+-----+
| D       |
+-----+
```

```
mysql> SELECT ORD('a');
```

```
+-----+
| ORD('a') |
+-----+
|          97 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT ORD('b');
```

```
+-----+
| ORD('b') |
+-----+
|          98 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT HEX(97);
```

```
+-----+
| HEX(97) |
+-----+
| 61      |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT HEX(98);
```

```
+-----+
| HEX(98) |
+-----+
| 62      |
+-----+
```

```
1 row in set (0.00 sec)
```

```
mysql> SELECT HEX('ab');
```

```
+-----+
| HEX('ab') |
+-----+
| 6162      |
+-----+
```

```
1 row in set (0.00 sec)
```

INSERT

```
INSERT(cadena, posición, longitud, nueva_cadena)
```

Sustituye la **parte** de la cadena que empieza en **posición** y continua la **longitud** de caracteres con **nueva_cadena**. La longitud de **nueva_cadena** y la **longitud** especificada **pueden** ser diferentes, en cuyo **caso** cambiará la longitud de la cadena original.

La función es segura para multibytes.

Por ejemplo:

```
mysql> SELECT INSERT('MySQL',1,0,'What is ');
```

```
+-----+
| INSERT('MySQL',1,0,'What is ') |
+-----+
| What is MySQL                  |
+-----+
```

```
mysql> SELECT INSERT('MySQL',1,1,'P');
```

```
+-----+
| INSERT('MySQL',1,1,'P') |
+-----+
| PySQL                    |
+-----+
```

```
mysql> SELECT INSERT('MySQL',1,1,'Py');
```

```
+-----+
| INSERT('MySQL',1,1,'Py') |
+-----+
| PyySQL                   |
+-----+
```

INSTR

```
INSTR(cadena, subcadena)
```

Busca en la **cadena**, sin distinguir entre mayúsculas y minúsculas (a **menos** que una de las **cadena**s sea binaria), la **primera** instancia de **subcadena** y **devuelve** la **posición** o devuelve **0** si **subcadena** no se encuentra. La **primera** letra se encuentra en la posición 1. Por ejemplo:

```
mysql> SELECT INSTR('MySQL','My');
```

```

+-----+
| INSTR('MySQL','My') |
+-----+
|                      1 |
+-----+
mysql> SELECT INSTR('Cecilia','i');
+-----+
| INSTR('Cecilia','i') |
+-----+
|                      4 |
+-----+

```

LCASE

`LCASE (cadena)`

Sinónimo de `LOWER()`.

LEFT

`LEFT (cadena, longitud)`

Devuelve los caracteres `longitud` que se encuentran mas a la izquierda en la cadena. Es segura para multibytes.

Por ejemplo:

```

mysql> SELECT LEFT('abc',2);
+-----+
| LEFT('abc',2) |
+-----+
| ab           |
+-----+

```

LENGTH

`LENGTH (cadena)`

Devuelve la longitud de la `cadena` en caracteres. Convierte el argumento en cadena si puede.

Por ejemplo:

```

mysql> SELECT LENGTH('MySQL');
+-----+
| LENGTH('MySQL') |
+-----+
|                 5 |
+-----+
mysql> SELECT LENGTH(99);
+-----+
| LENGTH(99) |

```

```

+-----+
|           2 |
+-----+

```

CHAR_LENGTH(), CHARACTER_LENGTH() y OCTET_LENGTH() son sinónimos, a excepción de que los caracteres multibyte solamente se cuentan una vez con CHAR_LENGTH() y CHARACTER_LENGTH().

LOAD-FILE

```
LOAD-FILE(nombre_de_archivo)
```

Lee el archivo y devuelve los contenidos del mismo como cadena. El archivo debe encontrarse en el servidor, debe especificar la ruta completa al archivo y debe tener el privilegio FILE. El archivo debe ser legible para todos y ser mas pequeño que max_allowed_packet. Si el archivo no existe o no se puede leer por alguna de las razones anteriores, la función devuelve NULL.

Por ejemplo, si el archivo /home/iang/test.txt contiene el texto 123456, con LOAD_FILE() se devolvería lo siguiente:

```

mysql> SELECT LOAD-FILE('/home/iang/test.txt');
+-----+
| LOAD_FILE('/home/iang/test.txt') |
+-----+
| 123456                            |
+-----+

```

Normalmente se utiliza LOAD_FILE() para cargar BLOB en la base de datos. Por ejemplo:

```

mysql> INSERT INTO table-with_blob(id,image)
VALUES(1,LOAD_FILE('/images/pic.jpg'));

```

LOCATE

```
LOCATE(subcadena, cadena [,posición])
```

Busca la primera instancia de subcadena en la cadena sin distinguir entre mayúsculas y minúsculas (a menos que una de las cadenas sea binaria) y devuelve la posición o 0 si no encuentra la subcadena. Si se proporciona el argumento opcional posición, la búsqueda empieza desde ese punto. La primera letra se encuentra en la posición 1.

Por ejemplo:

```

mysql> SELECT LOCATE('My','MySQL');
+-----+
| LOCATE('My','MySQL') |
+-----+
|                       1 |

```



```

+-----+
mysql> SELECT LOCATE('C','Cecilia',2);
+-----+
| LOCATE('C','Cecilia',2) |
+-----+
|                          3 |
+-----+

```

Es similar a la función `INSTR()` pero con los argumentos invertidos.

LOWER

`LOWER(cadena)`

Devuelve una **cadena** con todos los caracteres convertidos a minúsculas (en función del conjunto de caracteres actual). La función es segura para multibytes.

Por ejemplo:

```

mysql> SELECT LOWER('AbC');
+-----+
| LOWER('AbC') |
+-----+
| abc          |
+-----+

```

La función `LCASE()` es un sinónimo de esta.

LPAD

`LPAD(cadena, longitud, cadena_relleno)`

Rellena la cadena a la izquierda con **cadena_relleno** hasta que el resultado tenga los caracteres indicados en **longitud**. Si la cadena es mayor que la longitud, se reducirá en la **cantidad** de caracteres indicados en **longitud**.

Por ejemplo:

```

mysql> SELECT LPAD('short',7,'-');
+-----+
| LPAD('short',7,'-') |
+-----+
| -short              |
+-----+
mysql> SELECT LPAD('too-long',7,' ');
+-----+
| LPAD('too-long',7,' ') |
+-----+
| too-lon            |
+-----+
mysql> SELECT LPAD('a',4,'12');
+-----+
| LPAD('a',4,'12') |

```

```

+-----+
| 121a          |
+-----+

```

LTRIM

`LTRIM(cadena)`

Elimina los espacios situados por delante de la cadena y devuelve el resultado.

Por ejemplo:

```

mysql> SELECT LTRIM('  Yes');
+-----+
| LTRIM('  Yes') |
+-----+
| Yes             |
+-----+

```

MAKE_SET

`MAKE-SET(numero, cadena1 [, cadena2, ...])`

Devuelve un **conjunto** (una cadena en la que los elementos **están** separados por comas) con las **cadena**s que coinciden con el numero convertido a binario. La **primera** cadena aparece si se configura el bit 0, la segunda si se configura el bit 1 y así sucesivamente. Si el **argumento** de bits se define con el valor 3, se devuelven las dos **primeras** cadenas ya que 3 es 11 en binario. Por ejemplo:

```

mysql> SELECT MAKE-SET(3,'a','b','c');
+-----+
| MAKE_SET(3,'a','b','c') |
+-----+
| a,b                     |
+-----+
mysql> SELECT MAKE_SET(5,'a','b','c');
+-----+
| MAKE-SET(5,'a','b','c') |
+-----+
| a,c                     |
+-----+

```

OCT

`OCT(numero)`

Devuelve el valor octal (una **representación de cadenas**) del numero **BIGINT** especificado, 0 si el numero no se puede convertir (la **función** lo intentara desde la **parte** mas a la izquierda) o **NULL** si es **nulo**.

Por ejemplo:

```
mysql> SELECT OCT(09);
+-----+
| OCT(09) |
+-----+
| 11      |
+-----+
mysql> SELECT OCT('a1');
+-----+
| OCT('a1') |
+-----+
| 0         |
+-----+
mysql> SELECT OCT('13b');
+-----+
| OCT('13b') |
+-----+
| 15        |
+-----+
```

Esta funcion equivale a `CONV(numero, 10, 8)`.

OCTET_LENGTH

Sinónimo de la funcion `LENGTH`.

ORD

`ORD(cadena)`

Devuelve el valor ASCII del primer caracter de la cadena (el que se encuentra más a la izquierda), 0 si la cadena esta vacia y NULL si la cadena es nula. Es similar a la funcion `ASCII`, a excepción de que el caracter es un caracter multibyte, en cuyo caso el valor se calcula como un numero de base 256, es decir, cada byte vale 256 veces mas que el siguiente. Por ejemplo, la formula para un caracter de 2 bytes seria la siguiente: $(\text{byte_1_código ASCII} * 256) + (\text{byte_2_código ASCII})$.

Por ejemplo:

```
mysql> SELECT ORD("a");
+-----+
| ORD("a") |
+-----+
| 97      |
+-----+
mysql> SELECT ORD("az");
+-----+
| ORD("az") |
+-----+
| 97      |
```

+-----+

Puede utilizar **las funciones** `BIN()`, `OCT()` y `HEX()` para **convertir** numeros decimales en binarios, octales y hexadecimales respectivamente.

POSITION

`POSITION(subcadena IN cadena)`

Busca, sin distinguir entre mayusculas y minusculas (a **menos** que uno de los argumentos sea una cadena binaria), la **primera** instancia de `subcadena` en `cadena` y devuelve la **posición** (empezando desde 1) o 0 si no encuentra `subcadena`. La funcion es segura para multibytes.

Por ejemplo:

```
mysql> SELECT POSITION('i' IN 'Cecilia');
+-----+
| POSITION('i' IN 'Cecilia') |
+-----+
|                          | 4 |
+-----+
```

QUOTE

`QUOTE(cadena)`

Utiliza la conversion de escape correspondiente a **los** caracteres de comilla simple ('), de comillas dobles ("), NULL ASCII y Control-Z, y rodea la cadena con comillas simples para que se pueda utilizar con seguridad en una **instrucción SQL**. No es necesario utilizar comillas simples si el argumento es NULL.

Por ejemplo:

```
mysql> SELECT QUOTE("What's Up?");
+-----+
| QUOTE("What's Up?") |
+-----+
| 'What\'s Up?'      |
+-----+
```

REPEAT

`REPEAT(cadena, numero)`

Repite el argumento de la cadena tantas veces **como** indique `numero` y **devuelve** el resultado, devuelve una cadena vacia si `numero` no es positivo o devuelve NULL si alguno de los argumentos es **nulo**. Por ejemplo:

```
mysql> SELECT REPEAT('a',4);
+-----+
```

```

| REPEAT('a',4) |
+-----+
| aaaa          |
+-----+
mysql> SELECT REPEAT('a',-1);
+-----+
| REPEAT('a',-1) |
+-----+
|                 |
+-----+
mysql> SELECT REPEAT('a',NULL);
+-----+
| REPEAT('a',NULL) |
+-----+
| NULL              |
+-----+

```

REPLACE

`REPLACE(cadena, de_string, a_string)`

Sustituye todas las instancias de `de_cadena` que encuentre en la cadena por `a_cadena` y devuelve el resultado. La función es segura para multibytes.

Por ejemplo:

```

mysql> SELECT REPLACE('ftp://test.host.co.za','ftp','http');
+-----+
| REPLACE('ftp://test.host.co.za','ftp','http') |
+-----+
| http://test.host.co.za                        |
+-----+

```

REVERSE

`REVERSE(cadena)`

Invierte el orden de los caracteres de `cadena` y devuelve el resultado. Esta función es segura para multibytes.

Por ejemplo:

```

mysql> SELECT REVERSE('abc');
+-----+
| REVERSE('abc') |
+-----+
| cba             |
+-----+

```

RIGHT

`RIGHT(cadena, longitud)`

Devuelve el número de caracteres especificados en `longitud` situados más a la derecha. Esta **función** es segura para multibytes.

Por ejemplo:

```
mysql> SELECT RIGHT('abc',2);
+-----+
| RIGHT('abc',2) |
+-----+
| bc              |
+-----+
```

RPAD

`RPAD(cadena, longitud, cadena_relleno)`

Rellena la cadena a la derecha con `cadena_relleno` hasta que el resultado tenga **los** caracteres especificados en `longitud`. Si la cadena es mayor que la longitud, se reduce el número de caracteres especificados en `longitud`.

Por ejemplo:

```
mysql> SELECT RPAD('short',7,'-');
+-----+
| RPAD('short',7,'-') |
+-----+
| short-              |
+-----+
mysql> SELECT RPAD('too-long',7,' ');
+-----+
| RPAD('too-long',7,' ') |
+-----+
| too-lon            |
+-----+
mysql> SELECT RPAD('a',4,'12');
+-----+
| RPAD('a',4,'12') |
+-----+
| a121              |
+-----+
```

RTRIM

`RTRIM(cadena)`

Elimina **los** espacios situados **al** final de la cadena y devuelve el resultado.

Por ejemplo:

```
mysql> SELECT CONCAT('a',RTRIM('b          '), 'c');
+-----+
| CONCAT('a',RTRIM('b          '), 'c') |
+-----+
```

```
| abc
+-----+
```

SOUNDEX

SOUNDEX(*cadena*)

Devuelve una cadena **soundex**, una cadena fonetica diseiada para indexar errores ortograficos de **forma** mas eficaz. Las **cadena**s que suenan de la misma **forma** **tendrán** las mismas **cadena**s **soundex**. Normalmente tiene una longitud de cuatro caracteres **pero** esta funcion devuelve una cadena de longitud arbitraria. Utilice la funcion **SUBSTRING()** **sobre** **SOUNDEX()** para devolver una cadena **soundex** **estándar**. Los caracteres no alfanumericos se ignoran y los caracteres alfabeticos no ingleses se tratan **como** vocales.

Por ejemplo:

```
mysql> SELECT SOUNDEX('MySQL');
+-----+
| SOUNDEX('MySQL') |
+-----+
| M240              |
+-----+
mysql> SELECT SOUNDEX('MySequ');
+-----+
| SOUNDEX('MySequ') |
+-----+
| M240              |
+-----+
```

SPACE

SPACE(*numero*)

Devuelve una cadena formada por el número de espacios. Por ejemplo:

```
mysql> SELECT "A",SPACE(10),"B";
+--+-----+--+
| A | SPACE(10) | B |
+--+-----+--+
| A |           | B |
+--+-----+--+
```

SUBSTRING

SUBSTRING(*cadena*, *posición* [,*longitud*])
SUBSTRING(*cadena* FROM *posición* [FOR *longitud*])

Devuelve una subcadena del **argumento** de cadena comenzado desde **posición** (que empieza en 1) y, opcionalmente, con la **longitud** especificada.

Por ejemplo:

```
mysql> SELECT SUBSTRING('MySQL',2);
+-----+
| SUBSTRING('MySQL',2) |
+-----+
| ySQL                  |
+-----+
mysql> SELECT SUBSTRING('MySQL' FROM 3);
+-----+
| SUBSTRING('MySQL' FROM 3) |
+-----+
| SQL                      |
+-----+
1 row in set (0.16 sec)
mysql> SELECT SUBSTRING('MySQL',1,2);
+-----+
| SUBSTRING('MySQL',1,2) |
+-----+
| My                      |
+-----+
1 row in set (0.22 sec)
```

La función es segura para multibytes. La función `MID(cadena, posición, longitud)` equivale a `SUBSTRING(cadena, posición, longitud)`.

SUBSTRING_INDEX

`SUBSTRING_INDEX(cadena, delimitador, número)`

Devuelve la subcadena de la cadena hasta llegar a `número` (si es positivo) o después de `número` (si es negativo) de instancias de delimitador.

La función es segura para multibytes.

Por ejemplo:

```
mysql> SELECT SUBSTRING_INDEX('a||b||c||d','||',3);
+-----+
| SUBSTRING_INDEX('a||b||c||d','||',3) |
+-----+
| a||b||c                               |
+-----+
mysql> SELECT SUBSTRING_INDEX('I am what I am','a',2);
+-----+
| SUBSTRING_INDEX('I am what I am','a',2) |
+-----+
| I am wh                                 |
+-----+
mysql> SELECT SUBSTRING_INDEX('I am what I am','a',-2);
+-----+
| SUBSTRING_INDEX('I am what I am','a',-2) |
```



```

+-----+
| t I am |
+-----+

```

TRIM

```

TRIM([ [BOTH | LEADING | TRAILING] [cadena_recorte] FROM]
cadena)

```

Si no se especifica ninguno de los **parámetros** opcionales, TRIM() **elimina los** espacios anteriores y posteriores. Puede indicar LEADING o TRAILING para eliminar solamente uno de los dos tipos o **utilizar la opción** predeterminada BOTH. También puede eliminar otros elementos **además** de espacios si especifica **cadena_recorte**. La **función** es segura para multibytes.

Por ejemplo:

```

mysql> SELECT TRIM(' What a waste of space ') AS t;
+-----+
| t |
+-----+
| What a waste of space |
+-----+
mysql> SELECT TRIM(LEADING '0' FROM '0001');
+-----+
| TRIM(LEADING '0' FROM '0001') |
+-----+
| 1 |
+-----+
mysql> SELECT TRIM(LEADING FROM ' 1');
+-----+
| TRIM(LEADING FROM ' 1') |
+-----+
| 1 |
+-----+
mysql> SELECT TRIM(BOTH 'abc' FROM 'abcabcaabbccabcabc');
+-----+
| TRIM(BOTH 'abc' FROM 'abcabcaabbccabcabc') |
+-----+
| aabbcc |
+-----+

```

UCASE

```

UCASE(cadena)

```

Sinonimo de UPPER().

UPPER

```

UPPER(cadena)

```

Devuelve una cadena con todos los caracteres convertidos en mayusculas (en funcion del conjunto de caracteres actual). La funcion es segura para multibytes. Por ejemplo:

```
mysql> SELECT UPPER('aBc');
+-----+
| UPPER('aBc') |
+-----+
| ABC          |
+-----+
```

Su sinonimo es la funcion UCASE().

Funciones numericas

Las funciones numericas trabajan con numeros y, basicamente, adoptan argumentos numericos y devuelven resultados numericos. En caso de que se produzca un error, devuelven NULL. Tendra que prestar especial atencion y no superar el ambito numérico de un numero; la mayoría de las funciones MySQL funcionan con BIGINT (2⁶³ con signo o 2⁶⁴ sin signo) y si se supera este ambito, MySQL devolvera NULL.

ABS

ABS (numero)

Devuelve el valor absoluto (valor positivo) de un numero. La funcion se puede utilizar con BIGINT.

Por ejemplo:

```
mysql> SELECT ABS(24-26);
+-----+
| ABS(24-26) |
+-----+
|           2 |
+-----+
```

ACOS

ACOS (numero)

Devuelve el arco coseno del numero (el coseno inverso). El numero debe estar comprendido entre -1 y 1 o la funcion devolvera NULL.

Por ejemplo:

```
mysql> SELECT ACOS(0.9);
+-----+
```

```
| ACOS(0.9) |
+-----+
| 0.451027 |
+-----+
```

ASIN

ASIN(número)

Devuelve el **arco seno** del **número** (el seno inverso). El número debe estar comprendido entre -1 y 1 o la función devolverá NULL.

Por ejemplo:

```
mysql> SELECT ASIN(-0.4);
+-----+
| ASIN(-0.4) |
+-----+
| -0.411517 |
+-----+
```

ATAN

ATAN(número1 [, número2])

Devuelve la tangente del número (la tangente **inversa**) o de dos números (el **punto** número1, número2). Por ejemplo:

```
mysql> SELECT ATAN(4);
+-----+
| ATAN(4) |
+-----+
| 1.325818 |
+-----+
mysql> SELECT ATAN(-4,-3);
+-----+
| ATAN(-4,-3) |
+-----+
| -2.214297 |
+-----+
```

ATAN2

ATAN2(número1, número2)

Sinónimo de ATAN(número1, número2).

CEILING

CEILING(número)

Redondea el número **al** entero más próximo y lo devuelve como **BGINT**.
Por ejemplo:

```
mysql> SELECT CEILING(2.98);
+-----+
| CEILING(2.98) |
+-----+
|                3 |
+-----+
mysql> SELECT CEILING(-2.98);
+-----+
| CEILING(-2.98) |
+-----+
|                -2 |
+-----+
```

Utilice **FLOOR()** para redondear **hacia** abajo y **ROUND()** para hacerlo **hacia** arriba o **hacia** abajo.

COS

COS(número_radianes)

Devuelve el coseno de **numero_radianes**.

Por ejemplo:

```
mysql> SELECT COS(51);
+-----+
| COS(51) |
+-----+
| 0.742154 |
+-----+
```

COT

COT(número_radianes)

Devuelve la cotangente de **numero_radianes**.

Por ejemplo:

```
mysql> SELECT COT(0.45);
+-----+
| COT(0.45) |
+-----+
| 2.07015736 |
+-----+
```

DEGREES

DEGREES(numero)

Convierte el numero de radianes a grados y devuelve el resultado.

Por ejemplo:

```
mysql> SELECT DEGREES(2);
+-----+
| DEGREES(2) |
+-----+
| 114.59155902616 |
+-----+
mysql> SELECT DEGREES(PI()/2);
+-----+
| DEGREES(PI()/2) |
+-----+
| 90 |
+-----+
```

EXP

EXP(numero)

Devuelve el numero e (la base de logaritmos naturales) elevado a la potencia especificada.

Por ejemplo:

```
mysql> SELECT EXP(1);
+-----+
| EXP(1) |
+-----+
| 2.718282 |
+-----+
mysql> SELECT EXP(2.3);
+-----+
| EXP(2.3) |
+-----+
| 9.974182 |
+-----+
mysql> SELECT EXP(0.3);
+-----+
| EXP(0.3) |
+-----+
| 1.349859 |
+-----+
```

FLOOR

FLOOR(número)

Redondea el numero hacia abajo hasta el entero más proximo y lo devuelve como BIGINT. Por ejemplo:

```
mysql> SELECT FLOOR(2.98);
```

```

+-----+
| FLOOR(2.98) |
+-----+
|           2 |
+-----+
mysql> SELECT FLOOR(-2.98);
+-----+
| FLOOR(-2.98) |
+-----+
|           -3 |
+-----+

```

Utilice **CEILING()** para redondear **hacia arriba** y **ROUND()** para redondear **hacia arriba** o **hacia abajo**.

FORMAT

FORMAT(numero,decimales)

Aplica un **formato** a un numero de **forma** que **cada** tres digitos se separen por una coma y redondea el resultado hasta el numero de posiciones indicado.

Por ejemplo:

```

mysql> SELECT FORMAT(88777634.232,2);
+-----+
| FORMAT(88777634.232,2) |
+-----+
| 88,777,634.23          |
+-----+

```

GREATEST

GREATEST(argument01, argument02 [, ...])

Devuelve el mayor de **los** argumentos. Los argumentos se comparan de **distintas formas** en funcion del **contexto** del valor devuelto o de **los** tipos de argumento, que pueden ser enteros, **reales** o **cadenas** (que distinguen entre mayusculas y **mi-** nusculas, y son la **opcion** predeterminada).

Por ejemplo:

```

mysql> SELECT GREATEST(-3,-4,5);
+-----+
| GREATEST(-3,-4,5) |
+-----+
|           5 |
+-----+
mysql> SELECT GREATEST('Pa','Ma','Ca');
+-----+
| GREATEST('Pa','Ma','Ca') |
+-----+

```

```
| Pa
+-----+
```

LEAST

```
LEAST(argument01, argument02 [, ...])
```

Devuelve el **menor** de los argumentos. Los argumentos se comparan de **distintas formas** en función del **contexto** del valor devuelto o de los tipos de argumento, que pueden ser enteros, **reales** o **cadenas** (que distinguen entre mayúsculas y minúsculas, y son la opción predeterminada).

Por ejemplo:

```
mysql> SELECT LEAST(-3,-4,5);
+-----+
| LEAST(-3,-4,5) |
+-----+
|                -4 |
+-----+
mysql> SELECT LEAST('Pa','Ma','Ca');
+-----+
| LEAST('Pa','Ma','Ca') |
+-----+
| Ca                    |
+-----+
```

LN

```
LN(numero)
```

Sinónimo de la función LOG (numero)

LOG

```
LOG(número1 [, número2])
```

Devuelve el logaritmo natural de **numero1** si hay un argumento. También puede utilizar un base arbitraria si proporciona un segundo argumento, en cuyo caso la función devuelve LOG (numero2) /LOG (numero1).

Por ejemplo:

```
mysql> SELECT LOG(2);
+-----+
| LOG(2) |
+-----+
| 0.693147 |
+-----+
mysql> SELECT LOG(2,3);
+-----+
```

```

| LOG(2,3) |
+-----+
| 1.584963 |
+-----+

```

LOG10

LOG10(*numero1*)

Devuelve el logaritmo de base 10 de *numero1*. Equivale a LOG(*numero1*) / LOG(10).

Por ejemplo:

```

mysql> SELECT LOG10(100);
+-----+
| LOG10(100) |
+-----+
| 2.000000 |
+-----+

```

LOG2

LOG2(*numero1*)

Devuelve el logaritmo de base 2 de *numero1*. Equivale a LOG(*numero1*) / LOG(2).

Por ejemplo:

```

mysql> SELECT LOG2(4);
+-----+
| LOG2(4) |
+-----+
| 2.000000 |
+-----+

```

MOD

MOD(*numero1*, *numero2*)

Devuelve el modulo de *numero1* y *numero2* (el resto de *numero1* dividido por *numero2*). Es similar al operador %. Se puede utilizar con BIGINT.

Por ejemplo:

```

mysql> SELECT MOD(15,4);
+-----+
| MOD(15,4) |
+-----+
|          3 |
+-----+
mysql> SELECT MOD(3,-2);

```



```

+-----+
| MOD(3,-2) |
+-----+
|          1 |
+-----+

```

PI

PI ()

Devuelve el valor de pi (o al menos la representación mas proxima). MySQL utiliza precision doble pero, de forma predeterminada, solamente devuelve cinco caracteres. Por ejemplo:

```

mysql> SELECT PI ();
+-----+
| PI ()      |
+-----+
| 3.141593   |
+-----+
mysql> SELECT PI () + 0.0000000000000000;
+-----+
| PI () + 0.0000000000000000 |
+-----+
|          3.1415926535897931 |
+-----+

```

POW

POW (número1, número2)

Esta funcion equivale a POWER (numerol, número2).

POWER

POWER (número1, número2)

Eleva numerol a la potencia de número2 y devuelve el resultado. Por ejemplo:

```

mysql> SELECT POWER(2,3);
+-----+
| POWER(2,3) |
+-----+
| 8.000000   |
+-----+

```

RADIANS

RADIANS (numerol)

Convierte el numero de grados a radianes y devuelve el resultado.

Por ejemplo:

```
mysql> SELECT RADIANS(180);
+-----+
| RADIANS(180) |
+-----+
| 3.1415926535898 |
+-----+
```

RAND

`RAND([numero])`

Devuelve un numero aleatorio (coma flotante) comprendido entre 0 y 1. El argumento es el generador de numeros aleatorios. Se suele utilizar la marca de tiempo **como** generador. Esta **función** se puede utilizar para devolver un **conjunto** de resultados en **orden** aleatorio.

Por ejemplo:

```
mysql> SELECT RAND();
+-----+
| RAND() |
+-----+
| 0.70100469486881 |
+-----+
mysql> SELECT RAND(20021010081523);
+-----+
| RAND(20021010081523) |
+-----+
| 0.80558716673924 |
+-----+
mysql> SELECT * FROM t1 ORDER BY RAND() LIMIT 1;
+----+
| f1 |
+----+
| 20 |
+----+
```

ROUND

`ROUND(numero1 [, numero2])`

Devuelve el argumento `numero1`, redondeado **al** entero mas proximo. Puede proporcionar un segundo argumento para **especificar** el numero de decimales que debe redondear (el predeterminado es 0, sin decimales). El comportamiento de redondeo para **los** numeros situados exactamente en el medio se basa en la **biblioteca** de C subyacente. Por ejemplo:

```
mysql> SELECT ROUND(2.49);
```

```

+-----+
| ROUND(2.49) |
+-----+
|           2 |
+-----+
mysql> SELECT ROUND(2.51);
+-----+
| ROUND(2.51) |
+-----+
|           3 |
+-----+
mysql> SELECT ROUND(-2.49,1);
+-----+
| ROUND(-2.49,1) |
+-----+
|          -2.5 |
+-----+

```

SIGN

`SIGN(numero)`

Devuelve -1, 0 o 1 en función de si el argumento es negativo, cero o no es un número, o positivo. Por ejemplo:

```

mysql> SELECT SIGN(-7);
+-----+
| SIGN(-7) |
+-----+
|        -1 |
+-----+
mysql> SELECT SIGN('a');
+-----+
| SIGN('a') |
+-----+
|          0 |
+-----+

```

SIN

`SIN(número_radianes)`

Devuelve el seno de `numero_radianes`.

Por ejemplo:

```

mysql> SELECT SIN(45);
+-----+
| SIN(45) |
+-----+
| 0.850904 |
+-----+

```

SQRT

SQRT(numero)

Devuelve la raíz cuadrada del argumento.

Por ejemplo:

```
mysql> SELECT SQRT(81);
+-----+
| SQRT(81) |
+-----+
| 9.000000 |
+-----+
```

TAN

TAN(número_radianes)

Devuelve la tangente de numero_radianes.

Por ejemplo:

```
mysql> SELECT TAN(66);
+-----+
| TAN(66) |
+-----+
| 0.026561 |
+-----+
```

TRUNCATE

TRUNCATE(numero,decimales)

Reduce (o aumenta) el numero al numero de decimales especificado.

Por ejemplo:

```
mysql> SELECT TRUNCATE(2.234,2);
+-----+
| TRUNCATE(2.234,2) |
+-----+
|                2.23 |
+-----+
mysql> SELECT TRUNCATE(2.4,5);
+-----+
| TRUNCATE(2.4,5) |
+-----+
|                2.40000 |
+-----+
mysql> SELECT TRUNCATE(2.998,0);
+-----+
| TRUNCATE(2.998,0) |
+-----+
```

```

|          2 |
+-----+
mysql> SELECT TRUNCATE(-12.43,1);
+-----+
| TRUNCATE(-12.43,1) |
+-----+
|          -12.4 |
+-----+

```

Funciones agregadas

Las **funciones agregadas** son las que trabajan con un grupo de datos (lo que **significa** que se pueden utilizar en una **cláusula GROUP BY**). Si no existe esta cláusula, se asume que el grupo es **todo el conjunto** de resultados y devuelven solamente un resultado. En los siguientes ejemplos, imagine que existe una sencilla tabla como esta:

```

mysql> SELECT * FROM table1;
+-----+
| field1 |
+-----+
|      4 |
|     12 |
|     12 |
|     20 |
+-----+
4 rows in set (0.00 sec)

```

AVG

AVG(expression)

Devuelve la media de las expresiones del grupo. Devuelve 0 si no es una expresión **numérica**. Por ejemplo:

```

mysql> SELECT AVG(field1) FROM table1;
+-----+
| AVG(field1) |
+-----+
|    12.0000 |
+-----+

```

BIT_AND

BIT-AND(expression)

Devuelve el operador AND en **orden** de bits de todos los bits de las expresiones del grupo (con una precisión de 64 bits).

```
mysql> SELECT BIT-AND(field1) FROM table1;  
+-----+  
| BIT_AND(field1) |  
+-----+  
|                4 |  
+-----+
```

BIT_OR

BIT-OR(*expression*)

Devuelve el operador OR de todos los bits de las expresiones del grupo (con una precisión de 64 bits).

Por ejemplo:

```
mysql> SELECT BIT-OR(field1) FROM table1;  
+-----+  
| BIT_OR(field1) |  
+-----+  
|                28 |  
+-----+
```

COUNT

COUNT([DISTINCT] *expresión1*, [*expresión2*])

Devuelve el numero de valores no nulos del grupo.

Si la expresion es un campo, devuelve el numero de filas que no contienen valores nulos en dicho campo. **COUNT** (*), el numero de todas las filas, nulas o no. La opción **DISTINCT** devuelve el numero de valores no nulos exclusivos (o una combinación, si se utiliza mas de una expresion).

```
mysql> SELECT COUNT(*) FROM table1;  
+-----+  
| COUNT(*) |  
+-----+  
|         4 |  
+-----+
```

MAX

MAX (*expression*)

Devuelve el mayor valor de las expresiones del grupo. La expresion puede ser numérica o una cadena.

Por ejemplo:

```
mysql> SELECT MAX(field1) FROM table1;  
+-----+  
| MAX(field1) |
```

```

+-----+
|          20 |
+-----+

```

MIN

MIN(expression)

Devuelve el valor mas **pequeño** de las expresiones del grupo. La expresion puede ser un numero o una cadena.

Por ejemplo:

```

mysql> SELECT MIN(field1) FROM table1;
+-----+
| MIN(field1) |
+-----+
|          4 |
+-----+

```

STD

STD(expression)

Devuelve la desviacion estandar de los valores de las expresiones del grupo.

Por ejemplo:

```

mysql> SELECT STD(field1) FROM table1;
+-----+
| STD(field1) |
+-----+
| 5.6569 |
+-----+

```

STDDEV

STDDEV(expression)

Sinonimo de la funcion STD()

SUM

SUM(expression)

Devuelve el valor mas pequefio de las expresiones del grupo o NULL si no hay filas. La expresion puede ser un numero o una cadena.

Por ejemplo:

```

mysql> SELECT SUM(field1) FROM table1;
+-----+

```

```

| MIN(field1) |
+-----+
|              | 48 |
+-----+

```

Otras funciones

Entre las siguientes funciones se incluyen funciones de cifrado, de comparación, de flujo de control y otros tipos de diversa naturaleza.

AES_DECRYPT

```
AES_DECRYPT(cadena_cifrada,cadena_clave)
```

Descifra el resultado de una función AES_ENCRYPT ().

AES_ENCRYPT

```
AES_ENCRYPT(cadena,cadena_clave)
```

Utiliza el algoritmo estándar de cifrado avanzado (Rijndael) para cifrar la cadena en función de **cadena_clave**. De forma predeterminada, utiliza una longitud de clave de 128 bits. AES_DECRYPT () descifra el resultado.

BENCHMARK

```
BENCHMARK(número,expresión)
```

Ejecuta la **expresión** un número de veces. Se utiliza principalmente para probar la velocidad a la que MySQL ejecuta una expresión. Siempre devuelve 0; el tiempo (en el cliente) que se muestra por debajo de la función es la parte útil del resultado.

Por ejemplo:

```

mysql> SELECT BENCHMARK(10000,SHA('how long'));
+-----+
| BENCHMARK(10000,SHA('how long')) |
+-----+
|                                     0 |
+-----+
1 row in set (0.95 sec)

```

CASE

```
CASE valor WHEN [valor1_comparación] THEN resultado1 [WHEN
[valor2_comparación]
```



```

THEN resultado2 ...] [ELSE resultado3] END
CASE WHEN [condicion1] THEN resultado1 [WHEN [condición2]
THEN resultado2 ...] [ELSE resultado3] END

```

La instrucción CASE tiene dos formas. La primera devuelve un resultado en función del valor. Compara el valor con los distintos **valores_comparacion** y devuelve el resultado asociado a dicho valor (por **detrás** de THEN), devuelve el resultado por detrás de ELSE si no encuentra nada o devuelve NULL si no hay resultados que devolver.

La segunda compara las distintas condiciones y devuelve el resultado asociado cuando encuentra una **condición** verdadera, devuelve el resultado por detrás de ELSE si no encuentra ninguna o devuelve NULL si no hay resultados que devolver.

Por ejemplo:

```

mysql> SELECT CASE 'a' WHEN 'a' THEN 'a it is' END;
+-----+
| CASE 'a' WHEN 'a' THEN 'a it is' END |
+-----+
| a it is                               |
+-----+
mysql> SELECT CASE 'b' WHEN 'a' THEN 'a it is' WHEN 'b' THEN 'b
it is' END;
+-----+
| CASE 'b' WHEN 'a' THEN 'a it is' WHEN 'b' THEN 'b it is' END
|
+-----+
| b it is                               |
|
+-----+
mysql> SELECT CASE 9 WHEN 1 THEN 'is 1' WHEN 2 THEN 'is 2' ELSE
'not found' END;
+-----+
| CASE 9 WHEN 1 THEN 'is 1' WHEN 2 THEN 'is 2' ELSE 'not found'
END |
+-----+
| not found                             |
|
+-----+
mysql> SELECT CASE 9 WHEN 1 THEN 'is 1' WHEN 2 THEN 'is 2' END;
+-----+
| CASE 9 WHEN 1 THEN 'is 1' WHEN 2 THEN 'is 2' END |
+-----+
| NULL                                   |
+-----+
mysql> SELECT CASE WHEN 1>2 THEN '1>2' WHEN 2=2 THEN 'is 2'
END;
+-----+
| CASE WHEN 1>2 THEN '1>2' WHEN 2=2 THEN 'is 2' END |
+-----+
| is 2                                   |

```

```

+-----+
mysql> SELECT CASE WHEN 1>2 THEN '1>2' WHEN 2<2 THEN '2<2' ELSE
'none' END;
+-----+
| CASE WHEN 1>2 THEN '1>2' WHEN 2<2 THEN '2<2' ELSE 'none' END
|
+-----+
| none
|
+-----+
mysql> SELECT CASE WHEN BINARY 'a' = 'A' THEN 'bin' WHEN
'a'='A' THEN 'text' END;
+-----+
| CASE WHEN BINARY 'a' = 'A' THEN 'bin' WHEN 'a'='A' THEN
'text' END |
+-----+
| text
|
+-----+
mysql> SELECT CASE WHEN BINARY 1=1 THEN '1' WHEN 2=2 THEN '2'
END;
+-----+
| CASE WHEN BINARY 1=1 THEN '1' WHEN 2=2 THEN '2' END |
+-----+
| 1
|
+-----+

```

El tipo de valor devuelto (INTEGER, DOUBLE o STRING) es igual que el tipo del primer valor devuelto (la expresion que aparece detras del primer THEN).

CAST

CAST(*expresion AS tipo*)

Convierte la expresion **al tipo** especificado y devuelve el resultado. Los tipos pueden ser uno de **los siguientes**: BINARY, DATETIME, SIGNED, SIGNED INTEGER, TIME, UNSIGNED y UNSIGNED INTEGER.

Normalmente, **MySQL** convierte **los tipos** automaticamente. Por **ejemplo**, si **añade** dos **cadenas** numericas, el resultado sera numerico. 0, si una parte de un calculo no tiene **firma**, el resultado no tendra **firma**. Puede utilizar CAST() para modificar este comportamiento.

Por ejemplo:

```

mysql> SELECT "4" + "3";
+-----+
| "4" + "3" |
+-----+
|          7 |
+-----+
mysql> SELECT CAST(("4"+"3") AS TIME);
+-----+

```

```

| CAST(("4"+"3") AS TIME) |
+-----+
| 7 |
+-----+
mysql> SELECT CAST(50-60 AS UNSIGNED INTEGER);
+-----+
| CAST(50-60 AS UNSIGNED INTEGER) |
+-----+
| 18446744073709551606 |
+-----+
mysql> SELECT CAST(50-60 AS SIGNED INTEGER);
+-----+
| CAST(50-60 AS SIGNED INTEGER) |
+-----+
| -10 |
+-----+

```

Utilice `CONVERT()` como sinónimo que utiliza sintaxis ODBC.

CONNECTIONID

`CONNECTION-ID()`

Devuelve el `id_de_subproceso` exclusivo de la conexión

Por ejemplo:

```

mysql> SELECT CONNECTION-ID();
+-----+
| CONNECTION-ID() |
+-----+
| 7 |
+-----+

```

CONVERT

`CONVERT(expresion,tipo)`

Sinónimo de `CAST(expresión AS tipo)` que es la sintaxis SQL99 ANSI.

DATABASE

`DATABASE()`

Devuelve el nombre de la base de datos actual o una cadena vacía en caso de que no haya ninguna.

Por ejemplo:

```

mysql> SELECT DATABASE();
+-----+

```

```
| DATABASE() |
+-----+
| test      |
+-----+
```

DECODE

DECODE(cadena_codificada,cadena_contraseña)

Descodifica la cadena codificada por medio de la cadena de contraseña y devuelve el resultado.

La cadena descodificada suele ser generada en primer lugar por la función ENCODE().

Por ejemplo:

```
mysql> SELECT DECODE('g','1');
+-----+
| DECODE('g','1') |
+-----+
| a                |
+-----+
mysql> SELECT DECODE('wer','1sz');
+-----+
| DECODE('wer','1sz') |
+-----+
| 8                    |
+-----+
```

DES_DECRYPT

DES_DECRYPT(cadena_cifrada [, cadena-clave])

Descodifica una cadena codificada con DES_ENCRYPT().

DES_ENCRYPT

DES_ENCRYPT(cadena [, (número_clave | cadena-clave)])

Utiliza el algoritmo DES para codificar la cadena y devuelve una cadena binaria. Si se omite el argumento de clave **opcional**, se utiliza la **primera** clave del archivo de claves de descodificación.

Si el argumento es un número (comprendido entre 0 y 9), se utiliza la correspondiente clave de archivo de claves de descodificación. Si el argumento es una cadena, se utilizará dicha clave.

Si los valores de clave cambian en el archivo de claves de descodificación, MySQL puede leer los nuevos valores cuando ejecute una instrucción FLUSH_DES_KEY_FILE, que requiere el **permiso reload**.

Esta función solamente funciona si MySQL es compatible con SSL.

ENCODE

ENCODE(cadena, cadena_contraseña)

Devuelve una cadena **binaria** codificada. Puede utilizar `DECODE()` con la misma `cadena_contraseña` para devolver la cadena original. Las **cadena**s codificada y **descodificada** tendrán la misma longitud. Por ejemplo:

```
mysql> SELECT ENCODE('a','1');
+-----+
| ENCODE('a','1') |
+-----+
| g                |
+-----+
mysql> SELECT ENCODE('ah','2');
+-----+
| ENCODE('ah','2') |
+-----+
| U                |
+-----+
```

ENCRYPT

ENCRYPT(cadena [, salt])

Codifica una cadena con la llamada del sistema `crypt()` de Unix y devuelve el resultado. El **argumento opcional** es una cadena utilizada en la **codificación**. Su comportamiento específico depende de la llamada del sistema subyacente.

Por ejemplo:

```
mysql> SELECT ENCRYPT('keepmeout');
+-----+
| ENCRYPT('keepmeout') |
+-----+
| V9tOly.dRY55k      |
+-----+
mysql> SELECT ENCRYPT('keepmeout','ab');
+-----+
| ENCRYPT('keepmeout','ab') |
+-----+
| abpr3o3DrHzJo      |
+-----+
```

FOUND_ROWS

FOUND_ROWS()

Devuelve el número de **filas** que **cumplen la consulta** `SELECT SQL_CALC_FOUND_ROWS` **anterior** (o que se habrían devuelto si no estuviera **limitado con una cláusula LIMIT**).

Por ejemplo:

```
mysql> SELECT SQL_CALC_FOUND_ROWS user FROM user LIMIT 1;
+-----+
| user |
+-----+
|      |
+-----+
1 row in set (0.00 sec)

mysql> SELECT FOUND-ROWS();
+-----+
| FOUND-ROWS() |
+-----+
|              | 4 |
+-----+
```

GET_LOCK

`GET_LOCK(cadena, tiempo_muerto)`

Intenta obtener un bloqueo denominado **cadena**, durante los segundos especificados en **tiempo_muerto**. Devuelve 1 si es satisfactorio, 0 si se queda sin tiempo o NULL si se produce algún otro error. El bloqueo se libera con `RELEASE_LOCK()`, una nueva función `GET_LOCK()` o si se termina el subproceso. Puede utilizar `IS_FREE_LOCK()` para comprobar si un bloqueo se ha liberado.

Se utiliza principalmente como mecanismo de bloqueo adicional en aplicaciones.

Por ejemplo:

```
mysql> SELECT GET-LOCK('one',1);
+-----+
| GET-LOCK('one',1) |
+-----+
|                   | 1 |
+-----+
```

IF

`IF(expresión1, expresión2, expresión3)`

Devuelve **expresión2** si **expresión1** es verdadera; en caso contrario, devuelve **expresión3**. Puede devolver un número o una cadena en función del contexto. **expresión1** se evalúa como entero, por lo que puede que las comparaciones reales no generen los resultados esperados.

Por ejemplo:

```
mysql> SELECT IF('a'='a',1,2);
```

```

+-----+
| IF('a'='a',1,2) |
+-----+
|                1 |
+-----+
mysql> SELECT IF(9<4,1,2);
+-----+
| IF(9<4,1,2) |
+-----+
|                2 |
+-----+
mysql> SELECT IF(NULL,'a','b');
+-----+
| IF(NULL,'a','b') |
+-----+
| b                |
+-----+
mysql> SELECT IF(16-6-10,'a',NULL);
+-----+
| IF(16-6-10,'a',NULL) |
+-----+
| NULL                |
+-----+

```

El siguiente ejemplo devuelve `false` porque el número real 0.49 se evalúa como el entero 0.

```

mysql> SELECT IF(0.49,'true','false');
+-----+
| IF(0.49,'true','false') |
+-----+
| false                    |
+-----+

```

IFNULL

`IFNULL(expresión1,expresión2)`

Devuelve `expresión1` si no es `nula`; en caso contrario, devuelve `expresión2`. El resultado puede ser un número o una cadena en función del contexto.

Por ejemplo:

```

mysql> SELECT IFNULL(1,2);
+-----+
| IFNULL(1,2) |
+-----+
|                1 |
+-----+
mysql> SELECT IFNULL(NULL,'nothing here');
+-----+
| IFNULL(NULL,'nothing here') |
+-----+

```

```

| nothing here |
+-----+
mysql> SELECT IFNULL(RELEASE-LOCK('nonexistent'),'The lock
never existed');
+-----+
| IFNULL(RELEASE-LOCK('nonexistent'),'The lock never existed')
|
+-----+
|
| The lock never existed
|
+-----+

```

INET_ATON

INET_ATON(cadena_dirección_cuatro_octetos)

Devuelve una direccion de red entera de 4 o 8 bits desde la cadena de direccion de cuatro octetos.

Por ejemplo:

```

mysql> SELECT INET_ATON('196.26.90.168');
+-----+
| INET_ATON('196.26.90.168') |
+-----+
| 3290061480 |
+-----+

```

INET_NTOA

INET_NTOA(dirección_de_red)

Devuelve una direccion de Internet de cuatro octetos desde una direccion de red de 4 o 8 bits y devuelve una cadena de direccion de cuatro octetos que representa la direccion de Internet de cuatro octetos.

Por ejemplo:

```

mysql> SELECT INET_NTOA(3290061480);
+-----+
| INET_NTOA(3290061480) |
+-----+
| 196.26.90.168 |
+-----+

```

IS_FREE-LOCK

IS-FREE-LOCK(cadena)

Se utiliza para comprobar si un bloqueo denominado cadena, creado con GET_LOCK(), esta libre o no. Devuelve 1 si el bloqueo esta libre, 0 si e bloqueo está activo o NULL si se producen otros errores.

Por ejemplo:

```
mysql> SELECT GET-LOCK('one',1);
+-----+
| GET-LOCK('one',1) |
+-----+
|                   1 |
+-----+
mysql> SELECT IS-FREE-LOCK('one');
+-----+
| IS-FREE-LOCK('one') |
+-----+
|                   0 |
+-----+
mysql> SELECT GET-LOCK('two',1);
+-----+
| GET-LOCK('two',1) |
+-----+
|                   1 |
+-----+
mysql> SELECT IS-FREE-LOCK('one');
+-----+
| IS-FREE-LOCK('one') |
+-----+
|                   1 |
+-----+
```

LAST_INSERT_ID

`LAST_INSERT_ID([expresión])`

Devuelve el último valor **añadido** a un campo **AUTO_INCREMENT** desde esta **conexión** o 0 si no hay ninguna. Por ejemplo:

```
mysql> SELECT LAST-INSERT-ID();
+-----+
| last-insert-id() |
+-----+
|                   0 |
+-----+
```

MASTER_POS_WAIT

`MASTER_POS_WAIT(nombre_registro, posición_registro)`

Se utiliza para sincronizar la **duplicación**. Si se ejecuta en el esclavo, espera hasta que este haya realizado todas las actualizaciones hasta la **posición especificada** en el registro principal antes de continuar. Por ejemplo:

```
mysql> SELECT MASTER_POS_WAIT('g-bin.001',273);
+-----+
```


Por ejemplo:

```
mysql> SELECT PASSWORD('a');
+-----+
| PASSWORD('a') |
+-----+
| 60671c896665c3fa |
+-----+
mysql> SELECT PASSWORD(PASSWORD('a'));
+-----+
| PASSWORD(PASSWORD('a')) |
+-----+
| 772a81723a030f10 |
+-----+
```

ENCRYPT() convierte una cadena a una contraseña según el método Unix.

RELEASE_LOCK

RELEASE_LOCK(cadena)

Libera la cadena de bloqueo anterior obtenida con GET_LOCK(). Devuelve 1 si el bloqueo se libera, 0 si no se puede liberar debido a que esta conexión no lo ha creado o NULL si el bloqueo no existe (nunca se ha creado o ya se ha liberado).

Por ejemplo:

```
mysql> SELECT GET_LOCK('one',1);
+-----+
| GET_LOCK('one',1) |
+-----+
| 1 |
+-----+
mysql> SELECT RELEASE_LOCK('one');
+-----+
| RELEASE_LOCK('one') |
+-----+
| 1 |
+-----+
mysql> SELECT RELEASE_LOCK('one');
+-----+
| RELEASE_LOCK('one') |
+-----+
| NULL |
+-----+
```

SESSION_USER

SESSION_USER()

Devuelve el usuario y el equipo MySQL conectados mediante el subproceso actual.

Por ejemplo:

```
mysql> SELECT SESSION-USER();
+-----+
| SESSION-USER() |
+-----+
| root@localhost |
+-----+
```

SYSTEM_USER() y USER() son sinonimos.

SHA

SHA(cadena)

Utiliza el algoritmo SHA (de hash seguro) para calcular una comprobacion de 160 bits a partir de la cadena y devuelve el numero hexadecimal de 40 digitos resultante. Es una codificacion mas segura que la que se obtiene con la funcion MD5(). Por ejemplo:

```
mysql> SELECT SHA('how many more');
+-----+
| SHA('how many more') |
+-----+
| 38ccbb8146b0673fa91abba3239829af6f3e5a6b |
+-----+
```

SHA1

SHA1(cadena)

Sinonimo de SHA().

SYSTEM-USER

SYSTEM-USER()

Sinónimo de SESSION_USER()

USER

USER()

Sinónimo de SESSION_USER()

VERSION

VERSION()

Devuelve la version del servidor MySQL en forma de cadena y adjunta -log si se ha activado el registro.

Por ejemplo:

```
mysql> SELECT VERSION();
+-----+
| VERSION()          |
+-----+
| 4.0.3-beta-log    |
+-----+
```



API PHP

PHP es uno de los lenguajes mas conocidos que se utilizan con MySQL, especialmente en entornos Web. En este apendice describiremos todas las funciones **PHP** que funcionan con MySQL, incluyendo algunas que todavia no se incluyen en la version comercial de **PHP**.

Opciones de configuracion PHP

El archivo de configuracion **PHP** se denominaphp.ini y cuenta con algunas opciones especificas para MySQL, como mostramos a **continuación**:

- **mysql.allow_persistent** boolean. Se define como On si se **permiten** conexiones permanentes a MySQL. El valor predeterminado es On. No es aconsejable desactivarlo.
- **mysql.max_persistent** integer. Numero maximo de conexiones **permanentes** de cada proceso. El valor predeterminado es -1 (sin limite).
- **mysql.max_links** integer. Numero maximo de conexiones MySQL de cualquier tipo por cada proceso. El valor predeterminado es -1 (sin limite).

- **mysql.default_port** string. Numero de puerto TCP predeterminado para conectarse a MySQL. PHP utiliza la variable de **entorno** `MYSQL_TCP_PORT` si no se configura una predeterminada. Unix **también** puede **utilizar**, en **orden**, la entrada `mysql-tcp` en `/etc/services` o la constante de tiempo de **compilación** `MYSQL_PORT`. El valor predeterminado es `NULL`.
- **mysql_default_socket** string. Nombre de socket Unix predeterminado utilizado para conectarse a MySQL. El valor predeterminado es `NULL`.
- **mysql.default_host** string. Nombre de anfitrión predeterminado utilizado para conectarse a MySQL. El **modo** seguro invalida esta opción. El valor predeterminado es `NULL`.
- **mysql.default_user** string. Nombre de usuario predeterminado que se utiliza para conectarse a MySQL. No se aplica en **modo** seguro, ya que invalidaría esta opción. El valor predeterminado es `NULL`.
- **mysql.default_password** string. Contraseña predeterminada utilizada para conectarse a MySQL. No se aplica en **modo** seguro, ya que invalidaría esta opción. El valor predeterminado es `NULL`. No es aconsejable que lo utilice para almacenar contraseñas.
- **mysql.connect_timeout** integer. Tiempo muerto de **conexión** expresado en segundos.

Funciones MySQL PHP

Las funciones PHP están íntimamente relacionadas con las funciones del API C. Las que enumeramos a **continuación** son las funciones **propias** de PHP. Al mismo tiempo, **existen** una serie de bibliotecas que proporcionan un cierto nivel de **abstracción** al utilizar la interfaz PHP en MySQL, entre las que destacamos ADODB, PEAR, **Metabase** y la antigua **PHPLib**.

mysql_affected_rows

```
int mysql_affected_rows([recurso conexión_mysql])
```

Devuelve el número de filas afectadas por la última **instrucción** que haya modificado los datos (`INSERT`, `UPDATE`, `DELETE`, `LOAD DATA`, `REPLACE`) o `-1` si la **consulta** ha **fallado**. Recuerde que `REPLACE INTO` afectará a dos **filas** por **cada** fila de la tabla original afectada (una `DELETE` y otra `INSERT`). Si la **conexión** a la base de datos no se especifica, se utiliza la última **conexión** que se haya abierto.

Si utiliza transacciones, invoque `mysql_affected_rows` antes de **invocar** `COMMIT`.

Para devolver el número de filas devuelto por una instrucción `SELECT`, utilice `mysql_num_rows()`.

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// actualice un número desconocido de campos en la tabla de la
//base de datos
mysql_query("UPDATE Stable SET field1=2 WHERE field1=3");

// almacene el número de filas actualizadas
$num_rows_updated = mysql_affected_rows();
```

mysql_change_user

```
boolean mysql_change_user(cadena nombre de usuario, cadena
contraseiia
[, cadena base de datos [, recurso conexion- mysql]])
```

Cambia el usuario MySQL actual (el que se haya conectado) por otro (es necesario especificar el nombre de usuario y la contraseña de este). También puede cambiar la base de datos al mismo tiempo o especificar una nueva conexión; en caso contrario, se utilizarán la conexión y la base de datos actuales. Devuelve `TRUE` si es satisfactoria y `FALSE` en caso contrario, manteniendo el usuario y los detalles existentes.

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

$change_succeeded = mysql_change_user($new_user, $new_password,
$database,
$connect);
```

mysql_client_encoding

```
int mysql_client_encoding ([recurso conexion-mysql])
```

Devuelve el conjunto de caracteres predeterminado (por ejemplo `latin1`) de la conexión especificada o la última conexión abierta que se haya abierto en caso de no especificar ninguna.

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

$charset = mysql_client_encoding($connect);
print "The current character set is $charset";
```


mysql_close

```
boolean mysql_close([recurso conexión_mysql])
```

Cierra la **conexión** especificada o la ultima **conexión** abierta que se haya **abierto**. No cierra conexiones permanentes. Por **ejemplo**:

```
// abra una conexión a la base de datos
$connect = mysql_connect($hostname, $username, $password);
// ... realiza algunos procesamientos
mysql_close($connect);
```

mysql_connect

```
mysql_connection mysql_connect([cadena nombre de anfitrión
[, cadena nombre de usuario
[, cadena contraseña [, nueva _ conexión booleana
[, indicadores-cliente int]]]])
```

Establece una **conexión** a un servidor MySQL (especificado, en caso de que sea necesario, por el nombre de servidor, nombre de usuario y contraseña) y devuelve un identificador de enlaces que utilizaran otras funciones. Si posteriormente se realiza una segunda llamada idéntica en el código, se obtiene el mismo identificador de enlaces, a **menos** que se configure el parámetro `nueva_ conexión` con el valor `true`.

El nombre de servidor también puede ser un puerto (que aparece, seguido por dos puntos, por detrás del nombre de servidor).

El parámetro final puede ser uno de los siguientes indicadores, que determinan elementos del comportamiento de MySQL cuando se conecta:

- **mysql_client_compress**. Utiliza un protocolo de impresión.
- **mysql_client_ignore_space**. Permite un espacio adicional por detrás de los nombres de funciones.
- **mysql_client_interactive**. Espera el valor de la variable `interactive_timeout` en lugar del de la variable `mysql_wait_timeout` antes de cerrar una **conexión** inactiva.

mysql_client_ssl. Utilice el protocolo SSL.

Por ejemplo:

```
// defina los parámetros de conexión (normalmente fuera de la
//secuencia de comandos)
$hostname = "localhost:3306";
$username = "guru2b";
$password = "g00r002b";

// abra una conexión a la base de datos
```

```
$connect = mysql_connect(Shostname, $username, Spassword,
MYSQL_CLIENT_COMPRESS);
```

mysql_create_db

```
boolean mysql_create_db (cadena base de datos [, recurso
conexión_mysql])
```

Crea una nueva base de datos en el servidor por medio de la **conexión** especificada o de la ultima **conexión abierta** si no es especifica ninguna. Devuelve **true** si es satisfactoria y **false** en **caso** contrario.

La nueva base de datos no se convierte en la base de datos activa. Tendra que **utilizar la función** `mysql_select_db()` para activarla.

Esta **función** reemplaza a la **obsoleta función** `mysql_createdb()` que todavia funciona.

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, Susername, $password);

if (mysql_create_db("new_db", $connect)) {
    print "Database new-db successfully created";
}
else {
    print "Database new-db was not created";
}
```

mysql_data_seek

```
boolean mysql_data_seek (recursos resultado_consulta, fila int)
```

Desplaza el **puntero de fila interno** (0 es la **primera fila**) asociado al resultado de la consulta a una nueva posicion. La siguiente fila que se **recupera** (por ejemplo desde) sera la fila especificada.

Devuelve **true** si el desplazamiento es satisfactorio y **false** si no lo es (normalmente porque el resultado de la consulta no tiene filas asociadas).

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, Susername, $password);

// seleccione la base de datos
mysql_select_db("database1", $connect);

// defina y ejecute la consulta
$sql = "SELECT field1,field2 FROM table1";
$result = mysql_query($sql, $connect);

// numero de filas devueltas
```

```

$x = mysql_num_rows($result);
// si existe la decima fila, desplacese hasta la misma
if ($x >= 10) {
    mysql_data_seek($result, 9);
}

// devuelva los datos de la decima fila
$row = mysql_fetch_array($result);
print "Field1: " . $row["field1"] . "<br>\n";
print "Field2: " . $row["field2"];

```

mysql_db_name

```

string mysql_db_name (recurso resultado_consulta, fila int
[, mixed unused])

```

Devuelve el nombre de una base de datos. El resultado de la consulta se devuelve de una invocación anterior a la función `mysql_list_dbs()`. La fila especifica qué elemento del conjunto de resultados de la consulta (que empiezan en 0) se devuelve.

La función `mysql_num_rows()` devuelve el numero de base de datos devuelto desde `mysql_list_dbs()`.

Por ejemplo:

```

// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// devuelva la lista de bases de datos de la conexión
$result = mysql_list_dbs($connect);

// procese una iteración por los resultados y devuelva los
// nombres de las bases de datos uno a uno
for ($i=0; $i < mysql_num_rows($result); $i++) {
    print mysql_db_name($result, $i) . "<br>\n";
}

```

mysql_db_query

```

query-result mysql_db_query ( cadena base de datos, cadena
consulta
[, recurso conexión_mysql])

```

Devuelve un recurso de resultado de la consulta si esta se procesa satisfactoriamente y `false` si la consulta falla.

La consulta se envía a la base de datos especificada por medio de la conexión especificada (o la ultima que se haya abierto, en caso de no especificar ninguna en concreto).

Esta función se ha quedado obsoleta, por lo que en su lugar debe utilizar `mysql_select_db()` y `mysql_query()`.

mysql_drop_db

```
boolean mysql_drop_db(string database [, recurso
conexion=mysql])
```

Elimina la base de datos especificada de la **conexión** especificada o la ultima que se haya abierto si no se especifica ninguna en concreto. Devuelve `true` si es satisfactoria y `false` si la base de datos no se puede eliminar.

Esta funcion, e **incluso** la funcion `mysql_dropdb()` que es mas antigua, se han quedado obsoletas. En su lugar debe **utilizar** la **función** `mysql_query()` para eliminar la base de datos. Por ejemplo:

```
// abra una conexión a la base de datos
Sconnect = mysql_pconnect($hostname, Susername, Spassword);

// elimine la base de datos old-db
if (mysql_drop_db("old_db", $connect)) {
    print "Database old-db is gone";
} else {
    print "Database old-db could not be dropped";
}
```

mysql_errno

```
int mysql_errno([recurso conexion])
```

Devuelve el numero de error de la ultima funcion MySQL que se haya ejecutado o cero si no se produjo ningun error. **Utiliza** la **conexión** especificada (o la ultima que se haya abierto en **caso** de no especificar ninguna **conexión** en concreto).

Esta funcion devolvera cero despues de ejecutar satisfactoriamente cualquier funcion relacionada con **MySQL**, a **excepción** de `mysql_error()` y `mysql_errno()`, que no cambian el valor.

Por ejemplo:

```
// abra una conexión permanente a la base de datos
Sconnect = mysql_pconnect($hostname, Susername, Spassword);

// intente utilizar una base de datos que acabe de eliminar
mysql_select_db("old-db", Sconnect);

// Muestra el codigo de error - 1049
if (mysql_errno()) {
    print "MySQL has thrown the following error: ".mysql_errno();
}
```

mysql_error

```
string mysql_error([recurso conexión_mysql])
```

Devuelve el texto del mensaje de error de la última función MySQL que se haya ejecutado o una cadena vacía (") si no se produjo ningún error. Utiliza la conexión especificada (o la última que se haya abierto en caso de no especificar ninguna conexión en concreto).

Esta función devuelve una cadena vacía después de ejecutar satisfactoriamente cualquier función relacionada con MySQL, a excepción de `mysql_error()` y `mysql_errno()`, que no cambian el valor.

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// intente utilizar una base de datos que acabe de eliminar
mysql_select_db("old-db", $connect);

// Muestra el texto de error - Base de datos 'old-db'
// desconocida
if (mysql_errno()) {
    print "MySQL has thrown the following error: ".mysql_error();
}
```

mysql_escape_string

```
string mysql_escape_string (nombre de la cadena)
```

Devuelve una cadena con todos los caracteres de conversión de escape que pueden dividir la consulta (con una barra invertida por delante de los mismos). Entre estos caracteres se incluyen los nulos (`\x00`), nueva línea (`\n`), retorno del carro (`\r`), barra invertida (`\`), corchillos simple (`'`), corchillos dobles (`"`) y Control-Z (`\x1A`).

No se aplica conversión de escape a los signos de porcentaje (`%`) y guion bajo (`_`).

De esta forma, la consulta resulta segura de utilizar. Siempre que se utilicen entradas del usuario en una consulta, es aconsejable utilizar esta función para garantizar la seguridad de la consulta.

También puede utilizar la función `addslashes()`, ligeramente **menos completa**.

Por ejemplo:

```
// cadena original, no segura
$field_value = "Isn't it true that the case may be";

// aplica conversión de escape a los caracteres especiales
$field_value = mysql_escape_string($field_value);

// ahora es segura y muestra: Isn\'t it true that the case may be
print "$field-value";
```

mysql-fetch-array

```
array mysql-fetch-array (recurso resultado_consulta  
[, tipo_matriz int])
```

Devuelve una matriz de **cadena**s basada en una fila de los resultados de la consulta devueltos desde una función como `mysql_query()`, y devuelve `false` si falla o no encuentra filas disponibles. La fila devuelta se basa en la posición del puntero de fila interno, que se **incrementa** en una unidad (el puntero de fila comienza en 0 justo después de ejecutar una consulta).

El segundo parámetro especifica cómo se devuelven los datos. Si el **tipo** de matriz se define como `MYSQL_ASSOC`, los datos se devuelven en forma de matriz asociativa (la misma que si se utiliza la función `mysql_fetch_assoc()`). Si el tipo de matriz se define como `MYSQL_NUM()`, los datos se devuelven como matriz **numérica** (la misma que si utiliza la función `mysql_fetch_row()`). La tercera opción, `MYSQL_BOTH`, es la predeterminada si no se especifica ninguna otra y le permite acceder a los datos como matriz asociativa o **numérica**.

La matriz asociativa solamente adopta como clave los nombres de los campos (y elimina todos los prefijos de tabla). Si hay nombres de campos duplicados, será necesario utilizar un alias; en caso contrario, el último valor mencionado reemplazará al anterior.

Por ejemplo:

```
// abra una conexión permanente a la base de datos  
$connect = mysql_pconnect($hostname, $username, $password);  
  
// seleccione la base de datos  
mysql_select_db("database1", $connect);  
  
// defina y ejecute la consulta  
$sql = "SELECT field1,field2 FROM table1";  
$result = mysql_query($sql, $connect);  
  
// devuelva los datos en matrices asociativas y numericas  
// (predeterminada)  
// procese una iteración por las filas para imprimir los datos  
while ($row = mysql_fetch_array($result)) {  
    print "Field1: ".$row["field1"]."<br>\n";  
    print "Field2: ".$row["field2"]."<br>\n";  
}
```

mysql_fetch_assoc

```
array mysql_fetch_assoc (recurso resultado_consulta)
```

Devuelve una matriz de **cadena**s basada en una fila de los resultados de la consulta devueltos por una función como `mysql_query()` y devuelve `false` si falla o no hay más filas disponibles. La fila devuelta se basa en la posición del

puntero de fila interno, que se incrementa en una unidad (el puntero de fila comienza en 0 justo después de ejecutar una consulta).

Los datos se devuelven en **forma** de matriz asociativa que solamente adopta como clave los nombres de los campos (y elimina todos los prefijos de tabla). Si hay nombres de campos duplicados, será necesario utilizar un alias; en caso contrario, el último valor mencionado reemplazará al anterior. Es lo mismo que utilizar `mysql_fetch_array()` con el parámetro `MYSQL_ASSOC`:

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// seleccione la base de datos
mysql_select_db("database1", $connect);

// defina y ejecute la consulta
$sql = "SELECT field1,field2 FROM table1";
$result = mysql_query($sql, $connect);

// devuelva los datos en formato de matriz asociativa y procese
// una iteración
// por el resultado para imprimir los valores de las filas
while ($row = mysql_fetch_assoc($result)) {
    print "Field1: ".$row["field1"]."<br>\n";
    print "Field2: ".$row["field2"]."<br>\n";
}
```

mysql_fetch_field

```
object mysql_fetch_field(recurso resultado_consulta
[, desplazamiento int ])
```

Devuelve un objeto que contiene información sobre un campo, basada en una fila de un resultado de consulta devuelto por una función como `mysql_query()`. Si el desplazamiento no se especifica, se devuelve el siguiente campo no recuperado (por lo que puede invocar varias veces esta función para obtener información sobre todos los campos); en caso contrario, será el determinado por el desplazamiento (0 para el primer campo).

Las propiedades del objeto son las siguientes:

- **name**. Nombre del campo.
- **table**. Nombre de la tabla a la que pertenece el campo.
- **max_length**. Longitud máxima del campo.
- **not-null**. 1 si el campo no puede contener nulos.
- **primary-key**. 1 si el campo es una clave principal.
- **unique-key**. 1 si el campo es una clave exclusiva.

- **multiple-key**. 1 si el campo es una clave no exclusiva.
- **numeric**. 1 si el campo es numerico.
- **blob**. 1 si el campo es un BLOB.
- **type**. El tipo del campo
- **unsigned**. 1 si el campo no tiene **firma**.
- **zerofill**. 1 si el campo se ha completado con ceros.

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// devuelve una lista de todos los campos de la
// basededatos1.table1
$result = mysql_list_fields("database1", "table1");

// procese una iteración por los campos y muestre el nombre, el
// tipo y
// la longitud maxima del campo
while ($row = mysql_fetch_field($result)) {
    $max_length = $row->max_length;
    $name = $row->name;
    $type = $row->type;
    print "Name:$name <br>\n";
    print "Type:$type <br>\n";
    print "Maximum Length:$max_length <br><br>\n\n";
}

```

mysql_fetch_lengths

```
array mysql_fetch_lengths(recurso resultado_consulta)
```

Devuelve una matriz de las longitudes de **cada** campo en la **última** fila obtenida de un resultado de una consulta (la longitud de dicho resultado, no la longitud maxima) y devuelve **false** si no ha sido satisfactorio.

Puede **utilizar** la funcion `mysql_fetch_lengths()` para devolver la maxima longitud de un campo.

Por ejemplo:

```
// abra una conexión a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// seleccione la base de datos
mysql_select_db("database1", $connect);

// defina y ejecute la consulta
$sql = "SELECT field1,field2 FROM table1";
$result = mysql_query($sql, $connect);

```



```

// devuelva los datos en formato de matriz asociativa y procese
// una iteracion por
// el resultado, para recuperar la longitud de los campos e
// imprimir
// los valores y longitudes de las filas
while ($row = mysql_fetch_assoc($result)) {
    $lengths = mysql_fetch_lengths($result);
    print "Field1: ".$row["field1"]."Length:
".$lengths[0]."<br>\n";
    print "Field2: ".$row["field2"]."Length:
".$lengths[1]."<br>\n";
}

```

mysql_fetch_object

```
object mysql_fetch_object(recurso resultado-consulta)
```

Devuelve un objeto con propiedades **basadas** en una fila de un resultado de una consulta devuelta por una funcion como `mysql_query()`. La fila devuelta se basa en la **posición del puntero de fila interno, que se incrementa** en una unidad (el puntero de fila empieza en 0 justo despues de ejecutar una consulta).

Cada una de las propiedades del objeto se basa en un nombre (un alias) del campo de la consulta.

Por ejemplo:

```

// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// seleccione la base de datos
mysql_select_db("database1", $connect);

// defina y ejecute la consulta
$sql = "SELECT field1,field2 FROM table1";
$result = mysql_query($sql, $connect);

// procese una iteracion por las filas para devolver cada una
// de ellas como un objeto
// y mostrar los campos
while ($row = mysql_fetch_object($result)) {
    print "Field1: ".$row->field1."<br>\n";
    print "Field2: ".$row->field2."<br>\n";
}

```

mysql_fetch_row

```
array mysql_fetch_row(recurso resultado-consulta)
```

Devuelve una matriz de **cadenas** basada en una fila de un resultado de una consulta devuelta por una funcion como `mysql_query()` o devuelve `false` si falla o no hay mas filas disponibles. La fila **devuelta** se basa en la **posición del**

puntero de fila interno, que se **incrementa** en una unidad (el puntero de fila comienza en 0 inmediatamente despues de ejecutar una consulta).

Los datos se devuelven en **forma** de matriz nurnerica (la misma que si se hubiera utilizado la funcion `mysql_fetch_array()` con el parametro `MYSQL_NUM`).

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// seleccione la base de datos
mysql_select_db("database1", $connect);

// defina y ejecute la consulta
$sql = "SELECT field1,field2 FROM table1";
$result = mysql_query($sql, $connect);

// procese una iteración por las filas para devolver cada una
// de ellas como matriz nurnerica
// y mostrar los campos
while ($row = mysql_fetch_row($result)) {
    print "Field1: ".$row[0]."<br>\n";
    print "Field2: ".$row[1]."<br>\n";
}
```

mysql_field_flags

```
string mysql_field_flags(recurso resultado-cadena,
desplazamiento int)
```

Devuelve una cadena que contiene indicadores del campo especificado **basado** en un resultado de una consulta **devuelto** por una funcion como `mysql_query()`. El desplazamiento determina **qué** campo se **examina** (0 para el **primer** campo).

Entre los indicadores se incluyen

La antigua funcion `mysql_field_flags()` **hace** lo mismo, **pero** se ha **quedado** obsoleta.

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// seleccione la base de datos
mysql_select_db("database1", $connect);

// defina y ejecute la consulta
$sql = "SELECT field1,field2 FROM table1";
$result = mysql_query($sql, $connect);

// Muestre las propiedades de los campos 1 y 2
print "Field1 flags: ".mysql_field_flags($result, 0)."<br>\n";
```

```
print "Field1 flags: ".mysql_field_flags($result, 1)."<br>\n";
```

mysql_field_len

```
int mysql_field_len (recurso resultado_cadena, desplazamiento  
int)
```

Devuelve la máxima longitud (determinada por la estructura de la base de datos) del campo especificado **basado** en una fila de un resultado de una consulta devuelto por una función como `mysql_query()`. El desplazamiento (que empieza en 0) determina el campo.

La antigua función `mysql_fieldlen()` hace lo mismo, pero se ha quedado obsoleta.

Puede utilizar la función `mysql_fetch_lengths()` para determinar la longitud concreta de un campo devuelto.

Por ejemplo:

```
// abra una conexión permanente a la base de datos  
$connect = mysql_pconnect($hostname, $username, $password);  
  
// seleccione la base de datos  
mysql_select_db("database1", $connect);  
  
// defina y ejecute la consulta  
$sql = "SELECT field1,field2 FROM table1";  
$result = mysql_query($sql, $connect);  
  
// Muestre las propiedades de los campos 1 y 2  
print "Field1 maximum length: ". mysql_field_len($result, 0).  
"<br>\n";  
print "Field1 maximum length: ". mysql_field_len($result, 1).  
"<br>\n";
```

mysql_field_name

```
string mysql_field_name(recurso resultado_consulta,  
desplazamiento int)
```

Devuelve el nombre del campo especificado **basado** en una fila de un resultado de una consulta devuelto por una función como `mysql_query()`. El desplazamiento (que empieza en 0) determina el campo. Por ejemplo:

```
// abra una conexión a la base de datos  
$connect = mysql_pconnect($hostname, $username, $password);  
  
// seleccione la base de datos  
mysql_select_db("database1", $connect);  
  
// defina y ejecute la consulta  
$sql = "SELECT * FROM table1";
```

```

$result = mysql_query($sql, $connect);

// procese una iteración por los campos y muestre el nombre
for($i=0; $i < mysql_num_fields($result); $i++) {
    print "Field name: ".mysql_field_name($result, $i). "<br>\n";
}

```

mysql_field_seek

```

boolean mysql_field_seek(recurso resultado-consulta,
desplazamiento int)

```

Desplaza el **puntero interno** hasta un nuevo campo del resultado de la consulta, en función del desplazamiento (que empieza en 0 con el primer campo). La siguiente **invocación** a la función `mysql_fetch_field()` comenzará con este desplazamiento. No resulta de gran **utilidad** ya **que** se puede desplazar **directamente** el **puntero** por medio de la función `mysql_fetch_field()`.

Por ejemplo:

```

// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// seleccione la base de datos
mysql_select_db("database1", $connect);

// defina y ejecute la consulta
$sql = "SELECT * FROM table1";
$result = mysql_query($sql, $connect);

// vaya hasta el segundo campo
mysql_field_seek($result, 1);

$field = mysql_fetch_field($result);
print "The name of the 2nd field is: " . $field->name;

```

mysql_field_table

```

string mysql_field_table(recurso resultado-consulta,
desplazamiento int)

```

Devuelve el nombre de la tabla a la que **hace** referencia el campo en un **resultado** de consulta determinado por el desplazamiento (que comienza en 0). **Devuelve false** si se produce un error. La obsoleta función `mysql_fieldtable()` es idéntica. Por ejemplo:

```

// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// seleccione la base de datos
mysql_select_db("database1", $connect);

```

```
// defina y ejecute la consulta
$sql = "SELECT field1,field2 FROM table1,table2 WHERE
field1=field2";
$result = mysql_query($sql, $connect);

// Obtenga el nombre de la tabla del campo1 (desplazamiento 0)
echo "field 1 belongs to the table:
".mysql_field_table($result, 0);
```

mysql_field_type

```
string mysql_field_type(recurso resultado-consultas,
desplazamiento int)
```

Devuelve el **tipo** de un campo de un resultado de una consulta determinado por el desplazamiento (que empieza en 0) o devuelve **false** si se produce un error. Entre los ejemplos de tipo de campo se incluyen `int`, `real`, `string` y `blob`.

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect(Shostname, $username, $password);

// seleccione la base de datos
mysql_select_db("databasel", $connect);

// defina y ejecute la consulta
$sql = "SELECT field1,field2 FROM table1,table2 WHERE
field1=field2";
$result = mysql_query($sql, $connect);

for($i=0;$i<mysql_num_fields($result);$i++) {
    echo "Field $i is of type: ".mysql_field_type($result, $i) .
"<br>\n";
}
```

mysql_free_result

```
boolean mysql_free_result(recurso resultado-consultas)
```

Libera toda la **memoria** utilizada por el **recurso** `query_result`, lo que permite volverlo a utilizar. Devuelve **true** si es **satisfactorio** y **false** si no lo es. La **memoria** se libera automáticamente al final de la secuencia de comandos incluso sin invocar esta función. La función `mysql_freeresult()`, ya obsoleta, es idéntica.

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect(Shostname, Susername, $password);

// seleccione la base de datos
```

```
mysql_select_db("database1", $connect);

// defina y ejecute la consulta
$sql = "SELECT field1,field2 FROM table1";
$result = mysql_query($sql, $connect);

// procese una iteración por las filas para devolver cada una
// de ellas como matriz numérica
// y muestre los campos
while ($row = mysql_fetch_row($result)) {
    print "Field1: ".$row[0]."<br>\n";
    print "Field2: ".$row[1]."<br>\n";
}

// libere los recursos asociados a la consulta
mysql_free_result($result);
```

mysql_get_client_info

```
string mysql_get_client_info()
```

Devuelve una cadena que contiene la version de la biblioteca cliente de MySQL (por ejemplo, 4.0.2).

Por ejemplo:

```
// muestra - La version de la biblioteca cliente es: 4.0.2 (por
// ejemplo)
print "La version de la biblioteca cliente es:
.mysql_get_client_info();
```

mysql_get_host_info

```
string mysql_get_host_info([recurso conexion-mysql])
```

Devuelve una cadena que contiene informacion sobre la **conexión** (por ejemplo "Servidor local a traves de un socket Unix). La informacion es de la **conexión** especificada (o de la ultima **conexión abierta** en caso de no especificar ninguna en concreto). Por ejemplo:

```
// muestra - Tipo de conexion: Servidor local a traves de un
// socket UNIX
// (por ejemplo)
print "Tipo de conexion: ".mysql_get_host_info();
```

mysql_get_proto_info

```
int mysql_get_proto_info([recurso conexion-mysql])
```

Devuelve un entero que contiene el protocolo de version (por ejemplo, 10) utilizado por la conexion. La informacion proviene de la **conexión** especificada

(o de la última que se haya abierto, en caso de no especificar ninguna en concreto).

Por ejemplo:

```
// muestra - Version de Protocolo: 10 (por ejemplo)
print "Version de Protocolo: ".mysql_get_proto_info();
```

mysql_get_server_info

```
string mysql_get_server_info([recurso conexión_mysql])
```

Devuelve una cadena que contiene la versión del servidor **MySQL** (por ejemplo, 4.0.3). La información se obtiene de la **conexión** especificada (o de la última **conexión** abierta, en caso de no especificar ninguna en concreto).

Por ejemplo:

```
// muestra - Version del servidor: 4.0.3-beta-log (por ejemplo)
print "Version del servidor: ".mysql_get_server_info();
```

mysql_info

```
string mysql_info ( [recurso conexión_mysql])
```

Devuelve una cadena que contiene información detallada **sobre** la consulta más reciente. Esta información **incluye** registros, las filas que coinciden, cambios y advertencias.

La información se obtiene de la **conexión** especificada (o de la última **conexión** abierta, en caso de no especificar una en concreto).

Por ejemplo:

```
// abra una conexión a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// seleccione la base de datos
mysql_select_db("database1", $connect);

// defina y ejecute la consulta
$sql = "UPDATE table1 set field1 = 2 WHERE field2=3";
$result = mysql_query($sql, $connect);

// muestra:
// Información sobre la consulta: Formato de cadena: Filas que
// coinciden: 19 Modificada: 19 Advertencias: 0
//(por ejemplo)
print "Query info: ".mysql_info();
```

mysql_insert_id

```
int mysql_insert_id([recurso conexión_mysql])
```

Devuelve un entero que contiene el valor `AUTO_INCREMENT` mas reciente de dicha conexion, o devuelve `false` si falla (no se han definido valores `AUTO_INCREMENT` para esa conexion). La **información** se obtiene de la **conexion especificada** (o de la **última conexión** abierta, en caso de no especificar ninguna en concreto).

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// seleccione la base de datos
mysql_select_db("database1", $connect);

// defina y ejecute la consulta
$sql = "INSERT INTO table1(field1, field2) VALUES(3,4)";
$result = mysql_query($sql, $connect);

// Muestra: valor AUTO-INCREMENT: 10 (por ejemplo)
print "valor AUTO-INCREMENT: ".mysql_insert_id();
```

mysql_list_dbs

```
query-result mysql_list_dbs([recurso conexion-mysql])
```

Devuelve un **recurso** que apunta a una lista de bases de datos disponibles en la conexion, o devuelve `false` en caso de fallo. La **información** se obtiene de la **conexion especificada** (o de la **ultima** que se haya abierto en caso de no **especificar** ninguna en concreto). El resultado se puede procesar con una funcion como `mysql_db_name()` o `mysql_result()`.

La **función** `mysql_list_dbs()`, ya obsoleta, es **idéntica**.

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// devuelva la lista de bases de datos de la conexión
$result = mysql_list_dbs($connect);

// procese una iteración por los resultados para devolver los
// nombres de bases de datos uno a uno
for ($i=0; $i < mysql_num_rows($result); $i++) (
    print mysql_db_name($result, $i) . "<br>\n";
)
```

mysql_list_fields

```
query-result mysql_list_fields(cadena base de datos, cadena
tabla
[, recurso conexión_mysql])
```


Devuelve un **recurso** que apunta a una lista de campos de una determinada base de datos y tabla o *false* en caso de fallo. La información proviene de la **conexión** especificada (o de la última que se haya abierto en caso de no especificar ninguna en concreto).

La función `mysql_list_fields()`, ya obsoleta, es idéntica.

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// devuelva una lista de todos los campos de la base de datos
// 1.tabla1
$result = mysql_list_fields("database1", "table1");

// procese una iteración por los campos y muestre el nombre,
// tipo y
// longitud máxima de los mismos
while ($row = mysql_fetch_field($result)) {
    $max_length = $row->max_length;
    $name = $row->name;
    $type = $row->type;
    print "Name:$name <br>\n";
    print "Type:$type <br>\n";
    print "Maximum Length:$max_length <br><br>\n\n";
}
}
```

mysql_list_processes

```
query_result mysql_list_processes ([recurso conexión-mysql])
```

Devuelve un **recurso** que contiene una lista de los procesos **MySQL** actuales o *false* en caso de fallo. Tras ello, puede utilizar una función como `mysql_fetch_assoc()` para devolver una matriz que contenga los elementos `Id`, `Host`, `db`, `Command` y `Time`. La **información** proviene de la **conexión** especificada (o de la última que se haya abierto, si no se especifica ninguna en concreto).

Por ejemplo:

```
// abra una conexión a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// devuelva todos los procesos
$result = mysql_list_processes($connect);

// procese una iteración por las filas para mostrar los
// elementos de los distintos procesos
while ($row = mysql_fetch_assoc($result)){
    print $row["Id"];
    print $row["Host"];
    print $row["db"];
}
```

```

    print $row["Command"];
    print $row["Time"];
    print "<br>\n"
}

```

mysql_list_tables

```

query-result mysql_list_tables(cadena base de datos [, recurso
conexion-mysql])

```

Devuelve un recurso que apunta a una lista de tablas de una determinada base de datos o false en caso de fallo. La información proviene de la conexión especificada (o de la última que se haya abierto, si no se especifica ninguna en concreto).

Por ejemplo:

```

// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// devuelve la lista de tablas
$result = mysql_list_tables("database1");

// procese una iteración por las filas de tablas y muestra los
// nombres
for($i=0; $i < mysql_num_rows($result); $i++) {
    print "Table name: ".mysql_tablename($result, $i)."<br>\n";
}

```

mysql_num_fields

```

int mysql_num_fields(recurso resultado-consultas)

```

Devuelve un entero que contiene un número de campos de un resultado de consulta o NULL si se produce un error. La función `mysql_numfields()`, ya obsoleta, es idéntica. Por ejemplo:

```

// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// devuelva una lista de todos los campos de
// basededatos1.tabla1
$result = mysql_list_fields("database1", "table1");

// Muestra: Campos numericos de basededatos1: 6 (por ejemplo)
print "Campos numericos de basededatos1:
".mysql_num_fields($result);

```

mysql_num_rows

```

int mysql_num_rows(recurso resultado-consultas)

```

Devuelve un entero que contiene el número de filas de un resultado de una consulta o NULL en **caso** de que se produzca un error. No funciona si el resultado de la consulta se ha obtenido con la función `mysql_unbuffered_query()`.

Debería utilizar `mysql_affected_rows()` **para** devolver **el número** de filas de datos modificados **por una consulta** (por ejemplo, **después** de INSERT o UPDATE).

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// devuelva la lista de bases de datos de la conexión
$result = mysql_list_dbs($connect);

// procese una iteración por los resultados para devolver, uno
// a uno, los nombres de las bases de datos
for ($i=0; $i < mysql_num_rows($result); $i++) {
    print mysql_db_name($result, $i) . "<br>\n";
}
```

mysql_pconnect

```
mysql_connection mysql_pconnect([cadena nombre_anfitrión
[, cadena nombre_usuario [, cadena contraseña
[, indicadores_cliente int]])])
```

Establece una **conexión** permanente (una que se puede reutilizar) con un **servidor** MySQL (especificado por el nombre de servidor, el nombre de usuario y la contraseña) y devuelve un identificador de enlaces que pueden utilizar otras **funciones**. Si ya existe uno, se reutilizará. El parámetro final puede ser uno o varios de los siguientes indicadores, que determinan elementos del comportamiento de MySQL cuando se conecta:

- **mysql_client_compress**. Utiliza un protocolo de compresión.
- **mysql_client_ignore_space**. Permite que haya espacio **detrás** de los nombres de funciones.
- **mysql_client_interactive**. Espera el valor de la variable `interactive_timeout` en lugar del de la variable `mysql_wait_timeout` antes de cerrar una **conexión** inactiva.
- **mysql_client_ssl**. Utiliza el protocolo SSL.

MySQL cierra conexiones permanentes después de los segundos especificados en `wait_timeout` (una variable `mysql`) o después de que se cierre el proceso que ha **iniciado** la **conexión**. Por ejemplo, su proceso de servidor Web puede utilizar la misma **conexión** para varias secuencias de comandos, **pero** la **conexión** se cerrará una vez que termine dicho proceso.

Por ejemplo:

```
// defina los parametros de conexión (normalmente se hace fuera
// de la secuencia de comandos)
Shostname = "localhost:3306";
Susername = "guru2b";
Spassword = "g00r002b";

// abra una conexión permanente a la base de datos
Sconnect = mysql_pconnect(Shostname, Susername, Spassword);
```

mysql_ping

```
boolean mysql_ping ([recurso conexion-mysql])
```

Devuelve `true` si el servidor MySQL esta en ejecucion y `false` si no lo esta. El ping se intenta a traves de la **conexión** especificada (o de la ultima que se haya abierto, si no se especifica ninguna en concreto). Si falla, la secuencia de comandos intentara volverse a conectar con los mismos parametros.

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect(Shostname, Susername, $password);

// el tiempo pasa...

if (mysql_ping()) {
    print "Still connected";
}
else {
    print "Connection lost";
}
```

mysql_query

```
query-result mysql_query(cadena consulta [, recurso
conexion-mysql
[, modo_resultado int]])
```

Devuelve un resultado de consulta (si la consulta es la que produce el resultado como SELECT o DESCRIBE), devuelve `true` si la consulta no genero un resultado **pero fue** satisfactoria (como DELETE o UPDATE) y `false` si la consulta falla. La consulta se **envía** a una base de datos especificada por medio de la **conexión** especificada (o de la ultima que se haya abierto, si no se especifica ninguna en concreto).

El parametro **opcional** `modo_resultado` puede ser `MYSQL_USE_RESULT`, que hace que el resultado **no** se almacene en el **búfer**, al igual que con `mysql_unbuffered_query()`, o `MYSQL_STORE_RESULT` (el predeterminado).

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// seleccione la base de datos
mysql_select_db("databasel", $connect);

// defina y ejecute la consulta
$sql = "SELECT field1,field2 FROM tablel";
$result = mysql_query($sql, $connect);

// devuelva los datos en formato de matriz asociativa y procese
// una iteración por
// el resultado, para imprimir los valores de las filas
while ($row = mysql_fetch_assoc($result)) {
    print "Field1: ".$row["field1"]."<br>\n";
    print "Field2: ".$row["field2"]."<br>\n";
}
```

mysql_real_escape_string

```
string mysql_real_escape_string (cadena nombre de cadena
[, recurso conexión_mysql])
```

Devuelve una cadena con todos los caracteres a los que se ha aplicado conversión de escape que pueden dividir la consulta (una barra invertida por delante de los mismos).

Entre estos se incluyen los nulos (\x00), nueva línea (\n), retorno del carro (\r), barra invertida (\), comillas simples (') , comillas dobles (") y Control-Z (\x1A). No se aplica conversión de escape a los signos de porcentaje (%) y guion bajo (_).

De esta forma, la consulta resulta segura de utilizar. Difiere de `mysql_escape_string()` en que tiene en cuenta el conjunto de caracteres actual.

Por ejemplo:

```
// cadena original no segura
$field-value = "Isn't it true that the case may be";

// se aplica conversión de escape a los caracteres especiales
$field-value = mysql_real_escape_string($field_value);

// ahora es segura y muestra: Isn\'t it true that the case may
// be
print "$field-value";
```

mysql_result

```
mixed mysql_result(recurso resultado_consulta, fila int
[, especificador_campo mixed])
```

Devuelve los contenidos de un solo campo de un resultado de una consulta. `especificador_campo` puede ser un desplazamiento (empezando desde 0) o el nombre del **campo**, con o sin el especificador de tabla (es decir, `nombredelatabla.nombredelcampo` o **simplemente** `nombredelcampo`) si se proporciona con la consulta. Si el especificador de campo no se proporciona, se devolverá el primer campo.

Esta función es considerablemente más lenta que las funciones que devuelven toda la fila, como por ejemplo `mysql_fetch_row()` y `mysql_fetch_array()`, por lo que es aconsejable que utilice una de estas. Por otra parte, no mezcle esta función con funciones que devuelvan toda la fila.

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// seleccione la base de datos
mysql_select_db("database1", $connect);

// devuelva la media de campo1
$sql = "SELECT AVG(field2) FROM table1";
$result = mysql_query($sql, $connect);

// muestre el valor medio de este campo
print "Field2 average: ".mysql_result($result, 0);
```

mysql_select_db

```
boolean mysql_select_db(cadena base de datos [, recurso
conexión_mysql])
```

Cambia la base de datos actual por la base de datos especificada. Utiliza la **conexión** especificada (o la última que se haya abierto, en caso de no especificar una en concreto).

Si no hay conexiones abiertas, intentará invocar `mysql_connect()` sin parámetros de conexión. Devuelve **true** si es **satisfactoria** y **false** en caso contrario.

La función `mysql_selectdb()`, ya obsoleta, es idéntica.

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// seleccione la base de datos
mysql_select_db("database1", $connect);
```

mysql_stat

```
string mysql_stat ([recurso conexión_mysql])
```

Devuelve una cadena que contiene el estado del servidor. Contiene el tiempo en ejecución, subprocessos, preguntas, consultas **lentas**, consultas abiertas, tablas vacías, tablas abiertas y consultas medias por segundo. Utiliza la **conexión especificada** (o la última que se haya abierto en **caso** de no especificar ninguna en concreto).

Por ejemplo:

```
// muestra (por ejemplo):
// Tiempo en ejecución: 109 Subprocesos: 2 Preguntas: 199
// Consultas lentas: 1 Abiertas: 4
// Tablas vacías: 1 Tablas abiertas: 2 Consultas medias por
// segundo: 1.826
print "Estado del servidor: ".mysql_stat();
```

mysql_tablename

```
string mysql_tablename(recurso resultado_consulta, fila int)
```

Devuelve el nombre de tabla de un resultado de una consulta devuelta por la función `mysql_list_tables()` basada en la fila (empezando desde 0) o devuelve `false` en caso de que haya un error.

Puede devolver el número de filas del resultado de la consulta con `mysql_num_rows()`. Estas funciones de hecho no tienen alias de `mysql_result()` pero no es una buena práctica de programación utilizarla de la misma forma ya que su nombre es específico de tablas y si la utiliza de otra forma puede resultar confuso.

Por ejemplo:

```
// abra una conexión permanente a la base de datos
$connect = mysql_pconnect($hostname, $username, $password);

// devuelva la lista de tablas
$result = mysql_list_tables("databasel");

// procese una iteración por las filas de tablas y muestra los
// nombres
for($i=0; $i < mysql_num_rows($result); $i++) {
    print "Table name: ".mysql_tablename($result, $i)."<br>\n";
}
```

mysql_thread_id

```
int mysql_thread_id ([recurso conexión_mysql])
```

Devuelve un entero que contiene el Id. del subprocesso actual.

Por ejemplo:

```
// muestra - Id. de subprocesso: 2394 (por ejemplo)
print "Id. de subprocesso: ".mysql_thread_id();
```

mysql_unbuffered_query

```
query-result mysql_unbuffered_query(cadena consulta  
[, recuso conexion=mysql [, modo_resultado int]])
```

Devuelve un resultado de consulta no almacenado en bufer (si la consulta es la que produce el resultado, como por **ejemplo** SELECT o DESCRIBE), devuelve **true** e **si** la consulta no produce un resultado **pero** es satisfactoria (como DELETE o UPDATE) y **false** si la consulta falla. La consulta se **envía** a través de la **conexión** especificada (o de la última **conexión** abierta en **caso** de no especificar ninguna en concreto).

La diferencia entre esta **función** y `mysql_query()` es que, como el **resultado** no se almacena en **búfer**, utiliza **menos memoria** y puede trabajar con los resultados tras recuperar la **primera** fila. El inconveniente es que no puede utilizar `mysql_num_rows()`. Se utiliza principalmente en consultas **lentas** de gran **tamaño**.

El **parámetro** opcional `modo_resultado` puede ser `MYSQL_USE_RESULT` (el predeterminado) o `MYSQL_STORE_RESULT`, que **almacena** el resultado en **búfer**, al igual que `mysql_query()`.

Por **ejemplo**:

```
// abra una conexión permanente a la base de datos  
$connect = mysql_pconnect($hostname, $username, $password);  
  
// seleccione la base de datos  
mysql_select_db("database1", $connect);  
  
// defina y ejecute la consulta  
$sql = "SELECT field1,field2 FROM table1";  
$result = mysql_unbuffered_query($sql, $connect);  
  
// devuelva los datos tanto en matrices asociativas como  
// numericas (predeterminado)  
// procese una iteración por las filas para imprimir los datos  
while ($row = mysql_fetch_array($result)) {  
    print "Field1: ".$row["field1"]."<br>\n";  
    print "Field2: ".$row["field2"]."<br>\n";  
}
```




DBI Perl

La forma recomendada para conectarse a una base de datos (no solamente a MySQL) en Perl es por medio del modulo **DBI**. Se trata de una interfaz **genérica** que le permite acceder a distintos tipos de bases de datos de la misma forma. Junto con el modulo **DBI**, necesita un modulo **DBD**.

Cada modulo **DBD** es para una base de datos concreta, por lo que debe utilizar el asociado a **MySQL**.

NOTA: Para instalar compatibilidad DBI Perl con **MySQL**, necesita los módulos **DBI**, **DBD-mysql**, **Data-Dumper** y **File-Spec**. Puede descargar las últimas versiones de www.perl.com/CPAN. El software incluye instrucciones completas.

A lo largo de este apéndice, encontrara las siguientes convenciones utilizadas con los nombres de las variables:

- **\$dbh**. Un objeto identificador de base de datos, devuelto por los metodos `connect()` o `connect_cached()`
- **\$sth**. Un objeto identificador de instrucción devuelto por el metodo `prepare()` entre otros

- **\$drh**. Un objeto identificador controlador (apenas se utiliza en aplicaciones)
- **\$h**. Un identificador de base de datos, instruccion o controlador
- **\$rc**. Un codigo de **devolución booleano** (verdadero si es satisfactoria o falso en **caso** de que falle)
- **\$rv**. Un valor devuelto de ordenaciones, normalmente un entero
- **@ary**. Una matriz de valores, normalmente una **fila** de datos devuelta desde una **consulta**
- **\$rows**. Numero de filas procesadas o -1 si es desconocido
- **\$fh**. Un identificador de **archivos**
- **undef**. Un valor NULL o indefinido
- **\%attributes**. Referencia a un hash de valores de atributos. Lo utilizan **los** metodos para distintos propositos

Para utilizar el DBI, debe **cargar** el modulo DBI **al** inicio de su secuencia de comandos, **como** se indica a **continuación**:

```
use DBI;
```

Tras **ello**, necesita devolver un identificador de base de datos, normalmente con el **método** `connect()` de la clase DBI. Seguidamente, el identificador accede a **los** metodos que pueden ejecutar consultas y devolver resultados, y que habitualmente devuelven un identificador de instruccion.

Metodos de la clase DBI

Los metodos de la clase DBI son **los** que se proporcionan de **forma** completa desde la clase. El mas importante es el **método** `connect()`, que, si es **satisfactorio**, devuelve un identificador de base de datos.

available_drivers

```
@ary = DBI->available_drivers[($quiet)];
```

Devuelve una lista de controladores disponibles (modulos DBD). Muestra una advertencia si hay controladores con el mismo nombre. Si configura el parametro **opcional** `$quiet` con el valor `true`, se detiene esta advertencia.

connect

```
$dbh = DBI->connect($datasource, $username, $password  
[, \%attributes]);
```

Crea una **conexión** a la base de datos a través del origen de datos, el nombre de usuario y la contraseña especificados, y devuelve un identificador de base de datos. El origen de datos está **formado** por el controlador DBI (en este **caso**, `dbi:mysql`), el nombre de la base de datos, un nombre de servidor **opcional** (`localhost` si no se especifica), un nombre de **puerto opcional** (`3306` si no se especifica) y un número de modificadores, **cada** uno separado por **punto y coma**.

```
mysql_read_default_file=nombre de archivo
```

El archivo especificado se utiliza como archivo de opciones (el archivo de configuración **MySQL**, normalmente `my.ini` o `my.cnf` en el servidor).

```
mysql_read_default_group=nombre de grupo
```

Al leer un archivo de opciones, el grupo predeterminado que se utiliza es [`cliente`]. Esto cambia el grupo por [`nombre de grupo`]. La siguiente opción **hace** que se comprima la comunicación entre el cliente y el servidor:

```
mysql_compression=1
```

La siguiente opción especifica la ruta al socket Unix utilizado para conectarse al servidor:

```
mysql_socket=/ruta/a/socket
```

El nombre de usuario y contraseña opcionales **adoptan** los valores de las variables de **entorno** `DBI_USER` y `DBI_PASS` si no se especifican. Si la **conexión falla**, devolverá `undef` y configurará `$DBI::err` y `$DBI::errstr`.

Puede utilizar el parámetro `\%attribute` para configurar los distintos parámetros, como `AutoCommit` (recomendado), `RaiseError` y `PrintError`.

Por ejemplo:

```
my $hostname = 'localhost';
my $database = 'firstdb';
my $username = 'guru2b';
my $password = 'g00r002b';

#Conéctese a la base de datos
my $dbh = DBI->connect("dbi:mysql:$database:$hostname", $username,
    $password, (AutoCommit => 0, RaiseError => 1, PrintError => 0))
or die $DBI::errstr;
```

connect-cached

```
$dbh = DBI->connect_cached($data_source, $username, $password
    [ , \%attributes])
```

Igual que `connect()`, a **excepción** de que los detalles del identificador de base de datos **también** se almacenan en una matriz hash. Este mismo identificador de base de datos se utiliza por invocaciones idénticas posteriores a `connect_`

`cached()`, si sigue siendo valido. El atributo `CachedKids` (al que se accede por medio de `$dbh->{Driver}->{CachedKids}`) contiene datos de cache. Es un método bastante nuevo y es muy probable que cambie. No es lo mismo que una conexión permanente `Apache:DBI`.

data-sources

```
@ary = DBI->data_sources($driver [, \%attributes]);
```

Devuelve una matriz de todas las bases de datos disponibles en el controlador indicado (en este caso, `mysql`).

trace

```
trace($trace_level [, $trace_filename])
```

Este método activa o desactiva el rastreo. Cuando se invoca como método de la clase `DBI`, afecta al rastreo de todos los identificadores de base de datos e instrucciones. Cuando se invoca como método de identificador de base de datos o instrucciones, afecta al rastreo de dicho identificador (y los que en el futuro se deriven del mismo).

El nivel de rastreo puede estar comprendido entre 0 y 9. En el 0 esta desactivado, en el 1 permite un análisis general, el 2 es el mas utilizado y el resto de metodos añaden mas detalles sobre el controlador y `DBI`.

De forma predeterminada, el resultado se almacena en `STDERR` o se adjunta al archivo de rastreo si lo especificamos.

Por ejemplo:

```
DBI->trace(2);           # rastrea todo
$dbh->trace(2, "/tmp/dbi_trace.out"); # rastrea el
# identificador de base de datos
# a /tmp/dbi_trace.out
$dbh->trace(2);         # rastrea el identificador de
# instrucción
```

Tambien puede activar el rastreo si configura la variable de entorno `DBI_TRACE` con un numero (entre 0 y 9) o un archivo, en cuyo caso el rastreo se configurará en el nivel 2 y se guardará en dicho archivo.

Metodos DBI comunes a todos los identificadores

Los siguientes metodos estan disponibles para identificadores de base de datos, instrucciones y controladores. Se suelen utilizar para procesar errores.

err

```
srv = $h->err;
```

Devuelve el código de error nativo desde el último **método** (normalmente un entero).

errstr

```
$error_string = $dbh->errstr;
```

Devuelve una cadena de error del **fallo** de la invocación anterior.

Por ejemplo:

```
my $hostname = 'localhost';
my $database = 'firstdb';
my $username = 'guru2b';
my $password = 'g00r002b';

#Conéctese a la base de datos
my $dbh = DBI->connect("dbi:mysql:$database:$hostname", $username,
    $password) or die $DBI::errstr;
```

func

```
$h->func(@func_arguments, $func_name);
```

Se utiliza para invocar otros métodos de controlador no estándar. Adopta una matriz de argumentos y el nombre **del método** como argumentos.

No desencadena los mecanismos de **detección de errores** convencionales (como `RaiseError` o `PrintError`) no borra un error anterior (como `$DBI::err` o `$DBI::errstr`).

set_err

```
$rv = $h->set_err($err, $errstr [, $state, $method [, $rv]]);
```

Un nuevo **método** utilizado principalmente por controladores y subclases DBI. Configura los valores `err`, `errstr` y `state` para el identificador (para activar el procesamiento de **errores** a través de `RaiseError`, etc.).

`$method` define un nombre de **método** más útil para la cadena de errores y `$rv` un valor de **devolución** (normalmente `undef`).

Por ejemplo:

```
sub doodle {
    # intente 'garabatear'
    or return $sth->set_err(1234, "Nope. Sorry. Out of luck. It all
```

```
went wrong", undef, "doodle");  
}
```

state

```
$rv = $h->state;
```

Devuelve un código de error en formato **SQLSTATE**. Habitualmente devuelve el código general S1000 cuando del controlador no es compatible con **SQLSTATE**.

trace

```
trace($trace_level [, $trace-filename])
```

Define el método `trace` anterior.

trace-msg

```
$h->trace_msg($message_text [, $minimum-level]);
```

Si se activa el rastreo, almacena el texto del mensaje en el **archivo** de rastreo. Si se define el nivel mínimo (1, de **forma** predeterminada), solamente almacena el mensaje si el nivel de rastreo se encuentra **al menos** a ese nivel.

Funciones de utilidad DBI

El paquete DBI también incluye las **funciones** de utilidad DBI.

hash

```
$hash-value = DBI::hash($buffer [, $type]);
```

Devuelve **un** valor entero de 32 bits, que es el resultado de un algoritmo hash especificado por `$type` ejecutado en el bufer. Un tipo 0 (el predeterminado) realiza un hash **Perl** 5.1, con un resultado negativo.

Si el tipo es 1, se utiliza el algoritmo **Fowler/Noll/Vo**.

looks_like_number

```
@bool = DBI::looks_like_number(@array);
```

Devuelve una matriz booleana, con `true` en cada **elemento** de la matriz original que se parece a un número, `false` en cada **elemento** que no se parece y `undef` si **los** elementos están vacíos o no se han definido.

neat

```
$neat_string = DBI::neat($value [, $maxlen]);
```

Aplica un **formato** y aplica cornillas a la cadena por motivos visuales, no para pasarla al servidor de bases de datos. Si se **excede** la longitud máxima, la cadena se reduce en la **cantidad** indicada en `$maxlen-4` y se añaden puntos suspensivos (...) al extremo final. Si no se especifica `$maxlen`, se **utiliza** `SDBI::neat_maxlen`, con un valor predeterminado de 400.

neat-list

```
$neat_string = DBI::neat_list(\@listref [, $maxlen  
[, $field_sep]);
```

Invoca la función `neat()` en **cada elemento** de la lista y devuelve una cadena con todos los elementos separados por `$field_sep` que, de forma predeterminada, es una coma (,).

Metodos de identificadores de base de datos

Estos métodos están disponibles para el identificador de base de datos, por lo que tendrá que abrir una **conexión** antes de poder utilizarlos. Algunos de estos métodos devuelven un identificador de **instrucción**, que podrá procesar posteriormente por medio de los métodos de procesamiento de identificadores de instrucciones que veremos más adelante.

begin_work

```
$rc = $dbh->begin_work o die $dbh->errstr;
```

Inicia una transacción y desactiva `AutoCommit` hasta que la transacción finaliza con `commit()` o `rollback()`.

column_info

```
$sth = $dbh->column_info($catalog,$schema,$table,$column);
```

Un **método** experimental que devuelve un identificador de instrucciones **activo** para obtener **información sobre columnas**.

commit

```
$rc = $dbh->commit;
```

Confirma la transacción actual. Es necesario desactivar el parametro `AutoCommit` para que tenga efecto.

disconnect

```
$rc = $dbh->disconnect;
```

Utiliza el identificador de bases de datos especificado para desconectarse de la base de datos. Devuelve `true` si la operación es satisfactoria o `false` en caso contrario.

El método no define si se confirman o invierten las transacciones actualmente abiertas, por lo que debe hacerlo de forma especifica en su aplicacion antes de invocar `disconnect`.

do

```
$rv = $dbh->do($statement [, \%attributes [, @bind_values]]);
```

Prepara y ejecuta una instrucción SQL que devuelve el numero de filas afectadas. Devuelve `0E0` (se trata como `true`) si no hay filas afectadas o `undef` si se produce un error.

Normalmente se utiliza en consultas que no devuelven resultados (como `INSERT` o `UPDATE`) y que no utilizan marcadores de posición. Este método es mas rapido que los metodos equivalentes `prepare()` y `execute()`.

Devuelve un entero.

Por ejemplo:

```
my $hostname = 'localhost';
my $database = 'firstdb';
my $username = 'guru2b';
my $password = 'g00r002b';
```

#Conéctese a la base de datos

```
my $dbh = DBI->connect("dbi:mysql:$database:$hostname", $username,
    $password) or die $DBI::errstr;
```

```
$sql = "INSERT INTO customer(id,surname) VALUES (11,'Sandman')";
$dbh->do($sql);
```

foreign-key-info

```
$sth = $dbh->foreign_key_info($pk_catalog, $pk_schema,
    $pk_table [, $fk_catalog, $fk_schema, $fk_table]);
```

Un método experimental que devuelve un identificador de instrucciones para obtener información sobre claves secundarias. Los argumentos `$pk_catalog`,

`$pk_schema` y `$pk_table` especifican la tabla de clave principal. Los argumentos `$fk_catalog`, `$fk_schema` y `$fk_table`, la tabla de clave secundaria.

El resultado devuelto depende de las tablas que se proporcionen. Si solamente se proporciona la tabla de clave secundaria (pasando `undef` como argumento de clave principal), los resultados contendrán todas las claves secundarias en dicha tabla y las claves principales asociadas. Si solamente se proporciona la tabla de clave principal, los resultados contienen la clave principal de dicha tabla y todas las claves secundarias asociadas. Si se proporcionan ambas tablas, los resultados contendrán la clave secundaria de la tabla de clave secundaria que hace referencia a la clave principal de la tabla de clave principal.

get_info

```
$value = $dbh->get_info( $info_type );
```

Un método experimental que devuelve información de implementación.

ping

```
$rc = $dbh->ping;
```

Comprueba si la base de datos sigue en ejecución y la conexión se encuentra activa.

prepare

```
$sth = $dbh->prepare($statement [, \%attributes])
```

Devuelve una referencia a un identificador de instrucciones y prepara una instrucción SQL para su ejecución (por medio del método `execute`). Normalmente se preparan instrucciones que van a devolver resultados, como `SELECT` y `DESCRIBE`.

prepare_cached

```
$sth = $dbh->prepare_cached($statement [, \%attributes,  
[ $allow_active]);
```

Igual que `prepare` a excepción de que el identificador de instrucciones se almacena en un hash para que posteriores invocaciones de los mismos argumentos devuelvan el mismo identificador. El argumento `&allow_active` tiene tres parámetros. El predeterminado, 0, genera una advertencia e invoca `finish()` en el identificador de instrucciones antes de devolverlo. 1 invoca `finish()` pero elimina la advertencia. Si se configura como 2, el DBI no invocará `finish()` antes de devolver la instrucción.

primary-key

```
@key_column_names = $dbh->primary_key($catalog, $schema, $table);
```

Una interfaz experimental del método `primary_key_info` que devuelve una matriz de nombres de campos que **forman** la clave principal, secuencialmente, de la tabla especificada.

primary-key-info

```
$sth = $dbh->primary_key_info($catalog, $schema, $table);
```

Un método experimental para obtener información sobre columnas de clave principal.

quote

```
$quoted_string = $dbh->quote($string [, $data_type])
```

Devuelve una cadena con todos los caracteres especiales con conversiones de escape (como comillas simples y dobles) y **añade** otros signos. Si se especifica el tipo de datos, **Perl** lo utilizara para determinar el comportamiento predeterminado del uso de comillas.

quote_identifier

```
$sql = $dbh->quote_identifier( $name1[ , $name2, $name3,  
\%attributes ]);
```

Realiza una conversión de escape de todos los caracteres especiales en un identificador (como por ejemplo un nombre de campo) para utilizarlo en una **consulta**.

rollback

```
$rc = $dbh->rollback;
```

Invierte la **transacción** actual. Es necesario desactivar el parametro `AutoCommit` para que tenga efecto.

selectall_arrayref

```
$ary_ref = $dbh->selectall_arrayref($statement [, \%attributes  
[, @bind_values]]);
```

Un método que combina los metodos `prepare()`, `execute()` y `fetchall_arrayref()` en uno, para **facilitar** su utilización. Devuelve una

referencia a una matriz que contiene una referencia a una matriz para cada fila de datos devuelta desde la consulta. La instrucción también puede ser un identificador de instrucciones que ya se haya preparado, en cuyo caso el **método** no utilizará el **método** `prepare()`.

Puede configurar otros argumentos para pasarlos al **método** `selectall_arrayred()` en `%attributes`.

selectall-hashref

```
$hash_ref = $dbh->selectall_hashref($statement, $key_field  
[, \%attributes [,@bind_values]]);
```

Método que combina los métodos `prepare()`, `execute()` y `fetchall_hashref()` en uno, para **facilitar** su utilización. Devuelve una referencia a un hash que contiene una entrada para cada fila devuelta de la consulta. La clave de cada campo se especifica por medio de `$key_field` y el valor es una referencia a un hash. La instrucción también puede ser un **identificador** de instrucciones que ya se haya preparado, en cuyo caso el **método** ignora la parte `prepare()`.

selectcol_arrayref

```
$ary_ref = $dbh->selectcol_arrayref($statement [, \%attributes  
[,@bind_values]]);
```

Un **método** que combina los métodos `prepare()` y `execute()` con la obtención de una o varias **columnas** de todas las filas devueltas desde la consulta. Devuelve una referencia a una matriz que contiene los valores de las **columnas** de cada fila. La instrucción también puede ser un identificador de instrucciones que ya haya sido preparado, en cuyo caso el **método** ignora la parte `prepare()`.

De forma predeterminada, devuelve la **primera columna** de cada fila, pero puede devolver más si utiliza el atributo `Columns`, que es una referencia a una matriz que contiene el número de **columnas** que se va a utilizar. Por ejemplo:

```
# ejecute una consulta y devuelva las dos columnas  
my $array_ref = $dbh->selectcol_arrayref("SELECT first-name,  
surname FROM customer", { Columns=>[1,2] });  
# cree el hash a partir de los pares clave-valor de forma que  
$hash{$first_name} => surname  
my %hash = @$array_ref;
```

selectrow_array

```
@row_ary = $dbh->selectrow_array($statement [, \%attributes  
[,@bind_values]]);
```

Un **método** que combina los metodos `prepare()`, `execute()` y `fetchrow_array()` en uno para facilitar su utilizacion. Devuelve la **primera** fila de **datos** devuelta desde la consulta. La **instrucción** también puede ser un identificador de instrucciones, en cuyo caso el **método** ignora la parte `prepare()`.

selectrow_arrayref

```
$ary_ref = $dbh->selectrow_arrayref($statement [, \%attributes  
[, @bind_values]]);
```

Un **método** que combina los metodos `prepare()`, `execute()` y `fetchrow_arrayref()` en uno para facilitar su utilizacion. La **instrucción** también puede ser un identificador de instrucciones que ya se haya preparado, en cuyo caso el **método** ignora la parte `prepare()`.

selectrow_hashref()

```
$hash_ref = $dbh->selectrow_hashref($statement [, \%attributes  
[, @bind_values]]);
```

Un **método** que combina los metodos `prepare()`, `execute()` y `fetchrow_hashref()` en uno para facilitar su utilizacion. Devuelve la **primera** fila de **datos** devueltos desde la consulta. La **instrucción** también puede ser un identificador de instrucciones que ya se haya preparado, en cuyo caso el **método** ignora la parte `prepare()`.

table_info

```
$sth = $dbh->table_info($catalog, $schema, $table, $type  
[, \%attributes]);
```

Un **método** experimental que devuelve un identificador de instrucciones activo para obtener **información** sobre tablas y vistas de la base de datos.

tables

```
@names = $dbh->tables($catalog, $schema, $table, $type);
```

Una interfaz experimental del **método** `table_info()` que devuelve una matriz de nombres de tabla.

type_info

```
@type_info = $dbh->type_info($data_type);
```

Un **método** experimental que devuelve una referencia a una matriz que contiene información sobre tipos de datos admitidos por la base de datos y el controlador.

Metodos de procesamiento de instrucciones

Estos metodos funcionan en el identificador de instrucciones, que se obtiene al invocar un método de procesamiento de base de datos como `prepare()`.

bind_col

```
$type_info_all = $dbh->type_info_all;
```

Vincula un campo (empezando por 1) del resultado de una instrucción SELECT a una variable. Vea `bind_columns` para más información.

bind_columns

```
$rc = $sth->bind_col($column_number, \$column_variable);
```

Invoca el método `bind_col()` en cada campo de una instrucción SELECT. El número de referencias debe coincidir con el número de campos.

Por ejemplo:

```
# configure RaiseError con el valor 1 para no tener que
# comprobar todas las llamadas de metodos
$dbh->{RaiseError} = 1;
$sth = $dbh->prepare(q{SELECT first_name,surname FROM
customer});
$sth->execute;
my ($first_name, $surname);
# Vincule variables Perl a las columnas:
$rv = $sth->bind_columns(\$first_name, \$surname);
while ($sth->fetch) {
    print "$first_name $surname\n";
}
```

bind_param

```
$rv = $sth->bind_param($bind_num, $bind_value [%attributes |
$bind_type]);
```

Se utiliza para vincular un valor a un marcador de posición, indicado por un signo de interrogación (?). Se utiliza un marcador de posición cuando planeamos ejecutar varias veces una sencilla consulta, en la que cada una de las veces sólo cambia un parametro.

Por ejemplo:

```
$sth = $dbh->prepare("SELECT fname, sname FROM tname WHERE
sname LIKE ?");
$sth->bind_param(1, "Vusi%"); # placeholders begin from 1
$sth->execute;
```

No se pueden utilizar marcadores de posición para reemplazar a un nombre de tabla o de campo, o para reemplazar cualquier cosa que no sea un solo valor escalar. Por ejemplo, los siguientes casos son usos incorrectos de marcadores de posición:

```
SELECT fname, ? FROM tname WHERE sname LIKE 'Vusi%'
SELECT fname, sname FROM ? WHERE sname LIKE 'Vusi%'
SELECT fname, sname FROM tname WHERE sname IN (?)
```

También puede utilizar el parámetro de tipo de vinculación opcional para indicar que tipo de marcador de posición debe tener. Por ejemplo:

```
$sth->bind_param(1, $bind_value, (TYPE => SQL_INTEGER));
```

o el método abreviado equivalente, que requiere la importación de DBI con `use DBIqw(:sql_types);`

```
$sth->bind_param(1, $bind_value, SQL_INTEGER);
```

Por otra parte, puede utilizar el parámetro `\%attributes` de esta forma:

```
$sth->bind_param(1, $bind_value, {TYPE => SQL_INTEGER});
```

Esto devuelve un entero. Por ejemplo:

```
my $hostname = 'localhost';
my $database = 'firstdb';
my $username = 'guru2b';
my $password = 'g00r002b';

# Conectese a la base de datos
my $dbh = DBI->connect("dbi:mysql:$database:$hostname", $username,
$password) or die $DBI::errstr;

# Cree la consulta con un símbolo ? para indicar el marcador de
# posición
my $query = 'SELECT first_name,surname FROM customer WHERE id=?';

# Prepare la consulta
my $sth = $dbh->prepare($query);

# Cree una matriz de Id. para sustituir al marcador de posición
my @ids = (1,4,5,6);

# Procese una iteración por la matriz y ejecute la consulta
for(@ids) {
```

```

$sth->bind_param(1, $_, SQL_INTEGER);
$sth->execute();

my( $first_name, $surname);
$sth->bind_columns(undef, \$first_name, \$surname);

# Procese una iteración por las filas devueltas y muestre los
# resultados
while( $sth->fetch() ) {
    print "$first-name $surname\n";
}
}
$sth->finish();

```

bind_param_array

```

$rc = $sth->bind_param_array($p_num, $array_ref_or_value
    [, \%attributes | $bind_type])

```

Se utiliza para vincular una matriz a un **marcador de posición** definido en la **instrucción preparada**, lista para su ejecución con el **método** `execute_array()`.

Por ejemplo:

```

# configure RaiseError con el valor 1 para no tener que
# comprobar todas las llamadas a metodos
$dbh->{RaiseError} = 1;
$sth = $dbh->prepare("INSERT INTO customer(first-name, surname)
VALUES(?, ?)");
# Cada matriz debe tener el mismo numero de elementos
$sth->bind_param_array(1, [ 'Lyndon', 'Nkosi', 'Buhle' ]);
$sth->bind_param_array(2, [ 'Khumalo', 'Battersby', 'Lauria'
]);
my %tuple_status;
$sth->execute_array(\%tuple_status);

```

bind_param_inout

```

$rv = $sth->bind_param_inout($p_num, \$bind_value,
    $max_len [, \%attributes | $bind_type]) or ...

```

Igual que el **método** `bind_param()` pero le **permite** actualizar valores (para procedimientos almacenados). Actualmente **MySQL** no lo **admite**.

dump_results

```

$rows = $sth->dump_results($max_len, $lsep, $fsep, $fh);

```

Convierte todas las filas del identificador de instrucciones a `$fh` (de **forma** predeterminada `STDOUT`) después de invocar `DBI::neat_list` en **cada** fila. El separador de filas es `$lsep` (con el valor **predeterminado** `\n`), el separador de

campos es `$fsep` (una coma de forma predeterminada) y un valor de 35 en `$max_len`.

ex

```
$rv = $sth->execute(@bind_values);
```

Ejecuta una instrucción preparada y devuelve el número de filas afectadas (en consultas que no devuelven datos, como INSERT o UPDATE). Devuelve OEO (tratado como true) si no hay filas afectadas o undef si se produce un error. Puede utilizar uno de los métodos de obtención para procesar los datos.

Devuelve un entero.

Por ejemplo:

```
my $hostname = 'localhost'
my $database = 'firstdb';
my $username = 'guru2b';
my $password = 'g00r002b';

# Conectese a la base de datos
my $dbh = DBI->connect("dbi:mysql:$database:$hostname",
    $username, $password);

# Cree la consulta con un símbolo ? para indicar el marcador de
# posición
my $query = 'SELECT first_name,surname FROM customer WHERE
id=2';

# Prepare y ejecute la consulta
my $sth = $dbh->prepare($query);
$sth->execute();

my( $first_name, $surname);
$sth->bind_columns(undef, \$first_name, \$surname);

# Procese una iteración por las filas devueltas y muestre los
# resultados
while( $sth->fetch() ) {
    print "$first-name $surname\n";
}

$sth->finish();
```

execute_array

```
$rv = $sth->execute_array(\%attributes[, @bind_values]);
```

Ejecuta una instrucción preparada en cada parámetro configurado con `bind_param_array()` o en `@bind_values` y devuelve el número total de filas afectadas.

fetch

Un alias de `fetchrow_arrayref()`.

fetchall_arrayref

```
$table = $sth->fetchall_arrayref [[$slice[, $max_rows]]];
```

Devuelve todas las filas devueltas desde la **consulta** como referencia a una matriz que contiene una referencia por fila.

Si no se devuelven filas, devuelve una referencia a una matriz vacía. Si se produce un error, devuelve los datos obtenidos hasta el error, en **caso** de que haya alguno. El parametro opcional `$slice` puede ser una referencia a una matriz o a un hash. Si se trata de una referencia de matriz, el **método** utiliza `fetchall_arrayref` para obtener cada fila como referencia de matriz. Si se especifica un **índice**, devolverá campos (empezando desde 0). Si no hay parametros o si no se define `$slice`, el **método** funciona como si se hubiera **pasado** una referencia a una matriz vacía.

Si `$slice` es una referencia a un hash, el **método** utiliza `fetchall_hashref` para obtener cada fila como referencia de hash. Los campos devueltos se basaran en las claves hash. El valor hash siempre debe ser 1.

Lo entendera mejor con algunos ejemplos. Los dos primeros ejemplos devuelven referencias a una matriz de referencias de matriz. En primer lugar, para devolver solamente el segundo campo de cada fila, utilice lo siguiente:

```
$tbl_ary_ref = $sth->fetchall_arrayref([1]);
```

Para devolver el tercero y el ultimo campo de cada fila, utilice lo siguiente:

```
$tbl_ary_ref = $sth->fetchall_arrayref([-3,-1]);
```

Los dos siguientes ejemplos devuelven una referencia a una matriz de referencias hash. En primer lugar, para obtener todos los campos de todas las filas como referencia hash, utilice lo siguiente:

```
$tbl_ary_ref = $sth->fetchall_arrayref({});
```

Para obtener solamente los campos con el nombre `fname` y `sname` de cada fila como referencia hash, con las claves `FNAME` y `SNAME`, utilice lo siguiente:

```
$tbl_ary_ref = $sth->fetchall_arrayref({ FNAME=>1, SNAME=>1 });
```

Si el parametro opcional `$max_rows` se define como entero positivo (puede ser cero), el **número** de filas devueltas se **limitará** a este **número**. Puede invocar `fetchall_arrayref` de nuevo para devolver mas filas. Lo utilizara si no dispone de **suficiente memoria** para devolver todas las filas de una vez pero quiere que el rendimiento se beneficie de `fetchall_arrayref`.

fetchall_hashref

```
$hash_ref = $dbh->fetchall_hashref($key_field);
```

Devuelve una referencia a un hash que contiene, como mucho, una entrada por fila. Si la **consulta** no devuelve filas, el **método** devuelve una referencia a un hash vacío. Si se produce un error, devuelve los datos obtenidos hasta que surge el error, en caso de que haya obtenido alguno.

El parametro `$key_fie ld` especifica el nombre del campo que almacena el valor que se utilizara como clave del hash devuelto, o puede ser un **número** que **corresponda** a un campo (empieza en 1, no en 0). El **método** devuelve un error si la clave no coincide con un campo, bien como nombre o como numero.

Habitualmente se utiliza cuando el valor del campo de clave de **cada** fila es **exclusivo**; en caso contrario, los valores de la segunda fila y las siguientes **reem-**plazan a los anteriores que tengan la misma clave.

Por ejemplo:

```
$dbh->{FetchHashKeyName} = 'NAME_lc';  
$sth = $dbh->prepare("SELECT id, fname, sname FROM tname");  
$hash_ref = $sth->fetchall_hashref('id');  
print "The surname for id 8: $hash_ref->{8}->{sname}";.
```

fetchrow_array

```
@row = $sth->fetchrow_array;
```

Devuelve una matriz de valores de campo de la siguiente fila de datos, por medio de un identificador de instrucciones previamente preparado. **A los elementos** de la fila se puede acceder como `$row[0]`, `$row[1]`, etc. De esta **forma** se desplaza el **puntero** de fila para que la siguiente invocacion de este **método** devuelva la siguiente fila.

fetchrow_arrayref

```
$row_ref = $sth->fetchrow_arrayref
```

Devuelve una referencia a una matriz de valores de campo de la siguiente fila de datos, por medio de un identificador de instrucciones previamente preparado. **A los elementos** de la fila se puede acceder **como** `$row_ref->[0]`, `$row_ref->[1]` y así sucesivamente.

De esta **forma** se desplaza el **puntero** de fila para que la siguiente invocacion de este **método** devuelva la siguiente fila.

fetchrow_hashref

```
$hash_ref = $sth->fetchrow_hashref[($name)];
```

Devuelve una referencia a una tabla hash con el nombre del campo como clave y los contenidos del mismo como valores, por medio de un identificador de instrucciones previamente preparado. A los elementos de la fila se puede acceder como `$row[0]`, `$row[1]`, y así sucesivamente.

De esta forma se desplaza el puntero de fila de forma que la siguiente invocación de este método devuelva la siguiente fila.

El parametro opcional `name` especifica el nombre que se asignara a los atributos. El predeterminado es `NAME`, pero le sugerimos `NAME_uc` o `NAME_lc` (mayúsculas o minúsculas) por motivos de portabilidad.

Los metodos `fetchrow_arrayref` y `fetchrow_array` son considerablemente mas rapidos.

finish

```
$rc = $sth->finish;
```

Libera recursos del sistema asociados a un identificador de instrucciones, indicando que no se devolveran mas datos del mismo.

Devuelve `true` si es satisfactorio o `false` en caso contrario.

rows

```
$rv = $sth->rows;
```

Devuelve el numero de filas modificadas por la ultima instruccion SQL (por ejemplo despues de una **instrucción** `UPDATE` o `INSERT`) o `-1` si el numero es desconocido.

Atributos DBI comunes a todos los identificadores

Estos atributos proporcionan **información** y se pueden aplicar a todos los identificadores. Los mas habituales son los que se utilizan para procesar errores, `RaiseError` y `PrintError`. Por ejemplo, `$h->{'RaiseError'}` hace que los errores en ese identificador inicien excepciones.

Active

`Active` (booleano, solo lectura)

Verdadero si el objeto identificador esta **activo** (conectado a una base de datos si se trata de un identificador de base de datos o con mas datos por conseguir si se trata de un identificador de instrucciones).

ActiveKids

`ActiveKids` (entero, solo lectura)

El numero de identificadores de base de datos activos (en un identificador de controlador) o el numero de identificadores de instrucciones actualmente activos en un identificador de base de datos. **Activo** significa conectado a una base de datos si se trata de un identificador de base de datos o con mas datos por obtener si se trata de un identificador de instrucciones.

CachedKids

`CachedKids` (referencia hash)

En identificadores de base de datos, contiene una referencia a un hash de identificadores de **instrucción** creados por el método `prepare_cached()`. En identificadores de controladores, contiene un hash de **identificadores** de base de datos creados por `connect_cached()`.

ChopBlanks

`ChopBlanks` (booleano, heredado)

Especifica si los distintos metodos de obtencion eliminaran espacios en blanco delanteros o traseros de campos CHAR. Se configura como `true` si lo hacen y como `false` si no lo hacen.

CompatMode

`CompatMode` (booleano, heredado)

Las capas de emulacion garantizan un comportamiento compatible en el controlador.

FreeHashKeyName

`FetchHashKeyName` (cadena, heredado)

Especifica que nombre de atributo **utiliza** el método `fetchrow_hashref()` para obtener los nombres de campo de las claves hash. El **predeterminado** es `NAME` pero puede configurarlo como `NAME_lc` o `NAME_uc` por motivos de portabilidad.

HandleError

`HandleError` (referencia codigo, heredado)

Le permite crear su propia técnica de procesamiento de errores. Puede configurarlo como una referencia a una subrutina, que se invocara cuando se detecte un error (en el mismo caso en que se invocarían `RaiseError` y `PrintError`). Si la subrutina devuelve `false`, se comprueban `RaiseError` y `PrintError`. La subrutina se invoca con tres parametros: la cadena del mensaje de error (la misma que utilizarían `RaiseError` y `PrintError`), el identificador DBI y el primer valor devuelto por el método que ha fallado.

InactiveDestroy

`InactiveDestroy` (booleano)

Diseñado para aplicaciones Unix que dividen procesos secundarios. `False`, el valor predeterminado, indica que el identificador se destruye automáticamente cuando sobrepasa el ámbito. `True` indica que el identificador no se destruye automáticamente.

Kids

`Kids` (entero, solo lectura)

Contiene el número de identificadores de base de datos actuales (en identificadores de controladores) o el número de identificadores de instrucción actual en un identificador de base de datos.

LongReadLen

`LongReadLen` (entero sin firma, heredado)

Controla la longitud máxima de campos largos (BLOB y TEXT). Debe ser ligeramente superior al campo más largo. Al configurarlo como 0, los datos largos no se obtienen automáticamente (es decir, `fetch()` devuelve `undef`, en lugar del valor real del campo, cuando procesa campos largos). Configurarlos con un valor demasiado elevado es una pérdida de memoria.

LongTruncOK

`LongTruncOk` (booleano, heredado)

`False`, el valor predeterminado, indica que el intento de obtener valores largos superiores a `LongReadLen` hace que fracase la operación. `True` devuelve un valor truncado.

PrintError

`PrintError` (booleano, heredado)

Cuando se define como **true** (el predeterminado) los errores de un método generan una advertencia. Normalmente se configura como **false** cuando **RaiseError** se configura como **true**.

private_*

```
private_*
```

Puede almacenar informacibn propia adicional como atributo privado en un identificador DBI si especifica un nombre que empiece por **private_**. Debe asignar un nombre con el formato **private_nombre_descriptivo_de_su_módulo** y utilizar un solo atributo. Debido al funcionamiento del mecanismo **tie** de Perl, no puede utilizar el operador `||=` directamente para inicializar el atributo. Para inicializarlo, utilice un enfoque de dos pasos como el que mostramos a continuación:

```
my $descriptive-name = $dbh->{ private_nombre_descriptivo
_de_su_módulo };
$descriptive-name ||= $dbh->{private_nombre_descriptivo
_de_su_módulo } = { ... };
```

No puede utilizar lo siguiente como esperaba:

```
my $descriptive-name = $dbh->{private_nombre_descriptivo
_de_su_módulo } ||= { ... };
```

Profile

```
Profile (heredado)
```

Permite informar sobre estadísticas de tiempo de invocación de metodos. En la documentación **DBI::Profile** encontrara mas detalles.

RaiseError

```
RaiseError (booleano, heredado)
```

De forma predeterminada es **false**; cuando se configura como **true**, hace que los errores inicien excepciones en lugar de devolver codigos de error. Normalmente **PrintError** se configura como **false** cuando **RaiseError** es **true**.

ShowErrorStatement

```
ShowErrorStatement (booleano, heredado)
```

Cuando es **true**, adjunta el texto de la instrucción al mensaje de error de **RaiseError** y **PrintError**. Se aplica a errores de identificadores de instrucciones y a los metodos de identificación de bases de datos **prepare()** y **do()**.

Taint

`Taint` (booleano, heredado)

Si se define como `true`, y Perl se ejecuta en **modo taint** (-T), todos los datos de la base de datos se tratan como si fueran de este modo, como lo son los argumentos de la mayoría de los métodos DBI.

El valor predeterminado es `false`. En una fase posterior se pueden tratar más datos como de **modo taint**, por lo que debe prestar especial atención al utilizar `Taint`.

Warn

`TraceLevel` (entero, heredado)

De forma predeterminada es `true`, lo que activa las advertencias. Puede utilizar `$SIG{_WARN_}` para capturar advertencias.

Atributos de identificadores de base de datos

Los atributos de identificadores de base de datos son los que solamente están disponibles para un identificador de base de datos.

AutoCommit

`Warn` (booleano, heredado)

Si se configura como `true`, las instrucciones SQL se confirman automáticamente.

Si se configura como `false`, de forma predeterminada forman parte de una transacción y deben confirmarse o invertirse.

Driver

`AutoCommit` (booleano)

Contiene el identificador del controlador principal.

Por ejemplo:

`Driver` (identificador)

Name

`Name` (cadena)

El nombre de la base de datos.

RowCacheSize

`RowCacheSize (entero)`

Tamaño ideal que debería tener la cache de filas o `undef` si esta no se ha implementado. Si se **configura** con un número negativo, especifica el tamaño de memoria que se utilizara para almacenar en cache, 0 **determina** automáticamente el tamaño, 1 desactiva la cache y un número positivo mayor equivale al tamaño de la cache en filas.

Statement

`Statement (cadena, solo lectura)`

La última instrucción SQL que se haya pasado a `prepare()`

Atributos de identificadores de instrucciones

Estos atributos se aplican a identificadores de instrucciones que se han devuelto desde una consulta preparada. La mayoría son de sólo lectura y son específicos del identificador de instrucciones.

CursorName

`CursorName (cadena, solo lectura)`

Nombre del cursor asociado al identificador de instrucciones o `undef` si no se puede obtener.

NAME

`NAME (referencia a matriz, sólo lectura)`

Una referencia a una matriz de nombres de campos. Los nombres pueden estar en mayúscula, minúscula o mezclados. Puede utilizar `NAME_lc` o `NAME_uc` para mejorar la portabilidad entre sistemas.

Por ejemplo, para mostrar la segunda columna, utilice lo siguiente:

```
Ssth = Sdbh->prepare("select * from customer");
$sth->execute;
@row = $sth->fetchrow_array;
print "Column 2: $sth->{NAME}->[1]";
```

NAME_hash

`NAME-hash (referencia a hash, solo lectura)`

Información sobre nombres de campos devuelta como referencia a un hash. Los nombres pueden estar en mayúscula, minúscula o mezclados. Las claves del hash son los nombres de los campos y los valores son el índice del campo (siendo 0 el primer campo). Puede utilizar `NAME_lc` o `NAME_uc` para mejorar la portabilidad entre sistemas.

Por ejemplo:

```
Ssth = Sdbh->prepare("select first-name, surname from
customer");
Ssth->execute;
@row = Ssth->fetchrow-array;
print "First name: $row[{$sth->{NAME_hash}{first-name}}]\n";
print "Surname: $row[{$sth->{NAME_hash}{surname}}]\n";
```

NAME_lc

`NAME_lc` (referencia a matriz, solo lectura)

Igual que `NAME` pero solamente devuelve nombres en minúscula.

NAME_lc_hash

`NAME_lc_hash` (referencia a hash, solo lectura)

Igual que `NAME_hash` pero solamente devuelve nombres en minúscula.

NAME_uc

`NAME_uc` (referencia a matriz, solo lectura)

Igual que `NAME` pero solamente devuelve nombres en mayúscula

NAME_uc_hash

`NAME_uc_hash` (referencia a hash, sólo lectura)

Igual que `NAME_hash` pero solamente devuelve nombres en mayúscula.

NULLABLE

`NULLABLE` (referencia a matriz, sólo lectura)

Una referencia a una matriz que indica si el campo puede contener nulos o no. Mostrara 0 para no, 1 para sí y 2 si es desconocido.

Por ejemplo:

```
print "Field 1 can contain a NULL" if $sth->{NULLABLE}->[0];
```

NUM_OF_FIELDS

`NUM_OF_FIELDS` (entero, solo lectura)

El numero de campos que devolvera la instruccion preparada. Sera 0 en instrucciones que no devuelven campos (INSERT, UPDATE, et~.).

NUM_OF_PARAMS

`NUM_OF_PARAMS` (entero, solo lectura)

Numero de marcadores de posicion en la instruccion preparada.

ParamValues

`ParamValues` (referencia a hash, solo lectura)

Una referencia a un hash que contiene los valores vinculados a los marcadores de posicion o undef si no estan disponibles.

PRECISION

`PRECISION` (referencia a matriz, solo lectura)

Referencia a una matriz de enteros de cada campo, que **hace** referencia a la longitud maxima del campo (campos no numericos) o **al** numero maximo de digitos con significado (no el tamaiio de **representación**; excluye el signo, la coma decimal, el caracter E, etc.).

RowsInCache

`RowsInCache` (entero, solo lectura)

Numero de filas sin conseguir en cache o undef si el controlador no es compatible con una cache de filas local.

SCALE

`SCALE` (referencia a matriz, solo lectura)

Devuelve una referencia a una matriz de valores enteros de cada **columna**. Los valores NULL (undef) indican **columnas** en las que no se puede aplicar la escala.

Statement

`Statement` (cadena, solo lectura)

La ultima consulta SQL pasada al método `prepare()`

TYPE

`TYPE` (referencia a matriz, sólo lectura)

Referencia a una matriz de valores enteros (que representa el tipo de datos) de cada campo.

Atributos dinamicos

Se trata de atributos que **tienen** un ciclo vital breve y que solamente estan disponibles justo despues de configurarlos. Se aplican al identificador que se acaba de devolver.

err

`$DBI::err`

Igual que `$handle->err`.

errstr

`$DBI::errstr`

Igual que `$handle->errstr`

lasth

`$DBI::lasth`

Devuelve el identificador DIB utilizado con la ultima invocacion de metodos DBI o el principal del identificador (si existe) si la invocacion del **método** es DESTROY.

rows

`$DBI::rows`

Igual que `$handle->rows`

state

`$DBI::state`

Igual que `$handle->state`

Breve ejemplo de DBI Perl

El listado D.1 se conecta a un servidor, prepara una consulta con un marcador de posición, vincula una serie de valores a este marcador de posición y tras ello procesa una iteración por cada fila y cada consulta para mostrar los resultados.

Listado D.1: Example.pl

```
#!/usr/bin/perl -w

use strict; # no es necesario utilizar strict, pero debería
use DBI;    # el modulo principal

# defina variables con los detalles de la conexión
my $hostname = 'localhost';
my $database = 'firstdb';
my $username = 'guru2b';
my $password = 'g00r002b';

#Conéctese a la base de datos
my $dbh = DBI->connect("dbi:mysql:$database:$hostname",
    $username, $password);

# Defina y prepare la consulta con el símbolo ? para
# especificar una variable de vinculación.
my $sql = q(SELECT first_name,surname FROM customer WHERE
id=?);
my $sth = $dbh->prepare($sql);

# Cree una matriz de Id. que utilizará para sustituir al
marcador de posición
my @ids = (1,4,5,6);

# Procese una iteración por la matriz y ejecute la consulta
for(@ids) {
    $sth->bind_param(1, $_, SQL_INTEGER);
    $sth->execute();

    my( $first_name, $surname);
    $sth->bind_columns(undef, \$first_name, \$surname);

    # Procese una iteración por las filas devueltas y muestre los
# resultados
    while( $sth->fetch() ) {
        print "$first-name $surname\n";
    }
}
$sth->finish();

$dbh->disconnect;
```

E

API de base de datos Phyton

Python utiliza el API DB independiente de base de datos para su conectividad de base de datos. En **concreto** para MySQL, utiliza el modulo MySQLdb. La ultima version del API (en el **momento** de escribir este libro) era Phyton Database API Specification 2.0. Es un API relativamente nuevo y muchas de las características disponibles en otros API todavia no se han implementado. Sin embargo, su simplicidad **hace** que resulte muy sencillo de aprender y utilizar.

Encontrara MySQLdb e instrucciones completas de instalacion en <http://sourceforge.net/projects/mysql-python/>.

Atributos

Los atributos pueden estar disponibles para **todo** el modulo o pueden ser especificos de un cursor. En este **apartado** describiremos los atributos disponibles en funcion de cómo lo esten.

Atributos de modulo

Son atributos disponibles para **todo** el modulo.

APILEVEL

Una constante de cadena que contiene la version compatible del API-DB (2.0 si utiliza la version 2.0, por ejemplo).

CONV

Asigna tipos MySQL a objetos Python. De forma predeterminada es `MySQLdb.converters.conversions`.

PARAMSTYLE

Una constante de cadena que contiene el tipo de formato de marcadores de parámetro (marcador de posición) que utiliza la interfaz. Por ejemplo puede ser `format`:

```
...WHERE nombre_de_campo=%s
```

o puede ser `pyformat`:

```
...WHERE nombre_de_campo=%(nombre)s
```

THREADSAFETY

Una constante entera que contiene el nivel de seguridad de subprocessos. Puede ser 0 (no se comparten subprocessos), 1 (los subprocessos solamente pueden compartir el modulo), 2 (los subprocessos pueden compartir el modulo y las conexiones) o 3 (los subprocessos pueden compartir el modulo, las conexiones y los cursores). El predeterminado es 1.

Atributos de cursor

Son atributos especificos de un objeto de cursor, devuelto desde el método `cursor()`.

ARRAYSIZE

Especifica el numero de filas devueltas por el método `fetchmany()` y afecta al rendimiento del método `fetchall()`. El valor predeterminado es 1 o una fila por vez.

DESCRIPTION

Este atributo es de sólo lectura y es una secuencia de secuencias que describen las columnas del conjunto de resultados actual, cada uno con varios elementos. Son: `name`, `type_code`, `display_size`, `internal_size`, `precision`, `scale` y `null_ok`. Los elementos `name` y `type_code` son obligatorios y el resto se configuran como `None` si no existen valores con sentido para ellos.

Se configura como `None` si la consulta no devuelve ninguna fila o si todavia no se ha asignado.

ROWCOUNT

Atributo de solo lectura que indica el número de filas de la última consulta afectada o devuelta, o devuelve -1 si el número de filas es desconocido o si no se ha invocado una consulta.

Metodos

Los métodos pueden estar disponibles para **todo** el módulo, para una **conexión** o para un cursor. En este **apartado** describiremos los distintos métodos en función de si se corresponden a un módulo, a una **conexión** o a un cursor.

Metodos de modulo

Los métodos de módulo están disponibles para **todo** el módulo. El más importante es el método `connect ()`:

```
dbh = MySQLdb.connect (parametros)
```

Conecta **al** servidor y a la base de datos especificados con el nombre de usuario y contraseña especificados, y devuelve un objeto de **conexión** (o identificador de base de datos). Entre los parámetros que utiliza se incluyen los siguientes:

- **host**. De forma predeterminada es el servidor local.
- **user**. De forma predeterminada es el usuario actual.
- **passwd**. No hay valor predeterminado (contraseña en blanco).
- **db**. No hay valor predeterminado.
- **conv**. Diccionario para asignar literales **al** tipo Python. El predeterminado es `MySQLdb.converters.conversions`.
- **cursorclass**. La clase que utiliza `cursor ()`. De forma predeterminada ■ `MySQLdb.Cursors.Cursor`.
- **compress**. Activa el protocolo de compresión.
- **named-pipe**. En Windows, se conecta con una **canalización** con nombre.
- **init-command**. Especifica una **instrucción** para ejecutar el servidor de bases de datos en cuanto se **crea** la conexión.
- **read-default-file**. El archivo de configuración **MySQL** que se va a utilizar.
- **read-default-group**. El grupo predeterminado que se lee.
- **unix_socket**. En Unix, se conecta por medio del socket especificado. De forma predeterminada utiliza TCP.
- **port**. De forma predeterminada es 3306.

Por ejemplo:

```
dbh = MySQLdb.connect(user='guru2b', passwd='g00r002b',  
host='test.host.co.za', db='firstdb')
```

Metodos de conexión

Estos metodos se utilizan en un objeto de conexion, devuelto desde el **método** `MySQLdb.connect()`. En la **mayoría** de las ocasiones se utilizan **los metodos** `cursor()` y `close()`. Los metodos `commit()` y `rollback()` solamente se utilizan con transacciones.

BEGIN

```
dbh.begin()
```

Inicia una transaccion y desactiva **AUTOCOMMIT** si se encuentra activado, hasta que la transaccion finaliza con una llamada a `commit()` o `rollback()`.

CLOSE

```
dbh.close()
```

Cierra la **conexión** y **libera** los recursos asociados

COMMIT

```
dbh.commit()
```

Confirma todas las transacciones abiertas.

CURSOR

```
dbh.cursor([cursorClass])
```

Devuelve un nuevo objeto de cursor (que proporciona metodos para acceder a los datos y manipularlos). Puede especificar una clase diferente si lo desea (de **forma** predeterminada es `cursorclass`, especificada en la conexion, que de **forma** predeterminada adopta la clase `Cursor`).

ROLLBACK

```
dbh.rollback()
```

Invierte todas las transacciones abiertas. **Al** cerrar la **conexión** sin invocar explícitamente este **método**, se invoca implícitamente `rollback` en todas las instrucciones abiertas.

Metodos de cursor

Estos metodos **sirven** para acceder y manipular datos; **funcionan** como un objeto de cursor, devuelto por el **método** `cursor()`.

CLOSE

```
cursor.close()
```

Libera inmediatamente los recursos asociados al cursor

EXECUTE

```
cursor.execute(consulta [,parámetros])
```

Prepara y ejecuta una consulta de base de datos. El método también le permite utilizar marcadores de posición para optimizar consultas repetidas de un tipo similar si especifica distintos parámetros. Los marcadores de posición se suelen indicar por medio de un signo de interrogación (?), pero MySQL no lo admite actualmente. Tendrá que utilizar %s para indicar la presencia de un marcador de posición (si el atributo paramstyle se configura como format) ya que MySQLdb trata todos los valores como cadenas independientemente del tipo que tengan los campos.

Por ejemplo:

```
cursor.execute('INSERT INTO customer(first_name,surname) VALUES (%s, %s)', ('Mike', 'Harksen'))
```

También puede utilizar una asignación Python como segundo argumento si configura MySQLdb.paramstyle='pyformat'.

Puede utilizar listas de tuplas como segundo parámetro, pero es una práctica obsoleta. En su lugar, utilice executemany.

EXECUTEMANY

```
cursor.executemany(consulta,secuencia_de_parámetros)
```

Prepara una consulta de base de datos y, tras ello, ejecuta varias instancias con marcadores de posición para optimizar la repetición de consultas similares.

Por ejemplo:

```
cursor.executemany('INSERT INTO customer(first_name,surname) VALUES (%s, %s)', (('Mike', 'Harksen'), ('Mndeni', 'Vidal'), ('John', 'Vilakazi')))
```

También puede utilizar una asignación Python como segundo conjunto de argumentos si configura MySQLdb.paramstyle='pyformat'.

FETCHALL

```
cursor.fetchall()
```

Obtiene todas las filas del resultado de una consulta (desde el puntero de fila actual) y las devuelve como una secuencia de secuencias (una lista de tuplas).

El atributo `arraysize` del cursor puede afectar al rendimiento del método. Si se produce un error, se iniciara una excepcion.

Por ejemplo:

```
cursor.execute("SELECT first_name, surname FROM customer")
for row in cursor.fetchall():
    print "Firstname: ", row[0]
    print "Surname: ", row[1]
```

FETCHMANY

```
cursor.fetchmany([size=cursor.arraysize]);
```

Obtiene un numero de filas del resultado de una consulta y devuelve una secuencia de secuencias (una lista de tuplas). Tendra que especificar el numero de filas con el parametro de tamaño **opcional** o con `arraysize` del cursor si no se especifica. El método no devolvera mas filas de las que esten disponibles.

Es aconsejable utilizar el atributo `arraysize` por motivos de rendimiento o conservar el mismo parametro de tamaño entre las llamadas a `fetchmany`.

Si se produce un error, iniciara una excepcion.

FETCHNONE

```
cursor.fetchone()
```

Devuelve la siguiente fila de un **conjunto** de resultados de una consulta. Si se produce un error, se iniciara una excepcion. Por ejemplo:

```
cursor.execute("SELECT first_name, surname FROM customer")
row = cursor.fetchone():
    print "Firstname: ", row[0]
    print "Surname: ", row[1]
```

INSERT-ID

```
cursor.insert-id()
```

Un método estandar que no es del API DB que devuelve el valor del campo `AUTO_INCREMENT` anterior.

NEXTSET, SETINPUTSIZES y SETOUTPUTSIZES

```
cursor.nextset()
```

Estos metodos estandar no se utilizan actualmente en MySQL.

Breve ejemplo de Phyton

El listado E.1 le muestra los conceptos basicos relacionados con la conexion, ejecucion de consultas y procesamiento de resultados.

Listado E.1. Example.py

```
#!/usr/bin/env python

import MySQLdb
dbh = None
try:
    dbh = MySQLdb.connect(user='guru2b', passwd='g00r002b',
        host='test.host.co.za', db='firstdb')
except:
    print "Could not connect to MySQL server."
    exit(0)

try:
    cursor = dbh.cursor()
    cursor.execute("UPDATE customer SET surname='Arendse' WHERE
        surname='Burger'")
    print "Rows updated: ", cursor.rowcount
    cursor.close()
except:
    print "Could not update the table."

try:
    cursor = dbh.cursor()
    cursor.execute("SELECT first-name,surname FROM customer")
    for row in cursor.fetchall():
        print "Name: ", row[0], row[1]
    dbh.close()
except:
    print "Failed to perform query"
    dbh.close()
```



API Java

Java utiliza el API JDBC para acceder a bases de datos. Existen dos controladores **MySQL** principales: el "oficial" **Connector/J MySQL** (antes conocido como **MM.MySQL**) que puede descargar en el sitio de **MySQL** y el controlador **Caucho JDBC MySQL**.

El procedimiento básico consiste en crear una instancia de un objeto de conexión, un objeto de instrucción y, tras ello, un objeto de conjunto de resultados.

En este apéndice describiremos brevemente los principales métodos utilizados para la funcionalidad de base de datos en Java. También incluiremos un sencillo ejemplo de inserción y selección de datos.

Metodos generales

En este apartado describiremos algunos de los métodos utilizados para realizar conexiones o para acceder a datos de configuración desde un archivo.

getBundle

```
bundle.getBundle(nombre de archivo)
```

Carga datos desde un archivo de propiedades denominado `Config.properties`. Aunque no es específico de JDBC, se puede utilizar para almacenar datos de conexión en un archivo de configuración.

Por ejemplo, `Config.properties` contiene lo siguiente:

```
Driver = com.mysql.jdbc.Driver
Conn = jdbc:mysql://test.host.com/
firstdb?user=guru2b&password=g00r002b
```

El programa principal contendrá, por tanto, lo siguiente:

```
ResourceBundle rb = ResourceBundle.getBundle("Conn");
String conn = rb.getString("Conn");
...
Class.forName(rb.getString("Driver"));
Connection = DriverManager.getConnection(conn);
```

getConnection

```
jdbc:mysql-caucho://nombre_anfitrión [:port]/
nombre_base_de_datos
```

Solicita una conexión con los detalles especificados y devuelve un objeto de conexión. En función del controlador que utilice, los detalles de conexión se especificarán de forma diferente. Si se trata del controlador **Caucho**, el formato será el siguiente:

```
jdbc:mysql-caucho://nombre_de_anfitrión [:port]/
nombre_de_base_de_datos
```

Por ejemplo:

```
Connection connection =
DriverManager.getConnection("jdbc:mysql-caucho://
test.host.co.za/firstdb", "guru2b", "g00r002b");
```

El controlador **Connector/J** utiliza un formato ligeramente distinto:

```
jdbc:mysql://[nombre_de_anfitrión][:puerto]/
nombre_de_base_de_datos [?propiedad1=valor1][&propiedad=valor2]
```

Las propiedades pueden ser cualquiera de las enumeradas en la tabla F.1, aunque habitualmente basta con utilizar el nombre de usuario y la contraseña.

Tabla F.1. Propiedades de conexión

Nombre	Descripción
<code>autoReconnect</code>	De forma automática intenta conectarse de nuevo cuando se pierde la conexión. El valor predeterminado es <code>false</code> .

Nombre	Descripción
<code>characterEncoding</code>	Cuando se utiliza el conjunto de caracteres Unicode, indica que se debe utilizar codificación Unicode.
<code>initialTimeout</code>	Tiempo de espera en segundos entre intentos de conexión. El valor predeterminado es 2.
<code>maxReconnects</code>	Numero maximo de intentos de conexión. El valor predeterminado es 3.
<code>maxRows</code>	Número maximo de filas que se devuelven de una consulta o 0 (el predeterminado) para todas las filas.
<code>password</code>	La contraseña del usuario (no hay valor predeterminado).
<code>useUnicode</code>	Indica que se debe utilizar Unicode como conjunto de caracteres para la conexión. El valor predeterminado es <code>false</code> .
<code>user</code>	El usuario con el que se conecta (no hay valor predeterminado).

Por ejemplo:

```
DriverManager.getConnection("jdbc:mysql://
test.host.co.za/firstdb?user=guru2b&password=g00r002b");
```

getString

```
bundle.getString(cadena)
```

Consulte `getBundle`. Encontrara un ejemplo de como leer datos desde un archivo de configuracion.

Metodos de conexión

Los metodos de conexión requieren una conexión valida, que se devuelve del método `getConnection()`.

clearWarnings

```
connection.clearWarnings()
```

Borra todas las advertencias de la conexión y la devuelve en blanco.

close

```
connection.close()
```

Cierra la **conexión** a la base de datos y **libera** todos los recursos de conexión, devolviéndola en blanco.

commit

```
connection.commit()
```

Confirma las transacciones abiertas.

createStatement

```
connection.createStatement([int resultSetType, int  
resultSetConcurrency])
```

Devuelve un objeto `statement`, que es un mecanismo para pasar consultas a la base de datos, y recibe **los** resultados a través de su objeto de conexión. Con los argumentos opcionales, los objetos `ResultSet` generados **tendrán** el tipo y la concurrencia especificados.

getAutoCommit

```
connection.getAutoCommit()
```

Devuelve `true` si se establece el **modo** `AutoCommit` para la **conexión** y `false` en **caso** contrario.

getMetaData

```
connection.getMetaData()
```

Devuelve un objeto de metadatos de base de datos que contiene metadatos acerca de la base de datos **sobre** la que se ha realizado la conexión.

getTransactionIsolation

```
connection.getTransactionIsolation()
```

Devuelve un entero que contiene el nivel de aislamiento de transacciones para la conexión. Los niveles de aislamiento de transacciones pueden ser uno de los siguientes: `TRANSACTION_READ_UNCOMMITTED`, `TRANSACTION_READ_COMMITTED`, `TRANSACTION_REPEATABLE_READ`, `TRANSACTION_SERIALIZABLE` o `TRANSACTION_NONE`.

getTypeMap

```
connection.getTypeMap
```

Devuelve un objeto `Map` asociado a la conexión.

isClosed

```
connection.isClosed()
```

Devuelve **true** si la conexión se ha cerrado y **false** si sigue abierta.

isReadOnly

```
connection.isReadOnly()
```

Devuelve **true** si la conexión es de sólo lectura o **false** en caso contrario.

nativeSQL

```
connection.nativeSQL(String sql)
```

Devuelve una cadena con la cadena proporcionada convertida al SQL propietario del sistema.

prepareStatement

```
connection.prepareStatement(String sql)
```

Prepara una instrucción que se enviara a la base de datos, lo que significa que puede utilizar marcadores de posición (o parametros).

Utilice los metodos `setInt()` y `setString()` para configurar los valores de los parametros.

rollback

```
connection.rollback()
```

Deshace todos los cambios de la transacción actual.

setAutoCommit

```
connection.setAutoCommit(boolean mode)
```

Establece el modo `AutoCommit` de la conexión (**true** si lo establece, **false** en caso contrario).

setReadOnly

```
connection.setReadOnly(modos booleano)
```

Al pasar el método **true** configura la conexión a modo de sólo lectura

setTransactionIsolation

```
connection.setTransactionIsolation(nivel intl)
```

El nivel puede ser uno de los siguientes: `connection.TRANSACTION_READ_UNCOMMITTED`, `connection.TRANSACTION_READ_COMMITTED`, `connection.TRANSACTION_REPEATABLE_READ` o `connection.TRANSACTION_SERIALIZABLE`.

setTypeMap

```
connection.setTypeMap(Map map)
```

Establece el tipo de objeto Map de la conexión.

Metodos de instrucciones y de instrucciones preparadas

Es necesario invocar estos metodos a través de un objeto `statement` o `PreparedStatement` válido.

La mayoría de estos metodos se aplican tanto a instrucciones como a instrucciones preparadas. Los que tienen `preparedstatement` como objeto solamente se pueden invocar mediante una instrucción preparada, mientras que los metodos con `statement` como objeto se pueden invocar con ambos tipos.

addBatch

```
statement.addBatch(String sql)
preparedstatement.addBatch()
```

Añade la instrucción SQL a una lista actual de instrucciones, que se puede ejecutar con el método `executeBatch()`.

clearBatch

```
statement.clearBatch()
```

Borra del lote la lista de instrucciones que se han añadido por medio del método `addBatch()`.

clearWarnings

```
statement.clearWarnings()
```

Borra todas las advertencias asociadas a la instrucción

close

```
statement.close()
```

Libera todos los recursos asociados a la instrucción.

execute

```
statement.execute(String sql [,int autoGeneratedKeys | int[] columnIndexes  
| String[] columnNames])  
preparedstatement.execute()
```

Ejecuta una instrucción SQL. Devuelve `true` si la consulta devuelve un conjunto de resultados (como con una instrucción `SELECT`) y `false` si no genera ningún conjunto de resultados (como con instrucciones `INSERT` o `UPDATE`).

Las opciones indican que las claves autogeneradas **deben** estar disponibles para su **recuperación**, bien todas ellas o todas las de las matrices de enteros o cadenas, respectivamente.

executeBatch

```
statement.executeBatch()
```

Ejecuta todas las instrucciones del lote (añadidas por `addBatch`) y devuelve una matriz entera de contadores de **actualización** o `false` si no se ejecutó correctamente ninguna instrucción.

executeQuery

```
statement.executeQuery(String sql)  
preparedstatement.executeQuery()
```

Ejecuta una consulta que devuelve datos (como `SELECT` o `SHOW`) y devuelve un solo conjunto de resultados.

executeUpdate

```
statement.executeUpdate(String sql)  
preparedstatement.executeUpdate()
```

Ejecuta una consulta que modifica datos (como `UPDATE`, `INSERT` o `ALTER`) y devuelve el número de filas afectadas.

getConnection

```
statement.getConnection()
```

Devuelve el objeto de **conexión** que ha creado la instrucción

getFetchSize

```
statement.getFetchSize()
```

Devuelve en forma de entero el numero del tamaño de obtencion **predeterminado** de un objeto `ResultSet` de esta instruccion.

getMaxFieldSize

```
statement.getMaxFieldSize()
```

Devuelve en forma de entero el numero maximo de bytes que se pueden devolver de valores de **columnas** de caracteres y **binarias** para un objeto `ResultSet` desde esta instruccion.

getMaxRows

```
statement.getMaxRows()
```

Devuelve en forma de entero el numero maximo de **filas** que puede contener un objeto `ResultSet` desde esta instruccion.

getMoreResults

```
statement.getMoreResults([actual int])
```

Se desplaza hasta el siguiente resultado de esta instruccion y devuelve `true` si hay otro `ResultSet` valido o `false` si no lo hay.

Si no hay parametros, se cierran todos los objetos `ResultSet` actuales; en caso contrario, se procesan en funcion del valor de `actual` (que puede ser `LOSE_CURRENT_RESULT`, `KEEP_CURRENT_RESULT` o `CLOSE_ALL_RESULTS`).

getQueryTimeout

```
statement.getQueryTimeout()
```

Devuelve el numero de segundos que esperara el controlador a que se ejecute la consulta antes de que expire el plazo.

getResultSet

```
statement.getResultSet()
```

Devuelve un conjunto de resultados de la instruccion actual.

getResultType

```
statement.getResultSetType()
```

Devuelve el tipo de objetos `ResultSet` de la instrucción actual

getUpdateCount

```
statement.getUpdateCount()
```

Recupera el resultado actual como número de actualización; si el resultado es un objeto `ResultSet` o no hay más resultados, devuelve -1.

setXXX

```
preparedstatement.setXXX(parámetro int, valor xxx)
```

Establece un parámetro en una instrucción preparada previamente. Los parámetros empiezan desde 1. El valor debe ser de uno de los tipos enumerados en la tabla F.2.

Tabla F.2. Tipos SQL y métodos Set equivalentes

Tipo SQL	Método Java
BIGINT	setLong()
BINARY	setBytes()
BIT	setBoolean()
BLOB	setBlob()
CHAR	setString()
DATE	setDate()
DECIMAL	setBigDecimal()
DOUBLE	setDouble()
FLOAT	setDouble()
INTEGER	setInt()
LONGVARBINARY	setBytes()
LONGVARCHAR	setString()
NUMERIC	setBigDecimal()
OTHER	setObject()
REAL	setFloat()

Tipo SQL	Método Java
SMALLINT	setShort()
TIME	setTime()
TIMESTAMP	setTimestamp()
TINYINT	setByte()
VARBINARY	setBytes()
VARCHAR	setString()

Por ejemplo:

```
preparedstatement = connection.prepareStatement("UPDATE customer
SET surname = ? WHERE id=?");
preparedstatement.setString(1, "Burger");
preparedstatement.setInt(2, 9);
preparedstatement.executeUpdate();
```

setCursorName

```
statement.setCursorName(cadena nombre del cursor)
```

Define el nombre de cursor SQL que se utilizara en posteriores metodos `execute()`.

setEscapeProcessing

```
statement.setEscapeProcessing(modo booleano)
```

Establece el procesamiento de conversion de escape (si el **modo** es `true`) o lo desactiva (si el **modo** es `false`). El valor predeterminado es `false`. El procesamiento de conversion de escape no funciona con objetos `PreparedStatement`.

setFetchSize

```
statement.setFetchSize(tamaño int)
```

Indica al controlador cuantas filas **deben** devolverse de la base de datos cuando se necesitan mas filas para la instruccion.

setMaxFieldSize

```
statement.setMaxFieldSize(límite int)
```

Establece el numero maximo de bytes de una **columna** `ResultSet` binaria o de caracteres.

setMaxRows

```
statement.setMaxRows(límite int)
```

Establece el número máximo de filas que puede contener `ResultSet`.

setQueryTimeout

```
statement.setQueryTimeout(segundos int)
```

Establece el número de segundos que espera el controlador a que se ejecute una consulta antes de que se agote el plazo.

Metodos ResultSet

Estos métodos requieren un objeto `ResultSet` válido, devuelto por el método `getResultSet()`.

absolute

```
resultset.absolute(fila int)
```

Desplaza el cursor hasta la fila especificada del **conjunto** de resultados (las filas empiezan en 1). Puede utilizar un número negativo para desplazar una fila que empiece al final del **conjunto** de resultados. Devuelve `true` si el cursor se encuentra en la fila y `false` en caso contrario.

afterLast

```
resultset.afterlast()
```

Desplaza el cursor hasta el final del **conjunto** de resultados.

beforeFirst

```
resultset.beforeFirst()
```

Desplaza el cursor hasta el inicio del **conjunto** de resultados

cancelRowUpdates

```
resultset.cancelRowUpdates()
```

Cancela las actualizaciones realizadas en la fila actual del **conjunto** de resultados.

close

```
resultset.close ()
```

Cierra el conjunto de resultados y libera todos los recursos asociados.

deleteRow

```
resultset.deleteRow()
```

Elimina la fila `ResultSet` actual de la base de datos (y el conjunto de resultados).

findColumn

```
resultset.findColumn(String field-name)
```

Asigna el nombre de campo a la columna del conjunto de resultados y devuelve el índice de la columna en forma de entero.

first

```
resultset.first ()
```

Desplaza el cursor hasta la primera fila del conjunto de resultados. Devuelve `true` si hay una primera fila válida y `false` en caso contrario.

getXXX

```
resultset.getXXX(cadena nombre_de_campo | índice_de_campo int)
```

Devuelve los contenidos de un campo del tipo especificado. Puede identificar el campo por su nombre o por su posición.

En la tabla F.3 se incluyen los tipos SQL y sus métodos Java equivalentes.

Tabla F.3. Tipos SQL y métodos Get equivalentes

Tipo SQL	Método Java
BIGINT	<code>getLong()</code>
BINARY	<code>getBytes()</code>
BIT	<code>getBoolean()</code>
BLOB	<code>getBlob()</code>
CHAR	<code>getString()</code>
DATE	<code>getDate()</code>
DECIMAL	<code>getBigDecimal()</code>

Tipo SQL	Método Java
DOUBLE	getDouble()
FLOAT	getDouble()
INTEGER	getInt()
LONGVARBINARY	getBytes()
LONGVARCHAR	getString()
NUMERIC	getBigDecimal()
OTHER	getObject()
REAL	getFloat()
SMALLINT	getShort()
TIME	getTime()
TIMESTAMP	getTimestamp()
TINYINT	getByte()
VARBINARY	getBytes()
VARCHAR	getString()

getCursorName

```
resultset.getCursorName()
```

Devuelve el nombre del cursor utilizado por el **conjunto** de resultados.

getFetchSize

```
resultset.getFetchSize()
```

Devuelve el tamaño de obtención del objeto de **conjunto** de resultados.

getMetaData

```
resultset.getMetaData()
```

Devuelve un objeto **ResultSetMetaData** con el número, tipo y propiedades de las columnas del **conjunto** de resultados.

getRow()

```
resultset.getRow()
```

Devuelve un entero que contiene el número de **fila** actual.

getStatement

```
resultSet.getStatement()
```

Devuelve el objeto de **instrucción** que ha creado el **conjunto** de resultados.

getType

```
resultSet.getType()
```

Devuelve el tipo del objeto de **conjunto** de resultados.

getWarnings

```
resultSet.getWarnings()
```

Devuelve la **primera** Warning desencadenada por una llamada desde un **metodo ResultSet** en este **conjunto** de resultados.

insertRow

```
resultSet.insertRow()
```

Añade los contenidos de la fila de **inserción** a la base de datos (y al **conjunto** de resultados).

isAfterLast

```
resultSet.isAfterLast()
```

Devuelve **true** si el cursor se encuentra por detras de la ultima fila del **conjunto** de resultados y **false** en caso contrario.

isBeforeFirst

```
resultSet.isBeforeFirst()
```

Devuelve **true** si el cursor se encuentra por delante de la **primera** fila del **conjunto** de resultados y **false** en caso contrario.

isFirst

```
resultSet.isFirst()
```

Devuelve **true** si el cursor se encuentra en la **primera** fila del **conjunto** de resultados y **false** en caso contrario.

isLast

```
resultset.islast()
```

Devuelve `true` si el cursor se encuentra en la ultima fila del **conjunto** de resultados, `false` en caso contrario.

last

```
resultset.last()
```

Desplaza el cursor hasta la ultima fila del **conjunto** de resultados. Devuelve `true` si hay una ultima fila valida o `false` en caso contrario.

moveToCurrentRow

```
resultset.moveToCurrentRow()
```

Desplaza el cursor a la posicion del cursor que recuerda, que suele ser la fila actual. No **funciona** si el cursor no se encuentra en la fila de insercion. Consulte el **método** `moveToInsertRow()`.

moveToInsertRow

```
resultset.moveToInsertRow()
```

Desplaza el cursor hasta la fila de insercion (un bufer en el que se puede añadir una nueva fila con un **método de actualización**). Recuerda la posicion actual del cursor, que se puede devolver con el **método** `moveToCurrentRow()`.

next

```
resultset.next()
```

Desplaza el cursor hasta la siguiente fila del **conjunto** de resultados y devuelve `true` si hay una siguiente fila. Devuelve `false` en caso contrario (se ha llegado al final). Por ejemplo:

```
connection = DriverManager.getConnection(url, "guru2b",
"q00r002b");
statement = connection.createStatement();
resultset = statement.executeQuery("SELECT first_name,surname
FROM customer");
while(resultset.next()) {
    String first-name = resultset.getString("first_name");
    String surname = resultset.getString("surname");
    System.out.print("Name: " + first-name + " " + surname);
}
```

previous

```
resultset.previous()
```

Desplaza el cursor hasta la fila anterior del **conjunto** de resultados y devuelve `true` si existe una fila anterior o `false` si no existe (se ha llegado al principio).

Por ejemplo:

```
while (resultset.previous()) {  
    ...  
}
```

refreshRow

```
resultset.refreshRow()
```

Actualiza la fila actual del **conjunto** de resultados con el valor más reciente de la base de datos.

relative

```
resultset.relative(filas int)
```

Desplaza el cursor **hacia delante** (si `filas` es positivo) o **hacia atrás** (si es negativo) el número de posiciones indicado en `filas`.

rowDeleted

```
resultset.rowDeleted()
```

Devuelve `true` si se ha detectado una fila como eliminada en el **conjunto** de resultados o `false` en caso contrario.

rowInserted

```
resultset.rowDeleted()
```

Devuelve `true` si se ha detectado una fila como insertada en el **conjunto** de resultados o `false` en caso contrario.

rowUpdated

```
resultset.rowUpdated()
```

Devuelve `true` si se ha detectado una fila como actualizada en el **conjunto** de resultados o `false` en caso contrario.

setFetchSize

```
resultset.setFetchSize(filas int)
```

Indica al controlador cuantas filas se **deben** devolver desde la base de datos cuando el conjunto de resultados necesite mas filas.

updateXXX

Actualiza el campo con un valor del tipo especificado. En la tabla F.4 se incluyen los tipos SQL y sus metodos Java equivalentes.

Tabla F.4. Tipos SQL y metodos Update equivalentes

Tipo SQL	Método Java
BIGINT	updateLong()
BINARY	updateBytes()
BIT	updateBoolean()
BLOB	updateBlob()
CHAR	updateString()
DATE	updateDate()
DECIMAL	updateBigDecimal()
DOUBLE	updateDouble()
FLOAT	updateDouble()
INTEGER	updateInt()
LONGVARBINARY	updateBytes()
LONGVARCHAR	updateString()
NUMERIC	updateBigDecimal()
NULL	updateNull()
OTHER	updateObject()
REAL	updateFloat()
SMALLINT	updateShort()
TIME	updateTime()
TIMESTAMP	updateTimestamp()
TINYINT	updateByte()
VARBINARY	updateBytes()
VARCHAR	updateString()

updateRow

```
resultSet.updateRow()
```

Actualiza la base de datos con los contenidos de la fila actual del conjunto de resultados.

wasNull

```
resultSet.wasNull()
```

Devuelve **true** si la lectura del campo anterior era un **NULL** de SQL y **false** en caso contrario.

Metodos ResultSetMetaData

Estos metodos requieren un objeto **ResultSetMetaData** valido devuelto desde el método `getMetaData()`. Obtienen información sobre los resultados. Las columnas empiezan en 1 por motivos de desplazamiento y del contador.

getColumnCount

```
resultSetmetadata.getColumnCount()
```

Devuelve el numero de columnas del conjunto de resultados.

getColumnDisplaySize

```
resultSetmetadata.getColumnDisplaySize(columna int)
```

Devuelve la anchura maxima de caracteres de la columna especificada.

getColumnName

```
resultSetmetadata.getColumnName(columna int)
```

Devuelve el nombre de campo de la columna especificada.

getColumnType

```
resultSetmetadata.getColumnType(columna int)
```

Devuelve el tipo SQL de la columna especificada.

getColumnTypeName

```
resultSetmetadata.getColumnTypeName(columna int)
```

Devuelve el nombre del tipo específico de base de datos de la columna especificada.

getPrecision

```
resultSetmetadata.getPrecision(columna int)
```

Devuelve el número de decimales de la columna especificada.

getScale

```
resultSetmetadata.getScale(columna int)
```

Devuelve el número de dígitos por detrás del punto decimal de la columna especificada.

getTableName

```
resultSetmetadata.getTableName(columna int)
```

Devuelve el nombre de la tabla a la que pertenece la columna especificada.

isAutoIncrement

```
resultSetmetadata.isAutoIncrement(columna int)
```

Devuelve `true` si la columna especificada es un campo de autoincremento y `false` en caso contrario.

isCaseSensitive

```
resultSetmetadata.isCaseSensitive(columna int)
```

Devuelve `true` si la columna especificada discrimina entre mayúsculas y minúsculas, y `false` en caso contrario.

isDefinitelyWritable

```
resultSetmetadata.isDefinitelyWritable(columna int)
```

Devuelve `true` si la escritura en la columna especificada es satisfactoria y `false` en caso contrario.

isNullable

```
resultSetmetadata.isNullable(columna int)
```

Devuelve el estado **nulo** de la columna especificada, que puede ser `columnNotNulls`, `columnNullable` o `columnNullableUnknown`

isReadOnly

```
resultSetmetadata.isReadOnly(columna int)
```

Devuelve `true` si la columna especificada es de sólo lectura y `false` en caso contrario.

isSearchable

```
resultSetmetadata.isSearchable(columna int)
```

Devuelve `true` si la columna especificada se puede utilizar en una cláusula `WHERE` y `false` en caso contrario.

isSigned

```
resultSetmetadata.isSigned(columna int)
```

Devuelve `true` si la columna especificada es una columna con **firma** y `false` en caso contrario.

isWritable

```
resultSetmetadata.isWritable(columna int)
```

Devuelve `true` si se puede escribir en la columna especificada y `false` en caso contrario.

Metodos SQLException

Estos métodos se utilizan cuando se crea un objeto `SQLException`.

getErrorCode

```
SQLException.getErrorCode()
```

Devuelve el código de error del proveedor

getMessage

```
SQLException.getMessage()
```

Heredado de `Throwable`. Devuelve la cadena de mensaje para `Throwable`.

getNextException

```
SQLException.getNextException()
```

Devuelve la siguiente `SQLException` o **nulo** en caso de que no haya ninguna.

getSQLState

```
SQLException.getSQLState()
```

Devuelve un identificador `SQLState`.

printStackTrace

```
SQLException.printStackTrace(PrintStream s)
```

Heredado de la clase `Throwable`. Este método imprime el rastreo de pila en el flujo de errores estandar.

setNextException

```
setNextException(SQLException e)
```

Añade una `SQLException`.

Metodos Warning

Se utilizan cuando se ha creado un objeto `SQLWarning`

getNextWarning

```
SQLWarning.getNextWarning()
```

Devuelve el siguiente `SQLWarning` o **nulo** en caso de que no haya ninguno.

setNextWarning

```
SQLWarning.setNextWarning(SQLWarning w)
```

Añade un `SQLWarning`.

Breve ejemplo de Java

El listado F.1 adopta tres parametros en la linea de comandos, que añade a la tabla customer. Ejecutelo de esta forma:

```
% /usr/java/j2sdk1.4.1/bin/java InsertSelect 10 Leon Wyk
```

Listado F.1: InsertSelect.Java

```
import java.sql.*;
import java.util.*;

// La clase insertRecord hace lo justo para insertar un registro
// y también realiza una resolución de excepciones minima
public class InsertRecord {

    public static void main(String argv[]) {
        Connection dbh = null;
        ResultSet resultset;

        try {
            Class.forName("com.mysql.jdbc.Driver");
        }
        catch (Exception e) {
            System.err.println("Unable to load driver.");
            e.printStackTrace();
        }

        try {
            Statement sth,sth2;
            // conectese a la base de datos por medio del
            // controlador Connector/J
            dbh = DriverManager.getConnection("jdbc:mysql://
            localhost/firstdb?user=mysql");

            // cree una instancia del objeto de instruccion y
            // ejecute la consulta (una
            // consulta de actualización)
            // los tres campos argv[] provienen de la linea de
            // comandos
            sth = dbh.createStatement();
            sth.executeUpdate("INSERT INTO customer(id,first_name,surname)
            VALUES(" + argv[0] + ", '" + argv[1] + "', '" + argv[2] + "'");
            sth.close();

            // cree una instancia del objeto de instruccion y
            // ejecute la consulta SELECT
            sth2 = dbh.createStatement();
            resultset = sth2.executeQuery("SELECT first_name,surname
            FROM customer");
```

```

    // procese una iteración por el conjunto de resultados y
    // muestre los resultados
    while(resultset.next()) {
        String first-name = resultset.getString("first_name");
        String surname = resultset.getString("surname");
        System.out.println("Name: " + first-name + " " +
            surname);
    }
    sth2.close();
}
catch( SQLException e ) {
    e.printStackTrace();
}
finally {
    if(dbh != null) {
        try {
            dbh.close();
        }
        catch(Exception e) {}
    }
}
}
}

```



El API C incorpora distribuciones MySQL y se incluye en la biblioteca `mysqlclient`. Muchos de los clientes MySQL se escriben en C y la mayoría de los API de otros lenguajes utilizan el API de C (lo puede apreciar, por ejemplo, en las semejanzas entre las funciones de C y de PHP).

También puede utilizar el API C en programación C++; en enfoques orientados a objetos, puede utilizar MySQL++, que se encuentra disponible en el sitio Web de MySQL (www.mysql.com):

Tipos de datos del API C

Los tipos de datos C representan los datos con los que trabajara al interactuar con el servidor de base de datos. Son muy variados; algunos son estructuras y otros simples booleanos. Para dominar el API C, tendrá que familiarizarse con estos tipos y con las funciones que devuelven o con la forma de utilizarlos como argumentos.

my_ulonglong

Un tipo numérico comprendido entre -1 y $1.84e19$, que se utiliza para devolver valores de funciones como `mysql_affected_rows()`, `mysql_num_rows()` y `mysql_insert_id()`.

MYSQL

Un identificador de base de datos (una **conexión al servidor** de base de datos). La variable se inicializa con la función `mysql_init()` y es utilizada por la mayoría de las funciones API.

MYSQL_FIELD

Datos de campos devueltos por la función `mysql_fetch_field()`, incluyendo el nombre de campo, el tipo y el tamaño. Los valores—de los campos se almacenan en la estructura `MYSQL_ROW`. En esta estructura encontrará los siguientes miembros:

- `char * name`. Nombre de campo en forma de cadena terminada en nulo.
- `char * table`. Tabla que contiene el campo o una cadena vacía en caso de que no haya ninguno (*¿¿ninguno?¿¿*) (como en el caso de un campo calculado).
- `char * def`. Valor predeterminado del campo en forma de cadena terminada en nulo. Solamente esta disponible si se ha utilizado la función `mysql_list_field()` para devolver `MYSQL_RES`.
- `enum enum_field_types type`. El tipo de campo. Este valor se corresponde al tipo de campo MySQL, como se indica en la tabla G.1.

Tabla G.1. Tipos de campos MySQL

Valor	Tipo MySQL
<code>FIELD_TYPE_TINY</code>	<code>TINYINT</code>
<code>FIELD_TYPE_SHORT</code>	<code>SMALLINT</code>
<code>FIELD_TYPE_LONG</code>	<code>INTEGER</code>
<code>FIELD_TYPE_INT24</code>	<code>MEDIUMINT</code>
<code>FIELD_TYPE_LONGLONG</code>	<code>BIGINT</code>
<code>FIELD_TYPE_DECIMAL</code>	<code>DECIMAL</code> o <code>NUMERIC</code>
<code>FIELD_TYPE_FLOAT</code>	<code>FLOAT</code>
<code>FIELD_TYPE_DOUBLE</code>	<code>DOUBLE</code> o <code>REAL</code>
<code>FIELD_TYPE_TIMESTAMP</code>	<code>TIMESTAMP</code>
<code>FIELD_TYPE_DATE</code>	<code>DATE</code>
<code>FIELD_TYPE_TIME</code>	<code>TIME</code>
<code>FIELD_TYPE_DATETIME</code>	<code>DATETIME</code>
<code>FIELD_TYPE_YEAR</code>	<code>YEAR</code>

Valor	Tipo MySQL
FIELD_TYPE_STRING	CHAR o VARCHAR
FIELD_TYPE_BLOB	BLOB o TEXT
FIELD_TYPE_SET	SET
FIELD_TYPE_ENUM	ENUM
FIELD_TYPE_NULL	De tipo NULL
FIELD_TYPE_CHAR	Obsoleto, sustituido por FIELD_TYPE_TINY

- **unsigned int length.** Anchura maxima del campo, determinada por la definición de la tabla.
- **unsigned int max_length.** Anchura maxima de un campo en el conjunto de resultados. Normalmente se confunde con el miembro `length`, pero `max_length` suele ser de menor tamaño. Contiene cero si la función `mysql_use_result()` ha devuelto `MYSQL_RES`.
- **unsigned int flags.** Se puede configurar cualquiera de los indicadores enumerados en la tabla G.2; proporcionan información adicional sobre el campo.

Tabla G.2. Indicadores

Indicador	Descripción
NOT_NULL_FLAG	Se define como NOT NULL
PRI_KEY_FLAG	Parte de la clave principal
UNIQUE_KEY_FLAG	Parte de una clave exclusiva
MULTIPLE_KEY_FLAG	Parte de una clave no exclusiva
UNSIGNED_FLAG	Se define como UNSIGNED
ZEROFILL_FLAG	Se define con ZEROFILL
BINARY_FLAG	Se define como BINARY
AUTO_INCREMENT_FLAG	Se define como campo AUTO_INCREMENT
ENUM_FLAG	Se define como un ENUM (obsoleto; utilice FIELD_TYPE_ENUM)
SET_FLAG	Se define como SET (obsoleto; utilice FIELD_TYPE_SET)
BLOB_FLAG	Se define como BLOB o TEXT (obsoleto; utilice FIELD_TYPE_BLOB)

Indicador	Descripción
TIMESTAMP_FLAG	Se define como <code>TIMESTAMP</code> (obsoleto; utilice <code>FIELD_TYPE_TIMESTAMP</code>)

A **continuación** incluimos un ejemplo de uso de una de estas opciones:

```
if (field->flags & MULTIPLE_KEY_FLAG) {
    /* debe hacer algo, ya que el campo es una clave multiple */
}
```

Las macros que se incluyen en la tabla G.3 facilitan la labor de **probar** algunos de los valores de los indicadores.

Tabla G.3. Macros

Macro	Descripción
<code>IS_NOT_NULL(indicadores)</code>	Verdadero si el campo se define como NOT NULL
<code>IS_PRIMARY_KEY(indicadores)</code>	Verdadero si el campo es o forma parte de una clave primaria
<code>IS_BLOB(indicadores)</code>	Verdadero si el campo es de tipo BLOB o TEXT (obsoleto; utilice <code>FIELD_TYPE</code>)
<code>IS_NUM(indicadores)</code>	Verdadero si es un campo numérico

- `unsigned int decimals`. Numero de posiciones decimales utilizados por un campo numerico.

MYSQL_FIELD_OFFSET

Representa la **posición** del puntero de campos dentro de una lista de campos, comenzado desde 0 para el primer campo. Lo utiliza la función `mysql_field_seek()`.

MYSQL_RES

Una estructura que contiene los resultados de una **consulta** que devuelve datos (como SELECT, DESCRIBE, SHOW o EXPLAIN).

MYSQL_ROW

Una sola fila de datos que se obtiene de la función `mysql_fetch_row()`.

Todos los datos se representan como una matriz de **cadenas** que puede **contener** bytes nulos si algun dato es binario.

Funciones del API C

Las funciones del API C **permiten** abrir y cerrar conexiones **al servidor**, ejecutar consultas, analizar resultados de consultas, depurar y ejecutar **tareas administrativas**. Para dominar el API C, debe conocerlas en profundidad y saber como interactuan con **los tipos de datos del API C**.

mysql_affected_rows

```
my_ulonglong mysql_affected_rows(MYSQL *mysql)
```

Devuelve el numero de filas afectadas por la ultima **consulta** (por ejemplo, el numero de filas eliminadas con una **instrucción DELETE** o el numero de filas devueltas por una **instrucción SELECT**, en cuyo caso es **igual** que la función `mysql_num_rows`). Devuelve -1 si se produce un error.

Con **instrucciones UPDATE**, la **fila** no se cuenta como afectada si coincide con la **condición** pero no se han realizado cambios, a **menos** que el indicador `CLIENT_FOUND_ROWS` se configure al conectarse con `mysql_real_connect()`. Por ejemplo:

```
/* Actualice la tabla de clientes y devuelva el numero de
registros afectados */
mysql_query(&mysql, "UPDATE customer SET first_name='Jackie' WHERE
surname='Wood'");
affected-rows = mysql_affected_rows(&mysql);
```

mysql_change_user

```
my_bool mysql_change_user(MYSQL *mysql, char
*username, char *password, char *database)
```

Cambia el usuario **MySQL** actual (el que se ha conectado) por otro (especificando el nombre de usuario y la contraseña de este). Tambien puede cambiar **al mismo tiempo** la base de datos o abrir una nueva conexion; en caso contrario, se utilizaran la **conexión** y la base de datos actuales. Devuelve `true` si es satisfactorio o `false` si no lo es, en cuyo caso se mantienen el usuario y los detalles existentes.

Por ejemplo:

```
if (!mysql_change_user(&mysql, 'guru2b', 'g00r002b',
'firstdb')) {
    printf("Unable to change user and or database!");
}
```

mysql_character_set_name

```
char *mysql_character_set_name(MYSQL *mysql)
```

Devuelve el nombre del **conjunto** de caracteres predeterminado (normalmente ISO-8859-1 o Latin1).

Por ejemplo:

```
printf("The default character set is: %s \n",
mysql_character_set_name(&mysql));
```

mysql_close

```
void mysql_close(MYSQL *mysql)
```

Cierra la **conexión** y libera los recursos.

Por ejemplo:

```
mysql_close(&mysql);
```

mysql_connect

```
MYSQL *mysql_connect(MYSQL *mysql, const char *host, const char *user,
const char *passwd)
```

Para conectarse a **MySQL**. La función ha quedado obsoleta, por lo que en su lugar debe utilizar `mysql_real_connect()`.

mysql_create_db

```
int mysql_create_db(MYSQL *mysql, const char *db)
```

Para crear una base de datos. La función ha quedado obsoleta, por lo que en su lugar debe utilizar `mysql_query()`.

mysql_data_seek

```
void mysql_data_seek(MYSQL_RES *res, desplazamiento int sin
firma)
```

Desplaza a una nueva **posición** el puntero de fila interno (0 es la primera fila) asociado a los resultados devueltos desde `mysql_store_result()`. El desplazamiento es la fila a la que se mueve, empezando en 0.

Por ejemplo:

```
mysql_data_seek(results, mysql_num_rows(results)-1);
```

mysql_debug

```
mysql_debug(char *debug)
```

Para utilizarlo, el cliente **MySQL** debe haberse compilado con la depuración activada. Utiliza la biblioteca de depuración Fred Fish.

Por ejemplo:

```
/* Realiza el seguimiento de la actividad de la aplicación en
el archivo debug.out */
mysql_debug("d:t:0 debug.out");
```


mysql_drop_db

```
int mysql_drop_db(MYSQL *mysql, const char *db)
```

Permite eliminar una base de datos. La funcion ha quedado obsoleta, por lo que en su lugar deberia utilizar `mysql_query()`.

mysql_dump_debug_info

```
int mysql_dump_debug_info(MYSQL *mysql)
```

Escribe **información** de depuracion de la **conexión** en el registro. La **conexión** necesita el **privilegio SUPER** para **ello**. Devuelve 0 si es satisfactoria o, en **caso** contrario, un resultado que no sea cero.

Por ejemplo:

```
result = mysql_dump_debug_info(&mysql);
```

mysql_eof

```
my_bool mysql_eof(MYSQL_RES *result)
```

Comprueba si se ha **leído** la ultima **fila**. La funcion ha quedado obsoleta, por lo que en su lugar debe utilizar `mysql_err()` o `mysql_errno()`.

mysql_errno

```
unsigned int mysql_errno(MYSQL *mysql)
```

Devuelve el codigo de error de la funcion API mas reciente o 0 si no ha habido errores. Puede recuperar el texto del error por medio de la funcion `mysql_error()`.

Por ejemplo:

```
error = mysql_errno(&mysql);
```

mysql_error

```
char *mysql_error(MYSQL *mysql)
```

Devuelve el mensaje de error (en el idioma actual del servidor) de la funcion API mas reciente o una cadena vacia si no ha habido errores. Si no ha habido errores en la conexion, la funcion devuelve 0. Por ejemplo:

```
printf("Error: '%s'\n", mysql_error(&mysql));
```

mysql_escape_string

```
unsigned int mysql_escape_string(char *to, const char *from,  
unsigned int length)
```

Devuelve una cadena con todos los caracteres que pueden dividir la **consulta** a la que se ha aplicado la conversion de escape (con una barra invertida por delante

de los mismos). En su lugar debe utilizar `mysql_real_escape_string()` ya que respeta el conjunto de caracteres actual.

mysql_fetch_field

```
MYSQL_FIELD *mysql_fetch_field(MYSQL_RES *resultado)
```

Devuelve los datos del campo actual. Puede invocar varias veces esta función para devolver datos de los siguientes campos en el resultado. Devuelve un valor nulo cuando no hay mas campos por devolver. Por ejemplo:

```
while((campo = mysql_fetch_field(resultados))) {
    /* .. procesa los resultados mediante el acceso a campo-
    >nombre, campo->longitud etc. */
}
```

mysql_fetch_field_direct

```
MYSQL_FIELD * mysql_fetch_field_direct(MYSQL_RES * resultado,
numero_campo int sin firma)
```

Devuelve datos del campo especificado (que empieza en 0). Por ejemplo:

```
/* Devuelve el segundo campo */
field = mysql_fetch_field_direct(results, 1);
```

mysql_fetch_fields

```
MYSQL_FIELD *mysql_fetch_fields(MYSQL_RES * resultado)
```

Devuelve una matriz de datos de cada uno de los campos del resultado. Por ejemplo:

```
unsigned int num_fields;
unsigned int i;
MYSQL_FIELD *fields;

/* Devuelve el numero de campos del resultado */
num_fields = mysql_num_fields(result);

/* Devuelve una matriz de datos de campos */
fields = mysql_fetch_fields(result);

/* ... Accede a los datos de campos como campos [0].nombre,
campos [1].tabla, etc. */
```

mysql_fetch_lengths

```
unsigned long *mysql_fetch_lengths(MYSQL_RES *resultado)
```

Devuelve una matriz de las longitudes de los campos de la fila actual (denominada `mysql_fetch_row()`) del conjunto del resultados o nulo si se produce un error.

Es la única función que devuelve correctamente la longitud de campos binarios (por ejemplo, campos BLOB).

Por ejemplo:

```
unsigned long *lengths;

/* Devuelve la siguiente fila de datos */
row = mysql_fetch_row(results);

/* Devuelve la matriz de longitudes */
length_array = mysql_fetch_lengths(results);

/* ... Accede a las longitudes como longitud_matriz[0],
longitud_matriz[1] , etc. */
```

mysql_fetch_row

```
MYSQL_ROW mysql_fetch_row(MYSQL_RES *resultado)
```

Devuelve la siguiente fila del resultado o null si no hay más filas o se produce un error. Por ejemplo:

```
MYSQL_ROW row;

row = mysql_fetch_row(results);
/* Accede a los datos de filas como fila[0], fila[1] , etc.
```

mysql_field_count

```
unsigned int mysql_field_count(MYSQL *mysql)
```

Devuelve el número de campos de la última consulta ejecutada. Le permite determinar si un NULL devuelto desde `mysql_use_result()` o `mysql_store_result()` se debe a un error o a que no debería haberse devuelto un resultado (una consulta de tipo no SELECT). Para comprobar el número de campos en un conjunto de resultados satisfactorio, utilice `mysql_num_fields()`. Por ejemplo:

```
results = mysql_store_result(&mysql);
/* pruebe si no se encuentran resultados */
if (results == NULL) {
    /* si no hay resultados, pruebe si el número de campos es
    cero o no */
    if (mysql_field_count(&mysql) > 0) {
        /* la consulta es de tipo SELECT, por lo que el resultado
        almacenado nulo es un error */
    }
    else {
        /* no hay error, ya que la consulta es de tipo INSERT, por
        lo que no devuelve campos */
    }
}
```

mysql_field_seek

```
MYSQL_FIELD_OFFSET mysql_field_seek(MYSQL_RES *resultado,  
MYSQL_FIELD_OFFSET offset)
```

Desplaza el puntero de campo **interno** (que empieza en 0) al campo especificado. La siguiente llamada a `mysql_fetch_field()` sera el campo especificado. Devuelve el puntero de **posición del campo anterior**.

mysql_field_tell

```
MYSQL_FIELD_OFFSET mysql_field_tell(MYSQL_RES *resultado)
```

Devuelve la posición actual del puntero de campo.

Por ejemplo:

```
/* Registre la posición actual */  
current-pos = mysql_field_tell(results);
```

mysql_free_result

```
void mysql_free_result(MYSQL_RES *resultado)
```

Libera los recursos asignados a un conjunto de resultados.

mysql_get_client_info

```
char *mysql_get_client_info(void)
```

Devuelve una cadena que contiene la versión de biblioteca **MySQL** del cliente.

Por ejemplo:

```
/* Muestra - La versión de la biblioteca cliente es: 4.0.2 (por  
ejemplo) */  
printf("La versión de la biblioteca cliente es: %s\n",  
mysql_get_client_info());
```

mysql_get_host_info

```
char *mysql_get_host_info(MYSQL *mysql)
```

Devuelve una cadena que contiene **información sobre la conexión**.

Por ejemplo:

```
/* Muestra - Tipo de conexión: Equipo local a través de socket  
de UNIX (por ejemplo) */  
printf("Tipo de conexión: %s", mysql_get_host_info(&mysql));
```

mysql_get_proto_info

```
unsigned int mysql_get_proto_info(MYSQL *mysql)
```

Devuelve un entero que contiene la versión del protocolo (por ejemplo, 10) utilizado por la conexión.

Por ejemplo:

```
/* muestra - Version del protocolo: 10 (por ejemplo) */
printf("Version del protocolo: %d\n",
mysql_get_proto_info(&mysql));
```

mysql_get_server_info

```
char *mysql_get_server_info(MYSQL *mysql)
```

Devuelve una cadena que contiene la version del servidor MySQL (por ejemplo, 4.0.3).

Por ejemplo:

```
/* muestra - Version del servidor: 4.0.3-beta-log (por ejemplo) */
printf("Version del servidor: %s\n",
mysql_get_server_info(&mysql));
```

mysql_info

```
char *mysql_info(MYSQL *mysql)
```

Devuelve una cadena que contiene informacion detallada sobre la consulta mas reciente. Esta informacion incluye registros, filas que coinciden, cambios y advertencias.

Por ejemplo:

```
/* Información sobre la consulta en formato de cadena: Filas
encontradas: 19 Modificadas: 19 Advertencias: 0
(por ejemplo) */
printf("Información sobre la consulta: %s\n",
mysql_info(&mysql));
```

mysql_init

```
MYSQL *mysql_init(MYSQL *mysql)
```

Devuelve un identificador MySQL inicializado, listo para una funcion `mysql_real_connect()`.

Este **argumento** puede ser un **puntero nulo**, en cuyo caso se creara una estructura o un puntero a una estructura MYSQL existente. La funcion `mysql_close()` libera los recursos de la estructura si la ha creado. Si ha pasado una estructura existente, tendra que liberar personalmente los recursos una vez cerrada la conexion.

mysql_insert_id

```
my_ulonglong mysql_insert_id(MYSQL *mysql)
```

Devuelve un valor que contiene el ultimo valor AUTO INCREMENT añadido o 0 si la última consulta añadió un valor autoincrementado.

Por ejemplo:

```
last-auto-increment = mysql_insert_id(&mysql);
```

mysql_kill

```
int mysql_kill(MYSQL *mysql, id-proceso long sin firma)
```

Solicita que MySQL cancele el subprocesso especificado por `id_proceso`. Devuelve 0 si la operación es satisfactoria o un valor distinto a cero si **ha fracasado**.

Requiere el **privilegio** PROCESS.

Por ejemplo:

```
kill = mysql_kill(&mysql, 1293);
```

mysql_list_dbs

```
MYSQL-RES *mysql_list_dbs(MYSQL *mysql, const char *wild)
```

Devuelve un **conjunto** de resultados que contiene los nombres de las bases de datos del servidor que coinciden con la expresión de comodín habitual (equivale a la instrucción SQL `SHOW DATABASES LIKE 'comodin'`) o **nulo** si se produce un error. Devuelve todas las bases de datos si se pasa un **puntero nulo**.

Por ejemplo:

```
MYSQL-RES nombres_de_bases_de_datos;
```

```
/* devuelve una lista de todas las bases de datos que incluyan  
'db' en el nombre */
```

```
nombres_de_bases_de_datos = mysql_list_dbs(&mysql, "%db%");
```

```
/* ... No olvide liberar los recursos posteriormente  
mysql_free_result(nombres_de_bases_de_datos);
```

mysql_list_fields

```
MYSQL-RES *mysql_list_fields(MYSQL *mysql, const char *table,  
const char *wild)
```

Devuelve un **conjunto** de resultados que contiene los nombres de los campos de la tabla especificada que coinciden con la expresión de comodín habitual (equivale a la instrucción SQL `SHOW COLUMNS FROM nombre_de_tabla LIKE 'comodin'`) o **nulo** si se produce un error. Devuelve **todos** los campos si se pasa un **puntero nulo**.

Por ejemplo:

```
MYSQL-RES nombres_de_campos;
```

```
/* devuelve una lista de todos los campos que incluyan 'nombre'  
en su nombre */
```

```
nombres_de_campos = mysql_list_fields(&mysql, "customer", "%nombre%");
```

```
/* ... No olvide liberar los recursos posteriormente
mysql_free_result(nombres_de_campos);
```

mysql_list_processes

```
MYSQL_RES *mysql_list_processes(MYSQL *mysql)
```

Devuelve un **conjunto** de resultados que contiene una **descripción** de los subprocesos que se ejecutan actualmente en el servidor de bases de datos o NULL si se produce un error. Devuelve la misma información que se obtiene a través de la instrucción `SHOW PROCESSLIST` (es decir, el Id. del proceso, el nombre de usuario, el nombre del servidor, la **acción**, la hora, el estado y la información). De esta **forma, como** de costumbre, puede pasar el resultado a `mysql_fetch_row()` para acceder a los resultados. Por ejemplo:

```
MYSQL_RES *lista_de_subprocesos;
MYSQL_ROW row
Lista_de_subprocesos= mysql_list_processes(&mysql);

row = mysql_fetch_row(threadlist);
/* Accede a los datos de subprocesos como fila[0], fila[1] ,
etc.

/* ... No olvide liberar los recursos posteriormente
mysql_free_result(lista_de_subprocesos);
```

mysql_list_tables

```
MYSQL_RES *mysql_list_tables(MYSQL *mysql, const char *wild)
```

Devuelve un **conjunto** de resultados que contiene los nombres de las tablas de la base de datos actual que coinciden con la expresión de comodín habitual (equivale a **la instrucción SQL** `SHOW TABLES LIKE 'comodin'`) o NULL si se produce un error. Devuelve todos los campos si se pasa un **puntero nulo**.

Por ejemplo:

```
MYSQL_RES lista_de_tablas;

/* devuelve una lista de todas las tablas que incluyan
'customer' en su nombre */
lista_de_tablas = mysql_list_tables(&mysql, "%customer%");

/* ... No olvide liberar los recursos posteriormente
mysql_free_result(lista_de_tablas);
```

mysql_num_fields

```
unsigned int mysql_num_fields(MYSQL_RES *resultado)
```

Devuelve el número de campos del resultado de la consulta. Puede utilizar la función `mysql_field_count()` para **buscar errores** y esta para comprobar el **número de campos** de un **conjunto** de resultados satisfactorio.

Por ejemplo:

```
num_fields = mysql_num_fields(resultados);
```

mysql_num_rows

```
int mysql_num_rows(MYSQL_RES *resultado)
```

Devuelve el número de filas de un **conjunto** de resultados (solo los resultados hasta la fecha si se ha utilizado `mysql_user_result()` para obtener el conjunto de resultados). Por ejemplo:

```
num_rows = mysql_num_rows(resultados);
```

mysql_options

```
int mysql_options(MYSQL *mysql, enum opción opcion_mysql, void *valor)
```

Define opciones de **conexión** adicionales para la **conexión** que se va a realizar. Se puede invocar varias veces y se invoca después de `mysql_init()` y antes de `mysql_real_connect()`. Devuelve 0 si es satisfactoria o un valor que no sea cero si se pasa una opción no válida. Las opciones son las siguientes:

- `MYSQL_OPT_CONNECT_TIMEOUT`. `int *` sin firma que especifica el tiempo muerto de **conexión** en segundos.
- `MYSQL_OPT_COMPRESS`. Utiliza el protocolo comprimido cliente/servidor.
- `MYSQL_OPT_LOCAL_INFILE`. Un puntero opcional a `uint`. Se activa `LOAD DATA LOCAL` si el puntero apunta a un entero sin firma que no sea cero o si no se proporciona ninguno.
- `MYSQL_OPT_NAMED_PIPE`. En Windows NT; utiliza canalizaciones con nombre para conectarse al servidor.
- `MYSQL_INIT_COMMAND`. Un `char *` que especifica una **consulta** que se ejecutara cuando se establezca la **conexión** (incluyendo una reconexión automática).
- `MYSQL_READ_DEFAULT_FILE`. Un `char *` que hace que las opciones **provengan del** archivo de nombres en lugar del archivo de configuración habitual (`my.cnf` o `my.ini`).
- `MYSQL_READ_DEFAULT_GROUP`. Un `char *` que hace que las opciones se lean del grupo denominado dentro del archivo de configuración (`my.cnf` o `my.ini` o el que defina `MYSQL_READ_DEFAULT_FILE`).

Por ejemplo:

```
MYSQL mysql;  
mysql_init(&mysql);
```



```

/* utilice compresion y vacie las tablas al conectarse */
mysql_options(&mysql, MYSQL_OPT_COMPRESS, 0 );
mysql_options(&mysql, MYSQL_INIT_COMMAND, "FLUSH TABLES" );

/* ... continue y conectese */
if(!mysql_real_connect(&mysql, "localhost", "guru2b", "g00r002b",
"firstdb", 0,NULL,0)) {
    printf("The following connection error occurred %s\n",
        mysql_error(&mysql));
}

```

mysql_ping

```
int mysql_ping(MYSQL *mysql)
```

Devuelve 0 si el servidor MySQL esta en ejecucion o un valor que no sea cero en caso contrario.

Si falla, el programa intentara volver a conectarse.

mysql_query

```
int mysql_query(MYSQL *mysql, const char *consulta)
```

Ejecuta la consulta especificada. Devuelve 0 si es satisfactorio o un valor que no sea cero si se produce un error. Para ejecutar una consulta con datos binarios (que **podría** contener el caracter **nulo**), tendra que utilizar `mysql_real_query()`. **También** debería utilizar esta **función** para eliminar y **crear bases** de datos, que sustituye a las funciones `mysql_create_db()` y `mysql_drop_db()`, ya obsoletas. Tras **ello** puede **recuperar el resultado**, en caso de que sea necesario, mediante las funciones `mysql_store_result()` o `mysql_use_result()`. Por ejemplo:

```
query-result = mysql_query(&mysql, "CREATE DATABASE seconddb");
```

mysql_real_connect

```
MYSQL *mysql_real_connect(MYSQL *mysql, const char *host, const
char *user, const char *passwd, const char *db, uint port,
const char *unix_socket, uint client_flag)
```

Establece una **conexión al** servidor MySQL con los argumentos especificados, como se indica a **continuación**:

- **MYSQL *mysql**. Una estructura MYSQL existente, creada al invocar `mysql_init()`.
- **const char *host**. Nombre de servidor o **dirección IP** del servidor MySQL. Puede ser una cadena vacía, igual que al conectarse a MySQL desde un cliente en el mismo equipo.

- **const char *user.** El nombre de usuario.
- **const char *passwd.** La contraseña del usuario especificado.
- **const char *db.** Base de datos a la que se conecta (puede ser **nulo**).
- **uint port.** Especifica el puerto para la **conexión TCP/IP** (si se utiliza). 0 es el puerto predeterminado.
- **const char *unix_socket.** Especifica el nombre de **archivo** del socket de Unix, o **canalización** con nombre, para realizar la **conexión** localmente. Puede ser **nulo** para aceptar el predeterminado.
- **CLIENT_FOUND_ROWS.** Especifica que `mysql_affected_rows()` devolverá el número de filas que coinciden con la **condición** de la consulta, no el número de filas que se han modificado realmente.
- **CLIENT_IGNORE_SPACE.** Le permite **añadir** espacios por detrás de los nombres de funciones (con la consecuencia de que todas las funciones se convierten en **palabras** reservadas).
- **CLIENT_INTERACTIVE.** Especifica que MySQL cierra la **conexión** después de `interactive_timeout` segundos en lugar de `wait_timeout` segundos (dos **variables mysql**). Normalmente se utiliza **cuando** el cliente espera más tiempo por entradas de usuarios interactivos antes de **ejecutar** una consulta.
- **CLIENT_NO_SCHEMA.** Se utiliza principalmente con ODBC y no permite la **sintaxis** `nombre-de-base-de-datos.nombre-de-tabla.nombre-de-campo`.
- **CLIENT_COMPRESS.** Garantiza que la **conexión** utilice compresión.
- **CLIENT_ODBC.** Indica a MySQL que el cliente es un cliente ODBC, lo que provoca cambios de comportamiento.
- **CLIENT_SSL.** Garantiza que la **conexión** utiliza cifrado SSL siempre que se haya compilado en el servidor.

Por ejemplo:

```
MYSQL mysql;

mysql_init(&mysql);
mysql_options(&mysql, MYSQL_OPT_COMPRESS, 0 );
mysql_options(&mysql, MYSQL_INIT_COMMAND, "FLUSH TABLES" );
if(!mysql_real_connect(&mysql, "localhost", "guru2b", "g00r002b",
    "firstdb", 0, NULL,0)) {
    printf("The following connection error occurred %s\n",
        mysql_error(&mysql));
}
```

mysql_real_escape_string

```
unsigned long mysql_real_escape_string(MYSQL *mysql, char
*cadena_nueva, char *cadena_antigua, longitud_cadena_antigua
long sin firma)
```

Aplica conversión de escape a una cadena (cadena_antigua de longitud longitud_cadena) para que pueda utilizarla en una consulta MySQL y ubicar el resultado en nueva_cadena (que debe tener al menos 1 byte más que el doble de la cadena original, en caso de que sea necesario aplicar conversión de escape a todos los caracteres, así como tener en cuenta la cadena nula). Los caracteres son NUL (ASCII 0), \n, \r, \', \" y Control-Z. Devuelve la longitud de la nueva cadena.

Por ejemplo:

```
/* la consulta original es de 4 bytes (a,b,c y el caracter
nulo) */
char *consulta_antigua = "abc\000";

/*la nueva longitud debe ser al menos de 4*2+1 bytes */
char nueva_consulta [9];
int nueva_longitud;
/* devuelve la nueva longitud */
nueva_longitud_de_consulta = mysql_real_escape_string(&mysql,
new-query, old-query, 4);
```

mysql_real_query

```
int mysql_real_query(MYSQL *mysql, const char *consulta,
longitud long sin firma)
```

Ejecuta la consulta (que también puede utilizar datos binarios) y también especifica la longitud (excluyendo un carácter nulo). Tras ello puede recuperar el resultado, en caso de que sea necesario, con las funciones mysql_store_result() o mysql_use_result().

Por ejemplo:

```
query-result = mysql_real_query(&mysql, "CREATE DATABASE
seconddb");
```

mysql_reload

```
int mysql_reload(MYSQL *mysql)
```

Una función obsoleta que recarga las tablas de permisos, asumiendo que el usuario conectado tenga permiso Reload. En su lugar, utilice la función mysql_query().

mysql_row_seek

```
MYSQL_ROW_OFFSET mysql_row_seek(MYSQL_RES *resultado,
desplazamiento MYSQL_ROW_OFFSET)
```

Desplaza el puntero de fila **interno** hasta la fila especificada y devuelve el puntero de fila original. `MYSQL_ROW_OFFSET` debe ser la estructura devuelta por la función `mysql_row_tell()` o por otra función `mysql_row_seek()`, no sólo un número de fila (en cuyo caso tendría que utilizar la función `mysql_data_seek()`).

Por ejemplo:

```
ubicación_actual=
mysql_row_seek(resultado,desplazamiento_fila);
```

mysql_row_tell

```
MYSQL_ROW_OFFSET mysql_row_tell(MYSQL_RES *resultado)
```

Devuelve la posición actual del puntero de fila. Puede utilizarlo con `mysql_row_seek()` para desplazarse hasta la fila especificada. Utilícelo después de `mysql_store_result()`, no de `mysql_use_result()`.

Por ejemplo:

```
MYSQL_ROW_OFFSET position-actual= mysql_row_tell(resultados);

/* Mas adelante..., vuelva a esta posición */
posición_cambiada = mysql_row_seek(resultado,posición_actual);
```

mysql_select_db

```
int mysql_select_db(MYSQL *mysql, const char *db)
```

Cambia la base de datos actual a la base de datos especificada (siempre que el usuario tenga **permiso** para hacerlo). Devuelve 0 si es satisfactorio y un valor que no sea cero si se produce un error.

Por ejemplo:

```
mysql_select_db(&mysql, "seconddb");
```

mysql_shutdown

```
int mysql_shutdown(MYSQL *mysql)
```

Solicita que se cierre el servidor **MySQL**. El usuario debe tener el privilegio **SHUTDOWN** para que esto funcione. Devuelve 0 si es satisfactorio o un valor que no sea cero si se produce un error. Por ejemplo:

```
mysql_shutdown(&mysql);
```

mysql_stat

```
char *mysql_stat(MYSQL *mysql)
```

Devuelve una cadena que contiene el estado del servidor. Contiene el tiempo en ejecución, subprocesos, preguntas, consultas **lentas**, abiertas, tablas vacías, tabla abiertas y el promedio de consultas por segundo.

Por ejemplo:

```
/* muestra (por ejemplo):
Tiempo en ejecucion: 109 Subprocesos: 2 Preguntas: 199
Consultas lentas: 1 Abiertas: 4
Tablas vacias: 1 Tablas abiertas: 2 Consultas medias por
segundo: 1.826b */
printf("Server status: %s\n", mysql_stat(&mysql));
```

mysql_store_result

```
MYSQL_RES *mysql_store_result(MYSQL *mysql)
```

En todas las consultas que devuelven datos, tendra que invocar esta funcion o `mysql_use_result()`. Almacena los resultados de la consulta en la estructura `MYSQL_RES` o devuelve `NULL` en caso de que se produzca un error o si la consulta no devuelve datos (como por ejemplo después de `CREATE DATABASE` o `INSERT`). Deberia utilizar `mysql_field_count()` para contar el numero de campos que se esperan de la consulta. Si no es cero (cuando no se espera que la consulta devuelva datos), se habrá producido un error.

Tras ello, libere los recursos.

Por ejemplo:

```
MYSQL_RES resultados;
mysql_query(&mysql, "SELECT first_name, surname FROM
customers");
results = mysql_store_result(&mysql);
/* posteriormente ... */
mysql_free_result(resultados);
```

mysql_thread_id

```
unsigned long mysql_thread_id(MYSQL *mysql)
```

Devuelve el Id. del subproceso actual de la conexion, normalmente para eliminarla con `mysql_kill()`.

Por ejemplo:

```
thread_id = mysql_thread_id(&mysql);
```

mysql_use_result

```
MYSQL_RES *mysql_use_result(MYSQL *mysql)
```

En todas las consultas que devuelven datos, tendra que invocar esta funcion o `mysql_store_result()`. Esta funcion lee los datos fila a fila, no simultáneamente como hace `mysql_store_result()`. Por ello es más rápida, pero no permite que se ejecuten otras consultas hasta que se hayan devuelto todos los datos, lo que complica los bloqueos mas de lo habitual. Devuelve un valor nulo en caso de que se produzca un error o si la consulta no ha devuelto datos (como por ejemplo despues de `CREATE DATABASE` o `INSERT`). Deberia utilizar

`mysql_field_count()` para **contar** el número de campos que se espera de una **consulta**. Si no es cero (cuando no se espera que la **consulta** devuelva datos), se **habrá** producido un error.

Por ejemplo:

```
MYSQL_RES resultados;
mysql_query(&mysql, "SELECT first_name,surname FROM customer");
results = mysql_use_result(&mysql);

/* ahora puede utilizar mysql_fetch_row() para acceder a los
datos de fila en fila */

/* posteriormente ... */
mysql_free_result(resultados);
```

Breve ejemplo del API C

El listado G.1 muestra un breve ejemplo de uso del API C con los componentes básicos para devolver datos desde la base de datos. Tendrá que compilarlo y ejecutarlo. Las instrucciones de **compilación** difieren de un sistema a otro, **pero** tendrá un **aspecto** similar al siguiente:

```
gcc -o example example.c -I/usr/local/mysql/include/mysql
-L/usr/local/mysql/lib/mysql -lmysqlclient -lz
```

Listado G.1. EXAMPLE.C

```
#include <stdio.h>
#include <mysql.h>
/* los dos basicos incluyen */

/* la funcion principal */
int main(char **args) {
    MYSQL_RES *query_result;
    MYSQL_ROW row;
    /* db_handle es la conexión a la base de datos y la utilizarán
muchas de las funciones posteriores */
    MYSQL *db_handle, mysql;
    int query_error;

    /* inicialice y abra la conexión */
    mysql_init(&mysql);
    db_handle = mysql_real_connect(&mysql,"localhost", "guru2b",
    "g00r002b", "firstdb", 0, 0, 0);

    /* si la conexión falla, muestre el error y salga. */
    if (db_handle == NULL) {
        printf(mysql_error(&mysql));
        return 1;
    }
}
```

```

    query-error = mysql_query(db_handle, "SELECT
first_name,surname FROM customer");

    /* si el error de la consulta no es 0 (no hay error), muestra
el error y sale */
    if (query-error != 0) {
        printf(mysql_error(db_handle));
        return 1;
    }

    /* Devuelva un resultado de consulta */
    query-result = mysql_store_result(db_handle);

    /* Procesa una iteración por el resultado de la consulta y
muestra cada una de las filas del mismo */
    while (( row = mysql_fetch_row(query-result)) != NULL ) {
        printf("Name: %s %s\n", (row[0] ? row[0] : "NULL"),
        (row[1] ? row[1] : "NULL"));
    }
    /* libere los recursos asociados al resultado de la consulta
*/
    mysql_free_result(query_result);

    /* cierre la conexión */
    mysql_close(db_handle);
1

```

H

ODBC y .NET

MySQL puede **conectarse** con **lenguajes** o **entornos** que no tengan **sus propias** interfaces de programación de **aplicaciones (API) desarrolladas** o controladores para interactuar con MySQL a través de la conectividad abierta de bases de **datos (ODBC)**. ODBC es un API muy utilizado que **permite conectarse** e interactuar con bases de **datos relacionales**. Es independiente del lenguaje y de la base de **datos**. Con MySQL, se utiliza principalmente para **conectarse** a **herramientas** como Microsoft Access o Visual Basic. Para ello, tendrá que **instalar** el controlador MyODBC que encontrará, junto a **las instrucciones de instalación**, en el sitio Web de MySQL (www.mysql.com/downloads/api-myodbc.html). Para instalarlo en Windows basta con descargar el **archivo ejecutable** y ejecutarlo.

En este apéndice **explicaremos** como configurar un origen de datos en Unix y en Windows, como exportar **datos desde Microsoft Access a MySQL**. Servirá como referencia a **las funciones MyODBC** para los programadores con experiencia y proporciona **sencillas secuencias de comandos** de muestra que le permitirán utilizar ODBC para conectar, **añadir** y **seleccionar** registros con **VB.NET, C# .NET, DAO, ADO y RDO**. El apéndice no **constituye** una completa **descripción** de ODBC y de como utilizarlo. Si necesita información adicional, **consulte** la **documentación ODBC** del sitio de Microsoft (www.microsoft.com/data/ODBC/), la **documentación ODBC** incluida en **su entorno de desarrollo** o la información que encontrará en el **sitio de MySQL** (www.mysql.com/products/myodbc).

Origenes de datos

Un *origen de datos* puede ser una ruta a una biblioteca de archivos o, en este caso, a una base de datos MySQL. La información de conexión esta asociada al origen de datos, almacenada por ejemplo en el registro de Windows. Para conectarse al origen de datos, el administrador de controladores ODBC busca la información de conexión asociada al DSN y la utiliza para conectarse. Para realizar la conexión a través de ODBC, no siempre es necesario disponer de un DSN existente. Puede especificar directamente el controlador. Todos los ejemplos que apareceran posteriormente en este apéndice, menos el ejemplo VB DAO, se conectan sin un DSN predeterminado.

Configuración de un origen de datos en Windows

El primer paso para que una aplicación de Windows se conecte a MySQL a través de ODBC consiste en configurar un origen de datos:

1. Seleccione Inicio>Configuración>Panel de control.
2. En función de su versión de Windows, puede seleccionar ODBC 32 bits u ODBC, o, también, Herramientas administrativas y Orígenes de datos (ODBC).
3. Pulse Agregar
4. En la lista que aparece en pantalla, seleccione el controlador MySQL que haya instalado con MyODBC y, tras ello, pulse Finalizar.
5. Se abra la pantalla de configuración predeterminada. Complete los detalles necesarios. El DSN de Windows puede ser cualquier valor que elija y los detalles relativos al servidor, base de datos, nombre de usuario, contraseña y puerto son los que normalmente se necesitan para conectarse a MySQL. También puede especificar un comando SQL para que se ejecute al realizar la conexión al servidor.
6. Puede pulsar el botón Opciones para seleccionar distintas opciones específicas de su aplicación (que no siempre son totalmente compatibles con ODBC). Por ejemplo, actualmente con Microsoft Access es necesario marcar la opción **Devolver filas** que coincidan. Puede que tenga que experimentar para obtener un correcto funcionamiento y también debería consultar el sitio de MySQL, que incluye las últimas opciones para la mayoría de las aplicaciones.
7. Pulse Aceptar para añadir el origen de datos.
8. En función de su versión de ODBC, podrá probar su conexión si pulsa el botón **Probar** origen de datos. Si la prueba falla, revise los detalles de su conexión.

Configuración de un origen de datos en Unix

En Unix, puede modificar directamente el archivo `ODBC.INI` para configurar sus orígenes de datos.

Por ejemplo:

```
[ODBC Data Sources]
myodbc          = MySQL ODBC 3.51 Driver DSN

[myodbc]
Driver          = /usr/local/lib/libmyodbc3.so
Description    = MySQL ODBC 3.51 Driver DSN
SERVER         = localhost
PORT           =
USER           = root
Password       = g00r002b
Database       = firstdb
OPTION         = 3
SOCKET         =
```

En Unix, para definir las distintas opciones para aplicaciones no compatibles con ODBC, debe configurar el valor `OPTION` en función de los que aparecen en la tabla H.1.

Configuración de opciones de conexión

Si no dispone de una interfaz gráfica, puede utilizar los valores de opción enumerados en la tabla H.1 para realizar la conexión. Para combinar opciones, basta con añadir los valores de forma conjunta (por lo que 3 es una combinación de 1 y 2).

Tabla H.1. Opciones de conexión

Bit	Descripción
1	No puede procesar la recepción del ancho real de una columna.
2	No puede procesar la recepción del número real de filas afectadas (en su lugar, se devuelve el número de filas encontradas).
4	Realiza un registro de depuración en <code>c:\myodbc.log</code> o <code>/tmp/myodbc.log</code> .
8	Establece un límite de paquetes ilimitados para resultados y parámetros.
16	No realiza preguntas.

Bit	Descripción
32	Cambia la compatibilidad con el cursor dinámico (lo activa o lo desactiva).
64	Ignora el nombre de la base de datos en una estructura como <code>nombre_de_base_de_datos.nombre_de_tabla.nombre_de_archivo</code> .
128	Opción experimental que obliga a utilizar cursores de administrador ODBC.
256	Opción experimental que inhabilita el uso de búsqueda extendida.
512	Los campos CHAR se rellenan con la longitud total del campo.
1024	la función <code>SQLDescribeCol()</code> devuelve nombres de columna completos.
2048	Utiliza un protocolo comprimido.
4096	Hace que el servidor ignore espacios entre el nombre de la función y el corchete de apertura y, como resultado, convierte a todos los nombres de función en palabras clave.
8192	Utiliza canalizaciones con nombre para conectarse a un servidor que se ejecute en NT/2000/XP.
16384	Los campos <code>LONGLONG</code> se convierten en campos INT.
32768	Opción experimental que devuelve <code>user</code> como <code>Table_qualifier</code> y <code>Table_owner</code> de la función <code>SQLTables()</code> .
65536	Lee el archivo de configuración MySQL para los parámetros de grupo <code>client y odbc</code> .
131072	Añade comprobaciones de seguridad adicionales.
262144	Inhabilita las transacciones.
524288	En modo depuración, activa el registro de consultas en el archivo <code>c:\myodbc.sql o /tmp/myodbc.sql</code> .

Exportación de datos desde Microsoft Access a MySQL

Muchos usuarios noveles de bases de datos empiezan con Microsoft Access y uno de los usos más habituales de ODBC consiste en pasar a MySQL y exportar

datos desde Access. Para ello, ejecute los pasos descritos a continuación (los pasos 1 y 2 solo son necesarios si es la primera vez que realiza esta operación):

1. Instale el controlador MyODBC.
2. **Configure** un origen de datos que apunte al servidor MySQL al que quiera **exportar los datos**, como describimos en el apartado anterior.
3. Inicie Microsoft Access.
4. Abra la ventana de bases de datos de Microsoft Access y seleccione la tabla que quiera exportar.
5. Seleccione **Archivo>Exportar** y escoja la opción **Bases de datos ODBC()** de la lista desplegable **Guardar como**.
6. Seleccione un nuevo nombre si quiere modificar el nombre de la tabla y pulse **Aceptar**.
7. Seleccione el origen de datos que haya definido en el paso 2 y pulse **Aceptar**.
8. **Si los detalles de conexión del origen de datos no son correctos**, tendrá que cambiarlos. **operación que puede realizar en este momento**. Recuerde que es necesario conceder permisos MySQL para poder acceder (consulte un capítulo anterior).

Uso de ODBC

En los siguientes ejemplos veremos como insertar un registro en una base de datos MySQL y como seleccionar registros de la misma por medio de ODBC en diferentes entornos de programación. Los primeros ejemplos muestran la conexión **realizada** directamente y el ejemplo DAO como se establece una conexión a través de un origen de datos.

Para que estos ejemplos funcionen, debe instalar MyODBC, un DSN preinstalado (solamente para el ejemplo DAO) y el entorno correcto (.NET: Visual Basic, etc.).

ADVERTENCIA: Debe mantener el mismo formato o los ejemplos no funcionarán.

Ejemplo de VB.NET

En el listado H.1 utilizamos VB.NET y ODBC para conectarnos a un servidor MySQL, insertar un registro y seleccionar e imprimir los resultados. Para compilar

el código, tendrá que especificar parámetros apropiados para su entorno .NET. Veamos un ejemplo (la pausa es para que las opciones de compilación sean visibles:

```
set path=%path%;C:\WINNT\Microsoft.NET\Framework\v1.0.3705
C:\WINNT\Microsoft.NET\Framework\v1.0.3705\csc /t:exe
/out:odbc_cnet.exe odbc_cnet.cs /r:"C:\Program
Files\Microsoft.NET\Odbc.Net\Microsoft.Data.Odbc.dll"
pause
```

Listado H.1. DBNET.VB

```
Imports Microsoft.Data.Odbc
Imports System
Module mysql_vbnet
    Sub Main()
        Try

            'Defina los argumentos para conectarse a una base de
            'datos MySQL firstdb con MyODBC 3.51
            Dim MySQLConnectionArgs As String = " DRIVER={MySQL
            ODBC 3.51 Driver}; SERVER=www.testhost.co.za;DATABASE=
            firstdb;UID=guru2b;PASSWORD=g00r002b;OPTION=0"

            'Abra la conexión ODBC y el comando ODBC
            Dim MySQLConnection As New
            OdbcConnection(MySQLConnectionArgs)
            MySQLConnection.Open()
            Dim MySQLCommand As New OdbcCommand()
            MySQLCommand.Connection = MySQLConnection

            'Añada un registro a la tabla de clientes
            MySQLCommand.CommandText = "INSERT INTO customer
            (first-name, surname) VALUES ('Frank', 'Weiss')"
            MySQLCommand.ExecuteNonQuery()

            'seleccione un registro de la tabla de clientes,
            'devuelva los resultados,
            'procese una iteración por los resultados para
            'mostrarlos
            MySQLCommand.CommandText = "SELECT
            id,first_name,surname FROM          customer"
            Dim MySQLDataReader As OdbcDataReader
            MySQLDataReader = MySQLCommand.ExecuteReader
            While MySQLDataReader.Read
                Console.WriteLine (CStr(MySQLDataReader("id")) & ":" &
                CStr(MySQLDataReader("first_name")) & " " &
                CStr(MySQLDataReader("surname")))
            End While

            'Si hay una excepción ODBC, capturela
            Catch MySQLOdbcException As OdbcException
```

```

    Console.WriteLine (MySQLOdbcException.ToString)

    'Si hay una excepción de programa, capturela
    Catch AnyException As Exception
        Console.WriteLine (AnyException.ToString)
    End Try
End Sub
End Module

```

Ejemplo de C#.NET

En el listado H.2 se utiliza C#.NET y ODBC para establecer la **conexión** a un servidor MySQL, insertar un registro y seleccionar e imprimir **los** resultados. Para compilar el código, tendrá que especificar **parámetros** apropiados para su entorno .NET.

Veamos un ejemplo (la pausa es para que las opciones de compilación sean visible~):

```

C:\WINNT\Microsoft.NET\Framework\v1.0.3705\csc /t:exe
/out:odbc_cnet.exe odbc_cnet.cs /r:"C:\Program
Files\Microsoft.NET\Odbc.Net\Microsoft.Data.Odbc.dll"
pause

```

Listado H.2. DBNET.CS

```

using Console = System.Console;
using Microsoft.Data.Odbc;

namespace MyODBC {
    class MySQLCSharp {
        static void Main(string[] args) {
            try {
                // Defina los argumentos para conectarse a una base
                // de datos MySQL firstdb con MyODBC 3.51

                string MySQLConnectionArgs = "DRIVER={MySQL ODBC
                3.51 Driver};SERVER=www.testhost.co.za;DATABASE=
                firstdb;UID=guru2b; PASSWORD=g00r002b;OPTION=0";

                // Abra la conexión ODBC y el comando ODBC
                OdbcConnection MySQLConnection = new
                OdbcConnection(MySQLConnectionArgs);
                MySQLConnection.Open();

                // Añada un registro a la tabla de clientes
                OdbcCommand MySQLCommand = new OdbcCommand("INSERT INTO
                customer (first-name, surname) VALUES('Frank', 'Weiss')",
                MySQLConnection);
                MySQLCommand.ExecuteNonQuery();
            }
        }
    }
}

```



```

'Abra una conexión por medio de ADODB y defina la cadena de
'conexión
'para conectarse a una base de datos MySQL firstdb con MyODBC
'3.51
Set MySQLConnection = New ADODB.Connection
MySQLConnection.ConnectionString = "DRIVER={MySQL ODBC 3.51 Driver};
SERVER=www.testhost.co.za;DATABASE=customer;UID=guru2b;
PWD=g00r002b;OPTION=0" MySQLConnection.Open

Set Results = New ADODB.Recordset
Results.CursorLocation = adUseServer

'Hay dos formas para insertar registros - la primera es la
'inserción directa
SQLQuery = "INSERT INTO customer (first-name, surname) VALUES
('Werner', 'Christerson')"
MySQLConnection.Execute SQLQuery

'La segunda consiste en añadir un conjunto de resultados por
'medio del
'método AddNew. En primer lugar, devuelva un conjunto de
'resultados
Results.Open "SELECT * FROM customer", MySQLConnection,
adOpenDynamic, adLockOptimistic
Results.AddNew
Results!first_name = "Lance"
Results!surname = "Plaaaitjies"
Results.Update
Results.Close

'seleccione un registro de la tabla de clientes, devuelva los
'resultados,
'procese una iteración por los resultados para mostrarlos
Results.Open "SELECT id, first-name, surname FROM customer",
MySQLConnection
While Not Results.EOF
    Debug.Print Results!id & ";" & Results!first_name & " " &
Results!surname
    Results.MoveNext
Wend
Results.Close

MySQLConnection.Close
End Sub

```

Ejemplo de VB RDO

En el listado H.4 se utiliza VB y RDO para establecer la conexión a un servidor MySQL a través de ODBC, insertar un registro y seleccionar e imprimir los resultados.

Visual Basic **admite** RDO pero puede que en su **lugar** le interese utilizar el nuevo ADO.

Para acceder a los objetos RDO 2.0 en Visual Basic, debe establecer una referencia a la biblioteca de tipos RDO incluida en MSRDO20.DLL. Aparece en el cuadro de **dialogo Referencias** (al que se accede desde el menu **Proyecto**) como Objetos de datos remotos 2.0 de Microsoft. Es necesario que el código aparezca dentro de un formulario para que funcione el **método** Debug. Print. Por otra **parte**, puede cambiarlo a MsgBox para que el código sea ejecutable.

Listado H.4. DARDO.VB

```
Private Sub MySQLRDO()  
    Dim Results As rdoResultset  
    Dim MySQLConnection As New rdoConnection  
    Dim SQLQuery As String  
  
    'Abra una conexión por medio de RDO y defina la cadena de  
'conexión  
    'para conectarse a una base de datos MySQL firstdb con MyODBC  
'3.51  
  
    MySQLConnection.Connect = "DRIVER={MySQL ODBC 3.51 Driver};  
SERVER=www.testhost.co.za;DATABASE=firstdb;UID=guru2b;  
PWD=g00r002b;OPTION=0"  
  
    MySQLConnection.CursorDriver = rdUseOdbc  
    MySQLConnection.EstablishConnection rdDriverNoPrompt  
  
    'Hay dos formas de inserción - la primera es la inserción  
'directa  
    SQLQuery = "INSERT INTO customer (first-name, surname) VALUES  
('Lance', 'Plaaitjies')"  
    MySQLConnection.Execute SQLQuery, rdExecDirect  
  
    'La segunda consiste en añadir a un conjunto de resultados  
'por medio del  
'método AddNew. Primero devuelva un conjunto de resultados  
    SQLQuery = "SELECT * FROM customer"  
    Set Results = MySQLConnection.OpenResultset(SQLQuery, rdOpenStatic,  
rdConcurRowVer, rdExecDirect)  
    Results.AddNew  
    Results!first_name = "Werner"  
    Results!surname = "Christerson"  
    Results.Update  
    Results.Close  
  
    'seleccione un registro de la tabla de clientes, devuelva los  
'resultados,  
    'procese una iteración por los resultados para mostrarlos
```

```

SQLQuery = "select * from customer"
Set Results = MySQLConnection.OpenResultset(SQLQuery, rdOpenStatic,
rdConcurRowVer, rdExecDirect)
While Not Results.EOF
    Debug-Print Results!id & ":" & Results!first_name & " " &
Results!surname
    Results.MoveNext
Wend
'Libere el conjunto de resultados y la conexión
Results.Close
MySQLConnection.Close
End Sub

```

Ejemplo de VB DAO

En el listado **H.5** se utiliza VB y DAO para establecer la **conexión** a un servidor MySQL a través de ODBC, insertar un registro (**tanto** directamente como mediante la inclusión de un **conjunto** de resultados) y seleccionar e imprimir los resultados.

Visual Basic **admite** DAO pero puede que en su lugar le interese utilizar el nuevo ADO.

Para acceder a los objetos DAO en Visual Basic, debe establecer una **referencia** a la biblioteca de tipos DAO incluida en DAO360.DLL. Aparece en el cuadro de **diálogo** Referencias (**al** que se accede desde el menú Proyecto) como **Biblioteca** de objetos DAO 3.6 de **Microsoft**.

Es necesario que el código aparezca dentro de un formulario para que funcione el **método** Debug.Print. Por otra **parte**, puede cambiarlo a MsgBox para que el código sea ejecutable. En este ejemplo es necesario configurar un DSN para que funcione.

Listado H.5. DBDAO.VB

```

Private Sub MySQLDAO()
    Dim Works As Workspace
    Dim MySQLConnection As Connection
    Dim Results As Recordset
    Dim SQLQuery As String

    'Abra un espacio de trabajo por medio de DAO y defina la
'cadena de conexión
    'para conectar a una base de datos firstdb MySQL DSN con
'MyODBC 3.51
    Set Works = DBEngine.CreateWorkspace("MySQLWorkspace", "guru2b",
    "g00r002b", dbUseODBC)
    Set MySQLConnection = Works.OpenConnection("MySQLConn",
rdDriverCompleteRequired, False, "ODBC;DSN=MyDAO")
    'Hay dos formas de inserción - la primera es la inserción
'directa

```

```

SQLQuery = "INSERT INTO customer (first_name, surname) VALUES
('Lance', 'Plaaaitjies')" MySQLConnection.Execute SQLQuery

'La segunda consiste en añadir a un conjunto de resultados
'por medio del
'método AddNew. Primero devuelva un conjunto de resultados
Set Results = MySQLConnection.OpenRecordset("customer")
Results.AddNew
Results!first_name = "Werner"
Results!surname = "Christerson"
Results.Update
Results.Close

'Lea la tabla de clientes
Set Results = MySQLConnection.OpenRecordset("customer",
dbOpenDynamic)
While Not Results.EOF
    Debug.Print Results!id & ":" & Results!first_name & " " &
Results!surname
    Results.MoveNext
Wend
Results.Close

MySQLConnection.Close
Works.Close
End Sub

```

Funciones MyODBC

Los siguientes apartados constituyen una guía de referencia **sobre funciones** que los programadores con experiencia pueden utilizar. Las descripciones de este apéndice se aplican a MyODBC 3.5x.

SQLAllocConnect

Asigna memoria a un identificador de conexión. Esta función ha quedado obsoleta y se ha sustituido por `SQLAllocHandle()`, que se invoca con el argumento `SQL_HANDLE_DBC`.

SQLAllocEnv

Obtiene un identificador de **entorno** del controlador. Esta función ha quedado obsoleta y se ha sustituido por `SQLAllocHandle()` que se invoca con el argumento `SQL_HANDLE_ENV`.

SQLAllocHandle

```
SQLAllocHandle (tipo_de_identificador,  
identificador_de_entrada, puntero_de_identificador_de_salida);
```

Asigna un identificador (de conexión, descriptor, **entorno** o instrucción).

`tipo_de_identificador` puede ser `SQL_HANDLE_ENV` (identificador de **entorno**), `SQL_HANDLE_DBC` (identificador de **conexión**) o `SQL_HANDLE_STMT` (identificador de **instrucción**).

El `identificador_de_entrada` describe el **contexto** de asignación del nuevo identificador.

Sera `SQL_NULL_HANDLE` si el `tipo_de_identificador` es `SQL_HANDLE_ENV`, un identificador de **entorno** si `tipo_de_identificador` es `SQL_HANDLE_DBC` y un identificador de **conexión** si es `SQL_HANDLE_STMT`.

El `puntero_de_identificador_de_salida` es un puntero a un búfer desde el que se devuelve el identificador.

SQLAllocStmt

Asigna **memoria** a un identificador de instrucción. Esta función ha quedado obsoleta y se ha sustituido por `SQLAllocHandle()` que se invoca con el argumento `SQL_HANDLE_STMT`.

SQLBindParameter

```
SQLBindParameter(identificador_de_instrucción,  
numero_de_parametro, tipo_de_parametro, tipo_de_valor;  
tipo_sql; tamaño_de_columna, digitos_decimales,  
puntero_de_valor_de_parametro, longitud_de_búfer,  
puntero_de_longitud_de_cadena);
```

Vincula un **marcador** de parámetros a una **instrucción SQL**. `numero_de_parametro` empieza desde 1.

Por ejemplo:

```
SQLINTEGER id_ptr;  
SQLINTEGER idl_ptr;  
  
// Prepare SQL  
SQLPrepare(sth, "INSERT INTO customer(id) VALUES (?)", SQL_NTS);  
  
// Vincule id al parametro de la columna de Id.  
SQLBindParameter(sth, 1, SQL_PARAM_INPUT, SQL_C_ULONG,  
SQL_LONG, 0, 0, &id_ptr, 0, &idl_ptr);  
  
// ...  
SQLExecute(sth);
```

SQLBulkOperations

```
SQLBulkOperations(identificador_de_instrucción, operación);
```

Realiza operaciones de grandes volúmenes de datos.

SQLCancel

```
SQLCancel(identificador_de_instrucción)
```

Cancela operaciones en el identificador de instrucción especificado.

SQLCloseCursor

```
SQLCloseCursor(identificador_de_instrucción);
```

Cierra todos los **cursores** abiertos para el identificador de instrucción especificado.

SQLColAttribute

```
SQLColAttribute (identificador_de_instrucción,  
número_de_registro, identificador_de_campo,  
puntero_de_atributo_de_caracteres, longitud_de_búfer,  
puntero_de_longitud_de_cadena, puntero_de_atributo_numérico);
```

Describe atributos de un campo del **conjunto** de resultados.

El **argumento** `numero_de_registro` es el número del registro, que empieza en 1. El **argumento** `identificador_de_campo` especifica el campo que se va a devolver.

El **argumento** `puntero_de_atributo_de_caracteres` apunta a un búfer desde el que se **devolverá el valor** (si es una **cadena**; en caso contrario no se utiliza).

El **argumento** `longitud_de_bufer` puede contener uno de los siguientes valores:

- La longitud del `puntero_de_atributo_de_caracteres` (o `SQL_NTS`) si apunta a una **cadena**.
- El resultado de `SQL_LEN_BINARY_ATTR(longitud)` si el `puntero_de_atributo_de_caracteres` apunta a un búfer binario.
- `SQL_IS_INTEGER`, `SQL_IS_UNINTEGER`, `SQL_SMALLINT` o `SQLUSMALLINT` si `puntero_de_atributo_de_caracteres` apunta a un tipo de datos **específico de longitud fija**.
- `SQL_IS_POINTER` si `puntero_de_atributo_de_caracteres` apunta a otro puntero.

El argumento `puntero_de_longitud_de_cadena` apunta a un bufer desde el que se devolvera el numero total de bytes de `puntero_de_atributo_de_caracteres` (excluyendo un byte nulo). Para datos de caracteres, si `longitud_de_búfer` es menor que el número de bytes que se van a devolver, los datos se recortan. En otros casos, se asume que es de 32 bits. El argumento `puntero_de_atributo_numérico` apunta a un bufer de enteros desde el que se devuelve un valor numérico. No es utiliza si el valor devuelto no es numérico.

SQLColAttributes

Describe atributos de un campo del resultado. Esta función ha quedado obsoleta y se ha sustituido por `SQLColAttribute()`.

SQLColumnPrivileges

```
SQLColumnPrivileges(identificador_de_instrucción,  
nombre_de_catálogo, longitud_de_nombre_de_catálogo,  
nombre_de_esquema, longitud_de_nombre_de_esquema,  
nombre_de_tabla, longitud_de_nombre_de_tabla,  
nombre_de_columna, longitud_de_nombre_de_cadena);
```

Devuelve una lista de campos y privilegios.

SQLColumns

```
SQLColumns(identificador_de_instrucción, nombre_de_catálogo,  
longitud_de_nombre_de_catálogo, nombre_de_esquema,  
longitud_de_nombre_de_esquema, nombre_de_tabla,  
longitud_de_nombre_de_tabla, nombre_de_columna,  
longitud_de_nombre_de_columna);
```

Devuelve una lista de nombres de columnas

SQLConnect

```
SQLConnect(identificador_de_conexión, nombre_origen_de_datos,  
longitud_nombre_de_origen_de_datos, nombre_de_usuario,  
longitud_nombre_de_usuario, contraseña,  
longitud_de_contraseña);
```

Realiza la conexión al origen de datos con el nombre de usuario y contraseña especificados.

SQLDataSources

Implementada por el Administrador de controladores, esta funcion devuelve una lista de origenes de datos disponibles.

SQLDescribeCol

```
SQLDescribeCol(identificador_de_instrucción, número_de_columna,  
nombre_de_columna, longitud_de_búfer,  
puntero_de_longitud_de_nombre, puntero_de_tipo_de_datos,  
puntero_de_número_de_columna, puntero_de_dígitos_decimales,  
puntero_nulo);
```

Describe una columna del **conjunto** de resultados.

El argumento `número_de_columna` es el número de la columna del conjunto de resultados, **empezando desde 1**.

El argumento `nombre_de_columna` apunta a un bufer desde el que se devuelve el nombre de la **columna (leído desde SQL_DESC_NAME)**. Devuelve una cadena vacía si el nombre no está disponible.

El argumento `longitud_de_búfer` es la longitud en caracteres del bufer `nombre_de_columna`.

El argumento `puntero_de_longitud_de_búfer` apunta a un bufer desde el que se devuelve el **número de bytes disponibles** en `nombre_de_columna` (excluyendo bytes nulos). Si la longitud que se devuelve es mayor que `longitud_de_búfer`, se recorta el nombre de la **columna**.

El argumento `puntero_de_tipo_de_datos` apunta a un bufer desde el que se devolverá el tipo de **datos SQL, obtenido de SQL_DESC_CONCISE_TYPE**. Devuelve `SQL_UNKNOWN_TYPE` si el tipo no está disponible.

El argumento `puntero_de_tamaño_de_columna` apunta a un bufer desde el que se **devolverá el tamaño de la columna o 0** si no está disponible.

El argumento `puntero_de_dígitos_decimales` apunta a un bufer desde el que se **devolverá el número de decimales de la columna o 0** si no está disponible.

El argumento `puntero_nulo` apunta a un bufer desde el que se devuelve la **nulidad** (`SQL_NO_NULLS`, `-SQL_NULLABLE` o `SQL_NULLABLE_UNKNOWN`).

SQLDescribeParam

```
SQLDescribeParam(identificador_de_instrucción,  
número_de_parámetro, puntero_de_tipo_de_datos,  
puntero_de_tamaño_de_parámetro, puntero_de_dígitos_decimales,  
puntero_nulo);
```

Devuelve una **descripción** del parámetro.

El argumento `número_de_parámetro` **especifica** el parámetro (**empezando desde 0**).

El `puntero_de_tipo_de_datos` apunta a un bufer desde el que se **devolverá el tipo de datos SQL**.

El argumento `puntero_de_tamaño_de_parámetro` apunta a un bufer desde el que se **devolverá el tamaño de la columna de parámetros**.

El argumento `puntero_de_dígitos_decimales` apunta a un bufer desde el que se **devolverá el número de decimales de la columna o 0** si no está disponible.

El argumento `puntero_nulo` apunta a un bufer desde el que se devolvera la **nulidad** (`SQL_NO_NULLS`, `SQL_NULLABLE` o `SQL_NULLABLE_UNKOWN`).

SQLDisconnect

```
SQLDisconnect (identificador_de_conexión);
```

Cierra la **conexión** especificada por el identificador de conexion.

SQLDriverConnect

```
SQLDriverConnect (identificador_de_conexión,  
identificador_de_ventana, entrada_de_conexión,  
longitud_de_entrada_de_conexión, salida_de_conexión,  
longitud_de_salida_de_conexión, longitud_de_búfer,  
indicador_de_línea_de_comandos);
```

Establece la **conexión** con un servidor.

En **su** lugar puede utilizar `SQLConnect` para conectarse sin un DSN, **información** de **conexión** especifica del controlador o solicite **al** usuario la **información** de conexion.

El **argumento** `identificador_de_ventana` puede ser el identificador de la ventana principal o un **puntero nulo** si no hay cuadros de **diálogo** o no se utiliza el identificador de ventana.

El **argumento** `entrada_de_conexión` puede ser una **conexión** completa, una cadena de **conexión** parcial o una cadena vacia.

El **argumento** `longitud_de_entrada_de_conexión` es la longitud en bytes de la cadena `entrada_de_conexión`.

El **argumento** `salida_de_conexión` apunta a un bufer desde el que se devolvera la cadena de conexion.

El **argumento** `longitud_de_salida_de_conexión` es la longitud del bufer `salida_de_conexión`.

El **argumento** `longitud_de_bufer` apunta a un bufer desde el que se devolvera el **número** de **caracteres disponibles**.

Si el numero de caracteres es mayor de `longitud_de_bufer`, se **recorta** `salida_de_conexión`.

El **argumento** `indicador_de_línea_de_comandos` especifica si el controlador debe solicitar **más información** para realizar la conexion. Puede ser `SQL_DRIVER_PROMPT`, `SQL_DRIVER_COMPLETE`, `SQL_DRIVER_COMPLETE_REQUIRED` o `SQL_DRIVER_NOPROMPT`.

SQLDrivers

Implementada por el Administrador de controladores, esta funcion devuelve detalles de los controladores instalados.

SQLEndTran

```
SQLEndTran(tipo_de_identificador, identificador,  
tipo_de_finalización);
```

Finaliza una **transacción** abierta e invoca su **confirmación** o la invierte.

El argumento `tipo_de_identificador` contiene `SQL_HANDLE_ENV` o `SQL_HANDLE_DBC` en función del tipo de identificador (de **entorno** o de **conexión**). El argumento `identificador` especifica el identificador.

`tipo_de_finalización` determina si las transacciones se finalizan con una **confirmación** o se invierten. Puede ser `SQL_COMMIT` o `SQL_ROLLBACK`.

SQLError

Esta función devuelve **información sobre errores**. Se ha quedado obsoleta. Puede utilizar `SQLGetDiagRec` o `SQLGetDiagField` para **reemplazarla**.

SQLExecDirect

```
SQLExecDirect(identificador_de_instrucción, sql, longitud_sql);
```

Ejecuta una instrucción SQL. Es más rápida que `SQLExecute` si la **instrucción sólo se va a ejecutar una vez** y no es necesario prepararla.

SQLExecute

```
SQLExecute(identificador_de_instrucción);
```

Ejecuta una instrucción preparada previamente (con `SQLPrepare`). Utilice `SQLExecDirect` si la **instrucción sólo se va a ejecutar una vez** y no es necesario prepararla.

SQLExtendedFetch

Esta función devuelve resultados **sobre los que se puede desplazar**. Se ha quedado obsoleta; en su lugar puede utilizar `SQLFetchScroll`.

SQLFetch

```
SQLFetch(identificador_de_instrucción);
```

Devuelve la siguiente fila de datos.

SQLFetchScroll

```
SQLFetchScroll(identificador_de_instrucción,  
tipo_de_recuperación, desplazamiento);
```

Devuelve datos de la fila especificada.

El argumento `tipo_de_recuperación` puede ser `SQL_FETCH_NEXT` (la siguiente fila, equivalente a la función `SQLFetch`), `SQL_FETCH_PRIOR` (la fila anterior), `SQL_FETCH_FIRST` (la primera fila), `SQL_FETCH_LAST` (la última fila), `SQL_FETCH_ABSOLUTE` (una fila desplazada desde la primera fila), `SQL_FETCH_RELATIVE` o `SQL_FETCH_BOOKMARK` (una fila desplazada desde la fila actual).

El argumento `desplazamiento` especifica la fila que se va a obtener, bien desde la primera o desde la fila actual, en función del argumento `tipo_de_recuperación`.

SQLFreeConnect

Libera el identificador de conexión. Esta función ha quedado obsoleta; en su lugar debe utilizar `SQLFreeHandle`.

SQLFreeEnv

Libera el identificador de entorno. Esta función ha quedado obsoleta; en su lugar debe utilizar `SQLFreeHandle`.

SQLFreeHandle

```
SQLFreeHandle(tipo_de_identificador, identificador);
```

Libera un identificador (ya sea de conexión, descriptor, de entorno o de instrucción).

El argumento `tipo_de_identificador` puede ser `SQL_HANDLE_ENV` (identificador de entorno), `SQL_HANDLE_DBC` (identificador de conexión), `SQL_HANDLE_STMT` (identificador de instrucción) o `SQL_HANDLE_DESC` (identificador descriptor). El argumento `identificador` es el identificador especificado que se va a liberar.

SQLFreeStmt

```
SQLFreeStmt(identificador_de_instrucción, opcion);
```

Detiene el procesamiento de la instrucción.

El argumento `opcion` puede ser `SQL_CLOSE` (cierra el cursor, igual que `SQLCloseCursor`, con la posibilidad de volver a abrirlo), `SQL_DROP` (libera el identificador de instrucción y cierra el cursor, aunque este uso se ha quedado obsoleto y en su lugar debe utilizar `SQLFreeHandle`), `SQL_UNBIND` (libera todos los búfer de columna vinculados a `SQLBindCol`) y `SQL_RESET_PARAMS` (libera todos los búfer de parámetros definidos por `SQLBindParameter`).

SQLForeignKeys

```
SQLForeignKeys (identificador_de_instrucción,  
nombre_de_catálogo_de_clave_principal,  
longitud_de_nombre_de_catálogo_de_clave_principal,  
nombre_de_esquema_de_clave_principal,  
longitud_de_nombre_de_esquema_de_clave_principal,  
nombre_de_tabla_de_clave_principal,  
longitud_de_nombre_de_tabla_de_clave_principal,  
nombre_de_catálogo_de_clave_secundaria,  
longitud_de_nombre_de_catálogo_de_clave_secundaria,  
nombre_de_esquema_de_clave_secundaria,  
longitud_de_nombre_de_esquema_de_clave_secundaria,  
nombre_de_tabla_de_clave_secundaria,  
longitud_de_nombre_de_tabla_de_clave_secundaria);
```

Devuelve claves secundarias de la tabla especificada y claves secundarias de otras tablas vinculadas a la tabla especificada.

SQLGetConnectAttr

```
SQLRETURN SQLGetConnectAttr (identificador_de_conexión,  
atributo,  
puntero_de_valor, longitud_de_búfer,  
puntero_de_longitud_de_cadena);
```

Devuelve el valor de un atributo de conexión. El argumento a `atributo` puede ser uno de los valores enumerados en la tabla H.2.

Tabla H.2. Atributos y contenidos asociados a `puntero_de_valor`

Atributo	Contenidos de puntero_de_valor
SQL_ATTR_AUTOCOMMIT	Indica si se usa el modo de confirmación automático o manual. Puede ser <code>SQL_AUTOCOMMIT_OFF</code> (en cuyo caso las transacciones terminarían con <code>SQLEndTran</code>) o <code>SQL_AUTOCOMMIT_ON</code> (el predeterminado).
SQL_ATTR_CONNECTION_DEAD	Puede ser <code>SQL_CD_TRUE</code> (la conexión está muerta) o <code>SQL_CD_FALSE</code> (la conexión sigue activa).
SQL_ATTR_CONNECTION_TIMEOUT	Numero de segundos que se espera a que se ejecute una instrucción SQL antes de agotar el tiempo muerto. Si se configura como 0 (el predeterminado), significa que no hay tiempo muerto.

Atributo	Contenidos de puntero_de_valor
SQL_ATTR_CURRENT_CATALOG	Nombre del catalogo.
SQL_ATTR_LOGIN_TIMEOUT	Número de segundos de espera durante la conexión antes de agotar el tiempo muerto. 0 indica que no hay tiempo muerto.
SQL_ATTR_ODBC_CURSORS	Indica cómo utiliza el Administrador de controladores la biblioteca de cursores.
SQL_ATTR_PACKET_SIZE	Indica el tamaño de paquetes de red, en bytes.
SQL_ATTR_QUIET_MODE	Identificador de ventana principal de la aplicacion o nulo si el controlador no muestra cuadros de dialogo.
SQL_ATTR_TRACE	Puede ser SQL_OPT_TRACE (el predeterminado; no se realiza seguimiento alguno) o SQL_OPT_TRACE (sí realiza seguimiento).
SQL_ATTR_TRACEFILE	Nombre del archivo de seguimiento.
SQL_ATTR_TRANSLATE_LIB	Nombre de la biblioteca que contiene las funciones SQLDriverToDataSource y SQLDataSourceToDriver.
SQL_ATTR_TRANSLATE_OPTION	Un valor indicador de 32 bits pasado a la DLL de traduccion.
SQL_ATTR_TXN_ISOLATION	Una mascara de bits de 32 bits que establece el nivel de aislamiento de transacciones. Es necesario finalizar la transacción con SQLEndTran antes de invocar SQLSetConnectAttr con esta opción .

El argumento puntero_de_valor es el puntero desde el que se devuelve el valor. El argumento longitud_de_bufer puede contener uno de los siguientes valores:

- **La longitud de puntero_de_valor (o SQL_NTS) si apunta a una cadena.**
- **El resultado de SQL_LEN_BINARY_ATTR(longitud) si puntero_de_valor apunta a un bufer binario.**
- **Puede ser SQL_IS_INTEGER, SQL_IS_UNINTEGER, SQL_SMALLINT o SQLUSMALLINT si puntero_de_valor apunta a un tipo de datos especifico de longitud fija.**

- `SQL_IS_POINTER` si `puntero_de_valor` apunta a otro puntero.

El argumento `puntero_de_longitud_de_cadena` apunta a un bufer desde el que se devolverá el número de caracteres disponibles (excluyendo bytes nulos).

Si el numero de caracteres es mayor que `longitud_de_bufer`, se recorta el valor de `puntero_de_valor`.

SQLGetConnectOption

Devuelve el valor de la opción de conexión. La función ha quedado obsoleta por lo que en su lugar debe utilizar `SQLGetConnectAttr`.

SQLGetCursorName

```
SQLGetCursorName(identificador_de_instrucción,
nombre_de_cursor, longitud_de_nombre_de_cursor,
puntero_de_longitud_de_nombre);
```

Devuelve el nombre del cursor asociado al identificador de instrucción.

El argumento `nombre_de_cursor` apunta a un bufer desde el que se devolverá el nombre del cursor.

El argumento `puntero_de_longitud_de_nombre` apunta a un bufer desde el que se devolverá el número de caracteres disponible. Si el número de caracteres es mayor que `puntero_de_longitud_de_nombre`, se recorta `nombre_de_cursor`.

SQLGetDiagField

```
SQLGetDiagField(tipo_de_identificador, identificador,
número_de_registro, identificador_de_diagnóstico,
puntero_de_identificador_de_diagnóstico, longitud_de_búfer,
puntero_de_longitud_de_identificador_de_diagnóstico);
```

Devuelve información sobre errores, advertencias y diagnóstico del estado.

El `tipo_de_identificador` puede ser `SQL_HANDLE_ENV` (entorno), `SQL_HANDLE_DBC` (conexión), `SQL_HANDLE_STMT` (instrucción) o `SQL_HANDLE_DESC` (descripción).

El argumento `de_identificador` contiene el identificador concreto de tipo `tipo_de_identificador`.

El argumento `numero_de_registro` es el registro (empezando desde 1) desde el que se devuelve información, en caso de que haya.

El `identificador_de_diagnóstico` puede ser cualquiera de los valores descritos en la tabla H.3.

Tabla H.3. El argumento `identificador_de_diagnóstico` de `SQLGetDialogField`

identificador_de_diagnóstico	Descripción
<code>SQL_DIAG_CLASS_ORIGIN</code>	Devuelve una cadena que indica el origen de la clase <code>SQLSTATE</code> (por ejemplo, ISO 9075 u ODBC 3.0).
<code>SQL_DIAG_COLUMN_NUMBER</code>	Devuelve el número de columna del conjunto de resultados o el número de parámetro del conjunto de parámetros, empezando desde 0. Si no se aplica ninguno de éstos, contendrá <code>SQL_NO_COLUMN_NUMBER</code> o <code>SQL_COLUMN_NUMBER_UNKNOWN</code> .
<code>SQL_DIAG_CONNECTION_NAME</code>	Devuelve una cadena que contiene el nombre de conexión a la que se refieren los diagnósticos.
<code>SQL_DIAG_CURSOR_ROW_COUNT</code>	Devuelve el número de filas del cursor.
<code>SQL_DIAG_MESSAGE_TEXT</code>	Devuelve una cadena con el mensaje de diagnóstico (error o advertencia).
<code>SQL_DIAG_NATIVE</code>	Devuelve un código de error nativo (un entero) o 0 si no hay ninguno.
<code>SQL_DIAG_NUMBER</code>	Devuelve el número de registros de estado disponibles para el identificador (actualmente el controlador devuelve 1).
<code>SQL_DIAG_RETURNCODE</code>	Devuelve el código de devolución de la función.
<code>SQL_DIAG_ROW_COUNT</code>	Devuelve el número de filas afectadas por una operación SQL que modifica datos; (como <code>INSERT</code> o <code>DELETE</code>) ejecutada por <code>SQLExecute</code> , <code>SQLExecDirect</code> , <code>SQLBulkOperations</code> o <code>SQLSetPos</code> .
<code>SQL_DIAG_ROW_NUMBER</code>	Devuelve el número de fila de un conjunto de resultados o el número de parámetro de un conjunto de parámetros, empezando desde 1. Si no se aplica ninguno de éstos, contendrá <code>SQL_NO_ROW_NUMBER</code> o <code>SQL_ROW_NUMBER_UNKNOWN</code> .
<code>SQL_DIAG_SERVER_NAME</code>	Devuelve una cadena que contiene el nombre del servidor al que hace referencia el diagnóstico.
<code>SQL_DIAG_SQLSTATE</code>	Devuelve una cadena de cinco caracteres; que contiene el código <code>SQLSTATE</code> .

Identificador_de_diagnóstico	Descripción
SQL_DIAG_SUBCLASS_ORIGIN	Devuelve una cadena que contiene el origen de la subclase SQLSTATE (por ejemplo, ISO 9075 u ODBC 3.0).

El puntero_de_id_de_diagnóstico apunta a un bufer desde el que se devolverán los datos de diagnóstico. El argumento longitud_de_búfer puede contener uno de los siguientes valores:

- La longitud de puntero_de_identificador_de_diagnóstico (o SQL_NTS) si este apunta a una cadena.
- El resultado de SQL_LEN_BINARY_ATTR (longitud) si puntero_de_identificador_de_diagnóstico apunta a un búfer binario.
- Puede ser SQL_IS_INTEGER, SQL_IS_UNINTEGER, SQL_SMALLINT o SQL_USMALLINT si puntero_de_identificador_de_diagnóstico apunta a un tipo de datos específico de longitud fija.
- SQL_IS_POINTER si puntero_de_identificador_de_diagnóstico apunta a otro puntero.

El argumento puntero_de_longitud_de_identificador_de_diagnóstico apunta a un búfer desde el que se devolverá el número de caracteres disponibles (excluyendo bytes nulos). Si el número de caracteres es mayor que longitud_de_bufer, se recorta puntero_de_identificador_de_diagnóstico.

SQLGetDiagField no se envía los mismos diagnósticos a sí mismo. Por el contrario, devuelve uno de los siguientes valores: SQL_SUCCESS, SQL_SUCCESS_WITH_INFO (satisfactorio pero los datos se recortan), SQL_INVALID_HANDLE, SQL_ERROR (si los argumentos no eran válidos, etc.) o SQL_NO_DATA.

SQLGetDiagRec

```
SQLGetDiagRec(tipo_de_identificador, identificador,
número_de_registro, estado_sql, puntero_de_error_nativo,
texto_del_mensaje, longitud_de_texto_del_mensaje,
puntero_de_longitud_del_texto);
```

Devuelve información de diagnóstico adicional. Normalmente se invoca cuando una llamada anterior a una función ha devuelto SQL_SUCCESS o SQL_SUCCESS_WITH_INFO. El tipo_de_identificador puede ser SQL_HANDLE_ENV (entorno), SQL_HANDLE_ODBC (conexión), SQL_HANDLE_STMT (instrucción) o SQL_HANDLE_DESC (descripción).

El argumento `_de_identificador` contiene el identificador especificado de tipo `tipo_de_identificador`.

El argumento `numero_de_registro` es el registro (empezando desde 1) desde el que se devuelve **información**, en caso de que haya.

El argumento `estado_sql` apunta a un bufer desde el que se devolvera el código SQLSTATE de **cinco** caracteres.

El argumento `puntero_de_error_nativo` apunta a un bufer desde el que se devolvera el código de **error nativo**.

El argumento `texto_del_mensaje` apunta a un bufer desde el que se devolverá el mensaje de **diagnóstico** (error o advertencia).

El argumento `longitud_del_texto_del_mensaje` contiene la **longitud** del **búfer** `texto_del_mensaje`.

El argumento `puntero_de_longitud_de_texto` apunta a un bufer desde el que se **devolverá** el **número** de caracteres **disponibles**. Si este **número** es mayor que `longitud_del_texto_del_mensaje`, se recortara `texto_del_mensaje`.

SQLGetEnvAttr

```
SQLGetEnvAttr(identificador_de_entorno, atributo,  
puntero_de_valor, longitud_de_búfer,  
puntero_de_longitud_de_cadena);
```

Devuelve el valor de los atributos de **entorno**.

El argumento `atributo` puede adoptar uno de los valores enumerados en la tabla H.4.

Tabla H.4. Argumento atributo de SQLGetEnvAttr

Atributo	Contenidos de puntero_de_valor
SQL_ATTR_CONNECTION_POOLING	Valor de 32 bits para activar o desactivar la agrupación de conexiones.
SQL_ATTR_CP_MATCH	Valor de 32 bits para determinar cómo se selecciona una conexión de la agrupación existente.
SQL_ATTR_ODBC_VERSION	Entero de 32 bits para determinar si el comportamiento es ODBC 2.x u ODBC 3.x. Puede ser <code>SQL_OV_ODBC3</code> o <code>SQL_OV_ODBC2</code> .

El argumento `puntero_de_valor` apunta a un bufer desde el que se devolvera el valor del atributo.

El argumento `longitud_de_buferes` es la longitud de `puntero_de_valor` si este apunta a una cadena; en caso contrario, no se utilizara.

El puntero_de_longitud_de_cadena apunta a un bufer desde el que se devolvera el numero de caracteres disponibles.

Si este numero es mayor que longitud_de_bufer, se recorta el puntero_de_valor

SQLGetFunctions

```
SQLGetFunctions(identificador_de_conexión, id_de_función,  
puntero_admitido);
```

Devuelve las funciones que **admite** el controlador.

El argumento id_de_función puede ser un Id. de funcion individual o puede ser SQL_API_ODBC3_ALL_FUNCTIONS o SQL_API_ALL_FUNCTIONS. El primero lo utiliza ODBC3 y el segundo, ODBC2.

El argumento puntero_admitido apunta a un valor que contiene SQL_FALSE o SQL_TRUE (si id de función era una sola funcion, lo que indica si la función es admitida o no) o una matriz de dichos valores (empezando desde 0).

SQLGetInfo

```
SQLGetInfo(identificador_de_conexión, tipo_de_información,  
puntero_de_valor_de_información, longitud_de_búfer,  
puntero_de_longitud_de_cadena);
```

Devuelve informacion sobre el controlador y el servidor.

El argumento tipo_de_información contiene el tipo de información (como SQL_DRIVER_HDESC o SQL_DRIVER_HSTMT).

El argumento puntero_de_valor_de_informacion apunta a un búfer desde el que se devolverá la información, en función del argumento tipo_de_información.

El argumento longitud_de_bufer contiene la longitud del bufer puntero_de_valor_de_informacion. Si este no apunta a una cadena, se ignorara longitud_de_bufer.

El puntero_de_longitud_de_cadena apunta a un bufer desde el que se devolvera el número total de bytes (excluyendo los nulos). Si la longitud supera al valor de longitud_de_bufer, se recorta el valor de puntero_de_valor_de_información.

SQLGetStmtAttr

```
SQLGetStmtAttr(identificador_de_instrucción, atributo,  
puntero_de_valor, longitud_de_búfer,  
puntero_de_longitud_de_cadena);
```

Devuelve el valor del atributo de la instrucción.

El argumento a atributo puede ser una de las opciones admitidas que se enumeran en la tabla H.5.

Tabla H.5. Argumento atributo de SQLGetStmtAttr

Atributo	Contenidos de puntero_de_valor
SQL_ATTR_CURSOR_SCROLLABLE	Puede ser SQL_NONSCROLLABLE (los cursores desplazables no se requieren en el identificador de instruccion). Si se invoca SQLFetchScroll con este identificador, fetch_type solamente puede contener SQL_FETCH_NEXT (el predeterminado) o SQL_SCROLLABLE (se requieren cursores desplazables en el identificador de instruccion). Si SQLFetchScroll se invoca con este identificador, fetch_type puede contener cualquier identificador valido. Este atributo afecta a las llamadas a SQLExecDirect y SQLExecute.
SQL_ATTR_CURSOR_SENSITIVITY	Puede ser SQL_UNSPECIFIED (el predeterminado cuando el tipo de cursor no se especifica), SQL_SENSITIVE (los cursores muestran el conjunto de resultados sin mostrar ninguno de los cambios realizados por otros cursores) o SQL_INSENSITIVE (los cursores muestran todos los cambios realizados en un conjunto de resultados por otros cursores). Afecta a SQLExecute y SQLExecDirect.
SQL_ATTR_CURSOR_TYPE	Especifica el tipo de cursor y puede contener SQL_CURSOR_FORWARD_ONLY, SQL_CURSOR_STATIC, SQL_CURSOR_KEYSET_DRIVEN (el controlador utiliza las claves para el numero de filas especificado en el atributo de instruccion SQL_ATTR_KEYSET_SIZE) o SQL_CURSOR_DYNAMIC.
SQL_ATTR_KEYSET_SIZE	Numero de filas del conjunto de claves para un tipo de cursor SQL_CURSOR_KEYSET_DRIVEN.
SQL_ATTR_MAX_LENGTH	Cantidad maxima de datos que el controlador devuelve desde columnas de caracteres o binarias. Si el puntero_

	<p><code>de_valor</code> es menor que la cantidad de datos, <code>SQLFetch</code> y <code>SQLGetData</code> se recortaran y devuelven <code>SQL_SUCCESS_WITH_INFO</code>.</p>
<code>SQL_ATTR_MAX_ROWS</code>	Número máximo de filas que el controlador puede devolver (0 equivale a ilimitado).
<code>SQL_ATTR_NOSCAN</code>	Puede ser <code>SQL_NOSCAN_OFF</code> (el controlador examina la instrucción SQL en busca de conversiones de escape, el predeterminado) o <code>SQL_NOSCAN_ON</code> (el controlador no examina la instrucción).
<code>SQL_ATTR_PARAM_BIND_TYPE</code>	Puede ser <code>SQL_PARAM_BIND_BY_COLUMN</code> (el predeterminado, que indica la vinculación en orden de columnas) o la longitud de la estructura o búfer para la vinculación en orden de filas.
<code>SQL_ATTR_PARAM_OPERATION_PTR</code>	Apunta a una matriz que contiene <code>SQL_PARAM_PROCEEDS</code> <code>SQL_PARAM_IGNORE</code> que determina si un parámetro se ignorara durante la ejecución. También puede ser un puntero nulo, en cuyo caso no se devuelven valores de estado del parámetro.
<code>SQL_ATTR_PARAM_STATUS_PTR</code>	Apunta a una matriz de valores (uno para cada fila) con información de estado después de invocar <code>SQLExecute</code> o <code>SQLExecDirect</code> que contiene uno de los siguientes valores: <code>SQL_PARAM_SUCCESS</code> , <code>SQL_PARAM_SUCCESS_WITH_INFO</code> (satisfactorio con una advertencia), <code>SQL_PARAM_ERROR</code> , <code>SQL_PARAM_UNUSED</code> (normalmente porque se ha definido <code>SQL_PARAM_IGNORE</code>) o <code>SQL_PARAM_DIAG_UNAVAILABLE</code> . También se puede configurar como un puntero nulo, en cuyo caso los datos no se devuelven.
<code>SQL_ATTR_PARAMS_PROCESSED_PTR</code>	Apunta a un búfer desde el que se devuelve el número de conjuntos de parámetros procesados, incluyendo los errores. Puede ser un puntero nulo, en cuyo caso no se devolverá ningún número.

Atributo	Contenidos de puntero_de_valor
SQL_ATTR_PARAMSET_SIZE	Número de valores de cada parametro.
SQL_ATTR_QUERY_TIMEOUT	Numero de segundos que se espera a que se ejecute una instrucción SQL antes de vencer el tiempo muerto. Si el <code>puntero_de_valor</code> es 0 (el predeterminado), no habrá tiempo muerto.
SQL_ATTR_ROW_ARRAY_SIZE	Numero de filas devueltas por una llamada a <code>SQLFetch</code> o <code>SQLFetchScroll</code> . El predeterminado es 1.
SQL_ATTR_ROW_BIND_OFFSET_PTR	Apunta a un desplazamiento añadido a punteros para cambiar la vinculacion de los datos de columnas.
SQL_ATTR_ROW_BIND_TYPE	Determina la orientación de la vinculación . Puede ser <code>SQL_BIND_BY_COLUMN</code> (indica una vinculación en orden de columnas) o la longitud de la estructura o bufer al que se van a vincular los resultados (en vinculaciones en orden de filas).
SQL_ATTR_ROW_NUMBER	Numero de la fila actual o 0 si no se puede determinar.
SQL_ATTR_ROW_OPERATION_PTR	Apunta a una matriz de elementos (uno por cada fila) que contienen <code>SQL_ROW_PROCEED</code> o <code>SQL_ROW_IGNORE</code> que determina si la fila se va a incluir en una operación de gran volumen (no incluye <code>SQLBulkOperations</code>). Tambien se puede definir con un puntero nulo , en cuyo caso no se devolvera una matriz.
SQL_ATTR_ROW_STATUS_PTR	Apunta a una matriz de valores (uno por cada fila) que contiene valores de estado de fila despues de una llamada a <code>SQLFetch</code> o <code>SQLFetchScroll</code> . Tambien se puede definir como un puntero nulo , en cuyo caso el controlador no devolvera la matriz.
SQL_ATTR_ROWS_FETCHED_PTR	Apunta a un bufer desde el que se devuelve el numero de filas devueltas o afectadas por una llamada <code>SQLFetch</code> , <code>SQLFetchScroll</code> , <code>SQLSetPos</code> o <code>SQLBulkOperations</code> , incluyendo errores.

Atributo	Contenidos de puntero_de_valor
SQL_ATTR_SIMULATE_CURSOR	Puede ser SQL_SC_NON_UNIQUE (no es definitivo que esas instrucciones UPDATE O DELETE situadas de forma simulada afecten a una sola fila), SQL_SC_TRY_UNIQUE (el controlador intenta garantizar que las instrucciones UPDATE O DELETE situadas de forma simulada afectan a una sola fila) o SQL_SC_UNIQUE (es definitivo que las instrucciones UPDATE O DELETE situadas de forma simulada afectan a una sola fila).

El argumento `puntero_de_valor` apunta a un bufer desde el que se devolvera el valor de atributo.

El argumento `longitud_de_bufer` puede contener uno de los siguientes valores:

- La longitud de `puntero_de_valor` (o `SQL_NTS`) si este apunta a una cadena.
- El resultado de `SQL_LEN_BINARY_ATTR(longitud)` si `puntero_de_valor` apunta a un búfer binario.
- Puede ser `SQL_IS_INTEGER`, `SQL_IS_UNINTEGER`, `SQL_SMALLINT` o `SQLUSALLINT` si `puntero_de_valor` apunta a un tipo de datos específico de longitud fija.
- `SQL_IS_POINTER` si `puntero_de_valor` apunta a otro puntero.

El `puntero_de_longitud_de_cadena` apunta a un bufer desde el que se devolvera el numero total de bytes (excluyendo los nulos).

Si la longitud es mayor que `longitud_de_bufer`, se recortara el `puntero_de_valor`.

SQLGetStmtOption

Devuelve el valor de opción de la instrucción.

Esta función ha quedado obsoleta, por lo que en su lugar tendra que utilizar `SQLGetStmtAttr`.

SQLGetTypeInfo

```
SQLGetTypeInfo(identificador_de_instrucción,tipo_de_datos);
```

Devuelve un conjunto de resultados SQL con información sobre el tipo de datos especificado.

Al configurar `tipo_de_datos` como `SQL_ALL_TYPES`, se devuelve información sobre los tipos de datos devueltos por el servidor.

SQLNativeSql

```
SQLNativeSql(identificador_de_conexión, cadena_sql, longitud_de_cadena_sql, cadena_sql_modificada, longitud_de_cadena_sql_modificada, puntero_de_longitud_de_cadena);
```

Devuelve una cadena SQL modificada (no ejecutada) por el controlador.

El argumento `cadena_sql_modificada` apunta a un bufer desde el que se devolverá la instrucción SQL modificada.

El argumento `longitud_de_cadena_sql_modificada` apunta a un búfer desde el que se devolverá el número de bytes (excluyendo los nulos).

El puntero `de_longitud_de_cadena` apunta a un bufer desde el que se devolverá el número total de bytes (excluyendo los nulos). Si la longitud es mayor que el valor de `longitud_de_cadena_sql_modificada`, se recortará el valor de `cadena-sql-modificada`.

SQLNumParams

```
SQLNumParams(identificador_de_instrucción, puntero_a_número_de_parámetros);
```

Devuelve el número de parámetros de una instrucción.

El argumento `puntero_a_numero_de_parametros` apunta a un bufer desde el que se devolverá el número de parámetros.

SQLNumResultCols

```
SQLNumResultCols(identificador_de_instrucción, puntero_a_número_de_columnas);
```

Devuelve el número de columnas del conjunto de resultados.

El argumento `puntero_a_numero_de_columnas` apunta a un búfer desde el que se devolverá el número de columnas.

SQLParamData

Se utiliza en combinación con `SQLPutData` para proporcionar datos de parámetros en el momento de la ejecución (resulta muy útil para valores de datos largos).

SQLPrepare

```
SQLPrepare(identificador_de_instrucción, cadena_sql, longitud_de_cadena_sql);
```

Prepara una instrucción SQL para su posterior ejecución.

SQLPrimaryKeys

```
SQLPrimaryKeys(identificador_de_instrucción, nombre_de_catálogo, longitud_de_nombre_de_catálogo, nombre_de_esquema, longitud_de_nombre_de_esquema, nombre_de_tabla, longitud_de_nombre_de_tabla);
```

Devuelve las **columnas** de clave **primaria** de la tabla especificada.

SQLPutData

```
SQLPutData(identificador_de_instrucción, puntero_de_datos, longitud_de_puntero_de_datos);
```

Sirve para enviar datos de **columnas** o parámetros durante la ejecución.

El argumento **puntero_de_datos** apunta a un bufer que contiene los **datos** de **parámetros** o **columnas** (el tipo es el especificado por el argumento **tipo_de_valor** de `SQLBindParameter` o por el argumento **tipo_de_destino** de `SQLBindCol`).

El argumento **longitud_de_puntero_de_datos** especifica la longitud de los datos enviados a `SQLPutData` (`SQL_NTS`, `SQL_NULL_DATA` o `SQL_DEFAULT_PARAM`).

SQLRowCount

```
SQLRowCount(identificador_de_instrucción, puntero_de_número_de_filas);
```

Devuelve el número de **filas** afectadas por una **instrucción** SQL que modifica datos (por ejemplo `INSERT` o `DELETE`).

El argumento **puntero_de_número_de_filas** apunta a un bufer desde el que se **devolverá** el número de filas o -1 si no está disponible.

SQLSetConnectAttr

```
SQLSetConnectAttr(identificador_de_conexión, atributo, puntero_de_valor, longitud_de_cadena);
```

Define un atributo de **conexión**

Consulte `SQLGetConnectAttr`. Encontrará una lista y la descripción de los posibles atributos.

El argumento `puntero_de_valor` apunta al valor del atributo, cuyo tipo depende de `atributo`.

El argumento `longitud_de_cadena` puede contener uno de los siguientes valores:

- La longitud del `puntero_de_valor` (o `SQL_NTS`) si este apunta a una cadena.
- El resultado de `SQL_LEN_BINARY_ATTR(longitud)` si `puntero_de_valor` apunta a un búfer binario.
- Pueden ser `SQL_IS_INTEGER`, `SQL_IS_UNINTEGER`, `SQL_SMALLINT` o `SQLUSMALLINT` si `puntero_de_valor` apunta a un tipo de datos específico de longitud fija.
- `SQL_IS_POINTER` si `puntero_de_valor` apunta a otro puntero.

SQLSetConnectOption

Establece una opción de conexión. Esta función ha quedado obsoleta, por lo que en su lugar debe utilizar `SQLSetConnectAttr`.

SQLSetCursorName

```
SQLSetCursorName(identificador_de_instrucción,  
nombre_de_cursor, longitud_de_nombre_de_cursor);
```

Especifica un nombre de cursor

SQLSetEnvAttr

```
SQLSetEnvAttr(identificador_de_entorno, atributo,  
puntero_de_valor, puntero_de_longitud_de_cadena);
```

Define un atributo de entorno. En la descripción de `SQLGetEnvAttr` encontrará una lista de posibles atributos.

SQLSetPos

```
SQLSetPos(identificador_de_instrucción, número_de_fila,  
operación, tipo_de_bloqueo);
```

Desplaza un cursor a una posición de un bloque obtenido de datos y también puede actualizar datos del conjunto de filas o actualizar y eliminar los datos subyacentes.

El argumento `número_de_fila` selecciona la fila del conjunto de resultados afectado por la operación (empezando desde 1). Si se configura como 0, la operación se aplica a todas las filas.

El argumento `operacion` especifica la operación que se debe realizar, que puede ser `SQL_POSITION`, `SQL_REFRESH`, `SQL_UPDATE` o `SQL_DELETE`. En la tabla H.6 se describe el argumento `operacion`.

Tabla H.6. El argumento `operacion`

Operación	Descripción
<code>SQL_POSITION</code>	El controlador sitúa el cursor en la fila <code>número_de_fila</code> .
<code>SQL_REFRESH</code>	El controlador sitúa el cursor en la fila <code>número_de_fila</code> y actualiza datos en dicha fila. Los datos de la fila no se vuelven a obtener, lo que difiere de una llamada de actualización a <code>SQLFetchScroll</code> con <code>tipo_de_recuperación</code> .
<code>SQL_UPDATE</code>	El controlador sitúa el cursor en la fila <code>número_de_fila</code> y actualiza los datos asociados a los valores de los bufer de conjunto de fila del argumento <code>TargetValuePtr</code> en <code>SQLBindCol</code> .
<code>SQL_DELETE</code>	El controlador sitúa el cursor en la fila <code>número_de_fila</code> y elimina los datos asociados.

El argumento `tipo_de_bloqueo` especifica la operación de bloqueo de fila después de ejecutar la operación y puede ser `SQL_LOCK_NO_CHANGE`, `SQL_LOCK_EXCLUSIVE` o `SQL_LOCK_UNLOCK`.

SQLSetScrollOptions

Define opciones que afectan al comportamiento de los cursores. Esta función ha quedado obsoleta, por lo que en su lugar debe utilizar `SQLSetStmtAttr`.

SQLSetStmtAttr

```
SQLGetStmtAttr(identificador_de_instrucción, atributo,
puntero_de_valor, longitud_de_cadena);
```

Define un atributo de instrucción.

Encontrará la lista de posibles valores de atributo en la descripción de `SQLGetStmtAttr`.

El argumento `puntero_de_valor` apunta al valor del atributo, cuyo tipo depende de `atributo`.

El argumento `longitud_de_cadena` puede contener uno de los siguientes valores:

- La longitud de `puntero_de_valor` (o `SQL_NTS`) si este apunta a una cadena.
- El resultado de `SQL_LEN_BINARY_ATTR(longitud)` si `puntero_de_valor` apunta a un búfer binario.
- Puede ser `SQL_IS_INTEGER`, `SQL_IS_UNINTEGER`, `SQL_SMALLINT` o `SQLUSMALLINT` si `puntero_de_valor` apunta a un tipo de datos específico de longitud fija.
- `SQL_IS_POINTER` si `puntero_de_valor` apunta a otro puntero.

SQLSetStmtOption

Define una opción de instrucción. Esta función se ha quedado obsoleta; en su lugar, utilice `SQLSetStmtAttr`.

SQLSpecialColumns

```
SQLSpecialColumns(identificador_de_instrucción,  
tipo_de_identificador, nombre_de_catálogo,  
longitud_de_catálogo, nombre_de_esquema,  
longitud_de_nombre_de_esquema, nombre_de_tabla,  
longitud_de_nombre_de_tabla, ambito, nulo);
```

Devuelve información de columnas que identifica de forma exclusiva a una fila de la tabla especificada o las columnas actualizadas automáticamente cuando una transacción actualiza un valor del registro.

El argumento `tipo_de_identificador` contiene el tipo de columna que se va a devolver. Debe ser `SQL_BEST_ROWID` o `SQL_ROWOVER`. `SQL_BEST_ID` devuelve el conjunto más pequeño de columnas que identifican a un registro de forma exclusiva (las columnas devueltas se pueden diseñar para habilitarlo). `SQL_ROWOVER` devuelve las columnas que se actualizan automáticamente cuando una transacción actualiza un valor del registro.

El argumento `ambito` es el ámbito mínimo exigido del Id. de la fila. Puede ser `SQL_SCOPE_CURROW` (el Id. de fila es válido solamente para esa fila), `SQL_SCOPE_TRANSACTION` (el Id. de fila es válido mientras dure la transacción actual) o `SQL_SCOPE_SESSION` (el Id. de fila es válido durante toda la sesión).

El argumento `nulo` indica si se incluyen columnas que puedan contener valores nulos. Puede ser `SQL_NO_NULLS` (se excluyen las columnas que contengan nulos) o `SQL_NULLABLE` (se incluyen las columnas que puedan contener nulos).

SQLStatistics

```
SQLStatistics(identificador_de_instrucción, nombre_de_catálogo,  
longitud_de_nombre_de_catálogo, nombre_de_esquema,  
longitud_de_nombre_de_esquema, nombre-de-tabla,  
longitud_de_nombre_de_tabla, tipo_de_índice, reservado);
```

Devuelve estadísticas sobre tablas e índices asociados.

El argumento `tipo_de_índice` puede ser `SQL_INDEX_UNIQUE` o `SQL_INDEX_ALL`.

El argumento `reservado` puede ser `SQL_ENSURE` (devuelve las columnas `CARDINALITY` y `PAGES`) o `SQL_QUICK` (solo devuelve las columnas `CARDINALITY` y `PAGES` si están disponibles).

SQLTablePrivileges

```
SQLTablePrivileges(identificador_de_instrucción,  
nombre_de_catálogo, longitud_de_catálogo, nombre_de_esquema,  
longitud_de_nombre_de_esquema, nombre-de-tabla,  
longitud_de_nombre_de_tabla);
```

Devuelve una lista de tablas y privilegios asociados.

SQLTables

```
SQLTables(identificador_de_instrucción, nombre_de_catálogo,  
longitud_de_catálogo, nombre_de_esquema,  
longitud_de_nombre_de_esquema, nombre-de-tabla,  
longitud_de_nombre_de_tabla, tipo_de_tabla,  
longitud_de_tipo_de_tabla);
```

Devuelve una lista de tablas, `catálogo` o nombres de esquema y tipos de tablas.

SQLTransact

Finaliza una **transacción**. Esta función ha quedado obsoleta, por lo que en su lugar debe utilizar `SQLEndTran`.



Contenido del CD-ROM

El CD-ROM que acompaña a este libro, contiene una gran cantidad de programas y de archivos que le ahorrarán mucho tiempo durante su trabajo con el libro.

Las siguientes secciones describen lo que puede encontrar en el CD-ROM.

Código fuente

El CD-ROM, contiene **todo** el código fuente desarrollado a lo largo del libro. Para trabajar con los códigos le recomendamos que copie todos los archivos a su disco duro.

MySQL

Se incluye la última versión del servidor de bases de datos, **tanto** para Linux como para Windows.

Apache

Se incluye el código fuente de la última versión del servidor Apache.

Java

En este directorio encontrara la ultima version comercializada 1.4 del Kit de desarrollo de Software (SDK) de Sun Microsystems, compatible con Windows, y Linux.

Perl

Contiene la distribucion fuente del lenguaje de script Perl, asi como los binarios. Para instalar Perl en su sistema **Unix/Linux**, extraiga la distribucion fuente en un directorio como **/usr/local/src** de su disco duro y lea el **archivo INSTALL**.

Los usuarios de Windows deberian bajarse la distribucion **binaria** de la siguiente **dirección Web**: <http://www.perl.com/CPAN-local/README.html>

PHP

El CD-ROM incluye la distribucion de la fuente de PHP.

Python

En este directorio se incluye la ultima version **dePython**.

En el fichero **leame.txt**, situado en el directorio raiz del CD-ROM, encontrara una **descripción** mas detallada de todas las utilidades y programas que se **incluyen**.

Indice alfabetico

Access, 519
Active, 761
ActiveKids, 762
Actualización, 363
Actualizaciones, 238
Acuerdo sobre el ambito, 323
Adabas, 519
AdabasD, 519
ADD, 616
Addgroup, 585
Addison-Wesley, 316
Addn, 274
Addslashes, 722
Adduser, 585
Administrador, 350
ADO, 827
ADOBD, 716
AGAINST, 195
AID, 602
Aislamiento, 344
AIX, 580
Alias, 726
ALL, 223
Alter, 67, 152, 174, 522, 572, 616
 table, 174
ALUES, 757
Ámbito, 323
AMD Duron. 324

Arg, 269-270
Args, 271
ARRAYSIZE, 774
AS, 71
ASC, 58
ASCII, 821
Atomicidad, 343
Atributos, 325
 dbi, 761
AUTO-INCREMENT, 465, 733
AUTOCOMMIT, 169, 765
AutoReconnect, 782
Available-drivers, 744
AVG. 62

B

Back-log. 484
BACKUP, 400
BAK, 500
Barras verticales, 418
Basedir, 351
BDB, 492, 583
Bdb_home, 351
BEGIN, 617
Begin-work, 749
BETWEEN, 123
Biblioteca Openssl, 572
Big5, 529
BIGINT, 88
Binaria, 137
Boyce-Codd, 309
BSDi, 580
Búfer de índice, 410

C

C, 244
C++, 244
C.J. Date, 316
CachedKids, 762
Calculos, 225
Campo id, 212
 rank, 212
Canalizaciones, 352
Cardinalidad, 328
Cardinality, 236
Carga de los datos, 333

Carpeta Inicio, 356
CGI, 479
CHANGE, 67
Changed, 379
CHAR, 91, 191
CHECK TABLE, 378
Chmod, 451
ChopBlanks, 762
Chown, 451
Ciclo de vida, 322
CIPHER 573
Clave primaria, 187
Claves externas, 332
CLIENT_INTERACTIVE, 497
CLOSE, 776
CNF, 362
Coherencia, 343
Collation, 236
Column_info, 749
Columna, 66
Coma binaria, 137
 decimal, 137
Compress, 99
CONCAT, 71
Conceptual, 325
Conectividad, 328
Connect, 744
Connect-cached, 745
Const, 222
CREATE, 618
 function, 266
 index, 174
 table, 48
Croat, 529
Ctype, 532
CURRENT-DATA, 70
CURSOR 776
CursorName, 766
CVS, 259
Czech, 529

D

Danish, 529
DAO, 827
DATA, 52, 419
Data-sources, 746
DATABASE, 65, 174

Datadir, 350,495
David Patterson, 602
DAYOFMONTH, 71
DAYOFWEEK, 98
Days–data, 570
Dbgconnet, 258
DB2, 519
DBD, 116,171
DBI, 514
DBMS, 94
Ddeinit, 272
Dec OSF, 580
Dec8, 529
DECIMAL, 89
Decimal, 137
Definición de los objetivos, 323
 de todos los problemas, 323
Deinit, 275
DELAYED, 499,504
Delayed–queue–size, 484
DELETE, 64, 621
DESC, 621
Directorio de datos, 359
DISABLE, 617
Disconnect, 750
Diseño, 322
DISTINCT, 146
Distintos, 61
Distribucion **binaria**, 587
 de Unix, 350
Distribucion fuente de Unix, 350
Division, 602
DMV, 287
DNS, 489
DO, 621,750

E.F. Codd, 309
Eliminaciones, 238
Empress, 519
ENABLE, 617
ENCODE, 703
Entidad debil, 327
Entidad-relacion, 325
Entidades, 325
ENUM, 93,619
Eq_ref, 223

Err, 747, 769
Errores, 363
 clave, 464
Errstr, 747
Esclavos, 441
Estonia, 529
Euc_kr, 529
Excesivos, 369
Exceso de actualizaciones, 462
Execute, 99, 758
EXPLAIN, 220
Extended, 379
Extension MYI, 214
Extra, 540

Fallo de hardware, 437
Fast, 379
Fecha, 70
Fetch, 759
FETCHALL, 777
FETCHMANY, 778
Fields, 735
FILE, 420, 572
Find, 351
Firma, 598
FLOAT, 88, 789
FLUSH, 450
 logs, 370
FNAME, 759
Force, 100
Foreign-key-info, 750
Forma normal, 306-307
FreeBSD, 43, 353
Frm, 352, 610
FROM, 53
 master, 444
FrontPage, 476
FULLTEXT, 494
Func, 747
Funcion Addn, 274
 db_pconnet, 258

G

Garth Gibson, 602
GB, 476

Gb2312,529
Gcc, 584
Gennanl, 529
Get-info, 751
GetTypeMap, 784
Gibson, 602
GLOBAL, 182.508
GNU, 584
GRANT, 335,551,572
Group, 525
Gzip, 584

H

Hamming, 603
HandleError, 762
Hash, 748
HEAP, 113,214
Hebrew, 529
Hex, 598
Hora, 70
HotBackup, 435
HP-UX, 580
Html, 100,255

Id. 212
IDE, 512
Identificadores, 761
Idioma, 528
IFNULL, 706
Implementación, 322
IN. 124
INDEX. 174,223,235
Índice, 187,410
 primario, 190
Información de una tabla, 53
Informix, 519
Iniciar, 358,584
Init, 268,274
Initd, 269
InitialTimeout, 783
InnoDB, 115,161,208
Innotest, 162
Inserciones, 238
INSERT, 52
 DELAYED, 504

INSTR, 676
Instrucción SELECT, 236-237
INT, 88
INTEGER, 88
Integridad referencial, 332
Intel, 438
Interfaz Web, 336
INTO, 51,416
IP, 486

Java, 244,802
Jerarquico, 284-285

K

KeepAlive, 479
Key-buffer-size, 478
Kids, 763
Kill, 364

L

LAST-INSERT-ID, 206
Lasth, 769
Latin1, 529
Latin1_de, 529
LCASE, 676
LEADING, 684
Length, 271,678
LIKE, 123
Límites de cardinalidad, 328
LINES TERMINATED BY, 422
Linux, 257
LOAD DATA, 52, 419
 FROM MASTER, 444
 LOCAL, 426,575
LOCAL, 426,575
Localhost, 519
LOCK, 175
 tables, 403
Log, 473
Log-uptate, 473
LOGS, 370,623
LOB, 92
LONGTEXT, 92
LONGVARBINARY, 789

LONGVARCHAR, 789
Looks-like-number, 748
LOW PRIORITY, 426
LOWER, 674

M

MacOS X, 580
Mantenimiento, 322
MASTER, 629
Master.info, 442
MATCH, 193
Matriz, 531
MAX, 60
Max-connections, 480
Max-connections-used, 480
MB, 476
Media, 62
Medium, 379
MEDIUMBLOB, 92
MEDIUMTEXT, 92
Menu Proyecto, 834
MERGE, 109,214
Mes, 66
Message, 270
MHz, 324
Microsoft, 356
Mimer, 519
MIN, 62
Mínimo, 62
Minus, 526
Modelo entidad-relacion, 325
 jerárquico, 284-285
MODIFY, 67
MONTH, 71
MS-SQL, 520
MsgBox, 834
MSYQL, 806
Mysql_db_name, 733
Mysql_fetch_row, 813
Mysql_result, 733
Mysqld, 472
Multibytes, 673
My.ini, 356
MYD, 500,610
MyISAM, 104
Myisamchk, 374,392
Mysql_kill, 816

Mysql_num_fields, 817
Mysql_options, 818
MYSQL_ROW, 808
Mysqldadmin, 350
Mysqlcheck, 350
Mysqld, 350
Mysqldhotcopy, 350,429
Mysqlexport, 350

N

Name, 765
NATIONAL, 91
Neat, 749
Neat-list, 749
NetMeeting, 476
Nombrebd, 435
Non_unique, 236
Normalización, 298
NOT, 120
 between, 123
 in, 124
 like, 123
 regexp, 124
NOW, 77
NT, 352
NUL, 821
NULL, 68,123
NULLABLE, 767
Numeric, 725

O

Objetivos, 323
Obligatoria, 326
ODBC, 582,827
 dbi, 514
Opción changed, 379
 extended, 379
 fast, 379
 medium, 379
 quick, 379
Opcional, 326
Operadores, 123
Optimización, 229
OPTIMIZE, 236
OPTIONALLY, 418
OR, 54

Oracle, 40, 519
ORDER BY, 216
Organización, 323
OTHER 789

P

Packed, 236
Pager, 100
Parametro arg, 269-270
 args, 271
 initd, 269
 length, 271
 message, 270
 result, 271
ParamValues, 768
Paridad, 602
Password, 536
Patrones, 56
Patterson, 602
Pconnect, 479
Perl, 244, 599
Permiso, 544
 reload, 821
Permisos, 537
Pg, 520
PHP, 244, 479
Phyton, 778
Ping, 751
Port, 474
Posibilidades y límites, 323
Prepare, 751
Prepare-cached, 751
Preparedstatement, 786
Previous, 796
Primario, 190
Privilegio SHUTDOWN, 822
PROCESS, 502, 572
Processgriv, 541
Profile, 764
Pruebas, 322
PSM, 526
Puesta en marcha, 322

Q

Query, 203, 481
Querylog, 370

Quick, 379,500
Quote, 752
Quote-identifier, 752

R

RAID, 495
RaiseError, 757, 761, 764
RAND, 599
Randy Kantz, 602
Rank, 212
Raw, 101
RDBMS, 40
RDO, 827
READ, 178
 uncommitted, 181
REFERENCES, 624
Referencial, 332
REGEXP, 124
Registros distintos, 61
 excesivos, 369
RENAME TABLE, 174
REPEATABLE READ, 181
REPLACE, 110, 425
REPLICATION SLAVE, 443
REQUIRE SSL, 573
Reset, 274
RESET MASTER, 366, 449
RESTORE, 404-405
Result, 271
Resultados, 58
ResultSet, 791
REVOKE, 335, 546
RIGHT, 78
RLIKE, 124
ROLLBACK, 161, 505
Rollback, 752
Row, 55
Rows, 735, 761
RPM, 476, 587
Run-all-tests, 512

SCALE, 768
SCO, 580
SCSI, 512
SELECT, 53, 236-237, 537

SERIALIZABLE, 182,527
Servidor DNS, 489
 Web Apache, 479
SESSION, 182,508,632
SESSION-USER, 711
SET, 60,508
Set-err, 747
SGI Iris, 580
SHOW INDEX, 235
 slave, 446
 tables, 49
SHUTDOWN, 559,572,822
Shutdowngriv, 541
Signo de **interrogación**, 755
Símbolo #, 361
 @, 332
Simultaneidad, 343
Sjis, 529
SLAVE, 446,629
 start, 461
SLED, 602
SMALLINT, 790
Socket, 101,474
Sockets, 352
Solaris, 580
Solid, 520
Sort_buffer, 484-485
SQL, 287
SQLAllocEnv, 838
SQLAllocStmt, 839
SQLSetPos, 859
SQLSetStmtAttr, 861
SQLSTATE, 849
SSL, 446,572
START, 461
State, 748,769

Tablas DBD, 171
 de **permiso**, 544
TABLE, 48,174
Tablename, 740
Tables, 49,403,735
Tabuladores, 418
Taint, 765
Tar, 584
Tcl, 244

TCP/IP, 352,488
Tee, 100
TEMPORARY, 620
Tenninadores, 418
TEXT, 92,191,619
THEN, 701
Threads-running, 480
THREADSAFETY, 774
TIMESTAMP, 97, 619
TIMESTAMP-FLAG, 808
TINYBLOB, 92
TINYINT, 209
TINYTEXT, 92
Tis620, 529
Tmpdir, 474
Total, 62
Trace, 746,748
Trace-msg, 748
TraceLevel, 765
TRAILING, 684
Transacciones, 343
Triggers, 526
TRUNCATE, 153,174
Type, 522

U

UDF, 263
Ujis, 529
UML, 257
Unbuffered, 100
UNCOMMITTED, 181
Undef, 751
UNION, 147
UNIX_TIMESTAMP, 98
UNLOCK, 175
Uno a uno, 326
Uno a varios, 326
UNSIGNED, 86, 88, 725
UPDATE, 103,758
UPPER 684
USAGE, 556
User, 102
USING, 144

Valor Max-connections, 480

- max_connections_used, 480
- VALUES, 51,757
- VARCHAR, 49, 91, 191
- VARIANCE, 525
- Varios a uno, 326
- Varios servidores, 591
- VBDAO, 828
- VI3.NET, 827
- Verbose, 102
- Vertical, 99
- Vinculacion simbolica, 609
- Vistas, 292-293
- Visual Basic, 244, 831
- Visual SourceSave, 259
- Volcados, 333

W

- Wait, 102
- Warn, 765
- Warning, 801
- WasNull, 798
- Web Apache, 479
- Winmysqladmin, 360
- Winsock 2, 356
- WRITE, 178
- WWW.openssl.org/, 572

X

- X509, 573
- XML, 414
- XP, 352

Y

- YEAR, 70

Z

- Zcat, 584
- Zerofill, 86, 88, 725
- Zubat, 481