EBook Gratis

APRENDIZAJE MySQL

Free unaffiliated eBook created from
Stack Overflow contributors.

#mysql

Tabla de contenido

Acerca de ... 1

Capítulo 1: Empezando con MySQL ... 2

Observaciones ... 2

Versiones ... 2

Examples ... 3

Empezando ... 3

Ejemplos de esquemas de información ... 7

Lista de procesos ... 7

Búsqueda de procedimientos almacenados ... 7

Capítulo 2: ACTUALIZAR ... 9

Sintaxis ... 9

Examples ... 9

Actualización básica ... 9

Actualizando una fila ... 9

Actualizando todas las filas ... 9

Actualizar con un patrón de unión ... 10

ACTUALIZAR CON ORDENAR Y LIMITAR ... 10

ACTUALIZACIÓN de tabla múltiple ... 11

Actualización masiva ... 11

Capítulo 3: Administrador de MySQL ... 13

Examples ... 13

Cambiar contraseña de root ... 13

Eliminar base de datos ... 13

RENOMBRE Atómico y Recarga de Mesa ... 13

Capítulo 4: Agrupación ... 14

Examples ... 14

Desambiguación ... 14

Capítulo 5: Agrupar por ... 15

Sintaxis ... 15

Parámetros ... 15

Observaciones 15
Examples 15
Grupo usando la función SUMA 15
Grupo usando la función MIN 16
GRUPO UTILIZANDO COUNT FUNCION 16
GRUPO POR USO QUE TIENE 16
Grupo utilizando Group Concat 16
GROUP BY con funciones AGREGADAS 17
Capítulo 6: ALTERAR MESA 20
Sintaxis 20
Observaciones 20
Examples 21
Cambio de motor de almacenamiento; tabla de reconstrucción cambiar file_per_table 21
ALTER COLUMNA DE MESA 21
Tabla ALTER añadir INDEX 22
Cambiar el valor de incremento automático 22
Cambiar el tipo de una columna de clave primaria 22
Cambiar definición de columna 22
Renombrando una base de datos MySQL 23
Intercambiando los nombres de dos bases de datos MySQL 23
Renombrando una tabla MySQL 24
Renombrando una columna en una tabla MySQL 24
Capítulo 7: Archivos de registro 26
Examples 26
Una lista 26
Registro de consultas lentas 26
Registro de consultas generales 27
Registro de errores 29
Capítulo 8: Aritmética 31
Observaciones 31
Examples 31
Operadores aritméticos 31

BIGINT ... 31
DOBLE ... 32
Constantes matemáticas ... 32
Pi ... 32
Trigonometría (SIN, COS) ... 32
Seno ... 32
Coseno ... 32
Tangente ... 33
Arco coseno (coseno inverso) ... 33
Seno del arco (seno siniestro) ... 33
Arco tangente (tangente inverso) ... 33
Cotangente ... 33
Conversión ... 34
Redondeo (REDONDO, PISO, CEIL) ... 34
Redondear un número decimal a un valor entero ... 34
Redondear un numero ... 34
Redondear hacia abajo un número ... 34
Redondear un número decimal a un número especificado de lugares decimales ... 35
Elevar un número a una potencia (POW) ... 35
Raíz cuadrada (SQRT) ... 35
Números aleatorios (RAND) ... 35
Generar un número aleatorio ... 35
Número aleatorio en un rango ... 35
Valor absoluto y signo (ABS, SIGNO) ... 36
Capítulo 9: Backticks ... 37
Examples ... 37
Uso de backticks ... 37
Capítulo 10: BORRAR ... 39
Sintaxis ... 39
Parámetros ... 39
Examples ... 39

Eliminar con la cláusula Where ... 39
Eliminar todas las filas de una tabla ... 40
LIMITAR eliminaciones ... 40
Eliminaciones de tablas múltiples ... 40
llaves extranjeras ... 41
Eliminación básica ... 42
DELETE vs TRUNCATE ... 42
Multi-mesa BORRAR ... 42
Capítulo 11: Búsqueda de texto completo ... 44
Introducción ... 44
Observaciones ... 44
Examples ... 44
Sencilla búsqueda en FULLTEXT ... 44
Búsqueda sencilla de BOOLEAN ... 44
Multi-columna de búsqueda en FULLTEXT ... 45
Capítulo 12: Cambia la contraseña ... 46
Examples ... 46
Cambiar la contraseña de root de MySQL en Linux ... 46
Cambiar la contraseña de root de MySQL en Windows ... 47
Proceso ... 47
Capítulo 13: Cliente MySQL ... 48
Sintaxis ... 48
Parámetros ... 48
Examples ... 48
Inicio de sesión base ... 48
Ejecutar comandos ... 49
Ejecutar comando desde una cadena ... 49
Ejecutar desde el archivo de script: ... 50
Escribe la salida en un archivo ... 50
Capítulo 14: Códigos de error ... 51
Examples ... 51
Código de error 1064: error de sintaxis ... 51

Código de error 1175: Actualización segura ... 51
Código de error 1215: No se puede agregar una restricción de clave externa ... 52
1045 Acceso denegado ... 53
1236 "posición imposible" en la replicación ... 53
2002, 2003 No se puede conectar ... 54
1067, 1292, 1366, 1411 - Valor incorrecto para el número, la fecha, el valor predeterminad ... 54
126, 127, 134, 144, 145 ... 54
139 ... 55
1366 ... 55
126, 1054, 1146, 1062, 24 ... 55
Capítulo 15: Comentar Mysql ... 57
Observaciones ... 57
Examples ... 57
Añadiendo comentarios ... 57
Comentar las definiciones de la tabla ... 57
Capítulo 16: Conectando con UTF-8 usando varios lenguajes de programación. ... 59
Examples ... 59
Pitón ... 59
PHP ... 59
Capítulo 17: Configuración de la conexión SSL ... 61
Examples ... 61
Configuración para sistemas basados en Debian ... 61
Generando una CA y claves SSL ... 61
Añadiendo las claves a MySQL ... 62
Probar la conexión SSL ... 62
Cumplimiento de SSL ... 63
Referencias y lecturas adicionales: ... 63
Configuración para CentOS7 / RHEL7 ... 63
Primero, inicie sesión en dbserver ... 64
TRABAJO LATERAL DEL FIN DEL SERVIDOR POR AHORA. ... 65
todavía en el cliente aquí ... 66

AHORA ESTAMOS LISTOS PARA PROBAR LA CONEXIÓN SEGURA 67
Todavía estamos en apuros aquí 67
Capítulo 18: Configuración y puesta a punto. 69
Observaciones 69
Examples 69
Rendimiento InnoDB 69
Parámetro para permitir la inserción de grandes datos. 69
Aumente el límite de cadena para group_concat 70
Configuración mínima de InnoDB 70
Cifrado seguro de MySQL 71
Capítulo 19: Conjuntos de caracteres y colaciones 72
Examples 72
Declaración 72
Conexión 72
¿Qué conjunto de personajes y colección? 72
Configuración de conjuntos de caracteres en tablas y campos 73
Capítulo 20: Consejos de rendimiento Mysql 74
Examples 74
Seleccione la optimización de la declaración 74
Optimización del diseño de almacenamiento para tablas InnoDB 74
Construyendo un índice compuesto 75
Capítulo 21: Consultas de pivote 77
Observaciones 77
Examples 77
Creando una consulta dinámica 77
Capítulo 22: Conversión de MyISAM a InnoDB 79
Examples 79
Conversión básica 79
Convertir todas las tablas en una base de datos 79
Capítulo 23: Copia de seguridad utilizando mysqldump 80
Sintaxis 80

Parámetros ... 80
Observaciones ... 81
Examples ... 81
Creación de una copia de seguridad de una base de datos o tabla ... 81
Especificando nombre de usuario y contraseña ... 82
Restaurar una copia de seguridad de una base de datos o tabla ... 82
mysqldump desde un servidor remoto con compresión ... 83
restaura un archivo mysqldump comprimido sin descomprimir ... 83
Copia de seguridad directa a Amazon S3 con compresión ... 83
Transferencia de datos de un servidor MySQL a otro ... 83
Base de datos de copia de seguridad con procedimientos almacenados y funciones ... 84
Capítulo 24: Creación de tablas ... 85
Sintaxis ... 85
Observaciones ... 85
Examples ... 85
Creación básica de tablas ... 85
Configuración de los valores predeterminados ... 86
Creación de tablas con clave primaria ... 86
Definiendo una columna como Clave Primaria (definición en línea) ... 87
Definir una clave primaria de varias columnas ... 87
Creación de tablas con clave externa ... 88
Clonando una tabla existente ... 89
CREAR TABLA DESDE SELECCIONAR ... 89
Mostrar estructura de tabla ... 90
Tabla Crear con la columna TimeStamp para mostrar la última actualización ... 90
Capítulo 25: Creando bases de datos ... 92
Sintaxis ... 92
Parámetros ... 92
Examples ... 92
Crear base de datos, usuarios y subvenciones ... 92
Mi base de datos ... 94

Bases de datos del sistema 95
Creando y Seleccionando una Base de Datos 95
Capítulo 26: Crear nuevo usuario 97
Observaciones 97
Examples 97
Crear un usuario de MySQL 97
Especifique la contraseña 97
Crear nuevo usuario y otorgar todos los privilegios al esquema 97
Renombrando usuario 98
Capítulo 27: Datos de carga infile 99
Sintaxis 99
Examples 99
usando LOAD DATA INFILE para cargar una gran cantidad de datos a la base de datos 99
Importar un archivo CSV en una tabla de MySQL 100
Cargar datos con duplicados 100
Datos de carga local 100
CARGAR DATOS INFILE 'fname' REEMPLAZAR 100
DATOS DE LA CARGA INFILE 'fname' IGNORE 101
Carga vía tabla intermedia 101
importación y exportación 101
Capítulo 28: ENUM 102
Examples 102
¿Por qué ENUM? 102
TINYINT como alternativa 102
VARCHAR como alternativa 103
Añadiendo una nueva opción 103
NULL vs NOT NULL 103
Capítulo 29: Error 1055: ONLY_FULL_GROUP_BY: algo no está en la cláusula GROUP BY ... 105
Introducción 105
Observaciones 105
Examples 106

Uso y mal uso de GROUP BY 106
Mal uso de GROUP BY para devolver resultados impredecibles: la ley de Murphy 106
Mal uso de GROUP BY con SELECT *, y cómo solucionarlo 107
ALGÚN VALOR() 108
Capítulo 30: Eventos 109
Examples 109
Crear un evento 109
Esquema para la prueba 109
Cree 2 eventos, 1º se ejecuta diariamente, 2º se ejecuta cada 10 minutos 109
Mostrar estados de eventos (diferentes enfoques) 110
Cosas al azar a considerar 111
Capítulo 31: Expresiones regulares 112
Introducción 112
Examples 112
REGEXP / RLIKE 112
Patrón ^ 112
Patrón $ ** 112
NO REGEXP 113
Regex contener 113
Cualquier caracter entre [] 113
Patrón o | 113
Contando coincidencias de expresiones regulares 113
Capítulo 32: Extraer valores de tipo JSON 115
Introducción 115
Sintaxis 115
Parámetros 115
Observaciones 115
Examples 115
Leer el valor de la matriz JSON 115
Operadores de extracto JSON 116
Capítulo 33: Gatillos 118

Sintaxis 118
Observaciones 118
POR CADA FILA 118
CREAR O REEMPLAZAR EL GATILLO 118
Examples 119
Disparador basico 119
Tipos de disparadores 119
Sincronización 119
Evento desencadenante 120
Antes de Insertar ejemplo de activador 120
Antes de actualizar el ejemplo de activación 120
Después de eliminar el ejemplo de activación 120
Capítulo 34: Índices y claves 122
Sintaxis 122
Observaciones 122
Conceptos 122
Examples 123
Crear índice 123
Crear un índice único 123
Índice de caída 123
Crear índice compuesto 123
Tecla AUTO_INCREMENT 123
Capítulo 35: información del servidor 125
Parámetros 125
Examples 125
MOSTRAR VARIABLES ejemplo 125
SHOW STATUS ejemplo 126
Capítulo 36: INSERTAR 127
Sintaxis 127
Observaciones 127
Examples 128

Inserto Básico ... 128
INSERTAR, ACTUALIZACIÓN CLAVE DUPLICADA ... 128
Insertando múltiples filas ... 128
Ignorando las filas existentes ... 129
INSERT SELECT (Insertando datos de otra tabla) ... 130
INSERTAR con AUTO_INCREMENT + LAST_INSERT_ID () ... 130
IDs AUTO_INCREMENT perdidos ... 132
Capítulo 37: Instalar el contenedor Mysql con Docker-Compose ... 134
Examples ... 134
Ejemplo simple con docker-compose ... 134
Capítulo 38: JSON ... 135
Introducción ... 135
Observaciones ... 135
Examples ... 135
Crear una tabla simple con una clave principal y un campo JSON ... 135
Insertar un simple JSON ... 135
Insertar datos mixtos en un campo JSON ... 135
Actualizando un campo JSON ... 136
Datos CAST a tipo JSON ... 136
Crear Json Object y Array ... 136
Capítulo 39: La optimización del rendimiento ... 138
Sintaxis ... 138
Observaciones ... 138
Examples ... 138
Agregue el índice correcto ... 138
Establecer el caché correctamente ... 139
Evitar construcciones ineficientes ... 139
Negativos ... 139
Tener un índice ... 139
No te escondas en función ... 140
O ... 140
Subconsultas ... 140

ÚNETE + GRUPO POR ... 141
Capítulo 40: Límite y compensación ... 142
Sintaxis ... 142
Observaciones ... 142
Examples ... 142
Relación de límite y compensación ... 142
Cláusula LIMIT con un argumento ... 142
Cláusula LIMIT con dos argumentos. ... 143
OFFSET palabra clave: sintaxis alternativa ... 144
Capítulo 41: Manejo de zonas horarias ... 145
Observaciones ... 145
Examples ... 145
Recupere la fecha y hora actual en una zona horaria particular ... 145
Convierte un valor `DATE` o` DATETIME` almacenado en otra zona horaria ... 145
Recupere los valores almacenados de `TIMESTAMP` en una zona horaria particular ... 146
¿Cuál es la configuración de zona horaria local de mi servidor? ... 146
¿Qué valores de time_zone están disponibles en mi servidor? ... 147
Capítulo 42: Mesa plegable ... 148
Sintaxis ... 148
Parámetros ... 148
Examples ... 148
Mesa plegable ... 148
Eliminar tablas de la base de datos ... 149
Capítulo 43: Mesas temporales ... 150
Examples ... 150
Crear tabla temporal ... 150
Drop Temporary Table ... 150
Capítulo 44: Motor myisam ... 152
Observaciones ... 152
Examples ... 152
MOTOR = MyISAM ... 152

Capítulo 45: MySQL LOCK TABLE 153
Sintaxis 153
Observaciones 153
Examples 153
Mysql Locks 153
Bloqueo de nivel de fila 154
Capítulo 46: MySQL Unions 157
Sintaxis 157
Observaciones 157
Examples 157
Operador sindical 157
Union all 158
UNION TODO CON DONDE 158
Capítulo 47: mysqlimport 160
Parámetros 160
Observaciones 160
Examples 160
Uso básico 160
Usando un delimitador de campo personalizado 161
Usando un delimitador de fila personalizado 161
Manejo de claves duplicadas 161
Importación condicional 162
Importar un csv estándar 162
Capítulo 48: NULO 163
Examples 163
Usos para NULL 163
Prueba de valores nulos 163
Capítulo 49: Operaciones de cuerdas 164
Parámetros 164
Examples 166
Encontrar elemento en la lista separada por comas 166
STR_TO_DATE - Convertir cadena a la fecha 167

LOWER () / LCASE () 167
REEMPLAZAR() 167
SUBSTRING () 167
SUPERIOR () / UCASE () 168
LONGITUD() 168
CHAR_LENGTH () 168
HEX (str) 168
Capítulo 50: Operaciones de fecha y hora 169
Examples 169
Ahora() 169
Aritmética de fecha 169
Pruebas contra un rango de fechas 170
SYSDATE (), NOW (), CURDATE () 170
Extraer la fecha de la fecha dada o la expresión de fecha y hora 170
Uso de un índice para una búsqueda de fecha y hora 170
Capítulo 51: ORDEN POR 172
Examples 172
Contextos 172
BASIC 172
Ascendiendo descendiendo 172
Algunos trucos 172
Capítulo 52: Palabras reservadas 174
Introducción 174
Observaciones 174
Examples 179
Errores debidos a palabras reservadas 179
Capítulo 53: Particionamiento 181
Observaciones 181
Examples 181
RANGO de particionamiento 181
Partición de la lista 182
Particionamiento HASH 183

Capítulo 54: Personalizar PS1 184
Examples 184
Personaliza el MySQL PS1 con la base de datos actual 184
PS1 personalizado a través del archivo de configuración de MySQL 184
Capítulo 55: Preparar declaraciones 185
Sintaxis 185
Examples 185
PREPARAR, EJECUTAR y DESALARCAR las declaraciones de PREPARACIÓN 185
Construir y ejecutar 185
Alterar tabla con añadir columna 186
Capítulo 56: Recuperar de la contraseña de root perdida 187
Examples 187
Establecer contraseña de root, habilitar usuario root para socket y acceso http 187
Capítulo 57: Recuperar y restablecer la contraseña de root predeterminada para MySQL 5.7+ 188
Introducción 188
Observaciones 188
Examples 188
¿Qué sucede cuando se inicia el servidor por primera vez? 188
Cómo cambiar la contraseña de root usando la contraseña predeterminada 188
restablecer la contraseña de root cuando "/ var / run / mysqld 'para el archivo socket UNI 189
Capítulo 58: Replicación 191
Observaciones 191
Examples 191
Maestro - Configuración de replicación de esclavos 191
Errores de replicación 194
Capítulo 59: Rutinas almacenadas (procedimientos y funciones) 196
Parámetros 196
Observaciones 196
Examples 196
Crear una función 196
Crear procedimiento con una preparación construida 197
Procedimiento almacenado con parámetros IN, OUT, INOUT 198

Cursores ... 199
Conjuntos de resultados múltiples ... 201
Crear una función ... 201
Capítulo 60: Se une ... 202
Sintaxis ... 202
Examples ... 202
Ejemplos de unión ... 202
ÚNETE con la subconsulta (tabla "Derivado") ... 202
Recuperar clientes con pedidos - variaciones en un tema ... 203
Unión externa completa ... 204
Unión interna para 3 mesas ... 205
Uniones visualizadas ... 206
Capítulo 61: Seguridad a través de GRANTs ... 208
Examples ... 208
Mejores prácticas ... 208
Host (del usuario @ host) ... 208
Capítulo 62: SELECCIONAR ... 210
Introducción ... 210
Sintaxis ... 210
Observaciones ... 210
Examples ... 210
SELECCIONAR por nombre de columna ... 210
SELECCIONAR todas las columnas (*) ... 211
SELECCIONA con DONDE ... 212
Consulta con un SELECT anidado en la cláusula WHERE ... 212
SELECCIONAR con LIKE (%) ... 212
SELECCIONAR con Alias (AS) ... 214
SELECT con una cláusula LIMIT ... 214
SELECCIONAR con DISTINTO ... 215
SELECCIONAR con LIKE (_) ... 216
SELECCIONAR con CASO o SI ... 216
SELECCIONAR CON ENTRE ... 217

SELECCIONAR con rango de fechas 218
Capítulo 63: Tabla de mapeo de muchos a muchos 219
Observaciones 219
Examples 219
Esquema tipico 219
Capítulo 64: Tabla dinámica de Un-Pivot usando una declaración preparada 220
Examples 220
Des-pivote un conjunto dinámico de columnas basado en condición 220
Capítulo 65: Tiempo con precisión subsecundaria. 223
Observaciones 223
Examples 223
Obtén la hora actual con milisegundos de precisión. 223
Obtenga la hora actual en un formulario que se parece a una marca de tiempo de Javascript 223
Crear una tabla con columnas para almacenar sub-segundo tiempo. 224
Convertir un valor de fecha / hora de precisión de milisegundos en texto. 224
Almacenar una marca de tiempo de Javascript en una columna TIMESTAMP 224
Capítulo 66: Tipos de datos 226
Examples 226
Fundición implícita / automática 226
VARCHAR (255) - o no 226
INT como AUTO_INCREMENT 227
Otros 227
Introducción (numérica) 228
Tipos enteros 228
Tipos de puntos fijos 229
Decimal 229
Tipos de punto flotante 229
Tipo de valor de bit 230
CHAR (n) 230
FECHA, DATETIME, TIMESTAMP, AÑO, Y HORA 230
Capítulo 67: Transacción 232
Examples 232

Iniciar Transacción ... 232
COMPROMISO, ROLLBACK y AUTOCOMMIT ... 233
Transacción utilizando el controlador JDBC ... 236
Capítulo 68: Tratar con datos escasos o faltantes ... 239
Examples ... 239
Trabajar con columnas que contienen valores NULL ... 239
Capítulo 69: UNE: Únete a la tabla 3 con el mismo nombre de ID. ... 242
Examples ... 242
Unir 3 tablas en una columna con el mismo nombre ... 242
Capítulo 70: UNIÓN ... 243
Sintaxis ... 243
Observaciones ... 243
Examples ... 243
Combinando sentencias SELECT con UNION ... 243
ORDEN POR ... 243
Paginación via OFFSET ... 244
Combinando datos con diferentes columnas. ... 244
UNION ALL Y UNION ... 244
Combinar y combinar datos en diferentes tablas de MySQL con las mismas columnas en filas ú ... 245
Capítulo 71: Uno a muchos ... 246
Introducción ... 246
Observaciones ... 246
Examples ... 246
Ejemplo de tablas de empresas ... 246
Haga que los empleados sean administrados por un solo gerente ... 247
Obtener el gerente para un solo empleado ... 247
Capítulo 72: Usando variables ... 248
Examples ... 248
Variables de configuración ... 248
Número de fila y grupo utilizando variables en la instrucción Select ... 249
Capítulo 73: VER ... 251
Sintaxis ... 251

Parámetros 251
Observaciones 251
Examples 252
Crear una vista 252
Una vista desde dos mesas 253
Actualización de una tabla a través de una vista 253
DROPPING A VIEW 253
Creditos 255

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: mysql

It is an unofficial and free MySQL ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official MySQL.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con MySQL

Observaciones

MySQL es un sistema de gestión de base de datos relacional de código abierto (RDBMS) desarrollado y soportado por Oracle Corporation.

MySQL es compatible con una gran cantidad de plataformas, incluidas las variantes de Linux, OS X y Windows. También tiene API para una gran cantidad de idiomas, incluidos C, C ++, Java, Lua, .Net, Perl, PHP, Python y Ruby.

MariaDB es una bifurcación de MySQL con un conjunto de características ligeramente diferente . Es totalmente compatible con MySQL para la mayoría de las aplicaciones.

Versiones

Versión	Fecha de lanzamiento
1.0	1995-05-23
3.19	1996-12-01
3.20	1997-01-01
3,21	1998-10-01
3,22	1999-10-01
3,23	2001-01-22
4.0	2003-03-01
4.1	2004-10-01
5.0	2005-10-01
5.1	2008-11-27
5.5	2010-11-01
5.6	2013-02-01

Versión	Fecha de lanzamiento
5.7	2015-10-01

Examples

Empezando

Creando una base de datos en MySQL

```
CREATE DATABASE mydb;
```

Valor de retorno:

> Consulta OK, 1 fila afectada (0.05 seg)

Usando la base de datos creada `mydb`

```
USE mydb;
```

Valor de retorno:

> Base de datos cambiada

Creando una tabla en MySQL

```
CREATE TABLE mytable
(
  id              int unsigned NOT NULL auto_increment,
  username        varchar(100) NOT NULL,
  email           varchar(100) NOT NULL,
  PRIMARY KEY     (id)
);
```

`CREATE TABLE mytable` creará una nueva tabla llamada `mytable` .

`id int unsigned NOT NULL auto_increment` crea la columna `id` , este tipo de campo asignará una ID numérica única a cada registro en la tabla (lo que significa que no hay dos filas que tengan la misma `id` en este caso), MySQL asignará automáticamente una nueva, valor único para el campo de `id` del registro (comenzando con 1).

Valor de retorno:

> Consulta OK, 0 filas afectadas (0.10 seg)

Insertar una fila en una tabla MySQL

```
INSERT INTO mytable ( username, email )
```

```
VALUES ( "myuser", "myuser@example.com" );
```

Ejemplo de valor de retorno:

> Consulta OK, 1 fila afectada (0.06 seg)

Las `strings varchar` aka también se pueden insertar utilizando comillas simples:

```
INSERT INTO mytable ( username, email )
VALUES ( 'username', 'username@example.com' );
```

Actualizar una fila en una tabla MySQL

```
UPDATE mytable SET username="myuser" WHERE id=8
```

Ejemplo de valor de retorno:

> Consulta OK, 1 fila afectada (0.06 seg)

El valor `int` se puede insertar en una consulta sin comillas. Las cadenas y las fechas deben estar entre comillas simples `'` o comillas dobles `"` .

Eliminar una fila en una tabla MySQL

```
DELETE FROM mytable WHERE id=8
```

Ejemplo de valor de retorno:

> Consulta OK, 1 fila afectada (0.06 seg)

Esto borrará la fila que tiene `id` es 8.

Seleccionando filas basadas en condiciones en MySQL

```
SELECT * FROM mytable WHERE username = "myuser";
```

Valor de retorno:

```
+----+----------+---------------------+
| id | username | email               |
+----+----------+---------------------+
|  1 | myuser   | myuser@example.com  |
+----+----------+---------------------+
```

> 1 fila en conjunto (0.00 seg)

Mostrar lista de bases de datos existentes

```
SHOW databases;
```

Valor de retorno:

```
+--------------------+
| Databases          |
+--------------------+
| information_schema|
| mydb               |
+--------------------+
```

2 filas en conjunto (0,00 seg)

Puede pensar en "esquema_información" como una "base de datos maestra" que proporciona acceso a los metadatos de la base de datos.

Mostrar tablas en una base de datos existente

```
SHOW tables;
```

Valor de retorno:

```
+----------------+
| Tables_in_mydb |
+----------------+
| mytable        |
+----------------+
```

1 fila en conjunto (0.00 seg)

Mostrar todos los campos de una tabla.

```
DESCRIBE databaseName.tableName;
```

o, si ya está utilizando una base de datos:

```
DESCRIBE tableName;
```

Valor de retorno:

```
+-----------+----------------+--------+---------+--------------------+-------+
| Field     | Type           | Null   | Key     | Default            | Extra |
+-----------+----------------+--------+---------+--------------------+-------+
| fieldname | fieldvaluetype | NO/YES | keytype | defaultfieldvalue |       |
+-----------+----------------+--------+---------+--------------------+-------+
```

`Extra` puede contener `auto_increment` por ejemplo.

`Key` refiere al tipo de clave que puede afectar el campo. Primario (PRI), Único (UNI) ...

    n fila en conjunto (0,00 seg)

Donde n es el número de campos en la tabla.

Creando usuario

Primero, debe crear un usuario y luego darle permisos de usuario en ciertas bases de datos / tablas. Al crear el usuario, también debe especificar desde dónde se puede conectar este usuario.

```
CREATE USER 'user'@'localhost' IDENTIFIED BY 'some_password';
```

Creará un usuario que solo puede conectarse en la máquina local donde está alojada la base de datos.

```
CREATE USER 'user'@'%' IDENTIFIED BY 'some_password';
```

Creará un usuario que puede conectarse desde cualquier lugar (excepto la máquina local).

Ejemplo de valor de retorno:

    Consulta OK, 0 filas afectadas (0.00 seg)

Añadiendo privilegios

Otorgue privilegios básicos comunes al usuario para todas las tablas de la base de datos especificada:

```
GRANT SELECT, INSERT, UPDATE ON databaseName.* TO 'userName'@'localhost';
```

Otorgue todos los privilegios al usuario para todas las tablas en todas las bases de datos (atención con esto):

```
GRANT ALL ON *.* TO 'userName'@'localhost' WITH GRANT OPTION;
```

Como se demostró anteriormente, `*.*` Apunta a todas las bases de datos y tablas, `databaseName.*` Apunta a todas las tablas de la base de datos específica. También es posible especificar la base de datos y la tabla como, por tanto, `databaseName.tableName` .

`WITH GRANT OPTION` debe `WITH GRANT OPTION` si el usuario no necesita poder otorgar privilegios a otros usuarios.

Los privilegios pueden ser **cualquiera**

```
ALL
```

o una combinación de los siguientes, cada uno separado por una coma (lista no exhaustiva).

```
SELECT
INSERT
UPDATE
DELETE
CREATE
DROP
```

Nota

En general, debe intentar evitar el uso de nombres de columnas o tablas que contengan espacios o usar palabras reservadas en SQL. Por ejemplo, es mejor evitar nombres como `table` o `first name` .

Si debe usar dichos nombres, póngalos entre los delimitadores `` tachar `` revés. Por ejemplo:

```
CREATE TABLE `table`
(
   `first name` VARCHAR(30)
);
```

Una consulta que contenga los delimitadores de marca regresiva en esta tabla podría ser:

```
 SELECT `first name` FROM `table` WHERE `first name` LIKE 'a%';
```

Ejemplos de esquemas de información

Lista de procesos

Esto mostrará todas las consultas activas y inactivas en ese orden y luego por cuánto tiempo.

```
SELECT * FROM information_schema.PROCESSLIST ORDER BY INFO DESC, TIME DESC;
```

Esto es un poco más detallado sobre los marcos de tiempo, ya que está en segundos de forma predeterminada

```
SELECT ID, USER, HOST, DB, COMMAND,
TIME as time_seconds,
ROUND(TIME / 60, 2) as time_minutes,
ROUND(TIME / 60 / 60, 2) as time_hours,
STATE, INFO
FROM information_schema.PROCESSLIST ORDER BY INFO DESC, TIME DESC;
```

Búsqueda de procedimientos almacenados

Busque fácilmente en todos los `Stored Procedures` palabras y los comodines.

```
SELECT * FROM information_schema.ROUTINES WHERE ROUTINE_DEFINITION LIKE '%word%';
```

Lea Empezando con MySQL en línea: https://riptutorial.com/es/mysql/topic/302/empezando-con-mysql

Capítulo 2: ACTUALIZAR

Sintaxis

- UPDATE [LOW_PRIORITY] [IGNORE] tableName SET column1 = expresión1, columna2 = expresión2, ... [DÓNDE condiciones]; // Actualización simple de una sola tabla
- UPDATE [LOW_PRIORITY] [IGNORE] tableName SET column1 = expresión1, columna2 = expresión2, ... [DÓNDE condiciones] [ORDEN POR expresión [ASC | DESC]] [LIMIT row_count]; // Actualizar con orden por y límite
- ACTUALIZAR [LOW_PRIORITY] [IGNORE] table1, table2, ... SET column1 = expresión1, column2 = expresión2, ... [DONDE condiciones]; // Actualización de tabla múltiple

Examples

Actualización básica

Actualizando una fila

```
UPDATE customers SET email='luke_smith@email.com' WHERE id=1
```

Esta consulta actualiza el contenido del `email` en la tabla de `customers` a la cadena `luke_smith@email.com` donde el valor de `id` es igual a 1. El contenido antiguo y nuevo de la tabla de la base de datos se ilustra a continuación a la izquierda y a la derecha, respectivamente:

customers

id	firstname	lastname	email
1	Luke	Smith	luke@example.com
2	Anna	Carey	anna@example.com
3	Todd	Winters	todd@example.com

customers

id	firstname	lastname	email
1	Luke	Smith	luke_smith@email.com
2	Anna	Carey	anna@example.com
3	Todd	Winters	todd@example.com

Actualizando todas las filas

```
UPDATE customers SET lastname='smith'
```

Esta consulta actualiza el contenido del `lastname` para cada entrada en la tabla de `customers` . Los contenidos antiguos y nuevos de la tabla de la base de datos se ilustran a continuación a la izquierda y a la derecha, respectivamente:

customers

id	firstname	lastname	email
1	Luke	Smith	luke@example.com
2	Anna	Carey	anna@example.com
3	Todd	Winters	todd@example.com

customers

id	firstname	lastname	email
1	Luke	Smith	luke@example.com
2	Anna	Smith	anna@example.com
3	Todd	Smith	todd@example.com

Aviso: Es necesario usar cláusulas condicionales (DONDE) en la consulta ACTUALIZAR. Si no utiliza ninguna cláusula condicional, se actualizarán todos los registros del atributo de esa tabla. En el ejemplo anterior, el nuevo valor (Smith) del apellido en la tabla de clientes se estableció en todas las filas.

Actualizar con un patrón de unión

Considere una tabla de producción llamada `questions_mysql` y una tabla `iwtQuestions` (tabla de trabajo importada) que representa el último lote de datos CSV importados de un `LOAD DATA INFILE` . La mesa de trabajo se trunca antes de la importación, los datos se importan y ese proceso no se muestra aquí.

Actualice nuestros datos de producción mediante una unión a nuestros datos de mesa de trabajo importados.

```
UPDATE questions_mysql q -- our real table for production
join iwtQuestions i -- imported worktable
ON i.qId = q.qId
SET q.closeVotes = i.closeVotes,
q.votes = i.votes,
q.answers = i.answers,
q.views = i.views;
```

Los alias `q` y `i` se utilizan para abreviar las referencias de la tabla. Esto facilita el desarrollo y la legibilidad.

`qId` , la clave principal, representa el ID de la pregunta de Stackoverflow. Se actualizan cuatro columnas para las filas coincidentes de la unión.

ACTUALIZAR CON ORDENAR Y LIMITAR

Si se especifica la cláusula `ORDER BY` en su instrucción SQL de actualización, las filas se actualizan en el orden especificado.

Si se especifica una cláusula `LIMIT` en su declaración SQL, eso coloca un límite en el número de filas que se pueden actualizar. No hay límite, si no se especifica la cláusula `LIMIT` .

`ORDER BY` y `LIMIT` no se pueden utilizar para la actualización de varias tablas.

La sintaxis de MySQL `UPDATE` con `ORDER BY` y `LIMIT` es,

```
UPDATE [ LOW_PRIORITY ] [ IGNORE ]
tableName
SET column1 = expression1,
```

```
    column2 = expression2,
    ...
[WHERE conditions]
[ORDER BY expression [ ASC | DESC ]]
[LIMIT row_count];

---> Example
UPDATE employees SET isConfirmed=1 ORDER BY joiningDate LIMIT 10
```

En el ejemplo anterior, se actualizarán 10 filas según el orden de los empleados que se `joiningDate` .

ACTUALIZACIÓN de tabla múltiple

En la `UPDATE` varias tablas, actualiza las filas en cada una de las tablas especificadas que satisfacen las condiciones. Cada fila coincidente se actualiza una vez, incluso si coincide con las condiciones varias veces.

En la tabla múltiple, `UPDATE` , `ORDER BY` y `LIMIT` no se pueden utilizar.

La sintaxis para tabla múltiple `UPDATE` es,

```
UPDATE [LOW_PRIORITY] [IGNORE]
table1, table2, ...
    SET column1 = expression1,
        column2 = expression2,
        ...
    [WHERE conditions]
```

Por ejemplo, considere dos tablas, `products` y `salesOrders` . En el caso, disminuimos la cantidad de un producto en particular del pedido de venta que ya se realizó. Entonces también necesitamos aumentar esa cantidad en nuestra columna de stock de la tabla de `products` . Esto se puede hacer en una sola instrucción de actualización de SQL como a continuación.

```
UPDATE products, salesOrders
  SET salesOrders.Quantity = salesOrders.Quantity - 5,
      products.availableStock = products.availableStock + 5
WHERE products.productId = salesOrders.productId
  AND salesOrders.orderId = 100 AND salesOrders.productId = 20;
```

En el ejemplo anterior, la cantidad '5' se reducirá de la tabla de `salesOrders` y la misma se incrementará en la tabla de `products` acuerdo con las condiciones `WHERE` .

Actualización masiva

Al actualizar varias filas con diferentes valores, es mucho más rápido utilizar una actualización masiva.

```
UPDATE people
SET name =
  (CASE id WHEN 1 THEN 'Karl'
           WHEN 2 THEN 'Tom'
```

```
            WHEN 3 THEN 'Mary'
    END)
WHERE id IN (1,2,3);
```

Mediante la actualización masiva, solo se puede enviar una consulta al servidor en lugar de una consulta para que se actualice cada fila. Los casos deben contener todos los parámetros posibles consultados en la cláusula `WHERE` .

Lea ACTUALIZAR en línea: https://riptutorial.com/es/mysql/topic/2738/actualizar

Capítulo 3: Administrador de MySQL

Examples

Cambiar contraseña de root

```
mysqladmin -u root -p'old-password' password 'new-password'
```

Eliminar base de datos

Útil para secuencias de comandos para eliminar todas las tablas y elimina la base de datos:

```
mysqladmin -u[username] -p[password] drop [database]
```

Utilizar con extrema precaución.

Para `DROP` base de datos como un script SQL (necesitará el privilegio DROP en esa base de datos):

```
DROP DATABASE database_name
```

o

```
DROP SCHEMA database_name
```

RENOMBRE Atómico y Recarga de Mesa

```
RENAME TABLE t TO t_old, t_copy TO t;
```

Ninguna otra sesión puede acceder a las tablas involucradas mientras se ejecuta RENAME TABLE, por lo que la operación de cambio de nombre no está sujeta a problemas de concurrencia.

Atomic Rename es especialmente para recargar completamente una tabla sin esperar a que `DELETE` y cargar para terminar:

```
CREATE TABLE new LIKE real;
load `new` by whatever means - LOAD DATA, INSERT, whatever
RENAME TABLE real TO old, new TO real;
DROP TABLE old;
```

Lea Administrador de MySQL en línea: https://riptutorial.com/es/mysql/topic/2991/administrador-de-mysql

Capítulo 4: Agrupación

Examples

Desambiguación

Desambiguación de "MySQL Cluster" ...

- NDB Cluster: un motor especializado, en su mayoría en memoria. No ampliamente utilizado.
- Galera Cluster también conocido como Percona XtraDB Cluster aka PXC aka MariaDB con Galera. - Una muy buena solución de alta disponibilidad para MySQL; va más allá de la replicación.

Vea las páginas individuales en esas variantes de "Cluster".

Para "índice agrupado" vea la (s) página (s) en la `PRIMARY KEY` .

Lea Agrupación en línea: https://riptutorial.com/es/mysql/topic/5130/agrupacion

Capítulo 5: Agrupar por

Sintaxis

1. SELECCIONAR expresión1, expresión2, ... expresión_n,
2. aggregate_function (expresión)
3. DE las tablas
4. [DÓNDE condiciones]
5. GRUPO POR expresión1, expresión2, ... expresión_n;

Parámetros

Parámetro	DETALLES
expresión1, expresión2, ... expresión_n	Las expresiones que no están encapsuladas dentro de una función agregada y deben incluirse en la cláusula GROUP BY.
función agregada	Una función como SUM, COUNT, MIN, MAX o AVG.
mesas	Las tablas de las que desea recuperar registros. Debe haber al menos una tabla en la cláusula FROM.
Donde las condiciones	Opcional. Las condiciones que deben cumplirse para que los registros sean seleccionados.

Observaciones

La cláusula MySQL GROUP BY se usa en una declaración SELECT para recopilar datos en varios registros y agrupar los resultados por una o más columnas.

Su comportamiento se rige en parte por el valor de la variable `ONLY_FULL_GROUP_BY` . Cuando esto está habilitado, las `SELECT` que se agrupan por cualquier columna que no esté en la salida devuelven un error. (Este es el valor predeterminado de 5.7.5). Tanto la configuración como la no configuración de esta variable pueden causar problemas para usuarios ingenuos o usuarios acostumbrados a otros DBMS.

Examples

Grupo usando la función SUMA

```
SELECT product, SUM(quantity) AS "Total quantity"
FROM order_details
GROUP BY product;
```

Grupo usando la función MIN

Supongamos una tabla de empleados en la que cada fila es un empleado que tiene un `name` , un `department` y un `salary` .

```
SELECT department, MIN(salary) AS "Lowest salary"
FROM employees
GROUP BY department;
```

Esto le diría qué departamento contiene el empleado con el salario más bajo y cuál es ese salario. Encontrar el `name` del empleado con el salario más bajo en cada departamento es un problema diferente, más allá del alcance de este Ejemplo. Consulte "groupwise max".

GRUPO UTILIZANDO COUNT FUNCION

```
SELECT department, COUNT(*) AS "Man_Power"
FROM employees
GROUP BY department;
```

GRUPO POR USO QUE TIENE

```
SELECT department, COUNT(*) AS "Man_Power"
FROM employees
GROUP BY department
HAVING COUNT(*) >= 10;
```

Usar `GROUP BY ... HAVING` para filtrar registros agregados es análogo a usar `SELECT ... WHERE` para filtrar registros individuales.

También podrías decir `HAVING Man_Power >= 10` ya que `HAVING` entiende "aliases".

Grupo utilizando Group Concat

Group Concat se usa en MySQL para obtener valores concatenados de expresiones con más de un resultado por columna. Es decir, hay muchas filas para volver a seleccionar para una columna como `Name(1):Score(*)`

Nombre	Puntuación
Adán	A +
Adán	UNA-
Adán	segundo
Adán	C +
Cuenta	RE-

Nombre	Puntuación
Juan	UNA-

```
SELECT Name, GROUP_CONCAT(Score ORDER BY Score desc SEPERATOR ' ') AS Grades
FROM   Grade
GROUP BY Name
```

Resultados:

```
+------+------------+
| Name | Grades     |
+------+------------+
| Adam | C+ B A- A+ |
| Bill | D-         |
| John | A-         |
+------+------------+
```

GROUP BY con funciones AGREGADAS

Tabla de pedidos

```
+---------+------------+----------+-------+--------+
| orderid | customerid | customer | total | items  |
+---------+------------+----------+-------+--------+
|       1 |          1 | Bob      |  1300 |     10 |
|       2 |          3 | Fred     |   500 |      2 |
|       3 |          5 | Tess     |  2500 |      8 |
|       4 |          1 | Bob      |   300 |      6 |
|       5 |          2 | Carly    |   800 |      3 |
|       6 |          2 | Carly    |  1000 |     12 |
|       7 |          3 | Fred     |   100 |      1 |
|       8 |          5 | Tess     | 11500 |     50 |
|       9 |          4 | Jenny    |   200 |      2 |
|      10 |          1 | Bob      |   500 |     15 |
+---------+------------+----------+-------+--------+
```

- **CONTAR**

Devuelve el **número de filas** que satisfacen un criterio específico en la cláusula `WHERE` .

Ej .: Número de pedidos para cada cliente.

```
SELECT customer, COUNT(*) as orders
FROM orders
GROUP BY customer
ORDER BY customer
```

Resultado:

```
+----------+--------+
| customer | orders |
+----------+--------+
| Bob      |      3 |
```

```
| Carly    |      2 |
| Fred     |      2 |
| Jenny    |      1 |
| Tess     |      2 |
+----------+--------+
```

- **SUMA**

Devuelve la **suma** de la columna seleccionada.

Ej .: Suma del total y artículos para cada cliente.

```
SELECT customer, SUM(total) as sum_total, SUM(items) as sum_items
FROM orders
GROUP BY customer
ORDER BY customer
```

Resultado:

```
+----------+-----------+-----------+
| customer | sum_total | sum_items |
+----------+-----------+-----------+
| Bob      |      2100 |        31 |
| Carly    |      1800 |        15 |
| Fred     |       600 |         3 |
| Jenny    |       200 |         2 |
| Tess     |     14000 |        58 |
+----------+-----------+-----------+
```

- **AVG**

Devuelve el valor **promedio** de una columna de valor numérico.

Ej .: Valor medio del pedido para cada cliente.

```
SELECT customer, AVG(total) as avg_total
FROM orders
GROUP BY customer
ORDER BY customer
```

Resultado:

```
+----------+-----------+
| customer | avg_total |
+----------+-----------+
| Bob      |       700 |
| Carly    |       900 |
| Fred     |       300 |
| Jenny    |       200 |
| Tess     |      7000 |
+----------+-----------+
```

- **MAX**

Devuelve el valor **más alto** de una determinada columna o expresión.

Por ejemplo: el mayor total de pedidos para cada cliente.

```
SELECT customer, MAX(total) as max_total
FROM orders
GROUP BY customer
ORDER BY customer
```

Resultado:

```
+----------+-----------+
| customer | max_total |
+----------+-----------+
| Bob      |      1300 |
| Carly    |      1000 |
| Fred     |       500 |
| Jenny    |       200 |
| Tess     |     11500 |
+----------+-----------+
```

- **MIN**

Devuelve el valor **más** bajo de una determinada columna o expresión.

Ej .: El pedido más bajo para cada cliente.

```
SELECT customer, MIN(total) as min_total
FROM orders
GROUP BY customer
ORDER BY customer
```

Resultado:

```
+----------+-----------+
| customer | min_total |
+----------+-----------+
| Bob      |       300 |
| Carly    |       800 |
| Fred     |       100 |
| Jenny    |       200 |
| Tess     |      2500 |
+----------+-----------+
```

Lea Agrupar por en línea: https://riptutorial.com/es/mysql/topic/3523/agrupar-por

Capítulo 6: ALTERAR MESA

Sintaxis

- ALTER [IGNORE] TABLE tbl_name [**alter_specification** [, alter_specification] ...] [partition_options]

Observaciones

```
    alter_specification: table_options
      | ADD [COLUMN] col_name column_definition [FIRST | AFTER col_name ]
      | ADD [COLUMN] (col_name column_definition,...)
      | ADD {INDEX|KEY} [index_name] [index_type] (index_col_name,...) [index_option] ...
      | ADD [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...) [index_option]
...
      | ADD [CONSTRAINT [symbol]] UNIQUE [INDEX|KEY] [index_name] [index_type]
(index_col_name,...) [index_option] ...
      | ADD FULLTEXT [INDEX|KEY] [index_name] (index_col_name,...) [index_option] ...
      | ADD SPATIAL [INDEX|KEY] [index_name] (index_col_name,...) [index_option] ...
      | ADD [CONSTRAINT [symbol]] FOREIGN KEY [index_name] (index_col_name,...)
reference_definition
      | ALGORITHM [=] {DEFAULT|INPLACE|COPY}
      | ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
      | CHANGE [COLUMN] old_col_name new_col_name column_definition [FIRST|AFTER col_name]
      | LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
      | MODIFY [COLUMN] col_name column_definition [FIRST | AFTER col_name]
      | DROP [COLUMN] col_name
      | DROP PRIMARY KEY
      | DROP {INDEX|KEY} index_name
      | DROP FOREIGN KEY fk_symbol
      | DISABLE KEYS
      | ENABLE KEYS
      | RENAME [TO|AS] new_tbl_name
      | RENAME {INDEX|KEY} old_index_name TO new_index_name
      | ORDER BY col_name [, col_name] ...
      | CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
      | [DEFAULT] CHARACTER SET [=] charset_name [COLLATE [=] collation_name]
      | DISCARD TABLESPACE
      | IMPORT TABLESPACE
      | FORCE
      | {WITHOUT|WITH} VALIDATION
      | ADD PARTITION (partition_definition)
      | DROP PARTITION partition_names
      | DISCARD PARTITION {partition_names | ALL} TABLESPACE
      | IMPORT PARTITION {partition_names | ALL} TABLESPACE
      | TRUNCATE PARTITION {partition_names | ALL}
      | COALESCE PARTITION number
      | REORGANIZE PARTITION partition_names INTO (partition_definitions)
      | EXCHANGE PARTITION partition_name WITH TABLE tbl_name [{WITH|WITHOUT} VALIDATION]
      | ANALYZE PARTITION {partition_names | ALL}
      | CHECK PARTITION {partition_names | ALL}
      | OPTIMIZE PARTITION {partition_names | ALL}
      | REBUILD PARTITION {partition_names | ALL}
      | REPAIR PARTITION {partition_names | ALL}
      | REMOVE PARTITIONING
```

```
      | UPGRADE PARTITIONING
   index_col_name: col_name [(length)] [ASC | DESC]
   index_type: USING {BTREE | HASH}
   index_option: KEY_BLOCK_SIZE [=] value
      | index_type
      | WITH PARSER parser_name
      | COMMENT 'string'
```

`table_options: table_option [[,] table_option] ... (see options)` CREAR TABLA `options)`

`partition_options: (see options)` CREAR TABLA `options)`

Ref: MySQL 5.7 Manual de referencia / ... / ALTER TABLE Sintaxis / 14.1.8 ALTER TABLE Sintaxis

Examples

Cambio de motor de almacenamiento; tabla de reconstrucción cambiar file_per_table

Por ejemplo, si `t1` es actualmente una tabla InnoDB, esta declaración cambia su motor de almacenamiento a InnoDB:

```
ALTER TABLE t1 ENGINE = InnoDB;
```

Si la tabla ya es InnoDB, esto reconstruirá la tabla y sus índices y tendrá un efecto similar a `OPTIMIZE TABLE` . Puede ganar algo de mejora de espacio en disco.

Si el valor de `innodb_file_per_table` es actualmente diferente del valor en vigencia cuando se construyó `t1` , esto se convertirá a (o desde) file_per_table.

ALTER COLUMNA DE MESA

```
CREATE DATABASE stackoverflow;

USE stackoverflow;

Create table stack(
    id_user int NOT NULL,
    username varchar(30) NOT NULL,
    password varchar(30) NOT NULL
);

ALTER TABLE stack ADD COLUMN submit date NOT NULL; -- add new column
ALTER TABLE stack DROP COLUMN submit; -- drop column
ALTER TABLE stack MODIFY submit DATETIME NOT NULL; -- modify type column
ALTER TABLE stack CHANGE submit submit_date DATETIME NOT NULL; -- change type and name of
column
ALTER TABLE stack ADD COLUMN mod_id INT NOT NULL AFTER id_user; -- add new column after
existing column
```

Tabla ALTER añadir INDEX

Para mejorar el rendimiento, es posible que desee agregar índices a las columnas

```
ALTER TABLE TABLE_NAME ADD INDEX `index_name` (`column_name`)
```

Alterar para agregar índices compuestos (múltiples columnas)

```
ALTER TABLE TABLE_NAME ADD INDEX `index_name` (`col1`,`col2`)
```

Cambiar el valor de incremento automático

Cambiar un valor de incremento automático es útil cuando no desea un espacio en una columna AUTO_INCREMENT después de una eliminación masiva.

Por ejemplo, ha publicado muchas filas (publicitarias) no deseadas en su tabla, las eliminó y desea corregir la brecha en los valores de incremento automático. Suponga que el valor MAX de la columna AUTO_INCREMENT es 100 ahora. Puede usar lo siguiente para arreglar el valor de incremento automático.

```
ALTER TABLE your_table_name AUTO_INCREMENT = 101;
```

Cambiar el tipo de una columna de clave primaria

```
ALTER TABLE fish_data.fish DROP PRIMARY KEY;
ALTER TABLE fish_data.fish MODIFY COLUMN fish_id DECIMAL(20,0) NOT NULL PRIMARY KEY;
```

Un intento de modificar el tipo de esta columna sin eliminar primero la clave principal daría como resultado un error.

Cambiar definición de columna

El cambio de la definición de una columna db, la consulta siguiente se puede usar, por ejemplo, si tenemos este esquema db

```
users (
    firstname varchar(20),
    lastname varchar(20),
    age char(2)
)
```

Para cambiar el tipo de columna de `age` de `char` a `int` , usamos la siguiente consulta:

```
ALTER TABLE users CHANGE age age tinyint UNSIGNED NOT NULL;
```

El formato general es:

```
ALTER TABLE table_name CHANGE column_name new_column_definition
```

Renombrando una base de datos MySQL

No hay un solo comando para cambiar el nombre de una base de datos MySQL, pero se puede usar una solución alternativa simple para lograr esto haciendo una copia de seguridad y restaurando:

```
mysqladmin -uroot -p<password> create <new name>
mysqldump -uroot -p<password> --routines <old name> | mysql -uroot -pmypassword <new name>
mysqladmin -uroot -p<password> drop <old name>
```

Pasos:

1. Copia las líneas de arriba en un editor de texto.
2. Reemplace todas las referencias a `<old name>` , `<new name>` y `<password>` (+ opcionalmente `root` para usar un usuario diferente) con los valores relevantes.
3. Ejecute uno por uno en la línea de comandos (suponiendo que la carpeta "bin" de MySQL esté en la ruta e ingrese "y" cuando se le solicite).

Pasos alternativos:

Renombra (mueve) cada tabla de una db a la otra. Haga esto para cada mesa:

```
RENAME TABLE `<old db>`.`<name>` TO  `<new db>`.`<name>`;
```

Puedes crear esas declaraciones haciendo algo como

```
SELECT CONCAT('RENAME TABLE old_db.', table_name, ' TO ',
                          'new_db.', table_name)
    FROM information_schema.TABLES
    WHERE table_schema = 'old_db';
```

Advertencia. No intente hacer ningún tipo de tabla o base de datos simplemente moviendo los archivos en el sistema de archivos. Esto funcionó bien en los viejos tiempos de solo MyISAM, pero en los nuevos días de InnoDB y tablespaces, no funcionará. Especialmente cuando el "Diccionario de datos" se mueve desde el sistema de archivos a las tablas InnoDB del sistema, probablemente en la próxima versión principal. En movimiento (en lugar de sólo `DROPping`) una `PARTITION` de una tabla InnoDB requiere el uso de tablas "transportables". En un futuro cercano, ni siquiera habrá un archivo que alcanzar.

Intercambiando los nombres de dos bases de datos MySQL

Los siguientes comandos se pueden usar para intercambiar los nombres de dos bases de datos MySQL (`<db1>` y `<db2>`):

```
mysqladmin -uroot -p<password> create swaptemp
mysqldump -uroot -p<password> --routines <db1> | mysql -uroot -p<password> swaptemp
mysqladmin -uroot -p<password> drop <db1>
```

```
mysqladmin -uroot -p<password> create <db1>
mysqldump -uroot -p<password> --routines <db2> | mysql -uroot -p<password> <db1>
mysqladmin -uroot -p<password> drop <db2>
mysqladmin -uroot -p<password> create <db2>
mysqldump -uroot -p<password> --routines swaptemp | mysql -uroot -p<password> <db2>
mysqladmin -uroot -p<password> drop swaptemp
```

Pasos:

1. Copia las líneas de arriba en un editor de texto.
2. Reemplace todas las referencias a `<db1>` , `<db2>` y `<password>` (+ opcionalmente `root` para usar un usuario diferente) con los valores relevantes.
3. Ejecute uno por uno en la línea de comandos (suponiendo que la carpeta "bin" de MySQL esté en la ruta e ingrese "y" cuando se le solicite).

Renombrando una tabla MySQL

El cambio de nombre de una tabla se puede hacer en un solo comando:

```
RENAME TABLE `<old name>` TO `<new name>`;
```

La siguiente sintaxis hace exactamente lo mismo:

```
ALTER TABLE `<old name>` RENAME TO `<new name>`;
```

Si se cambia el nombre de una tabla temporal, se debe usar la versión `ALTER TABLE` de la sintaxis.

Pasos:

1. Reemplace `<old name>` y `<new name>` en la línea de arriba con los valores relevantes. *Nota: Si la tabla se está moviendo a una base de datos diferente, el `dbname.tablename` sintaxis de nombre de `tablename` se puede usar para `<old name>` y / o `<new name>` .*
2. Ejecútelo en la base de datos correspondiente en la línea de comandos de MySQL o en un cliente como MySQL Workbench. *Nota: El usuario debe tener privilegios ALTER y DROP en la tabla anterior y CREATE e INSERT en la nueva.*

Renombrando una columna en una tabla MySQL

El cambio de nombre de una columna se puede hacer en una sola declaración, pero además del nuevo nombre, también se debe especificar la "definición de columna" (es decir, su tipo de datos y otras propiedades opcionales como nulabilidad, incremento automático, etc.).

```
ALTER TABLE `<table name>` CHANGE `<old name>` `<new name>` <column definition>;
```

Pasos:

1. Abra la línea de comandos de MySQL o un cliente como MySQL Workbench.
2. Ejecute la siguiente instrucción: `SHOW CREATE TABLE <table name>;` (reemplazando `<table name>` con el valor relevante).

3. Tome nota de la definición de la columna completa para la columna que se va a renombrar *(es decir, todo lo que aparece después del nombre de la columna pero antes de la coma que lo separa del nombre de la siguiente columna)* .
4. Reemplace `<old name>` , `<new name>` y `<column definition>` en la línea anterior con los valores relevantes y luego ejecútelo.

Lea ALTERAR MESA en línea: https://riptutorial.com/es/mysql/topic/2627/alterar-mesa

Capítulo 7: Archivos de registro

Examples

Una lista

- Registro general - todas las consultas - ver general_log VARIABLE
- Registro lento - consultas más lentas que long_query_time - slow_query_log_file
- Binlog - para replicación y copia de seguridad - log_bin_basename
- Registro de retransmisión - también para replicación
- errores generales - mysqld.err
- start / stop - mysql.log (no muy interesante) - log_error
- InnoDB rehacer registro - iblog *

Vea las variables `basedir` y `datadir` para la ubicación predeterminada para muchos registros

Algunos registros son activados / desactivados por otras VARIABLES. Algunos están escritos en un archivo o en una tabla.

(Nota para los revisores: Esto necesita más detalles y más explicación.)

Documentadores : incluya la ubicación y el nombre predeterminados para cada tipo de registro, tanto para Windows como para * nix. (O al menos tanto como puedas).

Registro de consultas lentas

El registro de consultas lentas consta de eventos de registro para consultas que `long_query_time` hasta segundos_tiempo_query en finalizar. Por ejemplo, hasta 10 segundos para completar. Para ver el umbral de tiempo establecido actualmente, emita lo siguiente:

```
SELECT @@long_query_time;
+-------------------+
| @@long_query_time |
+-------------------+
|         10.000000 |
+-------------------+
```

Se puede establecer como una variable GLOBAL, en el archivo `my.cnf` o `my.ini` . O puede establecerse por la conexión, aunque esto es inusual. El valor se puede configurar entre 0 y 10 (segundos). ¿Qué valor usar?

- 10 es tan alto que es casi inútil;
- 2 es un compromiso;
- 0.5 y otras fracciones son posibles;
- 0 captura todo; Esto podría llenar el disco peligrosamente rápido, pero puede ser muy útil.

La captura de consultas lentas está activada o desactivada. Y el archivo registrado también se

especifica. Lo siguiente captura estos conceptos:

```
SELECT @@slow_query_log; -- Is capture currently active? (1=On, 0=Off)
SELECT @@slow_query_log_file; -- filename for capture. Resides in datadir
SELECT @@datadir; -- to see current value of the location for capture file

SET GLOBAL slow_query_log=0; -- Turn Off
-- make a backup of the Slow Query Log capture file. Then delete it.
SET GLOBAL slow_query_log=1; -- Turn it back On (new empty file is created)
```

Para obtener más información, consulte la página del manual de MySQL El registro de consultas lentas

Nota: La información anterior sobre cómo activar / desactivar el slowlog se cambió en 5.6 (?); La versión anterior tenía otro mecanismo.

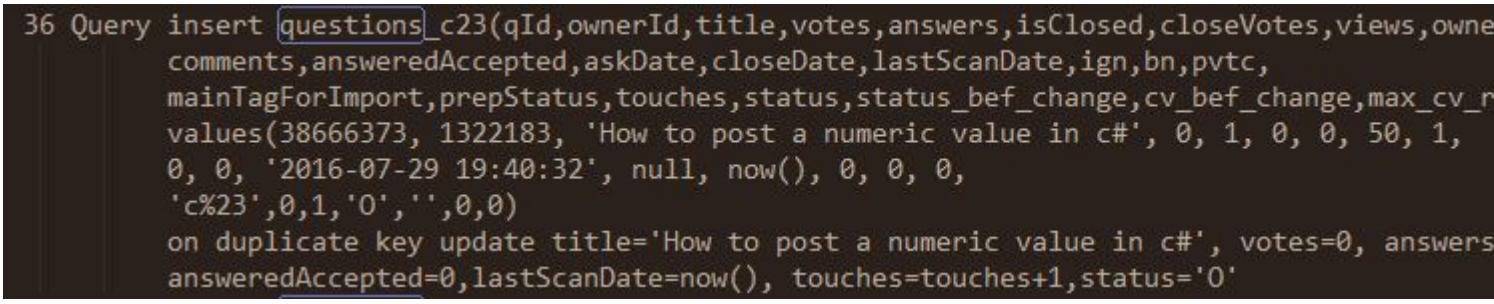
La "mejor" manera de ver lo que está ralentizando su sistema:

```
long_query_time=...
turn on the slowlog
run for a few hours
turn off the slowlog (or raise the cutoff)
run pt-query-digest to find the 'worst' couple of queries.  Or mysqldumpslow -s t
```

Registro de consultas generales

El Registro de consultas generales contiene una lista de información general de conexiones, desconexiones y consultas de clientes. Es invaluable para la depuración, sin embargo, representa un obstáculo para el rendimiento (¿citación?).

A continuación se muestra una vista de ejemplo de un registro de consultas generales:



Para determinar si el Registro general se está capturando actualmente:

```
SELECT @@general_log; -- 1 = Capture is active; 0 = It is not.
```

Para determinar el nombre del archivo de captura:

```
SELECT @@general_log_file; -- Full path to capture file
```

Si no se muestra la ruta completa al archivo, el archivo existe en el `datadir` .

Ejemplo de Windows:

```
+-----------------------------------------------------------+
| @@general_log_file                                        |
+-----------------------------------------------------------+
| C:\ProgramData\MySQL\MySQL Server 5.7\Data\GuySmiley.log |
+-----------------------------------------------------------+
```

Linux:

```
+----------------------------------+
| @@general_log_file               |
+----------------------------------+
| /var/lib/mysql/ip-ww-xx-yy-zz.log |
+----------------------------------+
```

Cuando se realizan cambios en la variable GLOBAL `general_log_file` , el nuevo registro se guarda en el `datadir` . Sin embargo, la ruta completa ya no puede reflejarse al examinar la variable.

En el caso de que no haya ninguna entrada para `general_log_file` en el archivo de configuración, se usará por defecto `@@hostname` .log en el `datadir` .

Las mejores prácticas son desactivar la captura. Guarde el archivo de registro en un directorio de respaldo con un nombre de archivo que refleje la fecha / hora de inicio / finalización de la captura. Eliminar el archivo anterior si no se produjo un *movimiento del* sistema de archivos de ese archivo. Establezca un nuevo nombre de archivo para el archivo de registro y active la captura (todo se muestra a continuación). Las mejores prácticas también incluyen una determinación cuidadosa si incluso desea capturar en este momento. Normalmente, la captura está activada solo con fines de depuración.

Un nombre de archivo de sistema de archivos típico para un registro de copia de seguridad podría ser:

```
/LogBackup/GeneralLog_20160802_1520_to_20160802_1815.log
```

donde la fecha y la hora son parte del nombre de archivo como un rango.

Para Windows, tenga en cuenta la siguiente secuencia con los cambios de configuración.

```
SELECT @@general_log; -- 0. Not being captured
SELECT @@general_log_file; -- C:\ProgramData\MySQL\MySQL Server 5.6\Data\GuySmiley.log
SELECT @@datadir; -- C:\ProgramData\MySQL\MySQL Server 5.7\Data\
SET GLOBAL general_log_file='GeneralLogBegin_20160803_1420.log'; -- datetime clue
SET GLOBAL general_log=1; -- Turns on actual log capture. File is created under `datadir`
SET GLOBAL general_log=0; -- Turn logging off
```

Linux es similar. Estos representarían cambios dinámicos. Cualquier reinicio del servidor recogerá la configuración del archivo de configuración.

En cuanto al archivo de configuración, considere las siguientes configuraciones de variables relevantes:

```
[mysqld]
general_log_file = /path/to/currentquery.log
general_log      = 1
```

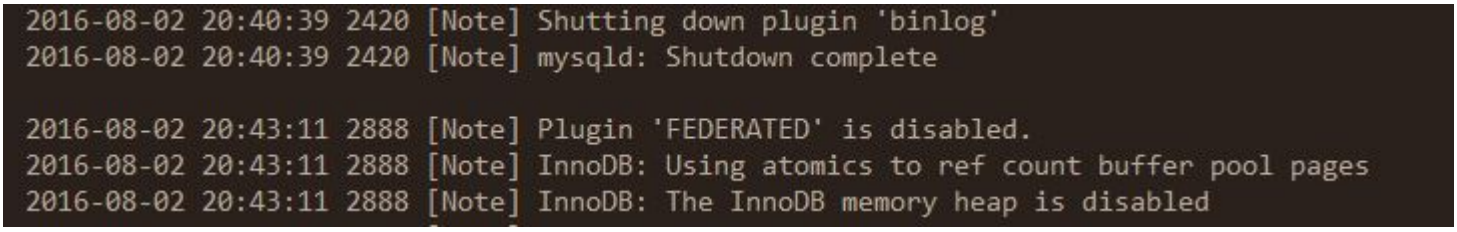
Además, la variable `log_output` se puede configurar para la salida de `TABLE` , no solo `FILE` . Para eso, por favor vea Destinos .

Por favor vea la página del manual de MySQL El Registro de consultas generales .

Registro de errores

El registro de errores se completa con información de inicio y detención, y eventos críticos encontrados por el servidor.

El siguiente es un ejemplo de su contenido:



La variable `log_error` contiene la ruta al archivo de registro para el registro de errores.

En ausencia de una entrada de archivo de configuración para `log_error` , el sistema predeterminará sus valores a `@@hostname` .err en el `datadir` . Tenga en cuenta que `log_error` no es una variable dinámica. Como tales, los cambios se realizan a través de los cambios de archivo cnf o ini y el reinicio del servidor (o al ver "Descarga y cambio de nombre del archivo de registro de errores" en el enlace de la página de manual en la parte inferior aquí).

El registro no puede ser deshabilitado por errores. Son importantes para la salud del sistema, mientras que la solución de problemas. Además, las entradas son poco frecuentes en comparación con el Registro de consultas generales.

La variable GLOBAL `log_warnings` establece el nivel de verbosidad que varía según la versión del servidor. El siguiente fragmento ilustra:

```
SELECT @@log_warnings; -- make a note of your prior setting
SET GLOBAL log_warnings=2; -- setting above 1 increases output (see server version)
```

`log_warnings` como se ve arriba es una variable dinámica.

Los cambios en el archivo de configuración en los archivos `cnf` e `ini` pueden parecerse a los siguientes.

```
[mysqld]
log_error        = /path/to/CurrentError.log
log_warnings     = 2
```

MySQL 5.7.2 expandió la verbosidad del nivel de advertencia a 3 y agregó el `log_error_verbosity` GLOBAL. De nuevo, se introdujo en 5.7.2. Puede configurarse dinámicamente y verificarse como una variable o establecerse a través de `cnf` o `ini` configuración de archivos de configuración.

A partir de MySQL 5.7.2:

```
[mysqld]
log_error           = /path/to/CurrentError.log
log_warnings        = 2
log_error_verbosity = 3
```

Consulte la página del manual de MySQL titulada El registro de errores, especialmente para vaciar y cambiar el nombre del archivo de registro de errores, y su sección `log_warnings` *registro de errores* con versiones relacionadas con `log_warnings` y `error_log_verbosity` .

Lea Archivos de registro en línea: https://riptutorial.com/es/mysql/topic/5102/archivos-de-registro

Capítulo 8: Aritmética

Observaciones

MySQL, en la mayoría de las máquinas, utiliza la aritmética de punto flotante IEEE 754 de 64 bits para sus cálculos.

En contextos enteros utiliza aritmética de enteros.

- `RAND()` no es un generador de números aleatorios perfecto. Se utiliza principalmente para generar rápidamente números pseudoaleatorios.

Examples

Operadores aritméticos

MySQL proporciona los siguientes operadores aritméticos

Operador	Nombre	Ejemplo
`+`	Adición	`SELECT 3+5;` -> 8 `SELECT 3.5+2.5;` -> 6.0 `SELECT 3.5+2;` -> 5.5
`-`	Sustracción	`SELECT 3-5;` -> -2
`*`	Multiplicación	`SELECT 3 * 5;` -> 15
`/`	División	`SELECT 20 / 4;` -> 5 `SELECT 355 / 113;` -> 3.1416 `SELECT 10.0 / 0;` -> NULL
`DIV`	División entera	`SELECT 5 DIV 2;` -> 2
`%` o `MOD`	Modulo	`SELECT 7 % 3;` -> 1 `SELECT 15 MOD 4` -> 3 `SELECT 15 MOD -4` -> 3 `SELECT -15 MOD 4` -> -3 `SELECT -15 MOD -4` -> -3 `SELECT 3 MOD 2.5` -> 0.5

BIGINT

Si los números en su aritmética son todos enteros, MySQL usa el tipo de datos entero `BIGINT` (con signo de 64 bits) para hacer su trabajo. Por ejemplo:

`select (1024 * 1024 * 1024 * 1024 *1024 * 1024) + 1` -> 1,152,921,504,606,846,977

y

`select (1024 * 1024 * 1024 * 1024 *1024 * 1024 * 1024` -> `BIGINT` error fuera de rango

DOBLE

Si algún número en su aritmética es fraccional, MySQL usa aritmética de punto flotante IEEE 754 de 64 bits . Debe tener cuidado al usar la aritmética de punto flotante, porque muchos números de punto flotante son, inherentemente, aproximaciones en lugar de valores exactos .

Constantes matemáticas

Pi

Lo siguiente devuelve el valor de `PI` formateado a 6 lugares decimales. El valor real es bueno para `DOUBLE` ;

```
SELECT PI();    -> 3.141593
```

Trigonometría (SIN, COS)

Los ángulos están en radianes, no en grados. Todos los cálculos se realizan en punto flotante de 64 bits IEEE 754 . Todos los cálculos de punto flotante están sujetos a pequeños errores, conocidos como errores de la máquina ε (épsilon) , así que evite intentar compararlos para la igualdad. No hay forma de evitar estos errores cuando se utiliza un punto flotante; Están incorporados a la tecnología.

Si usa valores `DECIMAL` en los cálculos trigonométricos, se convierten implícitamente a punto flotante y luego regresan a decimal.

Seno

Devuelve el seno de un número X expresado en radianes

```
SELECT SIN(PI()); -> 1.2246063538224e-16
```

Coseno

Devuelve el coseno de X cuando X se da en radianes

```
SELECT COS(PI()); -> -1
```

Tangente

Devuelve la tangente de un número X expresado en radianes. Observe que el resultado es muy cercano a cero, pero no exactamente a cero. Este es un ejemplo de máquina ε.

```
SELECT TAN(PI());   -> -1.2246063538224e-16
```

Arco coseno (coseno inverso)

Devuelve el coseno del arco de X si X está en el rango de `-1 to 1`

```
SELECT ACOS(1);    -> 0
SELECT ACOS(1.01); -> NULL
```

Seno del arco (seno siniestro)

Devuelve el seno de arco de X si X está en el rango de `-1 to 1`

```
SELECT ASIN(0.2); -> 0.20135792079033
```

Arco tangente (tangente inverso)

`ATAN(x)` devuelve la tangente de arco de un solo número.

```
SELECT ATAN(2); -> 1.1071487177941
```

`ATAN2(X, Y)` devuelve el arco tangente de las dos variables X e Y. Es similar al cálculo del arco tangente de Y / X. Pero es numéricamente más robusto: t funciona correctamente cuando X está cerca de cero y los signos de ambos argumentos se utilizan para determinar el cuadrante del resultado.

Las mejores prácticas sugieren escribir fórmulas para usar `ATAN2()` lugar de `ATAN()` siempre que sea posible.

```
 ATAN2(1,1);    -> 0.7853981633974483 (45 degrees)
 ATAN2(1,-1);   -> 2.356194490192345  (135 degrees)
 ATAN2(0, -1);  -> PI  (180 degrees)  don't try ATAN(-1 / 0)... it won't work
```

Cotangente

Devuelve la cotangente de X

```
SELECT COT(12); -> -1.5726734063977
```

Conversión

```
SELECT RADIANS(90) -> 1.5707963267948966
SELECT SIN(RADIANS(90)) -> 1
SELECT DEGREES(1), DEGREES(PI()) -> 57.29577951308232, 180
```

Redondeo (REDONDO, PISO, CEIL)

Redondear un número decimal a un valor entero

Para valores numéricos exactos (p. Ej., `DECIMAL`): si el primer lugar decimal de un número es 5 o más, esta función redondeará un número al siguiente entero que se encuentre *alejado de cero* . Si la posición decimal es 4 o inferior, esta función se redondeará al siguiente valor entero *más cercano a cero* .

```
SELECT ROUND(4.51) -> 5
SELECT ROUND(4.49) -> 4
SELECT ROUND(-4.51) -> -5
```

Para valores numéricos aproximados (p. Ej., `DOUBLE`): El resultado de la función `ROUND()` depende de la biblioteca C; en muchos sistemas, esto significa que `ROUND()` utiliza la *ronda a la* regla *par más cercana* :

```
SELECT ROUND(45e-1) -> 4  -- The nearest even value is 4
SELECT ROUND(55e-1) -> 6  -- The nearest even value is 6
```

Redondear un numero

Para redondear un número, use la función `CEIL()` o `CEILING()`

```
SELECT CEIL(1.23)    -> 2
SELECT CEILING(4.83) -> 5
```

Redondear hacia abajo un número

Para redondear un número, use la función `FLOOR()`

```
SELECT FLOOR(1.99) -> 1
```

PISO y CEIL van hacia / lejos del infinito:

```
SELECT FLOOR(-1.01), CEIL(-1.01) -> -2 and -1
SELECT FLOOR(-1.99), CEIL(-1.99) -> -2 and -1
```

Redondear un número decimal a un número especificado de lugares decimales.

```
SELECT ROUND(1234.987, 2) -> 1234.99
SELECT ROUND(1234.987, -2) -> 1200
```

La discusión de arriba contra abajo y "5" también se aplica.

Elevar un número a una potencia (POW)

Para elevar un número `x` a una potencia `y` , use las funciones `POW()` o `POWER()`

```
SELECT POW(2,2); => 4
SELECT POW(4,2); => 16
```

Raíz cuadrada (SQRT)

Utilice la función `SQRT()` . Si el número es negativo, se devolverá `NULL` .

```
SELECT SQRT(16); -> 4
SELECT SQRT(-3); -> NULL
```

Números aleatorios (RAND)

Generar un número aleatorio

Para generar un número de punto flotante pseudoaleatorio entre `0` y `1` , use la función `RAND()`

Supongamos que tiene la siguiente consulta

```
SELECT i, RAND() FROM t;
```

Esto devolverá algo como esto

yo	RAND ()
1	0.6191438870682
2	0.93845168309142
3	0.83482678498591

Número aleatorio en un rango

Para generar un número aleatorio en el rango a <= n <= b, puede usar la siguiente fórmula

```
FLOOR(a + RAND() * (b - a + 1))
```

Por ejemplo, esto generará un número aleatorio entre 7 y 12.

```
SELECT FLOOR(7 + (RAND() * 6));
```

Una forma sencilla de devolver aleatoriamente las filas de una tabla:

```
SELECT * FROM tbl ORDER BY RAND();
```

Estos son números **pseudoaleatorios** .

El generador de números pseudoaleatorios en MySQL no es criptográficamente seguro. Es decir, si usa MySQL para generar números aleatorios para usarlos como secretos, un adversario determinado que sabe que usó MySQL podrá adivinar sus secretos más fácilmente de lo que cree.

Valor absoluto y signo (ABS, SIGNO)

Devuelve el valor absoluto de un número.

```
SELECT ABS(2);   -> 2
SELECT ABS(-46); -> 46
```

El `sign` de un número lo compara con 0.

Firmar	Resultado	Ejemplo
-1	n <0	`SELECT SIGN(42);` -> 1
0	n = 0	`SELECT SIGN(0);` -> 0
1	n> 0	`SELECT SIGN(-3);` -> -1

```
SELECT SIGN(-423421); -> -1
```

Lea Aritmética en línea: https://riptutorial.com/es/mysql/topic/4516/aritmetica

Capítulo 9: Backticks

Examples

Uso de backticks

Hay muchos ejemplos en los que se utilizan acentos abiertos en el interior de una consulta, pero para muchos aún no está claro cuándo o dónde utilizar acentos abiertos `` .

Las comillas invertidas se utilizan principalmente para evitar un error llamado " *palabra reservada de MySQL* ". Al crear una tabla en PHPmyAdmin, a veces se le presenta una advertencia o alerta de que está utilizando una " *palabra reservada de MySQL* ".

Por ejemplo, cuando creas una tabla con una columna llamada " `group` " obtienes una advertencia. Esto es porque puedes hacer la siguiente consulta:

```
SELECT student_name, AVG(test_score) FROM student GROUP BY group
```

Para asegurarse de que no recibe un error en su consulta, debe utilizar backticks para que su consulta sea:

```
SELECT student_name, AVG(test_score) FROM student GROUP BY `group`
```

Mesa

No solo los nombres de columna pueden estar rodeados por comillas invertidas, sino también nombres de tablas. Por ejemplo, cuando necesita `JOIN` varias tablas.

```
SELECT `users`.`username`, `groups`.`group` FROM `users`
```

Mas facil de leer

Como puede ver, el uso de comillas en las tablas y los nombres de las columnas también facilita la lectura de la consulta.

Por ejemplo, cuando estás acostumbrado a escribir consultas todo en minúsculas:

```
select student_name, AVG(test_score) from student group by group
select `student_name`, AVG(`test_score`) from `student` group by `group`
```

Consulte la página del Manual de MySQL titulada Palabras clave y palabras reservadas . Los que tienen una (R) son palabras reservadas. Los otros son simplemente palabras clave. Los Reservados requieren especial precaución.

Lea Backticks en línea: https://riptutorial.com/es/mysql/topic/5208/backticks

Capítulo 10: BORRAR

Sintaxis

- ELIMINAR [LOW_PRIORITY] [QUICK] [IGNORE] FROM table [WHERE conditions] [ORDER BY expression [ASC | DESC]] [LIMIT number_rows]; /// Sintaxis para eliminar filas de una sola tabla

Parámetros

Parámetro	Detalles
BAJA PRIORIDAD	Si se proporciona `LOW_PRIORITY` , la eliminación se retrasará hasta que no haya procesos leyendo de la tabla
IGNORAR	Si se proporciona `IGNORE` , todos los errores encontrados durante la eliminación se ignoran
mesa	La tabla de la que va a borrar los registros.
Donde las condiciones	Las condiciones que deben cumplirse para que los registros sean eliminados. Si no se proporcionan condiciones, todos los registros de la tabla se eliminarán
ORDEN POR expresión	Si se proporciona `ORDER BY` , los registros se eliminarán en el orden dado
LÍMITE	Controla el número máximo de registros para eliminar de la tabla. Dados `number_rows` serán eliminados.

Examples

Eliminar con la cláusula Where

```
DELETE FROM `table_name` WHERE `field_one` = 'value_one'
```

Esto eliminará todas las filas de la tabla donde el contenido de `field_one` para esa fila coincida con 'value_one'

La cláusula `WHERE` funciona de la misma manera que una selección, por lo que se pueden usar cosas como `>` , `<` , `<>` o `LIKE` .

Aviso: Es necesario usar cláusulas condicionales (WHERE, LIKE) en la consulta de eliminación. Si no utiliza ninguna cláusula condicional, se eliminarán todos los datos de esa tabla.

Eliminar todas las filas de una tabla

```
DELETE FROM table_name ;
```

Esto borrará todo, todas las filas de la tabla. Es el ejemplo más básico de la sintaxis. También muestra que las sentencias `DELETE` deben usarse con mucho cuidado, ya que pueden vaciar una tabla, si se omite la cláusula `WHERE` .

LIMITAR eliminaciones

```
DELETE FROM `table_name` WHERE `field_one` = 'value_one' LIMIT 1
```

Esto funciona de la misma manera que en el ejemplo 'Eliminar con cláusula Where', pero detendrá la eliminación una vez que se haya eliminado el número limitado de filas.

Si está limitando filas para su eliminación de esta manera, tenga en cuenta que eliminará la primera fila que coincida con los criterios. Es posible que no sea la esperada, ya que los resultados pueden aparecer sin clasificar si no están ordenados explícitamente.

Eliminaciones de tablas múltiples

La sentencia `DELETE` de MySQL puede usar la construcción `JOIN` , permitiendo también especificar de qué tablas eliminar. Esto es útil para evitar consultas anidadas. Dado el esquema:

```
create table people
(    id int primary key,
    name varchar(100) not null,
    gender char(1) not null
);
insert people (id,name,gender) values
(1,'Kathy','f'),(2,'John','m'),(3,'Paul','m'),(4,'Kim','f');

create table pets
(    id int auto_increment primary key,
    ownerId int not null,
    name varchar(100) not null,
    color varchar(100) not null
);
insert pets(ownerId,name,color) values
(1,'Rover','beige'),(2,'Bubbles','purple'),(3,'Spot','black and white'),
(1,'Rover2','white');
```

carné de identidad	nombre	género
1	Kathy	F
2	Juan	metro
3	Pablo	metro
4	Kim	F

carné de identidad	ownerId	nombre	color
1	1	Vagabundo	beige
2	2	Burbujas	púrpura
4	1	Rover2	blanco

Si queremos eliminar a las mascotas de Paul, la declaración

```
DELETE p2
FROM pets p2
WHERE p2.ownerId in (
    SELECT p1.id
    FROM people p1
    WHERE p1.name = 'Paul');
```

Se puede reescribir como

```
DELETE p2    -- remove only rows from pets
FROM people p1
JOIN pets p2
ON p2.ownerId = p1.id
WHERE p1.name = 'Paul';
```

1 fila eliminada
El spot se ha eliminado de Pets.

`p1` y `p2` son alias para los nombres de tabla, especialmente útiles para nombres de tabla largos y facilidad de lectura.

Para eliminar tanto a la persona como a la mascota:

```
DELETE p1, p2     -- remove rows from both tables
FROM people p1
JOIN pets p2
ON p2.ownerId = p1.id
WHERE p1.name = 'Paul';
```

2 filas eliminadas
El spot se ha eliminado de Pets.
Paul es eliminado de People

llaves extranjeras

Cuando la instrucción DELETE incluye tablas con una clave extranjera, el optimizador puede procesar las tablas en un orden que no sigue la relación. Añadiendo, por ejemplo, una clave externa a la definición de `pets`

```
ALTER TABLE pets ADD CONSTRAINT `fk_pets_2_people` FOREIGN KEY (ownerId) references people(id)
```

```
ON DELETE CASCADE;
```

el motor puede intentar eliminar las entradas de las `people` antes que las `pets` , lo que provoca el siguiente error:

```
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails
(`test`.`pets`, CONSTRAINT `pets_ibfk_1` FOREIGN KEY (`ownerId`) REFERENCES `people` (`id`))
```

La solución en este caso es eliminar la fila de las `people` y confiar en las capacidades `ON DELETE` `InnoDB` para propagar la eliminación:

```
DELETE FROM people
WHERE name = 'Paul';
```

2 filas eliminadas
Paul es eliminado de People
El spot se borra en cascada de las mascotas

Otra solución es deshabilitar temporalmente la verificación de las teclas externas:

```
SET foreign_key_checks = 0;
DELETE p1, p2 FROM people p1 JOIN pets p2 ON p2.ownerId = p1.id WHERE p1.name = 'Paul';
SET foreign_key_checks = 1;
```

Eliminación básica

```
DELETE FROM `myTable` WHERE `someColumn` = 'something'
```

La cláusula `WHERE` es opcional pero sin ella se eliminan todas las filas.

DELETE vs TRUNCATE

```
TRUNCATE tableName;
```

Esto eliminará todos los datos y restablecerá el índice `AUTO_INCREMENT` . Es mucho más rápido que `DELETE FROM tableName` en un gran conjunto de datos. Puede ser muy útil durante el desarrollo / prueba.

Cuando ***trunca*** una tabla, el servidor SQL no borra los datos, elimina la tabla y la vuelve a crear, por lo que desasigna las páginas para que pueda recuperar los datos truncados antes de las páginas sobrescritas. (El espacio no se puede recuperar inmediatamente para `innodb_file_per_table=OFF`).

Multi-mesa BORRAR

MySQL permite especificar de qué tabla deben eliminarse las filas coincidentes

```
-- remove only the employees
DELETE e
FROM Employees e JOIN Department d ON e.department_id = d.department_id
WHERE d.name = 'Sales'
```

```
-- remove employees and department
DELETE e, d
FROM Employees e JOIN Department d ON e.department_id = d.department_id
WHERE d.name = 'Sales'
```

```
-- remove from all tables (in this case same as previous)
DELETE
FROM Employees e JOIN Department d ON e.department_id = d.department_id
WHERE d.name = 'Sales'
```

Lea BORRAR en línea: https://riptutorial.com/es/mysql/topic/1487/borrar

Capítulo 11: Búsqueda de texto completo

Introducción

MySQL ofrece búsqueda FULLTEXT. Busca tablas con columnas que contengan texto para las mejores coincidencias de palabras y frases.

Observaciones

`FULLTEXT` búsqueda en `FULLTEXT` funciona de manera extraña en tablas que contienen pequeñas cantidades de filas. Por lo tanto, cuando experimente con él, puede resultarle útil obtener una tabla de tamaño mediano en línea. Aquí hay una tabla de artículos de libros , con títulos y autores. Puedes descargarlo, descomprimirlo y cargarlo en MySQL.

`FULLTEXT` búsqueda `FULLTEXT` está diseñada para ser utilizada con ayuda humana. Está diseñado para generar más coincidencias que una operación de filtrado ordinaria de la `WHERE column LIKE 'text%'` .

`FULLTEXT` búsqueda `FULLTEXT` está disponible para las tablas `MyISAM` . También está disponible para tablas `InnoDB` en MySQL versión 5.6.4 o posterior.

Examples

Sencilla búsqueda en FULLTEXT

```
   SET @searchTerm= 'Database Programming';
SELECT MATCH (Title) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE) Score,
       ISBN, Author, Title
  FROM book
 WHERE MATCH (Title) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE)
 ORDER BY MATCH (Title) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE) DESC;
```

Dada una tabla llamada `book` con columnas llamadas `ISBN` , 'Título' y 'Autor', esto encuentra libros que coinciden con los términos `'Database Programming'` . Muestra los mejores partidos primero.

Para que esto funcione, debe estar disponible un índice de texto completo en la columna `Title` :

```
ALTER TABLE book ADD FULLTEXT INDEX Fulltext_title_index (Title);
```

Búsqueda sencilla de BOOLEAN

```
   SET @searchTerm= 'Database Programming -Java';
SELECT MATCH (Title) AGAINST (@searchTerm IN BOOLEAN MODE) Score,
       ISBN, Author, Title
  FROM book
 WHERE MATCH (Title) AGAINST (@searchTerm IN BOOLEAN MODE)
```

```
  ORDER BY MATCH (Title) AGAINST (@searchTerm IN BOOLEAN MODE) DESC;
```

Dada una tabla llamada `book` con columnas llamadas `ISBN` , `Title` y `Author` , esto busca libros con las palabras `'Database'` y `'Programming'` en el título, pero no la palabra `'Java'` .

Para que esto funcione, debe estar disponible un índice de texto completo en la columna Título:

```
ALTER TABLE book ADD FULLTEXT INDEX Fulltext_title_index (Title);
```

Multi-columna de búsqueda en FULLTEXT

```
    SET @searchTerm= 'Date Database Programming';
 SELECT MATCH (Title, Author) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE) Score,
        ISBN, Author, Title
   FROM book
  WHERE MATCH (Title, Author) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE)
  ORDER BY MATCH (Title, Author) AGAINST (@searchTerm IN NATURAL LANGUAGE MODE) DESC;
```

Dada una tabla llamada libro con columnas llamadas `ISBN` , `Title` y `Author` , esto encuentra libros que coinciden con los términos 'Programación de la base de datos de fechas'. Muestra los mejores partidos primero. Las mejores coincidencias incluyen libros escritos por el Prof. CJ Date.

(Pero, una de las mejores coincidencias también es *The Date Doctor's Guide to Dating: Cómo* pasar *de First Date a Perfect Mate* . Esto muestra una limitación de la búsqueda en FULLTEXT: no pretende entender cosas como partes del habla o El significado de las palabras indexadas.

Para que esto funcione, debe estar disponible un índice de texto completo en las columnas Título y Autor:

```
ALTER TABLE book ADD FULLTEXT INDEX Fulltext_title_author_index (Title, Author);
```

Lea Búsqueda de texto completo en línea: https://riptutorial.com/es/mysql/topic/8759/busqueda-de-texto-completo

Capítulo 12: Cambia la contraseña

Examples

Cambiar la contraseña de root de MySQL en Linux

Para cambiar la contraseña del usuario root de MySQL:

Paso 1: Detener el servidor MySQL.

- en Ubuntu o Debian:
  `sudo /etc/init.d/mysql stop`
- En CentOS, Fedora o Red Hat Enterprise Linux:
  `sudo /etc/init.d/mysqld stop`

Paso 2: Inicie el servidor MySQL sin el sistema de privilegios.

```
sudo mysqld_safe --skip-grant-tables &
```

o, si `mysqld_safe` no está disponible,

```
sudo mysqld --skip-grant-tables &
```

Paso 3: Conectar al servidor MySQL.

```
mysql -u root
```

Paso 4: Establecer una nueva contraseña para el usuario root.

5.7

```
FLUSH PRIVILEGES;
ALTER USER 'root'@'localhost' IDENTIFIED BY 'new_password';
FLUSH PRIVILEGES;
exit;
```

5.7

```
FLUSH PRIVILEGES;
SET PASSWORD FOR 'root'@'localhost' = PASSWORD('new_password');
FLUSH PRIVILEGES;
exit;
```

Nota: La sintaxis de `ALTER USER` se introdujo en MySQL 5.7.6.

Paso 5: Reinicie el servidor MySQL.

- en Ubuntu o Debian:
  `sudo /etc/init.d/mysql stop`

```
sudo /etc/init.d/mysql start
```

- En CentOS, Fedora o Red Hat Enterprise Linux:

```
sudo /etc/init.d/mysqld stop
sudo /etc/init.d/mysqld start
```

Cambiar la contraseña de root de MySQL en Windows

Cuando queremos cambiar la contraseña de root en Windows, debemos seguir los siguientes pasos:

Paso 1: Comience su Símbolo del sistema utilizando cualquiera de los siguientes métodos:

Perss `Crtl+R` o Goto `Start Menu > Run` y luego escriba `cmd` y presione enter

Paso 2: cambia tu directorio a donde está instalado `MYSQL` , en mi caso es

```
C:\> cd C:\mysql\bin
```

Paso 3: Ahora necesitamos iniciar el símbolo del sistema `mysql`

```
C:\mysql\bin> mysql -u root mysql
```

Paso 4: Encender la consulta para cambiar `root` contraseña de `root`

```
mysql> SET PASSWORD FOR root@localhost=PASSWORD('my_new_password');
```

Proceso

1. Detenga el proceso del servidor / daemon MySQL (mysqld).
2. Inicie el servidor MySQL procese la opción --skip-grant-tables para que no `mysqld_safe --skip-grant-tables &` una contraseña: `mysqld_safe --skip-grant-tables &`
3. Conéctese al servidor MySQL como usuario root: `mysql -u root`
4. Cambia la contraseña:

- (5.7.6 y posteriores): `ALTER USER 'root'@'localhost' IDENTIFIED BY 'new-password';`
- (5.7.5 y versiones anteriores, o MariaDB): `SET PASSWORD FOR 'root'@'localhost' = PASSWORD('new-password'); flush privileges; quit;`

5. Reinicie el servidor MySQL.

Nota: esto funcionará solo si está físicamente en el mismo servidor.

Doc en línea: http://dev.mysql.com/doc/refman/5.7/en/resetting-permissions.html

Lea Cambia la contraseña en línea: https://riptutorial.com/es/mysql/topic/2761/cambia-la-contrasena

Capítulo 13: Cliente MySQL

Sintaxis

- mysql [OPCIONES] [nombre_base_datos]

Parámetros

Parámetro	Descripción
`-D --database=name`	nombre de la base de datos
`--delimiter=str`	establecer el delimitador de la declaración. El predeterminado es ';'
`-e --execute='command'`	Ejecutar comando
`-h --host=name`	nombre de host para conectarse a
`-p --password=name`	*Nota de* contraseña *: no hay espacio entre* `-p` *y la contraseña*
`-p` (sin contraseña)	la contraseña será solicitada
`-P --port=#`	número de puerto
`-s --silent`	Modo silencioso, produce menos salida. Utilice `\t` como separador de columna
`-ss`	like `-s` , pero omita los nombres de columna
`-S --socket=path`	especifique el socket (Unix) o la canalización con nombre (Windows) que se usará al conectarse a una instancia local
`--skip-column-names`	omitir nombres de columnas
`-u --user=name`	nombre de usuario
`-U --safe-updates --i-am-a-dummy`	inicie sesión con la variable `sql_safe_updates=ON` . Esto permitirá solo `DELETE` y `UPDATE` que explícitamente use claves
`-V --version`	imprimir la versión y salir

Examples

Inicio de sesión base

Para acceder a MySQL desde la línea de comandos:

```
mysql --user=username --password=pwd --host=hostname test_db
```

Esto se puede reducir a:

```
mysql -u username -p password -h hostname test_db
```

Al omitir el valor de la `password` , MySQL solicitará cualquier contraseña requerida como primera entrada. Si especifica la `password` el cliente le dará una advertencia "insegura":

```
mysql -u=username -p -h=hostname test_db
```

Para las conexiones locales `--socket` se pueden utilizar para apuntar al archivo de socket:

```
mysql --user=username --password=pwd --host=localhost --socket=/path/to/mysqld.sock test_db
```

La omisión del parámetro `socket` hará que el cliente intente conectarse a un servidor en la máquina local. El servidor debe estar ejecutándose para conectarse a él.

Ejecutar comandos

Este conjunto de ejemplos muestra cómo ejecutar comandos almacenados en cadenas o archivos de script, sin la necesidad del indicador interactivo. Esto es especialmente útil cuando un script de shell necesita interactuar con una base de datos.

Ejecutar comando desde una cadena

```
$ mysql -uroot -proot test -e'select * from people'

+----+-------+--------+
| id | name  | gender |
+----+-------+--------+
|  1 | Kathy | f      |
|  2 | John  | m      |
+----+-------+--------+
```

Para formatear la salida como una cuadrícula separada por tabulaciones, use el parámetro `--silent` :

```
$ mysql -uroot -proot test -s -e'select * from people'

id      name    gender
1       Kathy   f
2       John    m
```

Para omitir los encabezados:

```
$ mysql -uroot -proot test -ss -e'select * from people'

1       Kathy   f
2       John    m
```

Ejecutar desde el archivo de script:

```
$ mysql -uroot -proot test < my_script.sql
```

```
$ mysql -uroot -proot test -e'source my_script.sql'
```

Escribe la salida en un archivo

```
$ mysql -uroot -proot test < my_script.sql > out.txt

$ mysql -uroot -proot test -s -e'select * from people' > out.txt
```

Lea Cliente MySQL en línea: https://riptutorial.com/es/mysql/topic/5619/cliente-mysql

Capítulo 14: Códigos de error

Examples

Código de error 1064: error de sintaxis

```
select LastName, FirstName,
from Person
```

Devuelve el mensaje:

> Código de error: 1064. Usted tiene un error en su sintaxis SQL; consulte el manual que corresponde a la versión de su servidor MySQL para conocer la sintaxis correcta para usar cerca de "de Persona" en la línea 2.

Obtener un mensaje de "error 1064" de MySQL significa que la consulta no se puede analizar sin errores de sintaxis. En otras palabras, no puede dar sentido a la consulta.

La cita en el mensaje de error comienza con el primer carácter de la consulta que MySQL no puede averiguar cómo analizar. En este ejemplo, MySQL no puede tener sentido, en contexto, de parte `from Person` . En este caso, hay una coma adicional inmediatamente anterior `from Person` . La coma le dice a MySQL que espere otra descripción de columna en la cláusula `SELECT`

Un error de sintaxis siempre dice `... near '...'` . La cosa al comienzo de las citas está muy cerca de donde está el error. Para localizar un error, mire el primer token en las comillas y el último token antes de las comillas.

A veces te pondrás `... near ''` ; Es decir, nada en las citas. Eso significa que el primer carácter que MySQL no puede entender es justo al final o al principio de la declaración. Esto sugiere que la consulta contiene comillas no balanceadas (`'` o `"`) o paréntesis no balanceados o que no terminó la declaración correctamente antes.

En el caso de una rutina almacenada, es posible que haya olvidado usar `DELIMITER` correctamente.

Entonces, cuando obtenga el Error 1064, mire el texto de la consulta y encuentre el punto mencionado en el mensaje de error. Inspeccione visualmente el texto de la consulta alrededor de ese punto.

Si le pide a alguien que lo ayude a solucionar el error 1064, es mejor proporcionar tanto el texto de toda la consulta como el texto del mensaje de error.

Código de error 1175: Actualización segura

Este error aparece al intentar actualizar o eliminar registros sin incluir la cláusula `WHERE` que utiliza la columna `KEY` .

Para ejecutar la eliminación o actualización de todos modos - escriba:

```
SET SQL_SAFE_UPDATES = 0;
```

Para volver a habilitar el modo seguro, escriba:

```
SET SQL_SAFE_UPDATES = 1;
```

Código de error 1215: No se puede agregar una restricción de clave externa

Este error se produce cuando las tablas no están estructuradas adecuadamente para manejar la verificación rápida de los requisitos de clave externa (`FK`) que el desarrollador exige.

```
CREATE TABLE `gtType` (
  `type` char(2) NOT NULL,
  `description` varchar(1000) NOT NULL,
  PRIMARY KEY (`type`)
) ENGINE=InnoDB;

CREATE TABLE `getTogethers` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `type` char(2) NOT NULL,
  `eventDT` datetime NOT NULL,
  `location` varchar(1000) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_gt2type` (`type`), -- see Note1 below
  CONSTRAINT `gettogethers_ibfk_1` FOREIGN KEY (`type`) REFERENCES `gtType` (`type`)
) ENGINE=InnoDB;
```

Nota 1: una clave como esta se creará automáticamente si es necesario debido a la definición de FK en la línea que la sigue. El desarrollador puede omitirlo, y se agregará la CLAVE (también conocido como índice) si es necesario. Un ejemplo de lo que fue omitido por el desarrollador se muestra a continuación en `someOther` .

Hasta aquí todo bien, hasta la llamada de abajo.

```
CREATE TABLE `someOther` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `someDT` datetime NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `someOther_dt` FOREIGN KEY (`someDT`) REFERENCES `getTogethers` (`eventDT`)
) ENGINE=InnoDB;
```

> Código de error: 1215. No se puede agregar una restricción de clave externa

En este caso, falla debido a la falta de un índice en la tabla *referenciada* `getTogethers` para manejar la búsqueda rápida de un `eventDT` . Para ser resuelto en la siguiente afirmación.

```
CREATE INDEX `gt_eventdt` ON getTogethers (`eventDT`);
```

La tabla `getTogethers` ha sido modificada, y ahora la creación de `someOther` tendrá éxito.

De la página del manual de MySQL usando restricciones de FOREIGN KEY :

MySQL requiere índices en claves externas y claves referenciadas para que las comprobaciones de claves externas puedan ser rápidas y no requieran un escaneo de tablas. En la tabla de referencia, debe haber un índice donde las columnas de clave externa se enumeran como las primeras columnas en el mismo orden. Dicho índice se crea automáticamente en la tabla de referencia si no existe.

Las columnas correspondientes en la clave externa y la clave a la que se hace referencia deben tener tipos de datos similares. El tamaño y el signo de los tipos enteros deben ser iguales. La longitud de los tipos de cadena no tiene por qué ser la misma. Para las columnas de cadena no binaria (carácter), el conjunto de caracteres y la intercalación deben ser los mismos.

InnoDB permite que una clave externa haga referencia a cualquier columna de índice o grupo de columnas. Sin embargo, en la tabla a la que se hace referencia, debe haber un índice donde las columnas a las que se hace referencia se enumeran como las primeras columnas en el mismo orden.

Tenga en cuenta que el último punto anterior sobre las primeras columnas (más a la izquierda) y la falta de un requisito de clave principal (aunque altamente recomendado).

Tras la creación exitosa de una tabla de *referencia* (secundaria), todas las claves que se crearon automáticamente para usted son visibles con un comando como el siguiente:

```
SHOW CREATE TABLE someOther;
```

Otros casos comunes de experimentar este error incluyen, como se mencionó anteriormente en los documentos, pero se debe resaltar:

- Diferencias aparentemente triviales en la `INT` que están firmadas, apuntando hacia `INT UNSIGNED` .
- Los desarrolladores tienen problemas para comprender las LLAVES de varias columnas (compuestas) y los primeros requisitos de pedido (de la izquierda).

1045 Acceso denegado

Ver discusiones en "GRANT" y "Recuperar contraseña de root".

1236 "posición imposible" en la replicación

Por lo *general,* esto significa que el Maestro se estrelló y que `sync_binlog` estaba DESACTIVADO. La solución es `CHANGE MASTER to POS=0` del siguiente archivo binlog (ver Maestro) en el Esclavo.

La causa: el Maestro envía elementos de replicación al Esclavo antes de descargar a su binlog (cuando `sync_binlog=OFF`). Si el maestro se bloquea antes de la descarga, el esclavo ya se ha movido lógicamente más allá del final del archivo en el binlog. Cuando el Maestro se inicia de nuevo, inicia un nuevo binlog, por lo que CAMBIAR al principio de ese binlog es la mejor solución disponible.

Una solución a largo plazo es `sync_binlog=ON` , si puede pagar la E / S adicional que causa.

(Si está ejecutando con GTID, ...?)

2002, 2003 No se puede conectar

Compruebe si hay un problema con el Firewall bloqueando el puerto 3306.

Algunos posibles diagnósticos y / o soluciones.

- ¿El servidor está funcionando realmente?
- "Service firewalld stop" y "systemctl disable firewalld"
- telnet master 3306
- Compruebe la `bind-address`
- compruebe `skip-name-resolve`
- compruebe el zócalo.

1067, 1292, 1366, 1411 - Valor incorrecto para el número, la fecha, el valor predeterminado, etc.

1067 Esto probablemente esté relacionado con los valores predeterminados de `TIMESTAMP` , que han cambiado con el tiempo. Consulte los `TIMESTAMP defaults` en la página Fechas y horarios. (que aún no existe)

1292/1366 DOBLE / Integer Compruebe si hay letras u otros errores de sintaxis. Compruebe que las columnas se alinean; quizás pienses que estás poniendo en un `VARCHAR` pero está alineado con una columna numérica.

1292 DATETIME Verifica demasiado lejos en el pasado o el futuro. Verifique entre las 2 y las 3 de la madrugada cuando cambie el horario de verano. Comprueba si hay una sintaxis incorrecta, como `+00` cosas de zona horaria.

1292 VARIABLE Verifique los valores permitidos para la `VARIABLE` que está intentando `SET` .

1292 DATOS DE LA CARGA Mire la línea que es 'mala'. Compruebe los símbolos de escape, etc. Mire los tipos de datos.

1411 STR_TO_DATE ¿Fecha con formato incorrecto?

126, 127, 134, 144, 145

Cuando intenta acceder a los registros de la base de datos MySQL, puede recibir estos mensajes de error. Estos mensajes de error ocurrieron debido a la corrupción en la base de datos MySQL. Los siguientes son los tipos

```
MySQL error code 126 = Index file is crashed
MySQL error code 127 = Record-file is crashed
MySQL error code 134 = Record was already deleted (or record file crashed)
MySQL error code 144 = Table is crashed and last repair failed
```

```
MySQL error code 145 = Table was marked as crashed and should be repaired
```

El error de MySQL, el ataque de virus, la falla del servidor, el apagado incorrecto, la tabla dañada son la razón detrás de esta corrupción. Cuando se corrompe, se vuelve inaccesible y ya no se puede acceder a ellos. Para obtener accesibilidad, la mejor manera de recuperar datos de una copia de seguridad actualizada. Sin embargo, si no tiene una copia de seguridad actualizada o válida, puede ir a la reparación de MySQL.

Si el tipo de motor de la mesa es `MyISAM` , aplique `CHECK TABLE` , luego `REPAIR TABLE` .

Luego piense seriamente en convertir a InnoDB, para que este error no vuelva a ocurrir.

Sintaxis

```
CHECK TABLE <table name> ////To check the extent of database corruption
REPAIR TABLE <table name> ////To repair table
```

139

El error 139 puede significar que el número y el tamaño de los campos en la definición de la tabla excede algún límite. Soluciones:

- Repensar el esquema
- Normalizar algunos campos.
- Partición vertical de la tabla

1366

Esto generalmente significa que el manejo del conjunto de caracteres no fue consistente entre el cliente y el servidor. Ver ... para más ayuda.

126, 1054, 1146, 1062, 24

(tomando un descanso) Con la inclusión de esos 4 números de error, creo que esta página habrá cubierto aproximadamente el 50% de los errores típicos que obtienen los usuarios.

(Sí, este 'Ejemplo' necesita revisión.)

24 No se puede abrir el archivo (demasiados archivos abiertos)

`open_files_limit` proviene de una configuración del sistema operativo. `table_open_cache` necesita ser menos que eso.

Estos pueden causar ese error:

- Fallo en `DEALLOCATE PREPARE` en un procedimiento almacenado.
- PARTICIÓN de tabla (s) con un gran número de particiones y innodb_file_per_table = ON. Recomienda no tener más de 50 particiones en una tabla dada (por varias razones).

(Cuando "Particiones nativas" estén disponibles, este consejo puede cambiar.)

La solución obvia es establecer aumentar el límite SO: Para permitir más archivos, cambiar `ulimit` o `/etc/security/limits.conf` o en `sysctl.conf` (kern.maxfiles y kern.maxfilesperproc) o algo más (depende del sistema operativo). Luego incremente `open_files_limit` y `table_open_cache` .

A partir de la versión 5.6.8, `open_files_limit` es de tamaño automático basado en `max_connections` , pero está bien cambiarlo del valor predeterminado.

1062 - Entrada duplicada

Este error ocurre principalmente debido a las siguientes dos razones

1. *Valor duplicado* - `Error Code: 1062. Duplicate entry '12' for key 'PRIMARY'`

   La columna de clave principal es única y no aceptará la entrada duplicada. Por lo tanto, cuando intente insertar una nueva fila que ya está presente en su tabla, se producirá este error.

   Para resolver esto, establezca la columna de clave principal como `AUTO_INCREMENT` . Y cuando intenta insertar una nueva fila, ignore la columna de clave principal o inserte el valor `NULL` en la clave principal.

```
CREATE TABLE userDetails(
  userId INT(10) NOT NULL AUTO_INCREMENT,
  firstName VARCHAR(50),
  lastName VARCHAR(50),
  isActive INT(1) DEFAULT 0,
  PRIMARY KEY (userId) );

--->and now while inserting
INSERT INTO userDetails VALUES (NULL ,'John', 'Doe', 1);
```

2. *Campo de datos únicos* - `Error Code: 1062. Duplicate entry 'A' for key 'code'`

   Puede asignar una columna como única e intentar insertar una nueva fila con un valor ya existente para esa columna producirá este error.

   Para superar este error, use `INSERT IGNORE` lugar de `INSERT` normal. Si la nueva fila que intenta insertar no duplica un registro existente, MySQL lo inserta como de costumbre. Si el registro es un duplicado, la `IGNORE` clave `IGNORE` descarta sin generar ningún error.

```
INSERT IGNORE INTO userDetails VALUES (NULL ,'John', 'Doe', 1);
```

Lea Códigos de error en línea: https://riptutorial.com/es/mysql/topic/895/codigos-de-error

Capítulo 15: Comentar Mysql

Observaciones

El `--` estilo de comentario, que requiere un espacio al final, se diferencia en el comportamiento del estándar SQL , que no requiere el espacio.

Examples

Añadiendo comentarios

Hay tres tipos de comentarios:

```
 # This comment continues to the end of line

-- This comment continues to the end of line

/* This is an in-line comment */

/*
This is a
multiple-line comment
*/
```

Ejemplo:

```
SELECT * FROM t1; -- this is comment

CREATE TABLE stack(
    /*id_user int,
    username varchar(30),
    password varchar(30)
    */
    id int
);
```

El método `--` requiere que un espacio siga a `--` antes de que comience el comentario, de lo contrario se interpretará como un comando y generalmente causará un error.

```
#This comment works
/*This comment works.*/
--This comment does not.
```

Comentar las definiciones de la tabla

```
CREATE TABLE menagerie.bird (
    bird_id INT NOT NULL AUTO_INCREMENT,
    species VARCHAR(300) DEFAULT NULL COMMENT 'You can include genus, but never subspecies.',
    INDEX idx_species (species) COMMENT 'We must search on species often.',
```

```
    PRIMARY KEY (bird_id)
) ENGINE=InnoDB COMMENT 'This table was inaugurated on February 10th.';
```

Usar un `=` después de `COMMENT` es opcional. (Documentos oficiales)

Estos comentarios, a diferencia de los otros, se guardan con el esquema y se pueden recuperar a través de `SHOW CREATE TABLE` o de `information_schema` .

Lea Comentar Mysql en línea: https://riptutorial.com/es/mysql/topic/2337/comentar-mysql

Capítulo 16: Conectando con UTF-8 usando varios lenguajes de programación.

Examples

Pitón

1ª o 2ª línea en código fuente (para tener literales en el código codificado en utf8):

```
# -*- coding: utf-8 -*-
```

Conexión:

```
db = MySQLdb.connect(host=DB_HOST, user=DB_USER, passwd=DB_PASS, db=DB_NAME,
        charset="utf8mb4", use_unicode=True)
```

Para páginas web, una de estas:

```
<meta charset="utf-8" />
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

PHP

En php.ini (este es el valor predeterminado después de PHP 5.6):

```
default_charset UTF-8
```

Al construir una página web:

```
header('Content-type: text/plain; charset=UTF-8');
```

Al conectarse a MySQL:

```
(for mysql:)   Do not use the mysql_* API!
(for mysqli:)  $mysqli_obj->set_charset('utf8mb4');
(for PDO:)     $db = new PDO('dblib:host=host;dbname=db;charset=utf8', $user, $pwd);
```

En el código, no utilice ninguna rutina de conversión.

Para la entrada de datos,

```
<form accept-charset="UTF-8">
```

Para JSON, para evitar `\uxxxx` :

```
$t = json_encode($s, JSON_UNESCAPED_UNICODE);
```

Lea Conectando con UTF-8 usando varios lenguajes de programación. en línea: https://riptutorial.com/es/mysql/topic/7332/conectando-con-utf-8-usando-varios-lenguajes-de-programacion-

Capítulo 17: Configuración de la conexión SSL

Examples

Configuración para sistemas basados en Debian

(Esto supone que MySQL se ha instalado y que se está utilizando `sudo`).

Generando una CA y claves SSL

Asegúrese de que OpenSSL y las bibliotecas estén instaladas:

```
apt-get -y install openssl
apt-get -y install libssl-dev
```

Luego haga e ingrese un directorio para los archivos SSL:

```
mkdir /home/ubuntu/mysqlcerts
cd /home/ubuntu/mysqlcerts
```

Para generar claves, cree una autoridad de certificación (CA) para firmar las claves (autofirmadas):

```
openssl genrsa 2048 > ca-key.pem
openssl req -new -x509 -nodes -days 3600 -key ca-key.pem -out ca.pem
```

Los valores ingresados en cada solicitud no afectarán la configuración. A continuación, cree una clave para el servidor y firme utilizando la CA de antes:

```
openssl req -newkey rsa:2048 -days 3600 -nodes -keyout server-key.pem -out server-req.pem
openssl rsa -in server-key.pem -out server-key.pem

openssl x509 -req -in server-req.pem -days 3600 -CA ca.pem -CAkey ca-key.pem -set_serial 01 -
out server-cert.pem
```

Luego crea una clave para un cliente:

```
openssl req -newkey rsa:2048 -days 3600 -nodes -keyout client-key.pem -out client-req.pem
openssl rsa -in client-key.pem -out client-key.pem
openssl x509 -req -in client-req.pem -days 3600 -CA ca.pem -CAkey ca-key.pem -set_serial 01 -
out client-cert.pem
```

Para asegurarse de que todo se haya configurado correctamente, verifique las claves:

```
openssl verify -CAfile ca.pem server-cert.pem client-cert.pem
```

Añadiendo las claves a MySQL

Abra el archivo de configuración de MySQL . Por ejemplo:

```
vim /etc/mysql/mysql.conf.d/mysqld.cnf
```

En la sección `[mysqld]` , agregue las siguientes opciones:

```
ssl-ca = /home/ubuntu/mysqlcerts/ca.pem
ssl-cert = /home/ubuntu/mysqlcerts/server-cert.pem
ssl-key = /home/ubuntu/mysqlcerts/server-key.pem
```

Reinicie MySQL. Por ejemplo:

```
service mysql restart
```

Probar la conexión SSL

Conéctese de la misma manera, pasando las opciones adicionales `ssl-ca` , `ssl-cert` y `ssl-key` , utilizando la clave de cliente generada. Por ejemplo, asumiendo `cd /home/ubuntu/mysqlcerts` :

```
mysql --ssl-ca=ca.pem --ssl-cert=client-cert.pem --ssl-key=client-key.pem -h 127.0.0.1 -u
superman -p
```

Después de iniciar sesión, verifique que la conexión sea segura:

```
superman@127.0.0.1 [None]> SHOW VARIABLES LIKE '%ssl%';
+---------------+-----------------------------------------+
| Variable_name | Value                                   |
+---------------+-----------------------------------------+
| have_openssl  | YES                                     |
| have_ssl      | YES                                     |
| ssl_ca        | /home/ubuntu/mysqlcerts/ca.pem          |
| ssl_capath    |                                         |
| ssl_cert      | /home/ubuntu/mysqlcerts/server-cert.pem |
| ssl_cipher    |                                         |
| ssl_crl       |                                         |
| ssl_crlpath   |                                         |
| ssl_key       | /home/ubuntu/mysqlcerts/server-key.pem  |
+---------------+-----------------------------------------+
```

También puedes consultar:

```
superman@127.0.0.1 [None]> STATUS;
...
SSL:                    Cipher in use is DHE-RSA-AES256-SHA
```

```
...
```

Cumplimiento de SSL

Esto es a través de `GRANT` , utilizando `REQUIRE SSL` :

```
GRANT ALL PRIVILEGES ON *.* TO 'superman'@'127.0.0.1' IDENTIFIED BY 'pass' REQUIRE SSL;
FLUSH PRIVILEGES;
```

Ahora, `superman` *debe* conectarse a través de SSL.

Si no desea administrar las claves de cliente, use la clave de cliente anterior y úsela automáticamente para todos los clientes. Abra el archivo de configuración de MySQL , por ejemplo:

```
vim /etc/mysql/mysql.conf.d/mysqld.cnf
```

En la sección `[client]` , agregue las siguientes opciones:

```
ssl-ca = /home/ubuntu/mysqlcerts/ca.pem
ssl-cert = /home/ubuntu/mysqlcerts/client-cert.pem
ssl-key = /home/ubuntu/mysqlcerts/client-key.pem
```

Ahora `superman` solo tiene que escribir lo siguiente para iniciar sesión a través de SSL:

```
mysql -h 127.0.0.1 -u superman -p
```

La conexión desde otro programa, por ejemplo en Python, normalmente solo requiere un parámetro adicional a la función de conexión. Un ejemplo de Python:

```
import MySQLdb
ssl = {'cert': '/home/ubuntu/mysqlcerts/client-cert.pem', 'key':
'/home/ubuntu/mysqlcerts/client-key.pem'}
conn = MySQLdb.connect(host='127.0.0.1', user='superman', passwd='imsoawesome', ssl=ssl)
```

Referencias y lecturas adicionales:

- https://www.percona.com/blog/2013/06/22/setting-up-mysql-ssl-and-secure-connections/
- https://lowendbox.com/blog/getting-started-with-mysql-over-ssl/
- http://xmodulo.com/enable-ssl-mysql-server-client.html
- https://ubuntuforums.org/showthread.php?t=1121458

Configuración para CentOS7 / RHEL7

Este ejemplo asume dos servidores:

1. dbserver (donde vive nuestra base de datos)
2. appclient (donde viven nuestras aplicaciones)

FWIW, ambos servidores están ejecutando SELinux.

Primero, inicie sesión en dbserver

Crear un directorio temporal para crear los certificados.

```
mkdir /root/certs/mysql/ && cd /root/certs/mysql/
```

Crear los certificados del servidor.

```
openssl genrsa 2048 > ca-key.pem
openssl req -sha1 -new -x509 -nodes -days 3650 -key ca-key.pem > ca-cert.pem
openssl req -sha1 -newkey rsa:2048 -days 730 -nodes -keyout server-key.pem > server-req.pem
openssl rsa -in server-key.pem -out server-key.pem
openssl x509 -sha1 -req -in server-req.pem -days 730  -CA ca-cert.pem -CAkey ca-key.pem -
set_serial 01 > server-cert.pem
```

Mueva los certificados del servidor a / etc / pki / tls / certs / mysql /

La ruta del directorio asume CentOS o RHEL (ajuste según sea necesario para otras distribuciones):

```
mkdir /etc/pki/tls/certs/mysql/
```

Asegúrese de establecer permisos en la carpeta y los archivos. mysql necesita plena propiedad y acceso.

```
chown -R mysql:mysql /etc/pki/tls/certs/mysql
```

Ahora configura MySQL / MariaDB

```
# vi /etc/my.cnf
# i
[mysqld]
bind-address=*
ssl-ca=/etc/pki/tls/certs/ca-cert.pem
ssl-cert=/etc/pki/tls/certs/server-cert.pem
ssl-key=/etc/pki/tls/certs/server-key.pem
# :wq
```

Entonces

```
systemctl restart mariadb
```

No olvide abrir su firewall para permitir conexiones desde appclient (usando IP 1.2.3.4)

```
firewall-cmd --zone=drop --permanent --add-rich-rule 'rule family="ipv4" source
address="1.2.3.4" service name="mysql" accept'
# I force everything to the drop zone.  Season the above command to taste.
```

Ahora reinicia firewalld

```
service firewalld restart
```

A continuación, inicie sesión en el servidor mysql de dbserver:

```
mysql -uroot -p
```

Emita lo siguiente para crear un usuario para el cliente. nota REQUIRE SSL en la sentencia GRANT.

```
GRANT ALL PRIVILEGES ON *.* TO 'iamsecure'@'appclient' IDENTIFIED BY 'dingdingding' REQUIRE
SSL;
FLUSH PRIVILEGES;
# quit mysql
```

Aún debe estar en / root / certs / mysql desde el primer paso. Si no, cd de nuevo a él para uno de los comandos a continuación.

Crear los certificados de cliente.

```
openssl req -sha1 -newkey rsa:2048 -days 730 -nodes -keyout client-key.pem > client-req.pem
openssl rsa -in client-key.pem -out client-key.pem
openssl x509 -sha1 -req -in client-req.pem -days 730 -CA ca-cert.pem -CAkey ca-key.pem -
set_serial 01 > client-cert.pem
```

Nota : usé el mismo nombre común para los certificados de servidor y cliente. YMMV.

Asegúrese de que todavía esté en / root / certs / mysql / para este próximo comando

Combine el certificado de CA del servidor y del cliente en un solo archivo:

```
cat server-cert.pem client-cert.pem > ca.pem
```

Asegúrese de ver dos certificados:

```
cat ca.pem
```

TRABAJO LATERAL DEL FIN DEL SERVIDOR POR AHORA.

Abre otra terminal y

```
ssh appclient
```

Como antes, crea un hogar permanente para los certificados del cliente.

```
mkdir /etc/pki/tls/certs/mysql/
```

Ahora, coloque los certificados de cliente (creados en dbserver) en appclient. Puede hacer una copia de ellos o simplemente copiar y pegar los archivos uno por uno.

```
scp dbserver
# copy files from dbserver to appclient
# exit scp
```

Una vez más, asegúrese de establecer permisos en la carpeta y los archivos. mysql necesita plena propiedad y acceso.

```
chown -R mysql:mysql /etc/pki/tls/certs/mysql
```

Debe tener tres archivos, cada uno de los cuales pertenece al usuario mysql:

```
/etc/pki/tls/certs/mysql/ca.pem
/etc/pki/tls/certs/mysql/client-cert.pem
/etc/pki/tls/certs/mysql/client-key.pem
```

Ahora edite la configuración MariaDB / MySQL de appclient en la sección `[client]` .

```
vi /etc/my.cnf
# i
[client]
ssl-ca=/etc/pki/tls/certs/mysql/ca.pem
ssl-cert=/etc/pki/tls/certs/mysql/client-cert.pem
ssl-key=/etc/pki/tls/certs/mysql/client-key.pem
# :wq
```

Reinicie el servicio mariadb de appclient:

```
systemctl restart mariadb
```

todavía en el cliente aquí

Esto debería devolver: ssl TRUE

```
mysql --ssl --help
```

Ahora, inicie sesión en la instancia de mysql de appclient

```
mysql -uroot -p
```

Debería ver SÍ a las dos variables a continuación

```
show variables LIKE '%ssl';
    have_openssl    YES
    have_ssl              YES
```

Inicialmente vi

```
 have_openssl NO
```

Una mirada rápida a mariadb.log reveló:

> Error de SSL: no se puede obtener el certificado de '/etc/pki/tls/certs/mysql/client-cert.pem'

El problema era que root era propiedad de client-cert.pem y la carpeta que lo contenía. La solución fue establecer la propiedad de / etc / pki / tls / certs / mysql / a mysql.

```
chown -R mysql:mysql /etc/pki/tls/certs/mysql
```

Reinicie mariadb si es necesario desde el paso inmediatamente anterior

AHORA ESTAMOS LISTOS PARA PROBAR LA CONEXIÓN SEGURA

Todavía estamos en apuros aquí

Intente conectarse a la instancia mysql de dbserver utilizando la cuenta creada anteriormente.

```
mysql -h dbserver -u iamsecure -p
# enter password dingdingding (hopefully you changed that to something else)
```

Con un poco de suerte debes iniciar sesión sin error.

Para confirmar que está conectado con SSL habilitado, emita el siguiente comando desde el indicador MariaDB / MySQL:

```
\s
```

Eso es una barra invertida, también conocido como estado

Eso mostrará el estado de tu conexión, que debería verse así:

```
Connection id:        4
Current database:
```

```
Current user:        iamsecure@appclient
SSL:            Cipher in use is DHE-RSA-AES256-GCM-SHA384
Current pager:        stdout
Using outfile:        ''
Using delimiter:    ;
Server:            MariaDB
Server version:        5.X.X-MariaDB MariaDB Server
Protocol version:    10
Connection:        dbserver via TCP/IP
Server characterset:    latin1
Db     characterset:    latin1
Client characterset:    utf8
Conn.  characterset:    utf8
TCP port:        3306
Uptime:            42 min 13 sec
```

Si obtiene un permiso denegado errores en su intento de conexión, verifique la declaración GRANT que se encuentra arriba para asegurarse de que no haya caracteres erráticos o marcas.

Si tiene errores de SSL, vuelva a leer esta guía para asegurarse de que los pasos estén ordenados.

Esto funcionó en RHEL7 y probablemente también funcionará en CentOS7. No se puede confirmar si estos pasos exactos funcionarán en otro lugar.

Espero que esto le ahorre a alguien un poco de tiempo y molestia.

Lea Configuración de la conexión SSL en línea:
https://riptutorial.com/es/mysql/topic/7563/configuracion-de-la-conexion-ssl

Capítulo 18: Configuración y puesta a punto.

Observaciones

La configuración se realiza en una de las 3 formas:

- opciones de línea de comando
- el archivo de configuración `my.cnf`
- configurando variables desde dentro del servidor

Las opciones de la línea de comandos toman la forma `mysqld --long-parameter-name=value --another-parameter` . Los mismos parámetros se pueden colocar en el archivo de configuración `my.conf` . *Algunos* parámetros son configurables usando variables del sistema desde MySQL. Consulte la documentación oficial para obtener una lista completa de los parámetros.

Las variables pueden tener guión `-` o subrayado `_` . Pueden existir espacios alrededor del `=` . Los números grandes pueden ser sufijados por `K` , `M` , `G` para kilo, mega y giga. Un ajuste por línea.

Indicadores: por lo general, `ON` y `1` son sinónimos, lo mismo para `OFF` y `0` . Algunas banderas no tienen nada tras ellas.

Al colocar la configuración en `my.cnf` , todas las configuraciones para el *servidor* deben estar en la sección `[mysqld]` , así que no agregue ciegamente la configuración al final del archivo. (Nota: para las herramientas que permiten que varias instancias de mysql compartan un my.cnf, los nombres de las secciones pueden ser diferentes).

Examples

Rendimiento InnoDB

Hay cientos de configuraciones que se pueden colocar en my.cnf. Para el usuario 'lite' de MySQL, no importarán tanto.

Una vez que su base de datos se convierte en no trivial, es recomendable establecer los siguientes parámetros:

```
innodb_buffer_pool_size
```

Esto debe configurarse en aproximadamente el 70% de la RAM *disponible* (si tiene al menos 4 GB de RAM; un porcentaje menor si tiene una máquina virtual pequeña o una máquina antigua). La configuración controla la cantidad de caché utilizada por el MOTOR InnoDB. Por lo tanto, es muy importante para el rendimiento de InnoDB.

Parámetro para permitir la inserción de grandes datos.

Si necesita almacenar imágenes o videos en la columna, debemos cambiar el valor según lo

requiera su aplicación

max_allowed_packet = 10M

M es Mb, G en Gb, K en Kb

Aumente el límite de cadena para group_concat

`group_concat` se utiliza para concatenar valores no nulos en un `group` . La longitud máxima de la cadena resultante se puede establecer utilizando la opción `group_concat_max_len` :

```
SET [GLOBAL | SESSION] group_concat_max_len = val;
```

La configuración de la variable `GLOBAL` garantizará un cambio permanente, mientras que la configuración de la variable `SESSION` establecerá el valor para la sesión actual.

Configuración mínima de InnoDB

Esta es una configuración mínima para los servidores MySQL que usan tablas InnoDB. Usando InnoDB, no se requiere el caché de consulta. Recupere espacio en el disco cuando una tabla o base de datos se `DROP` ed. Si está usando SSD, el lavado es una operación redundante (los SDD no son secuenciales).

```
default_storage_engine = InnoDB
query_cache_type = 0
innodb_file_per_table = 1
innodb_flush_neighbors = 0
```

Concurrencia

Asegúrese de que podamos crear más de los 4 hilos predeterminados configurando `innodb_thread_concurrency` en infinito (0); Esto permite que InnoDB decida basándose en la ejecución óptima.

```
innodb_thread_concurrency = 0
innodb_read_io_threads = 64
innodb_write_io_threads = 64
```

Utilización del disco duro

Configure la capacidad (carga normal) y la capacidad_max (máximo absoluto) de IOPS para MySQL. El valor predeterminado de 200 está bien para las unidades de disco duro, pero en estos días, con SSD con capacidad para miles de IOPS, es probable que desee ajustar este número. Hay muchas pruebas que puede ejecutar para determinar IOPS. Los valores anteriores deberían ser casi ese límite *si está ejecutando un servidor MySQL dedicado* . Si está ejecutando otros servicios en la misma máquina, debe distribuirlos según corresponda.

```
innodb_io_capacity = 2500
innodb_io_capacity_max = 3000
```

Utilización de RAM

Establecer la memoria RAM disponible para MySQL. Si bien la regla general es del 70-80%, esto realmente depende de si su instancia está dedicada o no a MySQL y de cuánta RAM está disponible. No *desperdicies* RAM (es decir, recursos) si tienes mucho disponible.

```
innodb_buffer_pool_size = 10G
```

Cifrado seguro de MySQL

El cifrado predeterminado `aes-128-ecb` utiliza el `aes-128-ecb` electrónico (ECB), que es inseguro y nunca debe usarse. En su lugar, agregue lo siguiente a su archivo de configuración:

```
block_encryption_mode = aes-256-cbc
```

Lea Configuración y puesta a punto. en línea:
https://riptutorial.com/es/mysql/topic/3134/configuracion-y-puesta-a-punto-

Capítulo 19: Conjuntos de caracteres y colaciones

Examples

Declaración

```
CREATE TABLE foo ( ...
    name CHARACTER SET utf8mb4
    ... );
```

Conexión

Es vital para el uso de conjuntos de caracteres decirle al servidor MySQL qué es lo que codifica los bytes del cliente. Aquí hay una forma:

```
SET NAMES utf8mb4;
```

Cada idioma (PHP, Python, Java, ...) tiene su propia forma en la que generalmente es preferible establecer `SET NAMES` .

Por ejemplo: `SET NAMES utf8mb4` , junto con una columna declarada `CHARACTER SET latin1` : se convertirá de latin1 a utf8mb4 al `INSERTing` y `INSERTing` convertir cuando `SELECTing` .

¿Qué conjunto de personajes y colección?

Hay docenas de juegos de caracteres con cientos de colaciones. (Una intercalación dada pertenece a un solo conjunto de caracteres). Vea la salida de `SHOW COLLATION;` .

Por lo general, solo hay 4 `CHARACTER SETs` que importan:

```
ascii -- basic 7-bit codes.
latin1 -- ascii, plus most characters needed for Western European languages.
utf8 -- the 1-, 2-, and 3-byte subset of utf8.  This excludes Emoji and some of Chinese.
utf8mb4 -- the full set of UTF8 characters, covering all current languages.
```

Todos incluyen caracteres ingleses, codificados idénticamente. utf8 es un subconjunto de utf8mb4.

Mejores prácticas...

- Use utf8mb4 para cualquier columna `TEXT` o `VARCHAR` que pueda tener una variedad de idiomas.
- Use ascii (latin1 está bien) para cadenas hexadecimales (UUID, MD5, etc.) y códigos simples (código de país, código postal, etc.).

utf8mb4 no existía hasta la versión 5.5.3, por lo que utf8 era el mejor disponible antes de eso.

Fuera de MySQL , "UTF8" significa lo mismo que utf8mb4 de MySQL, no utf8 de MySQL.

Las colaciones comienzan con el nombre del conjunto de caracteres y generalmente terminan con `_ci` para "insensibles a mayúsculas y minúsculas" o `_bin` para "simplemente comparar los bits.

La 'última' compilación utf8mb4 es `utf8mb4_unicode_520_ci` , basada en Unicode 5.20. Si está trabajando con un solo idioma, es posible que desee, por ejemplo, `utf8mb4_polish_ci` , que reorganizará las letras ligeramente, según las convenciones polacas.

Configuración de conjuntos de caracteres en tablas y campos

Puede establecer un conjunto de caracteres tanto por tabla como por campo individual utilizando las sentencias `CHARACTER SET` y `CHARSET` :

```
CREATE TABLE Address (
    `AddressID`    INTEGER NOT NULL PRIMARY KEY,
    `Street`       VARCHAR(80) CHARACTER SET ASCII,
    `City`         VARCHAR(80),
    `Country`      VARCHAR(80) DEFAULT "United States",
    `Active`       BOOLEAN DEFAULT 1,
) Engine=InnoDB default charset=UTF8;
```

`City` y `Country` usarán `UTF8` , ya que lo configuramos como el conjunto de caracteres predeterminado para la tabla. `Street` por otro lado, usará `ASCII` , como le hemos dicho específicamente para que lo haga.

La configuración del conjunto de caracteres correcto depende en gran medida de su conjunto de datos, pero también puede mejorar la portabilidad entre los sistemas que trabajan con sus datos.

Lea Conjuntos de caracteres y colaciones en línea:
https://riptutorial.com/es/mysql/topic/4569/conjuntos-de-caracteres-y-colaciones

Capítulo 20: Consejos de rendimiento Mysql

Examples

Seleccione la optimización de la declaración

A continuación hay algunos consejos para recordar mientras escribimos una consulta de selección en MySQL que puede ayudarnos y reducir nuestro tiempo de consulta:

1. Siempre que usemos en una tabla grande, debemos asegurarnos de que la columna en la cláusula donde está indexada o no. Ej .: - Seleccione * del empleado donde id_usuario> 2000. id_usuario si está indexado y luego acelerará la evaluación de la consulta. Los índices también son muy importantes durante las combinaciones y las claves externas.

2. Cuando necesite la sección más pequeña de contenido en lugar de recuperar datos completos de la tabla, intente usar el límite. En lugar de escribir Ex: - Seleccione * del empleado. Si solo necesita los primeros 20 empleados de lakhs, simplemente use el límite Ej .: - Seleccione * del límite de empleados 20.

3. También puede optimizar su consulta proporcionando el nombre de columna que desea en el conjunto de resultados. En lugar de escribir Ex: - Seleccione * del empleado. Solo mencione el nombre de la columna de la que necesita datos si su tabla tiene mucha columna y desea tener datos para algunos de ellos. Ej: - Seleccione id, nombre del empleado.

4. Columna de índice si está utilizando para verificar NULL en la cláusula where. Si tiene alguna declaración como SELECT * FROM tbl_name WHERE key_col IS NULL; luego, si key_col está indexado, la consulta se evaluará más rápido.

Optimización del diseño de almacenamiento para tablas InnoDB

1. En InnoDB, tener una LLAVE PRINCIPAL larga (ya sea una sola columna con un valor largo o varias columnas que forman un valor compuesto largo) desperdicia una gran cantidad de espacio en el disco. El valor de clave principal para una fila se duplica en todos los registros de índice secundarios que apuntan a la misma fila. Cree una columna AUTO_INCREMENT como clave principal si su clave principal es larga.
2. Utilice el tipo de datos VARCHAR en lugar de CHAR para almacenar cadenas de longitud variable o para columnas con muchos valores NULL. Una columna CHAR (N) siempre toma N caracteres para almacenar datos, incluso si la cadena es más corta o su valor es NULL. Las tablas más pequeñas encajan mejor en el grupo de búferes y reducen la E / S del disco.

   Cuando se utiliza el formato de fila COMPACT (el formato InnoDB predeterminado) y los juegos de caracteres de longitud variable, como utf8 o sjis, las columnas CHAR (N) ocupan una cantidad variable de espacio, pero aún así tienen al menos N bytes.

3. Para las tablas que son grandes, o que contienen gran cantidad de texto repetitivo o datos

numéricos, considere usar el formato de fila COMPRIMIDO. Se requiere menos E / S de disco para traer datos al grupo de búferes o para realizar exploraciones de tabla completas. Antes de tomar una decisión permanente, mida la cantidad de compresión que puede lograr utilizando el formato de fila COMPRIMIDO versus COMPACTO. *Advertencia: los* puntos de referencia rara vez se muestran mejor que la compresión 2: 1 y hay una gran cantidad de sobrecarga en el buffer_pool para COMPRESSED.

4. Una vez que sus datos alcancen un tamaño estable, o una tabla de crecimiento haya aumentado en decenas o en algunos cientos de megabytes, considere usar la instrucción OPTIMIZAR TABLA para reorganizar la tabla y compactar el espacio desperdiciado. Las tablas reorganizadas requieren menos E / S de disco para realizar exploraciones de tabla completas. Esta es una técnica sencilla que puede mejorar el rendimiento cuando otras técnicas, como mejorar el uso del índice o ajustar el código de la aplicación, no son prácticas. *Advertencia* : independientemente del tamaño de la mesa, OPTIMIZAR TABLA solo se debe realizar raramente. Esto se debe a que es costoso y rara vez mejora la tabla lo suficiente como para que valga la pena. InnoDB es razonablemente bueno para mantener sus árboles B + libres de una gran cantidad de espacio desperdiciado.

   OPTIMIZE TABLE copia la parte de datos de la tabla y reconstruye los índices. Los beneficios provienen de un mejor empaquetamiento de los datos dentro de los índices y una fragmentación reducida dentro de los espacios de tablas y en el disco. Los beneficios varían dependiendo de los datos en cada tabla. Puede encontrar que hay ganancias significativas para algunos y no para otros, o que las ganancias disminuyen con el tiempo hasta que luego optimice la tabla. Esta operación puede ser lenta si la tabla es grande o si los índices que se están reconstruyendo no encajan en el grupo de búferes. La primera ejecución después de agregar una gran cantidad de datos a una tabla suele ser mucho más lenta que las posteriores.

Construyendo un índice compuesto

En muchas situaciones, un índice compuesto funciona mejor que un índice con una sola columna. Para crear un índice compuesto óptimo, rellénelo con columnas en este orden.

- `=` columna (s) de la cláusula `WHERE` primero. (por ejemplo, `INDEX(a,b,...)` para `WHERE a=12 AND b='xyz' ...`)
- `IN` columna (s); El optimizador puede ser capaz de saltar a través del índice.
- Un "rango" (por ejemplo, `x BETWEEN 3 AND 9` , `name LIKE 'J%'`) No usará nada más allá de la primera columna de rango.
- Todas las columnas en `GROUP BY` , en orden.
- Todas las columnas en `ORDER BY` , en orden. Funciona solo si todos son `ASC` o todos son `DESC` o si está usando 8.0.

Notas y excepciones:

- No duplique ninguna columna.
- Omita cualquier caso que no aplique.
- Si no usa todas las columnas de `WHERE` , no es necesario pasar a `GROUP BY` , etc.
- Hay casos en los que es útil indexar solo las columnas `ORDER BY` , ignorando `WHERE` .

- No "esconda" una columna en una función (por ejemplo, `DATE(x) = ...` no se puede usar `x` en el índice).
- Es poco probable que la indexación de 'Prefijo' (por ejemplo, `text_col(99)`) sea útil; puede doler

Más detalles y consejos .

Lea Consejos de rendimiento Mysql en línea: https://riptutorial.com/es/mysql/topic/5752/consejos-de-rendimiento-mysql

Capítulo 21: Consultas de pivote

Observaciones

La creación de consultas dinámicas en MySQL se basa en la función `GROUP_CONCAT()` . Si se espera que el resultado de la expresión que crea las columnas de la consulta dinámica sea grande, el valor de la variable `group_concat_max_len` debe aumentarse:

```
set session group_concat_max_len = 1024 * 1024; -- This should be enough for most cases
```

Examples

Creando una consulta dinámica

MySQL no proporciona una forma integrada para crear consultas dinámicas. Sin embargo, estos pueden ser creados usando declaraciones preparadas.

Supongamos la tabla `tbl_values` :

Carné de identidad	Nombre	Grupo	Valor
1	Pete	UNA	10
2	Pete	segundo	20
3	Juan	UNA	10

Solicitud: cree una consulta que muestre la suma del `Value` para cada `Name` ; el `Group` debe ser el encabezado de la columna y el `Name` debe ser el encabezado de la fila.

```
-- 1. Create an expression that builds the columns
set @sql = (
    select group_concat(distinct
        concat(
            "sum(case when `Group`='", Group, "' then `Value` end) as `", `Group`, "`"
        )
    )
    from tbl_values
);

-- 2. Complete the SQL instruction
set @sql = concat("select Name, ", @sql, " from tbl_values group by `Name`");

-- 3. Create a prepared statement
prepare stmt from @sql;

-- 4. Execute the prepared statement
execute stmt;
```

Resultado:

Nombre	UNA	segundo
Juan	10	NULO
Pete	10	20

Importante: desasigne la declaración preparada una vez que ya no sea necesaria:

```
deallocate prepare stmt;
```

Ejemplo en SQL Fiddle

Lea Consultas de pivote en línea: https://riptutorial.com/es/mysql/topic/3074/consultas-de-pivote

Capítulo 22: Conversión de MyISAM a InnoDB

Examples

Conversión básica

```
ALTER TABLE foo ENGINE=InnoDB;
```

Esto convierte la tabla, pero no se ocupa de las diferencias entre los motores. La mayoría de las diferencias no importarán, especialmente para mesas pequeñas. Pero para tablas más ocupadas, otras consideraciones deben ser consideradas. *Consideraciones de conversión*

Convertir todas las tablas en una base de datos

Para convertir fácilmente todas las tablas en una base de datos, use lo siguiente:

```
SET @DB_NAME = DATABASE();

SELECT  CONCAT('ALTER TABLE `', table_name, '` ENGINE=InnoDB;') AS sql_statements
FROM    information_schema.tables
WHERE   table_schema = @DB_NAME
AND     `ENGINE` = 'MyISAM'
AND     `TABLE_TYPE` = 'BASE TABLE';
```

> **NOTA:** Debería estar conectado a su base de datos para que la función `DATABASE()` funcione, de lo contrario, devolverá `NULL` . Esto se aplica principalmente al cliente mysql estándar que se envía con el servidor, ya que permite conectarse sin especificar una base de datos.

Ejecute esta instrucción SQL para recuperar todas las tablas `MyISAM` en su base de datos.

Finalmente, copie la salida y ejecute las consultas SQL desde ella.

Lea Conversión de MyISAM a InnoDB en línea:
https://riptutorial.com/es/mysql/topic/3135/conversion-de-myisam-a-innodb

Capítulo 23: Copia de seguridad utilizando mysqldump

Sintaxis

- mysqldump -u [nombre de usuario] -p [contraseña] [otras opciones] db_name> dumpFileName.sql /// Para respaldar una base de datos única
- mysqldump -u [nombre de usuario] -p [contraseña] [otras opciones] db_name [tbl_name1 tbl_name2 tbl_name2 ...]> dumpFileName.sql /// Para respaldar una o más tablas
- mysqldump -u [nombre de usuario] -p [contraseña] [otras opciones] --databases db_name1 db_name2 db_name3 ...> dumpFileName.sql /// Para hacer una copia de seguridad de una o más bases de datos completas
- mysqldump -u [nombre de usuario] -p [contraseña] [otras opciones] --todos-bases de datos> dumpFileName.sql /// Para respaldar todo el servidor MySQL

Parámetros

Opción	Efecto
-	# Opciones de inicio de sesión del servidor
`-h` (`--host`)	Host (dirección IP o nombre de host) para conectarse. El valor predeterminado es `localhost` (`127.0.0.1`) Ejemplo: `-h localhost`
`-u` (`--user`)	Usuario de MySQL
`-p` (`--password`)	Contraseña MySQL. **Importante** : Al usar `-p` , no debe haber un espacio entre la opción y la contraseña. Ejemplo: `-pMyPassword`
-	# Opciones de volcado
`--add-drop-database`	Agregue una instrucción `DROP DATABASE` antes de cada instrucción `CREATE DATABASE` . Útil si quieres reemplazar bases de datos en el servidor.
`--add-drop-table`	Agregue una instrucción `DROP TABLE` antes de cada instrucción `CREATE TABLE` . Útil si quieres reemplazar tablas en el servidor.
`--no-create-db`	Suprimir las `CREATE DATABASE` en el volcado. Esto es útil cuando está seguro de que las bases de datos que está volcando ya existen en el servidor donde cargará el volcado.
`-t` (`--no-create-info`)	Suprimir todas las declaraciones `CREATE TABLE` en el volcado. Esto es útil cuando desea volcar solo los datos de las tablas y usará el archivo de volcado para rellenar tablas idénticas en otra base de datos / servidor.

Opción	Efecto
`-d` (`--no-data`)	No escriba información de la tabla. Esto solo volcará las `CREATE TABLE` . Útil para crear bases de datos de "plantillas"
`-R` (`--routines`)	Incluir procedimientos / funciones almacenados en el volcado.
`-K` (`--disable-keys`)	Desactive las claves para cada tabla antes de insertar los datos y habilite las claves después de insertar los datos. Esto acelera las inserciones solo en tablas MyISAM con índices no únicos.

Observaciones

La salida de una operación `mysqldump` es un archivo ligeramente comentado que contiene sentencias de SQL secuenciales que son compatibles con la versión de las utilidades de MySQL que se usaron para generarlo (con atención prestada a la compatibilidad con versiones anteriores, pero sin garantía para las futuras). Por lo tanto, la restauración de una base de datos `mysqldump` ed comprende la ejecución de esas declaraciones. En general, este archivo

- `DROP` es la primera tabla o vista especificada
- `CREATE` esa tabla o vista
- Para tablas volcadas con datos (es decir, sin la `--no-data`)
    - `LOCK` la mesa
    - `INSERT` s todas las filas de la tabla original en una declaración
- `UNLOCK TABLES`
- Repite lo anterior para todas las demás tablas y vistas.
- `DROP` s la primera rutina incluida
- `CREATE` s esa rutina
- Repite lo mismo para todas las demás rutinas.

La presencia del `DROP` antes de `CREATE` para cada tabla significa que si el esquema está presente, ya sea que esté vacío o no, el uso de un archivo `mysqldump` para su restauración completará o sobrescribirá los datos que contiene.

Examples

Creación de una copia de seguridad de una base de datos o tabla

Crear una instantánea de una base de datos completa:

```
mysqldump [options] db_name > filename.sql
```

Crea una instantánea de múltiples bases de datos:

```
mysqldump [options] --databases db_name1 db_name2 ... > filename.sql
mysqldump [options] --all-databases > filename.sql
```

Cree una instantánea de una o más tablas:

```
mysqldump [options] db_name table_name... > filename.sql
```

Cree una instantánea *excluyendo* una o más tablas:

```
mysqldump [options] db_name --ignore-table=tbl1 --ignore-table=tbl2 ... > filename.sql
```

La extensión de archivo `.sql` es totalmente una cuestión de estilo. Cualquier extensión funcionaría.

Especificando nombre de usuario y contraseña

```
> mysqldump -u username -p [other options]
Enter password:
```

Si necesita especificar la contraseña en la línea de comando (por ejemplo, en un script), puede agregarla después de la opción `-p` *sin* un espacio:

```
> mysqldump -u username -ppassword [other options]
```

Si la contraseña contiene espacios o caracteres especiales, recuerde utilizar el escape en función de su shell / sistema.

Opcionalmente la forma extendida es:

```
> mysqldump --user=username --password=password [other options]
```

(La Explicidad que especifica la contraseña en la línea de comandos no es Recomendada debido a problemas de seguridad).

Restaurar una copia de seguridad de una base de datos o tabla

```
mysql [options] db_name < filename.sql
```

Tenga en cuenta que:

- `db_name` debe ser una base de datos existente;
- su usuario autenticado tiene privilegios suficientes para ejecutar todos los comandos dentro de su `filename.sql` ;
- La extensión de archivo `.sql` es totalmente una cuestión de estilo. Cualquier extensión funcionaría.
- No puede especificar un nombre de tabla para cargar, aunque podría especificar uno para volcar desde. Esto debe hacerse dentro de `filename.sql` .

Alternativamente, cuando esté en la **herramienta de línea de comandos de MySQL** , puede restaurar (o ejecutar cualquier otro script) usando el comando de origen:

```
source filename.sql
```

o

```
\. filename.sql
```

mysqldump desde un servidor remoto con compresión

Con el fin de utilizar la compresión sobre el alambre para una transferencia más rápida, pasar el `--compress` opción de `mysqldump` . Ejemplo:

```
mysqldump -h db.example.com -u username -p --compress dbname > dbname.sql
```

Importante: si no desea bloquear la base de datos de *origen* , también debe incluir `--lock-tables=false` . Pero puede que no obtengas una imagen de db internamente consistente de esa manera.

Para guardar también el archivo comprimido, puede canalizar a `gzip` .

```
mysqldump -h db.example.com -u username -p --compress dbname | gzip --stdout > dbname.sql.gz
```

restaura un archivo mysqldump comprimido sin descomprimir

```
gunzip -c dbname.sql.gz | mysql dbname -u username -p
```

Nota: `-c` significa salida de escritura a la salida estándar.

Copia de seguridad directa a Amazon S3 con compresión

Si desea realizar una copia de seguridad completa de una gran instalación de MySql y no tiene suficiente almacenamiento local, puede volcarla y comprimirla directamente en un depósito de Amazon S3. También es una buena práctica hacer esto sin tener la contraseña DB como parte del comando:

```
mysqldump -u root -p --host=localhost --opt --skip-lock-tables --single-transaction \
        --verbose --hex-blob --routines --triggers --all-databases |
    gzip -9 | s3cmd put - s3://s3-bucket/db-server-name.sql.gz
```

Se le solicitará la contraseña, después de lo cual se iniciará la copia de seguridad.

Transferencia de datos de un servidor MySQL a otro

Si necesita copiar una base de datos de un servidor a otro, tiene dos opciones:

Opción 1:

1. Almacena el archivo de volcado en el servidor de origen

2. Copie el archivo de volcado a su servidor de destino
3. Cargue el archivo de volcado en su servidor de destino

En el servidor de origen:

```
mysqldump [options] > dump.sql
```

En el servidor de destino, copie el archivo de volcado y ejecute:

```
mysql [options] < dump.sql
```

Opcion 2:

Si el servidor de destino puede conectarse al servidor host, puede usar una canalización para copiar la base de datos de un servidor a otro:

En el servidor de destino

```
mysqldump [options to connect to the source server] | mysql [options]
```

Del mismo modo, el script podría ejecutarse en el servidor de origen, empujando hacia el destino. En cualquier caso, es probable que sea significativamente más rápido que la Opción 1.

Base de datos de copia de seguridad con procedimientos almacenados y funciones

De manera predeterminada, los procedimientos y funciones almacenados o no generados por `mysqldump` , deberá agregar el parámetro `--routines` (o `-R`):

```
mysqldump -u username -p -R db_name > dump.sql
```

Al utilizar `--routines` las `--routines` creación y cambio no se mantienen, en su lugar, debe volcar y volver a cargar el contenido de `mysql.proc` .

Lea Copia de seguridad utilizando mysqldump en línea:
https://riptutorial.com/es/mysql/topic/604/copia-de-seguridad-utilizando-mysqldump

Capítulo 24: Creación de tablas

Sintaxis

- CREATE TABLE table_name (column_name1 data_type (tamaño), column_name2 data_type (tamaño), column_name3 data_type (tamaño), ...); // Creación de tablas básicas
- CREATE TABLE table_name [SI NO EXISTE] (column_name1 data_type (tamaño), column_name2 data_type (tamaño), column_name3 data_type (tamaño), ...); // Comprobación de creación de tablas existente
- CREATE [TEMPORARY] TABLE table_name [IF NOT EXISTS] (column_name1 data_type (tamaño), column_name2 data_type (tamaño), column_name3 data_type (tamaño), ...); // Creación de tablas temporales
- CREAR TABLA new_tbl [AS] SELECT * FROM orig_tbl; // Creación de tablas desde SELECT

Observaciones

La `CREATE TABLE` debe terminar con una especificación `ENGINE` :

```
CREATE TABLE table_name ( column_definitions ) ENGINE=engine;
```

Algunas opciones son:

- `InnoDB` : (Predeterminado desde la versión 5.5.5) Es un motor seguro para las transacciones (compatible con ACID). Tiene compromiso de transacción y retroceso, y capacidades de recuperación de fallas y bloqueo a nivel de fila.
- `MyISAM` : (Predeterminado antes de la versión 5.5.5) Es un motor simple y rápido. No admite transacciones, ni claves externas, pero es útil para el almacenamiento de datos.
- `Memory` : Almacena todos los datos en la RAM para operaciones extremadamente rápidas, pero la fecha de la tabla se perderá al reiniciar la base de datos.

Más opciones de motor aquí .

Examples

Creación básica de tablas

La `CREATE TABLE` se usa para crear una tabla en una base de datos MySQL.

```
CREATE TABLE Person (
    `PersonID`      INTEGER NOT NULL PRIMARY KEY,
    `LastName`      VARCHAR(80),
    `FirstName`     VARCHAR(80),
```

```
    `Address`       TEXT,
    `City`          VARCHAR(100)
) Engine=InnoDB;
```

Cada definición de campo debe tener:

1. Nombre de campo: Un nombre de campo válido. Asegúrate de incluir los nombres en `-chars. Esto asegura que puede usar, por ejemplo, espacios en el nombre de campo.
2. Tipo de datos [Longitud]: si el campo es `CHAR` o `VARCHAR` , es obligatorio especificar la longitud del campo.
3. Atributos `NULL` | `NOT NULL` : Si se especifica `NOT NULL` , cualquier intento de almacenar un valor `NULL` en ese campo fallará.
4. Vea más sobre los tipos de datos y sus atributos aquí .

`Engine=...` es un parámetro opcional que se usa para especificar el motor de almacenamiento de la tabla. Si no se especifica ningún motor de almacenamiento, la tabla se creará utilizando el motor de almacenamiento de tablas predeterminado del servidor (generalmente InnoDB o MyISAM).

Configuración de los valores predeterminados

Además, donde tenga sentido, puede establecer un valor predeterminado para cada campo usando `DEFAULT` :

```
CREATE TABLE Address (
    `AddressID`   INTEGER NOT NULL PRIMARY KEY,
    `Street`      VARCHAR(80),
    `City`        VARCHAR(80),
    `Country`     VARCHAR(80) DEFAULT "United States",
    `Active`      BOOLEAN DEFAULT 1,
) Engine=InnoDB;
```

Si durante las inserciones no se especifica la `Street` , ese campo será `NULL` cuando se recupere. Cuando no se especifica ningún `Country` al insertarlo, se establecerá de forma predeterminada en "Estados Unidos".

Puede establecer valores predeterminados para todos los tipos de columna, excepto para los campos `BLOB` , `TEXT` , `GEOMETRY` y `JSON` .

Creación de tablas con clave primaria

```
CREATE TABLE Person (
    PersonID     INT UNSIGNED NOT NULL,
    LastName     VARCHAR(66) NOT NULL,
    FirstName    VARCHAR(66),
    Address      VARCHAR(255),
    City         VARCHAR(66),
```

```
    PRIMARY KEY (PersonID)
);
```

Una **clave principal** es un identificador único o de varias columnas `NOT NULL` que identifica de forma única una fila de una tabla. Se crea un índice , y si no se declara explícitamente como `NOT NULL` , MySQL los declarará de manera silenciosa e implícita.

Una tabla solo puede tener una `PRIMARY KEY` , y se recomienda que cada tabla tenga una. InnoDB creará automáticamente uno en su ausencia (como se ve en la documentación de MySQL), aunque esto es menos deseable.

A menudo, una `INT AUTO_INCREMENT` también conocida como "clave sustituta", se utiliza para la optimización de índice delgado y las relaciones con otras tablas. Este valor (normalmente) aumentará en 1 cada vez que se agregue un nuevo registro, a partir de un valor predeterminado de 1.

Sin embargo, a pesar de su nombre, no es su propósito garantizar que los valores sean incrementales, simplemente que sean secuenciales y únicos.

Un valor `INT` incremento automático no se restablecerá a su valor de inicio predeterminado si se eliminan todas las filas de la tabla, a menos que la tabla se trunca utilizando la instrucción `TRUNCATE TABLE` .

Definiendo una columna como Clave Primaria (definición en línea)

Si la clave principal consta de una sola columna, la cláusula `PRIMARY KEY` se puede colocar en línea con la definición de la columna:

```
CREATE TABLE Person (
    PersonID     INT UNSIGNED NOT NULL PRIMARY KEY,
    LastName     VARCHAR(66) NOT NULL,
    FirstName    VARCHAR(66),
    Address      VARCHAR(255),
    City         VARCHAR(66)
);
```

Esta forma del comando es más corta y más fácil de leer.

Definir una clave primaria de varias columnas

También es posible definir una clave principal que comprenda más de una columna. Esto se puede hacer, por ejemplo, en la tabla secundaria de una relación de clave externa. Una clave primaria de varias columnas se define enumerando las columnas participantes en una cláusula `PRIMARY KEY` separada. La sintaxis en línea no está permitida aquí, ya que solo una columna puede declararse `PRIMARY KEY` línea. Por ejemplo:

```
CREATE TABLE invoice_line_items (
    LineNum      SMALLINT UNSIGNED NOT NULL,
    InvoiceNum   INT UNSIGNED NOT NULL,
    -- Other columns go here
    PRIMARY KEY (InvoiceNum, LineNum),
    FOREIGN KEY (InvoiceNum) REFERENCES -- references to an attribute of a table
);
```

Tenga en cuenta que las columnas de la clave principal *deben* especificarse en orden de clasificación lógica, que *puede* ser diferente del orden en que se definieron las columnas, como en el ejemplo anterior.

Los índices más grandes requieren más espacio en disco, memoria y E / S. Por lo tanto, las claves deben ser lo más pequeñas posible (especialmente con respecto a las claves compuestas). En InnoDB, cada 'índice secundario' incluye una copia de las columnas de la `PRIMARY KEY` .

Creación de tablas con clave externa

```
CREATE TABLE Account (
    AccountID     INT UNSIGNED NOT NULL,
    AccountNo     INT UNSIGNED NOT NULL,
    PersonID    INT UNSIGNED,
    PRIMARY KEY (AccountID),
    FOREIGN KEY (PersonID) REFERENCES Person (PersonID)
) ENGINE=InnoDB;
```

Clave externa: una clave externa (`FK`) es una columna única o un compuesto de columnas de varias columnas en una tabla de *referencia* . Se confirma que este `FK` existe en la tabla *referenciada* . Se recomienda encarecidamente que la clave de la tabla a la que se hace *referencia* confirme que el `FK` sea una clave principal, pero no se aplica. Se utiliza como una búsqueda rápida en la *referencia* en la que no es necesario que sea único y, de hecho, puede ser un índice más a la izquierda.

Las relaciones de clave externa implican una tabla principal que contiene los valores de datos centrales y una tabla secundaria con valores idénticos que apuntan a su principal. La cláusula FOREIGN KEY se especifica en la tabla secundaria. Las tablas padre e hijo deben usar el mismo motor de almacenamiento. No deben ser tablas TEMPORALES .

Las columnas correspondientes en la clave externa y la clave a la que se hace referencia deben tener tipos de datos similares. El tamaño y el signo de los tipos enteros deben ser iguales. La longitud de los tipos de cadena no tiene por qué ser la misma. Para las columnas de cadena no binaria (carácter), el conjunto de caracteres y la intercalación deben ser los mismos.

Nota: las restricciones de clave foránea son compatibles con el motor de almacenamiento InnoDB (no MyISAM o MEMORY). Las configuraciones de bases de datos que usan otros motores aceptarán esta declaración `CREATE TABLE` pero no respetarán las restricciones de clave externa. (Aunque las versiones más nuevas de MySQL están predeterminadas en `InnoDB` , pero es una buena práctica ser explícitas).

Clonando una tabla existente

Una tabla se puede replicar de la siguiente manera:

```
CREATE TABLE ClonedPersons LIKE Persons;
```

La nueva tabla tendrá exactamente la misma estructura que la tabla original, incluidos los índices y los atributos de columna.

Además de crear una tabla manualmente, también es posible crear una tabla seleccionando datos de otra tabla:

```
CREATE TABLE ClonedPersons SELECT * FROM Persons;
```

Puede usar cualquiera de las características normales de una declaración `SELECT` para modificar los datos a medida que avanza:

```
CREATE TABLE ModifiedPersons
SELECT PersonID, FirstName + LastName AS FullName FROM Persons
WHERE LastName IS NOT NULL;
```

Las claves primarias y los índices no se conservarán al crear tablas desde `SELECT` . Debes redeclarlos:

```
CREATE TABLE ModifiedPersons (PRIMARY KEY (PersonID))
SELECT PersonID, FirstName + LastName AS FullName FROM Persons
WHERE LastName IS NOT NULL;
```

CREAR TABLA DESDE SELECCIONAR

Puede crear una tabla a partir de otra agregando una instrucción `SELECT` al final de la instrucción `CREATE TABLE` :

```
CREATE TABLE stack (
    id_user INT,
    username VARCHAR(30),
    password VARCHAR(30)
);
```

Crea una tabla en la misma base de datos:

```
-- create a table from another table in the same database with all attributes
CREATE TABLE stack2 AS SELECT * FROM stack;

-- create a table from another table in the same database with some attributes
CREATE TABLE stack3 AS SELECT username, password FROM stack;
```

Crear tablas desde diferentes bases de datos:

```
-- create a table from another table from another database with all attributes
CREATE TABLE stack2 AS SELECT * FROM second_db.stack;

-- create a table from another table from another database with some attributes
CREATE TABLE stack3 AS SELECT username, password FROM second_db.stack;
```

nótese bien

Para crear una tabla de otra tabla que exista en otra base de datos, debe especificar el nombre de la base de datos de esta manera:

```
FROM NAME_DATABASE.name_table
```

Mostrar estructura de tabla

Si desea ver la información de esquema de su tabla, puede utilizar uno de los siguientes:

```
SHOW CREATE TABLE child; -- Option 1

CREATE TABLE `child` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `fullName` varchar(100) NOT NULL,
  `myParent` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `mommy_daddy` (`myParent`),
  CONSTRAINT `mommy_daddy` FOREIGN KEY (`myParent`) REFERENCES `parent` (`id`)
      ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Si se usa desde la herramienta de línea de comandos mysql, esto es menos detallado:

```
SHOW CREATE TABLE child \G
```

Una forma menos descriptiva de mostrar la estructura de la tabla:

```
mysql> CREATE TABLE Tab1(id int, name varchar(30));
Query OK, 0 rows affected (0.03 sec)

mysql> DESCRIBE Tab1; -- Option 2

+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(11)     | YES  |     | NULL    |       |
| name  | varchar(30) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

Tanto **DESCRIBE** como **DESC** dan el mismo resultado.

Para ver `DESCRIBE` ejecutado en todas las tablas en una base de datos a la vez, vea este Ejemplo .

Tabla Crear con la columna TimeStamp para mostrar la última actualización

La columna TIMESTAMP mostrará cuándo se actualizó por última vez la fila.

```
CREATE TABLE `TestLastUpdate` (
    `ID` INT NULL,
    `Name` VARCHAR(50) NULL,
    `Address` VARCHAR(50) NULL,
    `LastUpdate` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)
COMMENT='Last Update'
;
```

Lea Creación de tablas en línea: https://riptutorial.com/es/mysql/topic/795/creacion-de-tablas

Capítulo 25: Creando bases de datos

Sintaxis

- CREAR {BASE DE DATOS | SCHEMA} [IF NOT EXISTS] db_name [create_specification] /// Para crear la base de datos
- DROP {BASE DE DATOS | SCHEMA} [IF EXISTS] db_name /// Para eliminar la base de datos

Parámetros

Parámetro	Detalles
Crear base de datos	Crea una base de datos con el nombre dado.
Crear un esquema	Este es un sinónimo para `CREATE DATABASE`
Si no existe	Se usa para evitar el error de ejecución, si la base de datos especificada ya existe
create_specification	`create_specification` opciones de `create_specification` especifican características de la base de datos como `CHARACTER SET` y `COLLATE` (`COLLATE` base de datos)

Examples

Crear base de datos, usuarios y subvenciones.

Crear una base de datos. Tenga en cuenta que la palabra abreviada SCHEMA se puede usar como sinónimo.

```
CREATE DATABASE Baseball; -- creates a database named Baseball
```

Si la base de datos ya existe, se devuelve el error 1007. Para evitar este error, intente:

```
CREATE DATABASE IF NOT EXISTS Baseball;
```

Similar,

```
DROP DATABASE IF EXISTS Baseball; -- Drops a database if it exists, avoids Error 1008
DROP DATABASE xyz; -- If xyz does not exist, ERROR 1008 will occur
```

Debido a las posibilidades de error anteriores, las declaraciones DDL se usan a menudo con `IF`

`EXISTS` .

Uno puede crear una base de datos con un conjunto de caracteres predeterminado y colación. Por ejemplo:

```
CREATE DATABASE Baseball CHARACTER SET utf8 COLLATE utf8_general_ci;

SHOW CREATE DATABASE Baseball;
+----------+-----------------------------------------------------------------+
| Database | Create Database                                                 |
+----------+-----------------------------------------------------------------+
| Baseball | CREATE DATABASE `Baseball` /*!40100 DEFAULT CHARACTER SET utf8 */ |
+----------+-----------------------------------------------------------------+
```

Vea sus bases de datos actuales:

```
SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| ajax_stuff         |
| Baseball           |
+--------------------+
```

Establezca la base de datos activa actualmente, y vea alguna información:

```
USE Baseball; -- set it as the current database
SELECT @@character_set_database as cset,@@collation_database as col;
+------+-----------------+
| cset | col             |
+------+-----------------+
| utf8 | utf8_general_ci |
+------+-----------------+
```

Lo anterior muestra el conjunto de caracteres y la intercalación predeterminados para la base de datos.

Crear un usuario:

```
CREATE USER 'John123'@'%' IDENTIFIED BY 'OpenSesame';
```

Lo anterior crea un usuario John123, capaz de conectarse con cualquier nombre de host debido al comodín `%` . La contraseña para el usuario se establece en 'OpenSesame', que está en hash.

Y crea otro:

```
CREATE USER 'John456'@'%' IDENTIFIED BY 'somePassword';
```

Muestre que los usuarios se han creado examinando la base de datos especial de `mysql` :

```
SELECT user,host,password from mysql.user where user in ('John123','John456');
```

```
+---------+------+-------------------------------------------+
| user    | host | password                                  |
+---------+------+-------------------------------------------+
| John123 | %    | *E6531C342ED87 ....................       |
| John456 | %    | *B04E11FAAAE9A ....................       |
+---------+------+-------------------------------------------+
```

Tenga en cuenta que en este punto, los usuarios se han creado, pero sin ningún permiso para usar la base de datos de béisbol.

Trabajar con permisos para usuarios y bases de datos. Otorgue derechos al usuario John123 para tener privilegios completos en la base de datos de Béisbol y solo SELECCIONE los derechos para el otro usuario:

```
GRANT ALL ON Baseball.* TO 'John123'@'%';
GRANT SELECT ON Baseball.* TO 'John456'@'%';
```

Verifique lo anterior:

```
SHOW GRANTS FOR 'John123'@'%';
+-----------------------------------------------------------------------------------------------------
-----------+
| Grants for John123@%
|
+-----------------------------------------------------------------------------------------------------
-----------+
| GRANT USAGE ON *.* TO 'John123'@'%' IDENTIFIED BY PASSWORD '*E6531C342ED87
.....................        |
| GRANT ALL PRIVILEGES ON `baseball`.* TO 'John123'@'%'
|
+-----------------------------------------------------------------------------------------------------
-----------+

SHOW GRANTS FOR 'John456'@'%';
+-----------------------------------------------------------------------------------------------------
-----------+
| Grants for John456@%
|
+-----------------------------------------------------------------------------------------------------
-----------+
| GRANT USAGE ON *.* TO 'John456'@'%' IDENTIFIED BY PASSWORD '*B04E11FAAAE9A
.....................        |
| GRANT SELECT ON `baseball`.* TO 'John456'@'%'
|
+-----------------------------------------------------------------------------------------------------
-----------+
```

Tenga en cuenta que el `GRANT USAGE` que siempre verá significa simplemente que el usuario puede iniciar sesión. Eso es todo lo que eso significa.

Mi base de datos

Debe crear su propia base de datos y no utilizar la escritura en ninguna de las bases de datos existentes. Es probable que esta sea una de las primeras cosas que se deben hacer después de

conectarse la primera vez.

```
CREATE DATABASE my_db;
USE my_db;
CREATE TABLE some_table;
INSERT INTO some_table ...;
```

Puede hacer referencia a su tabla calificando con el nombre de la base de datos: `my_db.some_table` .

Bases de datos del sistema

Las siguientes bases de datos existen para el uso de MySQL. Puede leerlos (`SELECT`), pero no debe escribir (`INSERT` / `UPDATE` / `DELETE`) las tablas en ellos. (Hay algunas excepciones.)

- `mysql` - repositorio de información `GRANT` y algunas otras cosas.
- `information_schema` : las tablas aquí son 'virtuales' en el sentido de que en realidad se manifiestan mediante estructuras en memoria. Sus contenidos incluyen el esquema para todas las tablas.
- `performance_schema` - ?? [por favor acepte, luego edite]
- ¿¿otros?? (para MariaDB, Galera, TokuDB, etc.)

Creando y Seleccionando una Base de Datos

Si el administrador crea su base de datos por usted al configurar sus permisos, puede comenzar a usarla. De lo contrario, necesitas crearlo tú mismo:

```
mysql> CREATE DATABASE menagerie;
```

En Unix, los nombres de las bases de datos distinguen entre mayúsculas y minúsculas (a diferencia de las palabras clave SQL), por lo que siempre debe referirse a su base de datos como gestor de archivos, no como Menagerie, MENAGERIE o alguna otra variante. Esto también es cierto para los nombres de tablas. (Bajo Windows, esta restricción no se aplica, aunque debe referirse a las bases de datos y tablas que usan el mismo caso de letras en una consulta determinada. Sin embargo, por una variedad de razones, la mejor práctica recomendada es usar el mismo tipo de letra que se usó cuando la base de datos fue creada.)

La creación de una base de datos no la selecciona para su uso; Debes hacerlo explícitamente. Para hacer que menagerie sea la base de datos actual, use esta declaración:

```
mysql> USE menagerie
Database changed
```

Su base de datos solo debe crearse una vez, pero debe seleccionarla para usarla cada vez que comience una sesión de mysql. Puede hacerlo emitiendo una declaración USE como se muestra en el ejemplo. Alternativamente, puede seleccionar la base de datos en la línea de comandos cuando invoque mysql. Simplemente especifique su nombre después de cualquier parámetro de conexión que deba proporcionar. Por ejemplo:

```
shell> mysql -h host -u user -p menagerie
Enter password: ********
```

Lea Creando bases de datos en línea: https://riptutorial.com/es/mysql/topic/600/creando-bases-de-datos

Capítulo 26: Crear nuevo usuario

Observaciones

Para ver una lista de usuarios de MySQL, usamos el siguiente comando:

```
SELECT User,Host FROM mysql.user;
```

Examples

Crear un usuario de MySQL

Para crear un nuevo usuario, debemos seguir los pasos simples que se detallan a continuación:

Paso 1: Iniciar sesión en `MySQL` como `root`

```
$ mysql -u root -p
```

Paso 2: Veremos el símbolo del sistema mysql

```
mysql> CREATE USER 'my_new_user'@'localhost' IDENTIFIED BY 'test_password';
```

Aquí, hemos creado con éxito un nuevo usuario, pero este usuario no tendrá ningún `permissions` . Por lo tanto, para asignar `permissions` al usuario, utilice el siguiente comando:

```
mysql> GRANT ALL PRIVILEGES ON my_db.* TO 'my_new_user'@'localhost' identified by
'my_password';
```

Especifique la contraseña

El uso básico es:

```
mysql> CREATE USER 'my_new_user'@'localhost' IDENTIFIED BY 'test_password';
```

Sin embargo, para situaciones en las que no es aconsejable codificar la contraseña en texto no cifrado, también es posible especificar directamente, utilizando la directiva `PASSWORD` , el valor de hash devuelto por la función `PASSWORD()` :

```
mysql> select PASSWORD('test_password'); -- returns *4414E26EDED6D661B5386813EBBA95065DBC4728
mysql> CREATE USER 'my_new_user'@'localhost' IDENTIFIED BY PASSWORD
'*4414E26EDED6D661B5386813EBBA95065DBC4728';
```

Crear nuevo usuario y otorgar todos los privilegios al esquema

```
grant all privileges on schema_name.* to 'new_user_name'@'%' identified by 'newpassword';
```

Atención: Esto se puede usar para crear un nuevo usuario root.

Renombrando usuario

```
rename user 'user'@'%' to 'new_name`@'%';
```

Si creas un usuario por error, puedes cambiar su nombre.

Lea Crear nuevo usuario en línea: https://riptutorial.com/es/mysql/topic/3508/crear-nuevo-usuario

Capítulo 27: Datos de carga infile

Sintaxis

1. DATOS DE CARGA [LOW_PRIORITY | CONCURRENTE] [LOCAL] INFILE 'nombre_archivo'
2. EN LA TABLA tbl_name
3. [CARACTER SET charset]
4. [{CAMPOS | COLUMNAS} [TERMINADO POR 'cadena'] [[OPCIONALMENTE] CERRADO POR 'char']]
5. [LÍNEAS [COMIENZO POR 'cadena'] [TERMINADO POR 'cadena']]
6. [IGNORAR número {LINEAS | FILAS}]
7. [(col_name_or_user_var, ...)]
8. [SET col_name = expr, ...]

Examples

usando LOAD DATA INFILE para cargar una gran cantidad de datos a la base de datos

Considere el siguiente ejemplo, suponiendo que tiene un CSV delimitado por ';' para cargar en su base de datos.

```
1;max;male;manager;12-7-1985
2;jack;male;executive;21-8-1990
.
.
.
1000000;marta;female;accountant;15-6-1992
```

Crear la tabla para la inserción.

```
CREATE TABLE `employee` ( `id` INT NOT NULL ,
                          `name` VARCHAR NOT NULL,
                          `sex` VARCHAR NOT NULL ,
                          `designation` VARCHAR NOT NULL ,
                          `dob` VARCHAR NOT NULL   );
```

Use la siguiente consulta para insertar los valores en esa tabla.

```
LOAD DATA INFILE 'path of the file/file_name.txt'
INTO TABLE employee
FIELDS TERMINATED BY ';' //specify the delimiter separating the values
LINES TERMINATED BY '\r\n'
(id,name,sex,designation,dob)
```

Considere el caso donde el formato de fecha no es estándar.

```
1;max;male;manager;17-Jan-1985
2;jack;male;executive;01-Feb-1992
.
.
.
1000000;marta;female;accountant;25-Apr-1993
```

En este caso, puede cambiar el formato de la columna `dob` antes de insertar así.

```
LOAD DATA INFILE 'path of the file/file_name.txt'
INTO TABLE employee
FIELDS TERMINATED BY ';' //specify the delimiter separating the values
LINES TERMINATED BY '\r\n'
(id,name,sex,designation,@dob)
SET date = STR_TO_DATE(@date, '%d-%b-%Y');
```

Este ejemplo de LOAD DATA INFILE no especifica todas las funciones disponibles.

Puedes ver más referencias en LOAD DATA INFILE aquí .

Importar un archivo CSV en una tabla de MySQL

El siguiente comando importa archivos CSV a una tabla de MySQL con las mismas columnas, respetando las reglas de cotización y escape de CSV.

```
load data infile '/tmp/file.csv'
into table my_table
fields terminated by ','
optionally enclosed by '"'
escaped by '"'
lines terminated by '\n'
ignore 1 lines; -- skip the header row
```

Cargar datos con duplicados.

Si usa el comando `LOAD DATA INFILE` para llenar una tabla con datos existentes, a menudo encontrará que la importación falla debido a duplicados. Hay varias formas posibles de superar este problema.

Datos de carga local

Si esta opción se ha habilitado en su servidor, puede usarse para cargar un archivo que existe en la computadora cliente en lugar del servidor. Un efecto secundario es que las filas duplicadas para valores únicos se ignoran.

```
LOAD DATA LOCAL INFILE 'path of the file/file_name.txt'
INTO TABLE employee
```

CARGAR DATOS INFILE 'fname' REEMPLAZAR

Cuando se usa la palabra clave de reemplazo, duplicar claves únicas o primarias resultará en que la fila existente se reemplace con otras nuevas

```
LOAD DATA INFILE 'path of the file/file_name.txt'
REPLACE INTO TABLE employee
```

DATOS DE LA CARGA INFILE 'fname' IGNORE

Lo contrario de `REPLACE` , las filas existentes se conservarán y las nuevas se ignorarán. Este comportamiento es similar al `LOCAL` descrito anteriormente. Sin embargo, el archivo no necesita existir en la computadora cliente.

```
LOAD DATA INFILE 'path of the file/file_name.txt'
IGNORE INTO TABLE employee
```

Carga vía tabla intermedia

A veces, ignorar o reemplazar todos los duplicados puede no ser la opción ideal. Es posible que deba tomar decisiones basadas en el contenido de otras columnas. En ese caso, la mejor opción es cargar en una tabla intermedia y transferir desde allí.

```
INSERT INTO employee SELECT * FROM intermediary WHERE ...
```

importación y exportación

importar

```
SELECT a,b,c  INTO OUTFILE 'result.txt' FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n' FROM table;
```

Exportar

```
LOAD DATA INFILE 'result.txt' INTO TABLE table;
```

Lea Datos de carga infile en línea: https://riptutorial.com/es/mysql/topic/2356/datos-de-carga-infile

Capítulo 28: ENUM

Examples

¿Por qué ENUM?

ENUM proporciona una forma de proporcionar un atributo para una fila. Los atributos con un pequeño número de opciones no numéricas funcionan mejor. Ejemplos:

```
reply ENUM('yes', 'no')
gender ENUM('male', 'female', 'other', 'decline-to-state')
```

Los valores son cadenas:

```
INSERT ... VALUES ('yes', 'female')
SELECT ... --> yes female
```

TINYINT como alternativa

Digamos que tenemos

```
type ENUM('fish','mammal','bird')
```

Una alternativa es

```
type TINYINT UNSIGNED
```

más

```
CREATE TABLE AnimalTypes (
    type TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
    name VARCHAR(20) NOT NULL  COMMENT "('fish','mammal','bird')",
    PRIMARY KEY(type),
    INDEX(name)
) ENGINE=InnoDB
```

que es muy parecido a una tabla de muchos a muchos.

Comparación, y si es mejor o peor que ENUM:

- (peor) INSERTAR: hay que buscar el `type`
- (peor) SELECCIONE: necesita UNIRSE para obtener la cadena (ENUM le da la cadena sin esfuerzo)
- (mejor) Agregando nuevos tipos: Simplemente insértelos en esta tabla. Con ENUM, necesitas hacer una ALTER TABLE.
- (igual) Cualquiera de las dos técnicas (para hasta 255 valores) toma solo 1 byte.

- (mixto) También hay un problema de integridad de datos: `TINYINT` admitirá valores no válidos; mientras que `ENUM` establece en un valor especial de cadena vacía (a menos que se habilite el modo SQL estricto, en cuyo caso se rechazan). Se puede lograr una mejor integridad de los datos con `TINYINT` al convertirla en una clave externa en una tabla de búsqueda: que, con las consultas / combinaciones apropiadas, pero aún existe el pequeño costo de llegar a la otra tabla. (`FOREIGN KEYs` no son gratuitas).

VARCHAR como alternativa

Digamos que tenemos

```
type ENUM('fish','mammal','bird')
```

Una alternativa es

```
type VARCHAR(20)  COMENT "fish, bird, etc"
```

Esto es bastante abierto ya que los nuevos tipos se agregan de forma trivial.

Comparación, y si es mejor o peor que ENUM:

- (mismo) INSERT: simplemente proporciona la cadena
- (¿peor?) En INSERTAR un error tipográfico pasará desapercibido
- (igual) SELECCIONAR: se devuelve la cadena real
- (Peor) Se consume mucho más espacio.

Añadiendo una nueva opción

```
ALTER TABLE tbl MODIFY COLUMN type ENUM('fish','mammal','bird','insect');
```

Notas

- Al igual que con todos los casos de MODIFICAR COLUMNA, debe incluir `NOT NULL` , y cualquier otro calificador que haya existido originalmente, de lo contrario, se perderán.
- *Si* agrega al *final* de la lista *y* la lista está en 256 elementos, `ALTER` se realiza simplemente cambiando el esquema. Es decir, no habrá una copia larga de la tabla. (Las versiones anteriores de MySQL no tenían esta optimización.)

NULL vs NOT NULL

Ejemplos de lo que sucede cuando NULL y 'mal valor' se almacenan en columnas que admiten nulos y que no admiten nulos. También muestra el uso de casting a numérico a través de `+0` .

```
CREATE TABLE enum (
    e     ENUM('yes', 'no')   NOT NULL,
    enull ENUM('x', 'y', 'z')     NULL
        );
INSERT INTO enum (e, enull)
```

```
    VALUES
        ('yes', 'x'),
        ('no',  'y'),
        (NULL,  NULL),
        ('bad-value', 'bad-value');
Query OK, 4 rows affected, 3 warnings (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 3

mysql>SHOW WARNINGS;
+---------+------+-------------------------------------------+
| Level   | Code | Message                                   |
+---------+------+-------------------------------------------+
| Warning | 1048 | Column 'e' cannot be null                 |
| Warning | 1265 | Data truncated for column 'e' at row 4    |
| Warning | 1265 | Data truncated for column 'enull' at row 4 |
+---------+------+-------------------------------------------+
3 rows in set (0.00 sec)
```

Lo que está en la tabla después de esas inserciones. Esto usa "+0" para convertir para ver numéricamente lo que está almacenado.

```
mysql>SELECT e, e+0 FROM enum;
+-----+-----+
| e   | e+0 |
+-----+-----+
| yes |   1 |
| no  |   2 |
|     |   0 |  -- NULL
|     |   0 |  -- 'bad-value'
+-----+-----+
4 rows in set (0.00 sec)

mysql>SELECT enull, enull+0 FROM enum;
+-------+---------+
| enull | enull+0 |
+-------+---------+
| x     |       1 |
| y     |       2 |
| NULL  |    NULL |
|       |       0 |  -- 'bad-value'
+-------+---------+
4 rows in set (0.00 sec)
```

Lea ENUM en línea: https://riptutorial.com/es/mysql/topic/4425/enum

Capítulo 29: Error 1055: ONLY_FULL_GROUP_BY: algo no está en la cláusula GROUP BY ...

Introducción

Recientemente, las nuevas versiones de los servidores MySQL han comenzado a generar 1055 errores para las consultas que solían funcionar. Este tema explica esos errores. El equipo de MySQL ha estado trabajando para retirar la extensión no estándar a `GROUP BY` , o al menos para dificultar que los desarrolladores de escritura de consultas se quemen con ella.

Observaciones

Durante mucho tiempo, MySQL ha contenido una extensión notoriamente no estándar de `GROUP BY` , que permite un comportamiento extraño en nombre de la eficiencia. Esta extensión ha permitido que innumerables desarrolladores de todo el mundo usen `GROUP BY` en el código de producción sin entender completamente lo que estaban haciendo.

En particular, es una mala idea usar `SELECT *` en una consulta `GROUP BY` , porque una cláusula `GROUP BY` estándar requiere enumerar las columnas. Muchos desarrolladores, lamentablemente, lo han hecho.

Lee esto. https://dev.mysql.com/doc/refman/5.7/en/group-by-handling.html

El equipo de MySQL ha estado tratando de solucionar este error sin arruinar el código de producción. `sql_mode` un indicador `sql_mode` en 5.7.5 llamado `ONLY_FULL_GROUP_BY` para obligar al comportamiento estándar. En una versión reciente, activaron esa bandera de forma predeterminada. Cuando actualizó su MySQL local a 5.7.14, la bandera se encendió y su código de producción, dependiendo de la extensión anterior, dejó de funcionar.

Si recientemente comenzó a recibir 1055 errores, ¿cuáles son sus opciones?

1. arregle las consultas SQL ofensivas, o haga que sus autores hagan eso.
2. vuelva a una versión de MySQL lista para usar con el software de la aplicación que utiliza.
3. cambie el `sql_mode` su servidor para deshacerse del modo `ONLY_FULL_GROUP_BY` recién configurado.

Puedes cambiar el modo haciendo un comando `SET` .

```
SET  sql_mode =
'STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENG
```

debe hacer el truco si lo hace justo después de que su aplicación se conecte a MySQL.

O bien, puede encontrar el archivo init en su instalación de MySQL , ubicar la línea `sql_mode=` y cambiarlo para omitir `ONLY_FULL_GROUP_BY` , y reiniciar su servidor.

Examples

Uso y mal uso de GROUP BY

```
SELECT item.item_id, item.name,     /* not SQL-92 */
       COUNT(*) number_of_uses
 FROM item
 JOIN uses ON item.item_id, uses.item_id
GROUP BY item.item_id
```

mostrará las filas en una tabla llamada `item` y mostrará el recuento de filas relacionadas en una tabla llamada `uses` . Esto funciona bien, pero desafortunadamente no es estándar SQL-92.

Por qué no? porque la cláusula `SELECT` (y la cláusula `ORDER BY`) en las consultas `GROUP BY` deben contener columnas que son

1. mencionado en la cláusula `GROUP BY` , o
2. funciones agregadas como `COUNT()` , `MIN()` y similares.

La cláusula `SELECT` este ejemplo menciona `item.name` , una columna que no cumple ninguno de esos criterios. MySQL 5.6 y anteriores rechazarán esta consulta si el modo SQL contiene `ONLY_FULL_GROUP_BY` .

Esta consulta de ejemplo se puede hacer para cumplir con el estándar SQL-92 cambiando la cláusula `GROUP BY` , de esta manera.

```
SELECT item.item_id, item.name,
       COUNT(*) number_of_uses
  FROM item
  JOIN uses ON item.item_id, uses.item_id
 GROUP BY item.item_id, item.name
```

El último estándar SQL-99 permite que una instrucción `SELECT` omita las columnas no agregadas de la clave de grupo si el DBMS puede probar una dependencia funcional entre ellas y las columnas de la clave de grupo. Debido a que `item.name` es funcionalmente dependiente de `item.item_id` , el ejemplo inicial es válido SQL-99. MySQL ganó un probador de dependencia funcional en la versión 5.7. El ejemplo original funciona en `ONLY_FULL_GROUP_BY` .

Mal uso de GROUP BY para devolver resultados impredecibles: la ley de Murphy

```
SELECT item.item_id, uses.category,   /* nonstandard */
       COUNT(*) number_of_uses
  FROM item
  JOIN uses ON item.item_id, uses.item_id
 GROUP BY item.item_id
```

mostrará las filas en una tabla llamada elemento y mostrará el recuento de filas relacionadas en una tabla llamada usos. También mostrará el valor de una columna llamada `uses.category` .

Esta consulta funciona en MySQL (antes de que `ONLY_FULL_GROUP_BY` indicador `ONLY_FULL_GROUP_BY`). Utiliza la extensión no estándar de MySQL para `GROUP BY` .

Pero la consulta tiene un problema: si varias filas en la tabla de `uses` coinciden con la condición `ON` en la cláusula `JOIN` , MySQL devuelve la columna de `category` de solo una de esas filas. Que fila El autor de la consulta y el usuario de la aplicación no lo saben de antemano. Hablando formalmente, es *impredecible* : MySQL puede devolver cualquier valor que desee.

Lo impredecible es como *aleatorio,* con una diferencia significativa. Uno podría esperar que una elección *aleatoria* cambie de vez en cuando. Por lo tanto, si una elección fuera aleatoria, podría detectarla durante la depuración o la prueba. El resultado *impredecible* es peor: MySQL devuelve el mismo resultado cada vez que usa la consulta, *hasta que no lo hace.* A veces es una nueva versión del servidor MySQL que causa un resultado diferente. A veces es una mesa cada vez mayor que causa el problema. Lo que puede salir mal, saldrá mal, y cuando no lo espere. Eso se llama la ley de Murphy .

El equipo de MySQL ha estado trabajando para que a los desarrolladores les resulte más difícil cometer este error. Las nuevas versiones de MySQL en la secuencia de 5,7 tienen un `sql_mode` bandera llamada `ONLY_FULL_GROUP_BY` . Cuando se establece ese indicador, el servidor MySQL devuelve el error 1055 y se niega a ejecutar este tipo de consulta.

Mal uso de GROUP BY con SELECT *, y cómo solucionarlo.

A veces, una consulta se ve así, con un `*` en la cláusula `SELECT` .

```
SELECT item.*,     /* nonstandard */
       COUNT(*) number_of_uses
 FROM item
 JOIN uses ON item.item_id, uses.item_id
GROUP BY item.item_id
```

Dicha consulta debe ser reformulada para cumplir con el estándar `ONLY_FULL_GROUP_BY` .

Para ello, necesitamos una subconsulta que utiliza `GROUP BY` correctamente para devolver el `number_of_uses` valor para cada `item_id` . Esta subconsulta es breve y dulce, porque solo necesita mirar la tabla de `uses` .

```
                              SELECT item_id, COUNT(*) number_of_uses
                                FROM  uses
                               GROUP BY item_id
```

Luego, podemos unir esa subconsulta con la tabla de `item` .

```
SELECT item.*, usecount.number_of_uses
  FROM item
  JOIN (
                              SELECT item_id, COUNT(*) number_of_uses
```

```
                              FROM  uses
                             GROUP BY item_id
        ) usecount ON item.item_id = usecount.item_id
```

Esto permite que la cláusula `GROUP BY` sea simple y correcta, y también nos permite usar el especificador `*` .

Nota: sin embargo, los desarrolladores inteligentes evitan el uso del especificador `*` en cualquier caso. Por lo general, es mejor enumerar las columnas que desea en una consulta.

ALGÚN VALOR()

```
SELECT item.item_id, ANY_VALUE(uses.tag) tag,
       COUNT(*) number_of_uses
 FROM item
 JOIN uses ON item.item_id, uses.item_id
GROUP BY item.item_id
```

muestra las filas en una tabla llamada `item` , el recuento de filas relacionadas y uno de los valores en la tabla relacionada llamados `uses` .

Puede pensar en esta función `ANY_VALUE()` como un tipo extraño de función agregada. En lugar de devolver un conteo, suma o máximo, le indica al servidor MySQL que elija, de forma arbitraria, un valor del grupo en cuestión. Es una forma de evitar el error 1055.

Tenga cuidado al usar `ANY_VALUE()` en consultas en aplicaciones de producción.

Realmente debería llamarse `SURPRISE_ME()` . Devuelve el valor de alguna fila en el grupo GRUPO POR. La fila que devuelve es indeterminada. Eso significa que depende completamente del servidor MySQL. Formalmente, devuelve un valor impredecible.

El servidor no elige un valor aleatorio, es peor que eso. Devuelve el mismo valor cada vez que ejecuta la consulta, hasta que no lo hace. Puede cambiar, o no, cuando una tabla crece o se encoge, o cuando el servidor tiene más o menos RAM, o cuando la versión del servidor cambia, o cuando Marte está en retroceso (lo que sea que eso signifique), o sin ninguna razón.

Usted ha sido advertido.

Lea Error 1055: ONLY_FULL_GROUP_BY: algo no está en la cláusula GROUP BY ... en línea: https://riptutorial.com/es/mysql/topic/8245/error-1055--only-full-group-by--algo-no-esta-en-la-clausula-group-by----

Capítulo 30: Eventos

Examples

Crear un evento

Mysql tiene su funcionalidad EVENTO para evitar interacciones cron complicadas cuando gran parte de lo que está programando está relacionado con SQL y menos con archivos. Vea la página del Manual aquí . Piense en los eventos como procedimientos almacenados que están programados para ejecutarse en intervalos recurrentes.

Para ahorrar tiempo en la depuración de problemas relacionados con eventos, tenga en cuenta que el controlador de eventos global debe estar activado para procesar eventos.

```
SHOW VARIABLES WHERE variable_name='event_scheduler';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| event_scheduler | OFF   |
+-----------------+-------+
```

Con el apagado, nada disparará. Así que enciéndelo:

```
SET GLOBAL event_scheduler = ON;
```

Esquema para la prueba

```
create table theMessages
(   id INT AUTO_INCREMENT PRIMARY KEY,
    userId INT NOT NULL,
    message VARCHAR(255) NOT NULL,
    updateDt DATETIME NOT NULL,
    KEY(updateDt)
);

INSERT theMessages(userId,message,updateDt) VALUES (1,'message 123','2015-08-24 11:10:09');
INSERT theMessages(userId,message,updateDt) VALUES (7,'message 124','2015-08-29');
INSERT theMessages(userId,message,updateDt) VALUES (1,'message 125','2015-09-03 12:00:00');
INSERT theMessages(userId,message,updateDt) VALUES (1,'message 126','2015-09-03 14:00:00');
```

Las inserciones anteriores se proporcionan para mostrar un punto de partida. Tenga en cuenta que los 2 eventos creados a continuación limpiarán las filas.

Cree 2 eventos, 1º se ejecuta diariamente, 2º se ejecuta cada 10 minutos

Ignora lo que realmente están haciendo (jugando uno contra el otro). El punto está en el

INTERVALO y la programación.

```
DROP EVENT IF EXISTS `delete7DayOldMessages`;
DELIMITER $$
CREATE EVENT `delete7DayOldMessages`
  ON SCHEDULE EVERY 1 DAY STARTS '2015-09-01 00:00:00'
  ON COMPLETION PRESERVE
DO BEGIN
   DELETE FROM theMessages
   WHERE datediff(now(),updateDt)>6; -- not terribly exact, yesterday but <24hrs is still 1
day

  -- Other code here

END$$
DELIMITER ;
```

...

```
DROP EVENT IF EXISTS `Every_10_Minutes_Cleanup`;
DELIMITER $$
CREATE EVENT `Every_10_Minutes_Cleanup`
  ON SCHEDULE EVERY 10 MINUTE STARTS '2015-09-01 00:00:00'
  ON COMPLETION PRESERVE
DO BEGIN
   DELETE FROM theMessages
   WHERE TIMESTAMPDIFF(HOUR, updateDt, now())>168; -- messages over 1 week old (168 hours)

   -- Other code here
END$$
DELIMITER ;
```

Mostrar estados de eventos (diferentes enfoques)

```
SHOW EVENTS FROM my_db_name; -- List all events by schema name (db name)
SHOW EVENTS;
SHOW EVENTS\G; -- <--------- I like this one from mysql> prompt

*************************** 1. row ***************************
                  Db: my_db_name
                Name: delete7DayOldMessages
             Definer: root@localhost
           Time zone: SYSTEM
                Type: RECURRING
          Execute at: NULL
      Interval value: 1
      Interval field: DAY
              Starts: 2015-09-01 00:00:00
                Ends: NULL
              Status: ENABLED
          Originator: 1
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: utf8_general_ci
*************************** 2. row ***************************
                  Db: my_db_name
                Name: Every_10_Minutes_Cleanup
```

```
             Definer: root@localhost
           Time zone: SYSTEM
                Type: RECURRING
          Execute at: NULL
      Interval value: 10
      Interval field: MINUTE
              Starts: 2015-09-01 00:00:00
                Ends: NULL
              Status: ENABLED
          Originator: 1
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: utf8_general_ci
2 rows in set (0.06 sec)
```

Cosas al azar a considerar

`DROP EVENT someEventName;` - Borra el evento y su código.

`ON COMPLETION PRESERVE` : cuando el evento haya terminado de procesarse, consérvelo. De lo contrario, se elimina.

Los eventos son como disparadores. No son llamados por el programa de un usuario. Más bien, están programados. Como tales, tienen éxito o fallan en silencio.

El enlace a la página del manual muestra bastante flexibilidad con las opciones de intervalo, que se muestran a continuación:

> intervalo:
>
> ```
> quantity {YEAR | QUARTER | MONTH | DAY | HOUR | MINUTE |
>           WEEK | SECOND | YEAR_MONTH | DAY_HOUR | DAY_MINUTE |
>           DAY_SECOND | HOUR_MINUTE | HOUR_SECOND | MINUTE_SECOND}
> ```

Los eventos son mecanismos poderosos que manejan tareas recurrentes y programadas para su sistema. Pueden contener tantas declaraciones, rutinas DDL y DML y uniones complicadas como usted desee razonablemente. Consulte la página del manual de MySQL titulada Restricciones en programas almacenados .

Lea Eventos en línea: https://riptutorial.com/es/mysql/topic/4319/eventos

Capítulo 31: Expresiones regulares

Introducción

Una expresión regular es una forma poderosa de especificar un patrón para una búsqueda compleja.

Examples

REGEXP / RLIKE

El operador `REGEXP` (o su sinónimo, `RLIKE`) permite la coincidencia de patrones en base a expresiones regulares.

Considere la siguiente tabla de `employee` :

```
+-------------+-------------+-------------+--------------+----------+
| EMPLOYEE_ID | FIRST_NAME  | LAST_NAME   | PHONE_NUMBER | SALARY   |
+-------------+-------------+-------------+--------------+----------+
|         100 | Steven      | King        | 515.123.4567 | 24000.00 |
|         101 | Neena       | Kochhar     | 515.123.4568 | 17000.00 |
|         102 | Lex         | De Haan     | 515.123.4569 | 17000.00 |
|         103 | Alexander   | Hunold      | 590.423.4567 |  9000.00 |
|         104 | Bruce       | Ernst       | 590.423.4568 |  6000.00 |
|         105 | David       | Austin      | 590.423.4569 |  4800.00 |
|         106 | Valli       | Pataballa   | 590.423.4560 |  4800.00 |
|         107 | Diana       | Lorentz     | 590.423.5567 |  4200.00 |
|         108 | Nancy       | Greenberg   | 515.124.4569 | 12000.00 |
|         109 | Daniel      | Faviet      | 515.124.4169 |  9000.00 |
|         110 | John        | Chen        | 515.124.4269 |  8200.00 |
+-------------+-------------+-------------+--------------+----------+
```

Patrón ^

Seleccione todos los empleados cuyo `FIRST_NAME` comience con **N.**

Consulta

```
SELECT * FROM employees WHERE FIRST_NAME REGEXP '^N'
-- Pattern start with----------------------------^
```

Patrón $ **

Seleccione todos los empleados cuyo `PHONE_NUMBER` finalice con **4569** .

Consulta

```
SELECT * FROM employees WHERE PHONE_NUMBER REGEXP '4569$'
-- Pattern end with---------------------------------^
```

NO REGEXP

Seleccione todos los empleados cuyo `FIRST_NAME` *no* comience con **N.**

Consulta

```
SELECT * FROM employees WHERE FIRST_NAME NOT REGEXP '^N'
-- Pattern does not start with---------------^
```

Regex contener

Seleccionar todos los empleados cuyos `LAST_NAME` contiene **y** cuya `FIRST_NAME` contiene `a` .

Consulta

```
SELECT * FROM employees WHERE FIRST_NAME REGEXP 'a' AND LAST_NAME REGEXP 'in'
-- No ^ or $, pattern can be anywhere ------------------------------------^
```

Cualquier caracter entre []

Seleccione todos los empleados cuyo `FIRST_NAME` comience con **A** o **B** o **C.**

Consulta

```
SELECT * FROM employees WHERE FIRST_NAME REGEXP '^[ABC]'
-------------------------------------------------^^---^
```

Patrón o |

Seleccione todos los empleados cuyo `FIRST_NAME` comience con **A** o **B** o **C** y termine con **r** , **e** , o **i** .

Consulta

```
SELECT * FROM employees WHERE FIRST_NAME REGEXP '^[ABC]|[rei]$'
-- ---------------------------------------------^^---^^^---^^
```

Contando coincidencias de expresiones regulares

Considere la siguiente consulta:

```
SELECT FIRST_NAME, FIRST_NAME REGEXP '^N' as matching FROM employees
```

`FIRST_NAME REGEXP '^N'` es *1* o *0* según el hecho de que `FIRST_NAME` coincida con `^N`

Para visualizarlo mejor:

```
SELECT
FIRST_NAME,
IF(FIRST_NAME REGEXP '^N', 'matches ^N', 'does not match ^N') as matching
FROM employees
```

Finalmente, cuente el número total de filas coincidentes y no coincidentes con:

```
SELECT
IF(FIRST_NAME REGEXP '^N', 'matches ^N', 'does not match ^N') as matching,
COUNT(*)
FROM employees
GROUP BY matching
```

Lea Expresiones regulares en línea: https://riptutorial.com/es/mysql/topic/9444/expresiones-regulares

Capítulo 32: Extraer valores de tipo JSON

Introducción

MySQL 5.7.8+ admite el tipo JSON nativo. Si bien tiene diferentes formas de crear objetos json, también puede acceder y leer miembros de diferentes formas.

La función principal es `JSON_EXTRACT` , por lo que los operadores `->` y `->>` son más amigables.

Sintaxis

- JSON_EXTRACT (json_doc, ruta [, ...])
- JSON_EXTRACT (json_doc, ruta)
- JSON_EXTRACT (json_doc, path1, path2)

Parámetros

Parámetro	Descripción
json_doc	documento JSON válido
camino	ruta de miembros

Observaciones

Mencionado en MySQL 5.7 Reference Manual

- Múltiples valores coincidentes por argumento (s) de ruta

  Si es posible que esos argumentos puedan devolver múltiples valores, los valores coincidentes se envuelven automáticamente como una matriz, en el orden correspondiente a las rutas que los produjeron. De lo contrario, el valor de retorno es el único valor coincidente.

- Resultado `NULL` cuando:
  - cualquier argumento es NULL
  - camino no emparejado

  Devuelve NULL si algún argumento es NULL o si no hay rutas de acceso ubican un valor en el documento.

Examples

Leer el valor de la matriz JSON

Cree la variable @myjson como tipo JSON (lea más):

```
SET @myjson = CAST('["A","B",{"id":1,"label":"C"}]' as JSON) ;
```

`SELECT` algunos miembros!

```
SELECT
  JSON_EXTRACT( @myjson , '$[1]' ) ,
  JSON_EXTRACT( @myjson , '$[*].label') ,
  JSON_EXTRACT( @myjson , '$[1].*' ) ,
  JSON_EXTRACT( @myjson , '$[2].*')
;
-- result values:
'\"B\"', '[\"C\"]', NULL, '[1, \"C\"]'
-- visually:
"B", ["C"], NULL, [1, "C"]
```

Operadores de extracto JSON

Extraiga la `path` mediante `->` o `->>` Operadores, mientras que `->>` es un valor NO ASUMIDO:

```
SELECT
  myjson_col->>'$[1]' , myjson_col->'$[1]' ,
  myjson_col->>'$[*].label' ,
  myjson_col->>'$[1].*' ,
  myjson_col->>'$[2].*'
FROM tablename ;
 -- visuall:
    B, "B" , ["C"], NULL, [1, "C"]
 __^^^ ^^^
```

Entonces `col->>path` es igual a `JSON_UNQUOTE(JSON_EXTRACT(col,path))` :

> Al igual que con ->, el operador - >> siempre se expande en la salida de EXPLAIN, como lo demuestra el siguiente ejemplo:

```
mysql> EXPLAIN SELECT c->>'$.name' AS name
    ->     FROM jemp WHERE g > 2\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: jemp
   partitions: NULL
         type: range
possible_keys: i
          key: i
      key_len: 5
          ref: NULL
         rows: 2
     filtered: 100.00
        Extra: Using where
1 row in set, 1 warning (0.00 sec)

mysql> SHOW WARNINGS\G
*************************** 1. row ***************************
```

```
  Level: Note
   Code: 1003
Message: /* select#1 */ select
json_unquote(json_extract(`jtest`.`jemp`.`c`,'$.name')) AS `name` from
`jtest`.`jemp` where (`jtest`.`jemp`.`g` > 2)
1 row in set (0.00 sec)
```

Lea sobre el extracto de ruta en línea (+)

Lea Extraer valores de tipo JSON en línea: https://riptutorial.com/es/mysql/topic/9042/extraer-valores-de-tipo-json

Capítulo 33: Gatillos

Sintaxis

- CREAR [DEFINER = {usuario | CURRENT_USER}] TRIGGER trigger_name trigger_time trigger_event ON tbl_name PARA CADA FILA [trigger_order] trigger_body
- trigger_time: {ANTES | DESPUÉS }
- trigger_event: {INSERT | Actualización | BORRAR }
- trigger_order: {SIGUE | PRECEDES} other_trigger_name

Observaciones

Dos puntos deben llamar su atención si ya usa desencadenantes en otros DB:

POR CADA FILA

`FOR EACH ROW` es una parte obligatoria de la sintaxis.

No se puede hacer un desencadenante de *declaración* (una vez por consulta) como lo hace Oracle. Es más un problema relacionado con el rendimiento que una característica faltante real

CREAR O REEMPLAZAR EL GATILLO

MySQL no admite `CREATE OR REPLACE`

MySQL no permite esta sintaxis, en su lugar tiene que usar lo siguiente:

```
DELIMITER $$

DROP TRIGGER IF EXISTS myTrigger;
$$
CREATE TRIGGER myTrigger
-- ...

$$
DELIMITER ;
```

Tenga cuidado, esto **no** es **una transacción atómica** :

- perderás el gatillo viejo si falla la `CREATE`
- en una carga pesada, otras operaciones pueden ocurrir entre el `DROP` y el `CREATE` , usar una `LOCK TABLES myTable WRITE;` primero para evitar inconsistencias de datos y `UNLOCK TABLES;` después de la `CREATE` para liberar la tabla

Examples

Disparador basico

Crear mesa

```
mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
Query OK, 0 rows affected (0.03 sec)
```

Crear gatillo

```
mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
    -> FOR EACH ROW SET @sum = @sum + NEW.amount;
Query OK, 0 rows affected (0.06 sec)
```

La declaración CREATE TRIGGER crea un desencadenante denominado ins_sum que está asociado con la tabla de cuentas. También incluye cláusulas que especifican el tiempo de acción del activador, el evento de activación y qué hacer cuando se activa el activador.

Insertar valor

Para usar el disparador, establezca la variable del acumulador (@sum) en cero, ejecute una instrucción INSERT y luego vea qué valor tiene la variable después:

```
mysql> SET @sum = 0;
mysql> INSERT INTO account VALUES(137,14.98),(141,1937.50),(97,-100.00);
mysql> SELECT @sum AS 'Total amount inserted';
+-----------------------+
| Total amount inserted |
+-----------------------+
| 1852.48               |
+-----------------------+
```

En este caso, el valor de @sum después de que se haya ejecutado la instrucción INSERT es 14.98 + 1937.50 - 100, o 1852.48.

Drop Trigger

```
mysql> DROP TRIGGER test.ins_sum;
```

Si suelta una tabla, también se eliminan los disparadores de la tabla.

Tipos de disparadores

Sincronización

Hay dos modificadores de tiempo de acción de disparo:

- `BEFORE` activador se active antes de ejecutar la solicitud,
- `AFTER` disparar el fuego después del cambio.

Evento desencadenante

Hay tres eventos que los desencadenantes se pueden adjuntar a:

- `INSERT`
- `UPDATE`
- `DELETE`

Antes de Insertar ejemplo de activador

```
DELIMITER $$

CREATE TRIGGER insert_date
    BEFORE INSERT ON stack
    FOR EACH ROW
BEGIN
        -- set the insert_date field in the request before the insert
    SET NEW.insert_date = NOW();
END;

$$
DELIMITER ;
```

Antes de actualizar el ejemplo de activación

```
DELIMITER $$

CREATE TRIGGER update_date
    BEFORE UPDATE ON stack
    FOR EACH ROW
BEGIN
        -- set the update_date field in the request before the update
    SET NEW.update_date = NOW();
END;

$$
DELIMITER ;
```

Después de eliminar el ejemplo de activación

```
DELIMITER $$

CREATE TRIGGER deletion_date
    AFTER DELETE ON stack
    FOR EACH ROW
```

```
BEGIN
        -- add a log entry after a successful delete
    INSERT INTO log_action(stack_id, deleted_date) VALUES(OLD.id, NOW());
END;

$$
DELIMITER ;
```

Lea Gatillos en línea: https://riptutorial.com/es/mysql/topic/3069/gatillos

Capítulo 34: Índices y claves

Sintaxis

- - Crear índice simple.

  CREAR *index_name* ÍNDICE DE LA *nombre_tabla (nombre1_columna* [, *COLUMN_NAME2,* ...])

- - Crear índice único

  CREAR *index_name* índice único en *nombre_tabla (nombre1_columna* [, *COLUMN_NAME2,* ...]

- - Índice de caída

  DROP INDEX **index_name** ON **tbl_name** [**algorithm_option** | **lock_option**] ...

  algorithm_option: ALGORITHM [=] {DEFAULT | INPLACE | COPY}

  lock_option: LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}

Observaciones

Conceptos

Un índice en una tabla MySQL funciona como un índice en un libro.

Supongamos que tiene un libro sobre bases de datos y desea encontrar información sobre, por ejemplo, almacenamiento. Sin un índice (suponiendo que no haya ninguna otra ayuda, como una tabla de contenido), tendría que recorrer las páginas una por una, hasta que encuentre el tema (que es un "análisis de tabla completa"). Por otro lado, un índice tiene una lista de palabras clave, por lo que debe consultar el índice y ver que el almacenamiento se menciona en las páginas 113-120, 231 y 354. Luego, puede ir directamente a esas páginas, sin buscarlas (eso es una búsqueda con un índice, algo más rápido).

Por supuesto, la utilidad del índice depende de muchas cosas, algunos ejemplos, utilizando el símil anterior:

- Si tenía un libro sobre bases de datos e indexó la palabra "base de datos", podría ver que se menciona en las páginas 1-59, 61-290 y 292-400. Eso es un montón de páginas, y en tal caso, el índice no es de mucha ayuda y podría ser más rápido recorrer las páginas una por una. (En una base de datos, esto es "mala selectividad".)
- Para un libro de 10 páginas, no tiene sentido hacer un índice, ya que puede terminar con un libro de 10 páginas con el prefijo de un índice de 5 páginas, lo cual es una tontería,

simplemente escanee las 10 páginas y listo. .
- El índice también debe ser útil; por lo general, no tiene sentido indexar, por ejemplo, la frecuencia de la letra "L" por página.

Examples

Crear índice

```
-- Create an index for column 'name' in table 'my_table'
CREATE INDEX idx_name ON my_table(name);
```

Crear un índice único

Un índice único impide la inserción de datos duplicados en una tabla. `NULL` valores `NULL` se pueden insertar en las columnas que forman parte del índice único (ya que, por definición, un valor `NULL` es diferente de cualquier otro valor, incluido otro valor `NULL`)

```
-- Creates a unique index for column 'name' in table 'my_table'
CREATE UNIQUE INDEX idx_name ON my_table(name);
```

Índice de caída

```
-- Drop an index for column 'name' in table 'my_table'
DROP INDEX idx_name ON my_table;
```

Crear índice compuesto

Esto creará un índice compuesto de ambas claves, `mystring` y `mydatetime` y acelerará las consultas con ambas columnas en la cláusula `WHERE` .

```
CREATE INDEX idx_mycol_myothercol ON my_table(mycol, myothercol)
```

Nota: ¡ El orden es importante! Si la consulta de búsqueda no incluye ambas columnas en la cláusula `WHERE` , solo puede usar el índice de la izquierda. En este caso, una consulta con `mycol` en el `WHERE` utilizará el índice, una consulta que `myothercol` **sin** buscar también `mycol` **no** . Para más información echa un vistazo a esta entrada de blog .

Nota: Debido a la forma en que funciona BTREE, las columnas que generalmente se consultan en rangos deben ir en el valor más a la derecha. Por ejemplo, las columnas `DATETIME` suelen consultar como `WHERE datecol > '2016-01-01 00:00:00'` . Los índices BTREE manejan los rangos de manera muy eficiente, pero solo si la columna que se consulta como rango es la última en el índice compuesto.

Tecla AUTO_INCREMENT

```
CREATE TABLE (
```

```
    id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    ...
    PRIMARY KEY(id),
    ...  );
```

Notas principales:

- Comienza con 1 e incrementa en 1 automáticamente cuando no puede especificarlo en `INSERT` , o lo especifica como `NULL` .
- Los identificadores siempre son distintos entre sí, pero ...
- No haga suposiciones (sin espacios, generados consecutivamente, no reutilizados, etc.) sobre los valores de la identificación que no sean únicos en un momento dado.

Notas sutiles:

- Al reiniciar el servidor, el valor 'siguiente' se 'calcula' como `MAX(id)+1` .
- Si la última operación antes de apagar o colapsar fue eliminar la ID más alta, esa ID *se* puede reutilizar (esto depende del motor). Por *lo* tanto, *no confíe en que auto_increments sea permanentemente único* ; Solo son únicos en cualquier momento.
- Para soluciones multi-master o agrupadas, vea `auto_increment_offset` y `auto_increment_increment` .
- Está bien tener otra cosa como la `PRIMARY KEY` y simplemente hacer `INDEX(id)` . (Esta es una optimización en algunas situaciones).
- El uso de `AUTO_INCREMENT` como la "clave de `PARTITION` " rara vez es beneficioso; hacer algo diferente.
- Varias operaciones *pueden* "quemar" valores. Esto sucede cuando asignan previamente los valores, luego no los use: `INSERT IGNORE` (con clave de duplicación), `REPLACE` (que es `DELETE` más `INSERT`) y otros. `ROLLBACK` es otra causa de lagunas en los identificadores.
- En la Replicación, no puede confiar en que los identificadores lleguen a los esclavos en orden ascendente. Aunque los identificadores se asignan en orden consecutivo, las declaraciones de InnoDB se envían a los esclavos en el orden `COMMIT` .

Lea Índices y claves en línea: https://riptutorial.com/es/mysql/topic/1748/indices-y-claves

Capítulo 35: información del servidor

Parámetros

Parámetros	Explicación
GLOBAL	Muestra las variables tal como están configuradas para todo el servidor. Opcional.
SESIÓN	Muestra las variables que están configuradas para esta sesión solamente. Opcional.

Examples

MOSTRAR VARIABLES ejemplo

Para obtener todas las variables del servidor, ejecute esta consulta en la ventana SQL de su interfaz preferida (PHPMyAdmin u otra) o en la interfaz CLI de MySQL

```
SHOW VARIABLES;
```

Puede especificar si desea las variables de sesión o las variables globales de la siguiente manera:

Variables de sesión:

```
SHOW SESSION VARIABLES;
```

Variables globales:

```
SHOW GLOBAL VARIABLES;
```

Al igual que cualquier otro comando SQL, puede agregar parámetros a su consulta, como el comando LIKE:

```
SHOW [GLOBAL | SESSION] VARIABLES LIKE 'max_join_size';
```

O, usando comodines:

```
SHOW [GLOBAL | SESSION] VARIABLES LIKE '%size%';
```

También puede filtrar los resultados de la consulta MOSTRAR utilizando un parámetro WHERE de la siguiente manera:

```
SHOW [GLOBAL | SESSION] VARIABLES WHERE VALUE > 0;
```

SHOW STATUS ejemplo

Para obtener el estado del servidor de la base de datos, ejecute esta consulta en la ventana SQL de su interfaz preferida (PHPMyAdmin u otra) o en la interfaz CLI de MySQL.

```
SHOW STATUS;
```

Puede especificar si desea recibir el estado de SESIÓN o GLOBAL de su servidor así: Estado de sesión:

```
SHOW SESSION STATUS;
```

Estado global:

```
SHOW GLOBAL STATUS;
```

Al igual que cualquier otro comando SQL, puede agregar parámetros a su consulta, como el comando LIKE:

```
SHOW [GLOBAL | SESSION] STATUS LIKE 'Key%';
```

O el comando Where:

```
SHOW [GLOBAL | SESSION] STATUS WHERE VALUE > 0;
```

La principal diferencia entre GLOBAL y SESSION es que con el modificador GLOBAL el comando muestra información agregada sobre el servidor y todas sus conexiones, mientras que el modificador SESSION solo mostrará los valores de la conexión actual.

Lea información del servidor en línea: https://riptutorial.com/es/mysql/topic/9924/informacion-del-servidor

Capítulo 36: INSERTAR

Sintaxis

1. INSERTAR [LOW_PRIORITY | RETRASADO | HIGH_PRIORITY] [IGNORE] [INTO] tbl_name [PARTITION (partition_name, ...)] [(col_name, ...)] {VALUES | VALOR} ({expr | DEFAULT}, ...), (...), ... [ACTUALIZACIÓN DE LA LLAVE DUPLICADA col_name = expr [, col_name = expr] ...]

2. INSERTAR [LOW_PRIORITY | RETRASADO | HIGH_PRIORITY] [IGNORE] [INTO] tbl_name [PARTITION (partition_name, ...)] SET col_name = {expr | DEFAULT}, ... [ON DUPLICATE KEY UPDATE col_name = expr [, col_name = expr] ...]

3. INSERTAR [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE] [INTO] tbl_name [PARTITION (partition_name, ...)] [(col_name, ...)] SELECT ... [ON DUPLICATE KEY UPDATE col_name = expr [, col_name = expr]]]

4. Una expresión expr puede hacer referencia a cualquier columna que se haya establecido anteriormente en una lista de valores. Por ejemplo, puede hacer esto porque el valor para col2 se refiere a col1, que se ha asignado previamente:
   Insertar en tbl_name (col1, col2) VALORES (15, col1 * 2);

5. Las instrucciones INSERT que utilizan la sintaxis de VALUES pueden insertar varias filas. Para hacer esto, incluya múltiples listas de valores de columna, cada uno entre paréntesis y separados por comas. Ejemplo:
   INSERTAR EN tbl_name (a, b, c) VALORES (1,2,3), (4,5,6), (7,8,9);

6. La lista de valores para cada fila debe estar entre paréntesis. La siguiente declaración es ilegal porque el número de valores en la lista no coincide con el número de nombres de columna:
   Insertar en tbl_name (a, b, c) VALORES (1,2,3,4,5,6,7,8,9);

7. INSERTAR ... SELECCIONAR Sintaxis
   INSERTAR [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE] [INTO] tbl_name [PARTITION (partition_name, ...)] [(col_name, ...)] SELECT ... [ON DUPLICATE KEY UPDATE col_name = expr, ...]

8. Con INSERT ... SELECT, puede insertar rápidamente muchas filas en una tabla de una o varias tablas. Por ejemplo:
   INSERTAR EN tbl_temp2 (fld_id) SELECCIONE tbl_temp1.fld_order_id DESDE tbl_temp1 DONDE tbl_temp1.fld_order_id> 100;

Observaciones

Sintaxis oficial de INSERT

Examples

Inserto Básico

```
INSERT INTO `table_name` (`field_one`, `field_two`) VALUES ('value_one', 'value_two');
```

En este ejemplo trivial, `table_name` es donde se agregarán los datos, `field_one` y `field_two` son campos para configurar los datos, y `value_one` y `value_two` son los datos que se deben hacer contra `field_one` y `field_two` respectivamente.

Es una buena práctica enumerar los campos en los que está insertando datos dentro de su código, ya que si la tabla cambia y se agregan nuevas columnas, su inserción se rompería si no estuvieran allí.

INSERTAR, ACTUALIZACIÓN CLAVE DUPLICADA

```
INSERT INTO `table_name`
    (`index_field`, `other_field_1`, `other_field_2`)
    VALUES
    ('index_value', 'insert_value', 'other_value')
    ON DUPLICATE KEY UPDATE
        `other_field_1` = 'update_value',
        `other_field_2` = VALUES(`other_field_2`);
```

Esto `INSERT` en `table_name` los valores especificados, pero si la clave única ya existe, actualizará el `other_field_1` para que tenga un nuevo valor.
A veces, al actualizar en clave duplicada resulta útil utilizar `VALUES()` para acceder al valor original que se pasó a `INSERT` lugar de establecer el valor directamente. De esta manera, puede establecer diferentes valores utilizando `INSERT` y `UPDATE` . Vea el ejemplo anterior donde `other_field_1` se establece en `insert_value` en `INSERT` o en `update_value` en `UPDATE` mientras que `other_field_2` siempre se establece en `other_value` .

Para que funcione Insertar en la actualización de la clave duplicada (IODKU), es crucial que el esquema contenga una clave única que indique un conflicto duplicado. Esta clave única puede ser una clave principal o no. Puede ser una clave única en una sola columna o una columna múltiple (clave compuesta).

Insertando múltiples filas

```
INSERT INTO `my_table` (`field_1`, `field_2`) VALUES
    ('data_1', 'data_2'),
    ('data_1', 'data_3'),
    ('data_4', 'data_5');
```

Esta es una manera fácil de agregar varias filas a la vez con una `INSERT` .

Este tipo de inserción 'por lotes' es mucho más rápida que insertar filas una por una. Por lo general, insertar 100 filas en una sola inserción de lotes de esta manera es 10 veces más rápido

que insertarlas todas individualmente.

Ignorando las filas existentes

Al importar conjuntos de datos grandes, puede ser preferible, bajo ciertas circunstancias, omitir filas que generalmente causan que la consulta falle debido a una restricción de columna, por ejemplo, claves primarias duplicadas. Esto se puede hacer utilizando `INSERT IGNORE` .

Considere la siguiente base de datos de ejemplo:

```
SELECT * FROM `people`;
--- Produces:
+----+------+
| id | name |
+----+------+
|  1 | john |
|  2 | anna |
+----+------+


INSERT IGNORE INTO `people` (`id`, `name`) VALUES
    ('2', 'anna'), --- Without the IGNORE keyword, this record would produce an error
    ('3', 'mike');

SELECT * FROM `people`;
--- Produces:
+----+--------+
| id |  name  |
+----+--------+
|  1 |  john  |
|  2 |  anna  |
|  3 |  mike  |
+----+--------+
```

Lo importante a recordar es que *INSERT IGNORE* también omitirá silenciosamente otros errores, aquí está lo que dice la documentación oficial de Mysql:

> Las conversiones de datos que podrían generar errores abortan la declaración si IGNORE no está> especificado. Con IGNORE, los valores no válidos se ajustan a los valores más cercanos y se insertan; Se producen advertencias pero la declaración no se cancela.

Nota: - La sección a continuación se agrega para completar, pero no se considera la mejor práctica (esto podría fallar, por ejemplo, si se agrega otra columna a la tabla).

Si especifica el valor de la columna correspondiente para todas las columnas de la tabla, puede ignorar la lista de columnas en la `INSERT` siguiente manera:

```
INSERT INTO `my_table` VALUES
        ('data_1', 'data_2'),
        ('data_1', 'data_3'),
        ('data_4', 'data_5');
```

INSERT SELECT (Insertando datos de otra tabla)

Esta es la forma básica de insertar datos de otra tabla con la instrucción SELECT.

```
INSERT INTO `tableA` (`field_one`, `field_two`)
   SELECT `tableB`.`field_one`, `tableB`.`field_two`
   FROM `tableB`
   WHERE `tableB`.clmn <> 'someValue'
   ORDER BY `tableB`.`sorting_clmn`;
```

Puede `SELECT * FROM` , pero la `tableA` y la `tableB` *deben* tener un recuento de columnas coincidente y los tipos de datos correspondientes.

Las columnas con `AUTO_INCREMENT` se tratan como en la cláusula `INSERT` with `VALUES` .

Esta sintaxis facilita el llenado de tablas (temporales) con datos de otras tablas, incluso más cuando los datos se filtran en la inserción.

INSERTAR con AUTO_INCREMENT + LAST_INSERT_ID ()

Cuando una tabla tiene una `AUTO_INCREMENT PRIMARY KEY` , normalmente uno no insertar en esa columna. En su lugar, especifique todas las demás columnas, luego pregunte cuál era la nueva identificación.

```
CREATE TABLE t (
    id SMALLINT UNSIGNED AUTO_INCREMENT NOT NULL,
    this ...,
    that ...,
    PRIMARY KEY(id) );

INSERT INTO t (this, that) VALUES (..., ...);
SELECT LAST_INSERT_ID() INTO @id;
INSERT INTO another_table (..., t_id, ...) VALUES (..., @id, ...);
```

Tenga en cuenta que `LAST_INSERT_ID()` está vinculado a la sesión, por lo que incluso si se insertan varias conexiones en la misma tabla, cada una con su propia ID.

Es probable que la API de su cliente tenga una forma alternativa de obtener `LAST_INSERT_ID()` sin realizar realmente una `SELECT` y devolverle el valor al cliente en lugar de dejarla en una `@variable` dentro de MySQL. Tal es generalmente preferible.

Más largo, más detallado, ejemplo.

El uso "normal" de IODKU es activar una "clave duplicada" basada en alguna tecla `UNIQUE` , no en la `AUTO_INCREMENT PRIMARY KEY` . Lo siguiente demuestra tal. Tenga en cuenta que *no* proporciona el `id` en el INSERT.

Configuración de ejemplos a seguir:

```
CREATE TABLE iodku (
    id INT AUTO_INCREMENT NOT NULL,
```

```
    name VARCHAR(99) NOT NULL,
    misc INT NOT NULL,
    PRIMARY KEY(id),
    UNIQUE(name)
) ENGINE=InnoDB;

INSERT INTO iodku (name, misc)
    VALUES
    ('Leslie', 123),
    ('Sally', 456);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0
+----+--------+------+
| id | name   | misc |
+----+--------+------+
|  1 | Leslie |  123 |
|  2 | Sally  |  456 |
+----+--------+------+
```

El caso de IODKU realizando una "actualización" y `LAST_INSERT_ID()` recuperando la `id` relevante:

```
INSERT INTO iodku (name, misc)
    VALUES
    ('Sally', 3333)            -- should update
    ON DUPLICATE KEY UPDATE    -- `name` will trigger "duplicate key"
    id = LAST_INSERT_ID(id),
    misc = VALUES(misc);
SELECT LAST_INSERT_ID();       -- picking up existing value
+------------------+
| LAST_INSERT_ID() |
+------------------+
|                2 |
+------------------+
```

El caso en el que IODKU realiza una "inserción" y `LAST_INSERT_ID()` recupera la nueva `id` :

```
INSERT INTO iodku (name, misc)
    VALUES
    ('Dana', 789)          -- Should insert
    ON DUPLICATE KEY UPDATE
    id = LAST_INSERT_ID(id),
    misc = VALUES(misc);
SELECT LAST_INSERT_ID();   -- picking up new value
+------------------+
| LAST_INSERT_ID() |
+------------------+
|                3 |
+------------------+
```

Contenido de la tabla resultante:

```
SELECT * FROM iodku;
+----+--------+------+
| id | name   | misc |
+----+--------+------+
|  1 | Leslie |  123 |
|  2 | Sally  | 3333 |  -- IODKU changed this
```

```
|  3 | Dana   |  789 |  -- IODKU added this
+----+--------+------+
```

IDs AUTO_INCREMENT perdidos

Varias funciones 'insertar' pueden "quemar" los identificadores. Aquí hay un ejemplo, usando InnoDB (otros motores pueden funcionar de manera diferente):

```
CREATE TABLE Burn (
    id SMALLINT UNSIGNED AUTO_INCREMENT NOT NULL,
    name VARCHAR(99) NOT NULL,
    PRIMARY KEY(id),
    UNIQUE(name)
        ) ENGINE=InnoDB;

INSERT IGNORE INTO Burn (name) VALUES ('first'), ('second');
SELECT LAST_INSERT_ID();          -- 1
SELECT * FROM Burn ORDER BY id;
  +----+--------+
  |  1 | first  |
  |  2 | second |
  +----+--------+

INSERT IGNORE INTO Burn (name) VALUES ('second');  -- dup 'IGNOREd', but id=3 is burned
SELECT LAST_INSERT_ID();          -- Still "1" -- can't trust in this situation
SELECT * FROM Burn ORDER BY id;
  +----+--------+
  |  1 | first  |
  |  2 | second |
  +----+--------+

INSERT IGNORE INTO Burn (name) VALUES ('third');
SELECT LAST_INSERT_ID();           -- now "4"
SELECT * FROM Burn ORDER BY id;    -- note that id=3 was skipped over
  +----+--------+
  |  1 | first  |
  |  2 | second |
  |  4 | third  |    -- notice that id=3 has been 'burned'
  +----+--------+
```

Piénselo (aproximadamente) de esta manera: primero, la inserción se ve para ver cuántas filas se *pueden* insertar. Luego, toma tantos valores del auto_increment para esa tabla. Finalmente, inserte las filas, utilizando los identificadores que sean necesarios y queme las sobras.

La única vez que los restos son recuperables es si el sistema se apaga y se reinicia. Al reiniciar, efectivamente se realiza `MAX(id)` . Esto puede reutilizar los identificadores que fueron quemados o que fueron liberados por `DELETE`s de los identificadores más altos.

Esencialmente, cualquier versión de `INSERT` (incluido `REPLACE` , que es `DELETE` + `INSERT`) puede grabar identificadores. En InnoDB, la variable global (¡no la sesión!) `innodb_autoinc_lock_mode` se puede usar para controlar algo de lo que está sucediendo.

Cuando se "normalizan" cadenas largas en un `AUTO INCREMENT id` , la grabación puede ocurrir fácilmente. Esto *podría* llevar a desbordar el tamaño de la `INT` que eligió.

Lea INSERTAR en línea: https://riptutorial.com/es/mysql/topic/866/insertar

Capítulo 37: Instalar el contenedor Mysql con Docker-Compose

Examples

Ejemplo simple con docker-compose

Este es un ejemplo simple para crear un servidor mysql con docker

1.- crear **docker-compose.yml** :

Nota: Si desea utilizar el mismo contenedor para todos sus proyectos, debe crear una RUTA en su PAPEL DE INICIO. Si desea crearlo para cada proyecto, puede crear un directorio **docker** en su proyecto.

```
version: '2'
services:
  cabin_db:
    image: mysql:latest
    volumes:
      - "./.mysql-data/db:/var/lib/mysql"
    restart: always
    ports:
      - 3306:3306
    environment:
      MYSQL_ROOT_PASSWORD: rootpw
      MYSQL_DATABASE: cabin
      MYSQL_USER: cabin
      MYSQL_PASSWORD: cabinpw
```

2.- ejecutarlo:

```
cd PATH_TO_DOCKER-COMPOSE.YML
docker-compose up -d
```

3.- conectarse al servidor

```
mysql -h 127.0.0.1 -u root -P 3306 -p rootpw
```

¡¡Hurra!!

4.- parar servidor

```
docker-compose stop
```

Lea Instalar el contenedor Mysql con Docker-Compose en línea:
https://riptutorial.com/es/mysql/topic/4458/instalar-el-contenedor-mysql-con-docker-compose

Capítulo 38: JSON

Introducción

A partir de MySQL 5.7.8, MySQL es compatible con un tipo de datos JSON nativo que permite un acceso eficiente a los datos en documentos JSON (Notación de Objetos de JavaScript). https://dev.mysql.com/doc/refman/5.7/en/json.html

Observaciones

A partir de MySQL 5.7.8, MySQL se envía con un tipo JSON. Muchos desarrolladores han estado guardando datos JSON en columnas de texto durante un tiempo de registro, pero el tipo JSON es diferente, los datos se guardan en formato binario después de la validación. Eso evita la sobrecarga de analizar el texto en cada lectura.

Examples

Crear una tabla simple con una clave principal y un campo JSON

```
CREATE TABLE table_name (
    id INT NOT NULL AUTO_INCREMENT,
    json_col JSON,
    PRIMARY KEY(id)
);
```

Insertar un simple JSON

```
INSERT INTO
    table_name (json_col)
VALUES
    ('{"City": "Galle", "Description": "Best damn city in the world"}');
```

Eso es simple como se puede obtener, pero tenga en cuenta que debido a que las claves del diccionario JSON deben estar entre comillas dobles, todo debe estar entre comillas simples. Si la consulta tiene éxito, los datos se almacenarán en un formato binario.

Insertar datos mixtos en un campo JSON.

Esto inserta un diccionario json donde uno de los miembros es una matriz de cadenas en la tabla que se creó en otro ejemplo.

```
INSERT INTO myjson(dict)
VALUES('{"opening":"Sicilian","variations":["pelikan","dragon","najdorf"]}');
```

Tenga en cuenta, una vez más, que debe tener cuidado con el uso de comillas simples y dobles.

Todo el asunto tiene que estar envuelto en comillas simples.

Actualizando un campo JSON

En el ejemplo anterior vimos cómo se pueden insertar tipos de datos mixtos en un campo JSON. ¿Qué pasa si queremos actualizar ese campo? Vamos a agregar *scheveningen* a la matriz denominada `variations` en el ejemplo anterior.

```
UPDATE
    myjson
SET
    dict=JSON_ARRAY_APPEND(dict,'$.variations','scheveningen')
WHERE
    id = 2;
```

Notas:

1. La matriz `$.variations` en nuestro diccionario json. El símbolo $ representa la documentación de json. Para obtener una explicación completa de las rutas json reconocidas por mysql, consulte https://dev.mysql.com/doc/refman/5.7/en/json-path-syntax.html
2. Dado que aún no tenemos un ejemplo sobre la consulta mediante los campos json, este ejemplo utiliza la clave principal.

Ahora si hacemos `SELECT * FROM myjson` veremos

```
+----+-------------------------------------------------------------------------------------
-+
| id | dict
|
+---+--------------------------------------------------------------------------------------
+
| 2  | {"opening": "Sicilian", "variations": ["pelikan", "dragon", "najdorf", "scheveningen"]}
|
+----+-------------------------------------------------------------------------------------
-+
1 row in set (0.00 sec)
```

Datos CAST a tipo JSON

Esto convierte las cadenas json válidas al tipo MySQL JSON:

```
 SELECT CAST('[1,2,3]' as JSON) ;
 SELECT CAST('{"opening":"Sicilian","variations":["pelikan","dragon","najdorf"]}' as JSON);
```

Crear Json Object y Array

`JSON_OBJECT` crea objetos JSON:

```
 SELECT JSON_OBJECT('key1',col1 , 'key2',col2 , 'key3','col3') as myobj;
```

`JSON_ARRAY` crea JSON Array también:

```
 SELECT JSON_ARRAY(col1,col2,'col3') as myarray;
```

Nota: myobj.key3 y myarray [2] son "col3" como cadena fija.

También mezclan datos JSON:

```
 SELECT JSON_OBJECT("opening","Sicilian",
"variations",JSON_ARRAY("pelikan","dragon","najdorf") ) as mymixed ;
```

Lea JSON en línea: https://riptutorial.com/es/mysql/topic/2985/json

Capítulo 39: La optimización del rendimiento

Sintaxis

- No utilice DISTINCT y GROUP BY en el mismo SELECT.
- No pagines a través de OFFSET, "recuerda dónde lo dejaste".
- DONDE (a, b) = (22,33) no se optimiza en absoluto.
- Diga explícitamente ALL o DISTINCT después de UNION: le recuerda que debe elegir entre el ALL más rápido o el DISTINCT más lento.
- No utilice SELECT *, especialmente si tiene columnas TEXT o BLOB que no necesita. Hay sobrecarga en tablas tmp y transmisión.
- Es más rápido cuando GROUP BY y ORDER BY pueden tener exactamente la misma lista.
- No uses el ÍNDICE DE FUERZA; Puede ayudar hoy, pero probablemente dolerá mañana.

Observaciones

Consulte también las discusiones sobre ORDENAR, COMO, REGEXP, etc. Nota: esto necesita edición con enlaces y más temas.

Libro de cocina sobre la construcción de índices óptimos .

Examples

Agregue el índice correcto

Este es un tema enorme, pero también es el problema más importante del "rendimiento".

La lección principal para un principiante es aprender de los índices "compuestos". Aquí hay un ejemplo rápido:

```
INDEX(last_name, first_name)
```

Es excelente para estos:

```
WHERE last_name = '...'
WHERE first_name = '...' AND last_name = '...'   -- (order in WHERE does not matter)
```

pero no para

```
WHERE first_name = '...'   -- order in INDEX _does_ matter
```

```
WHERE last_name = '...' OR first_name = '...'   -- "OR" is a killer
```

Establecer el caché correctamente

`innodb_buffer_pool_size` debería ser aproximadamente el 70% de la RAM disponible.

Evitar construcciones ineficientes.

```
x IN ( SELECT ... )
```

convertirse en un `JOIN`

Cuando sea posible, evite `OR` .

No "esconda" una columna indexada en una función, como la `WHERE DATE(x) = ...` ; reformular como `WHERE x = ...`

Por lo general, puede evitar `WHERE LCASE(name1) = LCASE(name2)` teniendo una intercalación adecuada.

No use `OFFSET` para "paginación", en lugar de eso, 'recuerde donde lo dejó'.

Evite `SELECT * ...` (a menos que esté depurando).

Nota para Maria Deleva, Barranka, Batsu: Este es un marcador de posición; Por favor, elimine estos elementos a medida que construye ejemplos a gran escala. Después de que hayas hecho lo que puedas, me moveré para elaborar el resto y / o lanzarlos.

Negativos

Aquí hay algunas cosas que probablemente no ayuden al rendimiento. Se derivan de información desactualizada y / o de ingenuidad.

- InnoDB ha mejorado hasta el punto de que es poco probable que MyISAM sea mejor.
- `PARTITIONing` rara vez proporciona beneficios de rendimiento; Incluso puede dañar el rendimiento.
- La configuración de `query_cache_size` mayor que 100M generalmente *afectará el* rendimiento.
- El aumento de muchos valores en `my.cnf` puede llevar a un "intercambio", que es un *grave* problema de rendimiento.
- Los "índices de prefijo" (como `INDEX(foo(20))`) son generalmente inútiles.
- `OPTIMIZE TABLE` es casi siempre inútil. (Y se trata de bloquear la mesa.)

Tener un índice

Lo más importante para acelerar una consulta en cualquier tabla no pequeña es tener un índice adecuado.

```
WHERE a = 12  --> INDEX(a)
WHERE a > 12  --> INDEX(a)

WHERE a = 12 AND b > 78  --> INDEX(a,b) is more useful than INDEX(b,a)
WHERE a > 12 AND b > 78  --> INDEX(a) or INDEX(b); no way to handle both ranges

ORDER BY x  --> INDEX(x)
ORDER BY x, y  --> INDEX(x,y) in that order
ORDER BY x DESC, y ASC  --> No index helps - because of mixing ASC and DESC
```

No te escondas en función

Un error común es ocultar una columna indexada dentro de una llamada de función. Por ejemplo, esto no puede ser ayudado por un índice:

```
WHERE DATE(dt) = '2000-01-01'
```

En cambio, dado el `INDEX(dt)` , estos pueden usar el índice:

```
WHERE dt = '2000-01-01'  -- if `dt` is datatype `DATE`
```

Esto funciona para `DATE` , `DATETIME` , `TIMESTAMP` e incluso `DATETIME(6)` (microsegundos):

```
WHERE dt >= '2000-01-01'
  AND dt  < '2000-01-01' + INTERVAL 1 DAY
```

O

En general `OR` mata la optimización.

```
WHERE a = 12 OR b = 78
```

no puede usar `INDEX(a,b)` , y puede o no puede usar `INDEX(a), INDEX(b)` través de "fusión de índice". La fusión de índices es mejor que nada, pero solo a medias.

```
WHERE x = 3 OR x = 5
```

se convierte en

```
WHERE x IN (3, 5)
```

que *puede* utilizar un índice con `x` en ella.

Subconsultas

Las subconsultas vienen en varios sabores y tienen un potencial de optimización diferente. Primero, tenga en cuenta que las subconsultas pueden ser "correlacionadas" o "no correlacionadas". Correlacionado significa que dependen de algún valor externo a la subconsulta.

Esto generalmente implica que la subconsulta *debe* ser reevaluada para cada valor externo.

Esta subconsulta correlacionada es a menudo bastante buena. Nota: Debe devolver como máximo 1 valor. A menudo es útil como una alternativa, aunque no necesariamente más rápida que una `LEFT JOIN` .

```
SELECT a, b, ( SELECT ... FROM t WHERE t.x = u.x ) AS c
    FROM u ...
SELECT a, b, ( SELECT MAX(x) ... ) AS c
    FROM u ...
SELECT a, b, ( SELECT x FROM t ORDER BY ... LIMIT 1 ) AS c
    FROM u ...
```

Esto generalmente no está correlacionado

```
SELECT ...
    FROM ( SELECT ... ) AS a
    JOIN b ON ...
```

Notas sobre el `FROM-SELECT` :

- Si vuelve 1 fila, genial.
- Un buen paradigma (nuevamente "1 fila") es que la subconsulta sea `( SELECT @n := 0 )` , inicializando así una `@variable para uso en el resto o la consulta.
- Si devuelve muchas filas *y* la `JOIN` también es `( SELECT ... )` con muchas filas, la eficiencia puede ser terrible. Pre-5.6, no había índice, por lo que se convirtió en una `CROSS JOIN` ; 5.6+ implica deducir el mejor índice en las tablas temporales y luego generarlo, solo para desecharlo cuando termine con `SELECT` .

ÚNETE + GRUPO POR

Un problema común que conduce a una consulta ineficiente es algo como esto:

```
SELECT ...
    FROM a
    JOIN b  ON ...
    WHERE ...
    GROUP BY a.id
```

Primero, el `JOIN` expande el número de filas; luego el `GROUP BY` lo reduce de nuevo el número de filas en `a` .

Puede que no haya ninguna buena elección para resolver este problema de explosión e implosión. Una opción posible es convertir el `JOIN` en una subconsulta correlacionada en `SELECT` . Esto también elimina el `GROUP BY` .

Lea La optimización del rendimiento en línea: https://riptutorial.com/es/mysql/topic/4292/la-optimizacion-del-rendimiento

Capítulo 40: Límite y compensación

Sintaxis

- SELECCIONAR column_1 [, column_2]
  FROM tabla_1
  ORDEN POR order_column
  LIMIT row_count [OFFSET row_offset]
- SELECCIONAR column_1 [, column_2]
  FROM tabla_1
  ORDEN POR order_column
  LIMIT [row_offset,] row_count

Observaciones

"Límite" podría significar "Número máximo de filas en una tabla".

"Offset" significa selección desde el número de `row` (no debe confundirse con el valor de la clave principal o cualquier valor de datos de campo)

Examples

Relación de límite y compensación

Teniendo en cuenta la siguiente tabla de `users` :

carné de identidad	nombre de usuario
1	Usuario1
2	Usuario2
3	Usuario3
4	Usuario4
5	Usuario5

Para restringir el número de filas en el conjunto de resultados de una consulta `SELECT` , la cláusula `LIMIT` se puede usar junto con uno o dos enteros positivos como argumentos (cero incluido).

Cláusula `LIMIT` con un argumento

Cuando se usa un argumento, el conjunto de resultados solo se limitará al número especificado

de la siguiente manera:

```
SELECT * FROM users ORDER BY id ASC LIMIT 2
```

carné de identidad	nombre de usuario
1	Usuario1
2	Usuario2

Si el valor del argumento es `0` , el conjunto de resultados estará vacío.

También tenga en cuenta que la cláusula `ORDER BY` puede ser importante para especificar las primeras filas del conjunto de resultados que se presentarán (al ordenar por otra columna).

Cláusula `LIMIT` con dos argumentos.

Cuando se usan dos argumentos en una cláusula `LIMIT` :

- el **primer** argumento representa la fila desde la cual se presentarán las filas del conjunto de resultados; este número se menciona a menudo como un **desplazamiento** , ya que representa la fila anterior a la fila inicial del conjunto de resultados restringido. Esto permite que el argumento reciba `0` como valor y, por lo tanto, tenga en cuenta la primera fila del conjunto de resultados no restringido.
- el **segundo** argumento especifica el número máximo de filas que se devolverán en el conjunto de resultados (de manera similar al ejemplo del argumento único).

Por lo tanto la consulta:

```
SELECT * FROM users ORDER BY id ASC LIMIT 2, 3
```

Presenta el siguiente conjunto de resultados:

carné de identidad	nombre de usuario
3	Usuario3
4	Usuario4
5	Usuario5

Observe que cuando el argumento de **compensación** es `0` , el conjunto de resultados será equivalente a una cláusula `LIMIT` un argumento. Esto significa que las siguientes 2 consultas:

```
SELECT * FROM users ORDER BY id ASC LIMIT 0, 2

SELECT * FROM users ORDER BY id ASC LIMIT 2
```

Producir el mismo conjunto de resultados:

carné de identidad	nombre de usuario
1	Usuario1
2	Usuario2

`OFFSET` palabra clave: sintaxis alternativa

Una sintaxis alternativa para la cláusula `LIMIT` con dos argumentos consiste en el uso de la palabra clave `OFFSET` después del primer argumento de la siguiente manera:

```
SELECT * FROM users ORDER BY id ASC LIMIT 2 OFFSET 3
```

Esta consulta devolvería el siguiente conjunto de resultados:

carné de identidad	nombre de usuario
3	Usuario3
4	Usuario4

Observe que en esta sintaxis alternativa los argumentos tienen sus posiciones cambiadas:

- el **primer** argumento representa el número de filas que se devolverán en el conjunto de resultados;
- El **segundo** argumento representa el desplazamiento.

Lea Límite y compensación en línea: https://riptutorial.com/es/mysql/topic/548/limite-y-compensacion

Capítulo 41: Manejo de zonas horarias

Observaciones

Cuando necesite manejar información de tiempo para una base de usuarios de todo el mundo en MySQL, use el tipo de datos TIMESTAMP en sus tablas.

Para cada usuario, almacene una columna de zona horaria de preferencia de usuario. VARCHAR (64) es un buen tipo de datos para esa columna. Cuando un usuario se registra para usar su sistema, solicite el valor de la zona horaria. El mío es el tiempo del Atlántico, `America/Edmonton` . El suyo puede o no ser `Asia/Kolkata` o `Australia/NSW` . Para una interfaz de usuario para esta configuración de preferencia de usuario, el software WordPress.org tiene un buen ejemplo.

Finalmente, siempre que establezca una conexión desde su programa host (Java, php, lo que sea) a su DBMS en nombre de un usuario, emita el comando SQL

```
SET SESSION time_zone='(whatever tz string the user gave you)'
```

Antes de manejar cualquier dato de usuario que involucre tiempos. Luego, todas las veces de `TIMESTAMP` que haya instalado se procesarán en la hora local del usuario.

Esto hará que todas las horas que ingresen a sus tablas se conviertan a UTC, y que todas las horas que salgan se traduzcan a local. Funciona correctamente para NOW () y CURDATE (). De nuevo, debe usar TIMESTAMP y no los tipos de datos DATETIME o DATE para esto.

Asegúrese de que el sistema operativo de su servidor y las zonas horarias de MySQL predeterminadas estén configuradas en UTC. Si no hace esto antes de comenzar a cargar información en su base de datos, será casi imposible de arreglar. Si utiliza un proveedor para ejecutar MySQL, insista en que lo hagan correctamente.

Examples

Recupere la fecha y hora actual en una zona horaria particular.

Esto obtiene el valor de `NOW()` en la hora local, en la hora estándar de la India, y luego nuevamente en UTC.

```
SELECT NOW();
SET time_zone='Asia/Kolkata';
SELECT NOW();
SET time_zone='UTC';
SELECT NOW();
```

Convierte un valor `DATE` o` DATETIME` almacenado en otra zona horaria.

Si tiene un `DATE` o `DATETIME` almacenado (en una columna en algún lugar) se almacenó con

respecto a alguna zona horaria, pero en MySQL la zona horaria *no se* almacena con el valor. Por lo tanto, si desea convertirlo en otra zona horaria, puede hacerlo, pero debe conocer la zona horaria original. Usando `CONVERT_TZ()` hace la conversión. Este ejemplo muestra las filas vendidas en California en la hora local.

```
SELECT CONVERT_TZ(date_sold,'UTC','America/Los_Angeles') date_sold_local
  FROM sales
 WHERE state_sold = 'CA'
```

Recupere los valores almacenados de `TIMESTAMP` en una zona horaria particular

Esto es realmente fácil. Todos los valores de `TIMESTAMP` se almacenan en tiempo universal, y siempre se convierten a la configuración actual de `time_zone` cuando se procesan.

```
SET SESSION time_zone='America/Los_Angeles';
SELECT timestamp_sold
  FROM sales
 WHERE state_sold = 'CA'
```

¿Por qué es esto? `TIMESTAMP` valores de `TIMESTAMP` se basan en el venerable tipo de datos UNIX `time_t` . Esos sellos de tiempo de UNIX se almacenan como un número de segundos desde `1970-01-01 00:00:00` UTC.

Note que los valores de `TIMESTAMP` se almacenan en tiempo universal. `DATE` valores `DATE` y `DATETIME` se almacenan en cualquier hora local que estuviera vigente cuando se almacenaron.

¿Cuál es la configuración de zona horaria local de mi servidor?

Cada servidor tiene una configuración de zona horaria global predeterminada, configurada por el propietario de la máquina del servidor. Puede averiguar la configuración de la zona horaria actual de esta manera:

```
SELECT @@time_zone
```

Desafortunadamente, eso generalmente produce el valor `SYSTEM` , lo que significa que la hora de MySQL se rige por la configuración de zona horaria del sistema operativo del servidor.

Esta secuencia de consultas (sí, es un pirateo) le devuelve la compensación en minutos entre la configuración de zona horaria del servidor y UTC.

```
CREATE TEMPORARY TABLE times (dt DATETIME, ts TIMESTAMP);
SET time_zone = 'UTC';
INSERT INTO times VALUES(NOW(), NOW());
SET time_zone = 'SYSTEM';
SELECT dt, ts, TIMESTAMPDIFF(MINUTE, dt, ts)offset FROM times;
DROP TEMPORARY TABLE times;
```

¿Como funciona esto? Las dos columnas en la tabla temporal con diferentes tipos de datos son la

clave. `DATETIME` tipos de datos `DATETIME` siempre se almacenan en hora local en tablas y `TIMESTAMP` en UTC. Por lo tanto, la `INSERT` , realizada cuando la zona horaria está configurada en UTC, almacena dos valores de fecha / hora idénticos.

Luego, la instrucción SELECT, se realiza cuando la time_zone se establece en la hora local del servidor. `TIMESTAMP` s siempre se traducen de su formulario UTC almacenado a la hora local en las declaraciones SELECT. `DATETIME` s no son. Por lo tanto, la operación `TIMESTAMPDIFF(MINUTE...)` calcula la diferencia entre la hora local y la universal.

¿Qué valores de time_zone están disponibles en mi servidor?

Para obtener una lista de los posibles valores de time_zone en su instancia del servidor MySQL, use este comando.

```
SELECT mysql.time_zone_name.name
```

Normalmente, esto muestra la lista de ZoneInfo de zonas horarias mantenida por Paul Eggert en la Autoridad de Números Asignados de Internet . En todo el mundo hay aproximadamente 600 zonas horarias.

Los sistemas operativos similares a Unix (por ejemplo, distribuciones de Linux, distribuciones BSD y distribuciones modernas de Mac OS) reciben actualizaciones de rutina. La instalación de estas actualizaciones en un sistema operativo permite que las instancias de MySQL que se ejecutan allí rastreen los cambios en la zona horaria y los cambios de horario de verano / horario estándar.

Si obtiene una lista mucho más corta de nombres de zona horaria, su servidor está configurado de forma incompleta o se está ejecutando en Windows. Aquí hay instrucciones para que el administrador de su servidor instale y mantenga la lista de ZoneInfo.

Lea Manejo de zonas horarias en línea: https://riptutorial.com/es/mysql/topic/7849/manejo-de-zonas-horarias

Capítulo 42: Mesa plegable

Sintaxis

- DROP TABLE table_name;
- DROP TABLE IF EXISTS nombre_tabla; - para evitar el error molesto en el script automatizado
- TABLA DE CAÍDA t1, t2, t3; - DROP tablas múltiples
- DROP TABLA TEMPORAL t; - TIRAR una tabla de CREAR TABLA TEMPORAL ...

Parámetros

Parámetros	Detalles
TEMPORAL	Opcional. Especifica que solo la tabla DROP TABLE debe eliminar las tablas temporales.
Si existe	Opcional. Si se especifica, la instrucción DROP TABLE no generará un error si una de las tablas no existe.

Examples

Mesa plegable

Drop Table se usa para borrar la tabla de la base de datos.

Creando tabla:

Creando una tabla llamada tbl y luego borrando la tabla creada

```
CREATE TABLE tbl(
    id INT NOT NULL AUTO_INCREMENT,
    title VARCHAR(100) NOT NULL,
    author VARCHAR(40) NOT NULL,
    submission_date DATE,
    PRIMARY KEY (id)
);
```

Tabla de caída:

```
DROP TABLE tbl;
```

> **TENGA EN CUENTA**
>
> La tabla eliminada eliminará completamente la tabla de la base de datos y toda su

información, y no se recuperará.

Eliminar tablas de la base de datos

```
DROP TABLE Database.table_name
```

Lea Mesa plegable en línea: https://riptutorial.com/es/mysql/topic/4123/mesa-plegable

Capítulo 43: Mesas temporales

Examples

Crear tabla temporal

Las tablas temporales podrían ser muy útiles para mantener datos temporales. La opción de tablas temporales está disponible en MySQL versión 3.23 y superior.

La tabla temporal se destruirá automáticamente cuando finalice la sesión o se cierre la conexión. El usuario también puede soltar la tabla temporal.

El mismo nombre de tabla temporal se puede usar en muchas conexiones al mismo tiempo, porque la tabla temporal solo está disponible y es accesible para el cliente que crea esa tabla.

La tabla temporal se puede crear en los siguientes tipos

```
--->Basic temporary table creation
CREATE TEMPORARY TABLE tempTable1(
       id INT NOT NULL AUTO_INCREMENT,
       title VARCHAR(100) NOT NULL,
       PRIMARY KEY ( id )
    );

--->Temporary table creation from select query
CREATE TEMPORARY TABLE tempTable1
    SELECT ColumnName1,ColumnName2,... FROM table1;
```

Puede agregar índices a medida que construye la tabla:

```
CREATE TEMPORARY TABLE tempTable1
        ( PRIMARY KEY(ColumnName2) )
    SELECT ColumnName1,ColumnName2,... FROM table1;
```

`IF NOT EXISTS` palabra clave `IF NOT EXISTS` se puede usar como se menciona a continuación para evitar *el* error *'ya existe la tabla'* . Pero en ese caso, la tabla no se creará si el nombre de la tabla que está utilizando ya existe en su sesión actual.

```
CREATE TEMPORARY TABLE IF NOT EXISTS tempTable1
    SELECT ColumnName1,ColumnName2,... FROM table1;
```

Drop Temporary Table

Eliminar tabla temporal se utiliza para eliminar la tabla temporal que se creó en su sesión actual.

```
DROP TEMPORARY TABLE tempTable1

DROP TEMPORARY TABLE IF EXISTS tempTable1
```

Utilice `IF EXISTS` para evitar que se produzcan errores en las tablas que pueden no existir

Lea Mesas temporales en línea: https://riptutorial.com/es/mysql/topic/5757/mesas-temporales

Capítulo 44: Motor myisam

Observaciones

A lo largo de los años, InnoDB ha mejorado hasta el punto de que casi siempre es mejor que MyISAM, al menos las versiones compatibles actualmente. En pocas palabras: no use MyISAM, excepto quizás para tablas que son verdaderamente temporales.

Una ventaja de MyISAM sobre InnoDB es que es 2x-3x más pequeño en el espacio requerido en el disco.

Cuando InnoDB salió por primera vez, MyISAM todavía era un motor viable. Pero con la llegada de XtraDB y 5.6, InnoDB se convirtió en "mejor" que MyISAM en la mayoría de los puntos de referencia.

Se rumorea que la próxima versión importante eliminará la necesidad de MyISAM al crear tablas realmente temporales de InnoDB y al mover las tablas del sistema a InnoDB.

Examples

MOTOR = MyISAM

```
CREATE TABLE foo (
     ...
) ENGINE=MyISAM;
```

Lea Motor myisam en línea: https://riptutorial.com/es/mysql/topic/4710/motor-myisam

Capítulo 45: MySQL LOCK TABLE

Sintaxis

- LOCK TABLES table_name [LEER | ESCRIBIR]; // Bloquear tabla
- TABLAS DE DESBLOQUEO; // Desbloquear Tablas

Observaciones

El bloqueo se utiliza para resolver problemas de concurrencia. El bloqueo se requiere solo cuando se ejecuta una transacción, que primero lee un valor de una base de datos y luego escribe ese valor en la base de datos. Los bloqueos nunca son necesarios para las operaciones de inserción, actualización o eliminación autónomas.

Hay dos tipos de cerraduras disponibles

READ LOCK - cuando un usuario solo lee de una tabla.

ESCRIBIR BLOQUEO: cuando un usuario está leyendo y escribiendo en una tabla.

Cuando un usuario tiene un `WRITE LOCK` en una tabla, ningún otro usuario puede leer o escribir en esa tabla. Cuando un usuario mantiene un `READ LOCK` en una tabla, otros usuarios también pueden leer o retener un `READ LOCK` , pero ningún usuario puede escribir o retener un `WRITE LOCK` en esa tabla.

Si el motor de almacenamiento predeterminado es InnoDB, MySQL usa automáticamente el bloqueo de nivel de fila para que múltiples transacciones puedan usar la misma tabla simultáneamente para lectura y escritura, sin que se hagan esperar.

Para todos los motores de almacenamiento que no sean InnoDB, MySQL usa el bloqueo de tablas.

Para más detalles sobre el bloqueo de la mesa, vea aquí.

Examples

Mysql Locks

Los bloqueos de tablas pueden ser una herramienta importante para `ENGINE=MyISAM` , pero rara vez son útiles para `ENGINE=InnoDB` . Si está tentado a utilizar bloqueos de tabla con InnoDB, debe reconsiderar cómo está trabajando con las transacciones.

MySQL permite que las sesiones de los clientes adquieran bloqueos de tablas explícitamente con el fin de cooperar con otras sesiones para acceder a las tablas, o para evitar que otras sesiones modifiquen las tablas durante los períodos en que una sesión requiere acceso exclusivo a ellas.

Una sesión puede adquirir o liberar bloqueos solo para sí misma. Una sesión no puede adquirir bloqueos para otra sesión o liberar bloqueos mantenidos por otra sesión.

Los bloqueos se pueden usar para emular transacciones o para obtener más velocidad al actualizar tablas. Esto se explica con más detalle más adelante en esta sección.

Comando: `LOCK TABLES table_name READ|WRITE;`

puede asignar solo el tipo de bloqueo a una sola tabla;

Ejemplo (READ LOCK):

```
LOCK TABLES table_name READ;
```

Ejemplo (BLOQUEO DE ESCRITURA):

```
LOCK TABLES table_name WRITE;
```

Para ver si el bloqueo está aplicado o no, use el siguiente comando

```
SHOW OPEN TABLES;
```

Para limpiar / eliminar todos los bloqueos, use el siguiente comando:

```
UNLOCK TABLES;
```

EJEMPLO:

```
LOCK TABLES products WRITE:
INSERT INTO products(id,product_name) SELECT id,old_product_name FROM old_products;
UNLOCK TABLES;
```

En el ejemplo anterior, cualquier conexión externa no puede escribir datos en la tabla de productos hasta que se desbloquee el producto de la tabla

EJEMPLO:

```
LOCK TABLES products READ:
INSERT INTO products(id,product_name) SELECT id,old_product_name FROM old_products;
UNLOCK TABLES;
```

En el ejemplo anterior, cualquier conexión externa no puede leer ningún dato de la tabla de productos hasta que se desbloquee el producto de la tabla

Bloqueo de nivel de fila

Si las tablas usan InnoDB, MySQL usa automáticamente el bloqueo de nivel de fila para que múltiples transacciones puedan usar la misma tabla simultáneamente para leer y escribir, sin que se hagan esperar.

Si dos transacciones intentan modificar la misma fila y ambas usan bloqueo a nivel de fila, una de las transacciones espera a que la otra se complete.

El bloqueo de nivel de fila también se puede obtener utilizando la `SELECT ... FOR UPDATE` para cada fila que se espera que se modifique.

Considere dos conexiones para explicar el bloqueo de nivel de fila en detalle

Conexión 1

```
START TRANSACTION;
SELECT ledgerAmount FROM accDetails WHERE id = 1 FOR UPDATE;
```

En la conexión 1, el bloqueo de nivel de fila obtenido por la `SELECT ... FOR UPDATE` .

Conexión 2

```
UPDATE accDetails SET ledgerAmount = ledgerAmount + 500 WHERE id=1;
```

Cuando alguien intente actualizar la misma fila en la conexión 2, eso esperará a que la conexión 1 finalice la transacción o se mostrará un mensaje de error según la configuración de `innodb_lock_wait_timeout` , que de manera predeterminada es de 50 segundos.

```
Error Code: 1205. Lock wait timeout exceeded; try restarting transaction
```

Para ver los detalles de este bloqueo, ejecute `SHOW ENGINE INNODB STATUS`

```
---TRANSACTION 1973004, ACTIVE 7 sec updating
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 360, 1 row lock(s)
MySQL thread id 4, OS thread handle 0x7f996beac700, query id 30 localhost root update
UPDATE accDetails SET ledgerAmount = ledgerAmount + 500 WHERE id=1
------- TRX HAS BEEN WAITING 7 SEC FOR THIS LOCK TO BE GRANTED:
```

Conexión 2

```
UPDATE accDetails SET ledgerAmount = ledgerAmount + 250 WHERE id=2;
1 row(s) affected
```

Pero mientras se actualiza alguna otra fila en la conexión 2 se ejecutará sin ningún error.

Conexión 1

```
UPDATE accDetails SET ledgerAmount = ledgerAmount + 750 WHERE id=1;
COMMIT;
1 row(s) affected
```

Ahora se libera el bloqueo de fila, porque la transacción se compromete en la Conexión 1.

Conexión 2

```
UPDATE accDetails SET ledgerAmount = ledgerAmount + 500 WHERE id=1;
1 row(s) affected
```

La actualización se ejecuta sin ningún error en la Conexión 2 después de que la Conexión 1 liberó el bloqueo de la fila al finalizar la transacción.

Lea MySQL LOCK TABLE en línea: https://riptutorial.com/es/mysql/topic/5233/mysql-lock-table

Capítulo 46: MySQL Unions

Sintaxis

- SELECCIONE column_name (s) FROM table1 UNION SELECT column_name (s) FROM table2;
- SELECCIONE column_name (s) FROM table1 UNION ALL SELECT column_name (s) FROM table2;
- SELECCIONE column_name (s) FROM table1 WHERE col_name = "XYZ" UNION ALL SELECT column_name (s) FROM table2 WHERE col_name = "XYZ";

Observaciones

`UNION DISTINCT` es lo mismo que `UNION` ; es más lento que `UNION ALL` debido a un pase de deduplicación. Una buena práctica es explicar siempre `DISTINCT` o `ALL` , indicando así que pensó en qué hacer.

Examples

Operador sindical

El operador UNION se utiliza para combinar el conjunto de resultados (*solo valores distintos*) de dos o más instrucciones SELECT.

Consulta: (Para seleccionar todas las diferentes ciudades (*solo valores distintos*) de las tablas "Clientes" y "Proveedores")

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;
```

Resultado:

```
Number of Records: 10

City
------
Aachen
Albuquerque
Anchorage
Annecy
Barcelona
Barquisimeto
Bend
Bergamo
Berlin
Bern
```

Union all

UNION ALL para seleccionar todas las ciudades (también valores duplicados) de las tablas "Clientes" y "Proveedores".

Consulta:

```
SELECT City FROM Customers
UNION ALL
SELECT City FROM Suppliers
ORDER BY City;
```

Resultado:

```
Number of Records: 12

City
-------
Aachen
Albuquerque
Anchorage
Ann Arbor
Annecy
Barcelona
Barquisimeto
Bend
Bergamo
Berlin
Berlin
Bern
```

UNION TODO CON DONDE

UNION ALL para seleccionar todas las ciudades alemanas (valores duplicados también) de las tablas "Clientes" y "Proveedores". Here `Country="Germany"` se debe especificar en la cláusula where.

Consulta:

```
SELECT City, Country FROM Customers
WHERE Country='Germany'
UNION ALL
SELECT City, Country FROM Suppliers
WHERE Country='Germany'
ORDER BY City;
```

Resultado:

```
Number of Records: 14
```

Ciudad	País

Aquisgrán	Alemania
Berlina	Alemania
Berlina	Alemania
Brandeburgo	Alemania
Cunewalde	Alemania
Cuxhaven	Alemania
Francfort	Alemania
Francfort aM	Alemania
Köln	Alemania
Leipzig	Alemania
Mannheim	Alemania
München	Alemania
Münster	Alemania
Stuttgart	Alemania

Lea MySQL Unions en línea: https://riptutorial.com/es/mysql/topic/5376/mysql-unions

Capítulo 47: mysqlimport

Parámetros

Parámetro	Descripción
`--delete -D`	Vacía la tabla antes de importar el archivo de texto.
`--fields-optionally-enclosed-by`	Definir el carácter que cita los campos.
`--fields-terminated-by`	terminador de campo
`--ignore -i`	ignorar la fila ingerida en caso de claves duplicadas
`--lines-terminated-by`	definir terminador de fila
`--password -p`	contraseña
`--port -P`	Puerto
`--replace -r`	sobrescribir la fila de entrada antigua en caso de claves duplicadas
`--user -u`	nombre de usuario
`--where -w`	especifique una condición

Observaciones

`mysqlimport` utilizará el nombre del archivo importado, después de eliminar la extensión, para determinar la tabla de destino.

Examples

Uso básico

Dado el archivo separado por pestañas `employee.txt`

1 `\t` Arthur Dent
2 `\t` Marvin
3 `\t` Zaphod Beeblebrox

```
$ mysql --user=user --password=password mycompany -e 'CREATE TABLE employee(id INT, name
VARCHAR(100), PRIMARY KEY (id))'
```

```
$ mysqlimport --user=user --password=password mycompany employee.txt
```

Usando un delimitador de campo personalizado

Dado el archivo de texto employee.txt

1 | Arthur Dent
2 | Marvin
3 | Zaphod Beeblebrox

```
$ mysqlimport --fields-terminated-by='|' mycompany employee.txt
```

Usando un delimitador de fila personalizado

Este ejemplo es útil para terminaciones similares a las ventanas:

```
$ mysqlimport --lines-terminated-by='\r\n' mycompany employee.txt
```

Manejo de claves duplicadas

Dada la mesa `Employee`

carné de identidad	Nombre
3	Yooden Vranx

Y el archivo `employee.txt`

1 `\t` Arthur Dent
2 `\t` Marvin
3 `\t` Zaphod Beeblebrox

La opción `--ignore` ignorará la entrada en claves duplicadas

```
$ mysqlimport --ignore mycompany employee.txt
```

carné de identidad	Nombre
1	Arthur Dent
2	Marvin
3	Yooden Vranx

La opción `--replace` sobrescribirá la entrada anterior

```
$ mysqlimport --replace mycompany employee.txt
```

carné de identidad	Nombre
1	Arthur Dent
2	Marvin
3	Zaphod Beeblebrox

Importación condicional

```
$ mysqlimport --where="id>2" mycompany employee.txt
```

Importar un csv estándar

```
$ mysqlimport
    --fields-optionally-enclosed-by='"'
    --fields-terminated-by=,
    --lines-terminated-by="\r\n"
    mycompany employee.csv
```

Lea mysqlimport en línea: https://riptutorial.com/es/mysql/topic/5215/mysqlimport

Capítulo 48: NULO

Examples

Usos para NULL

- Datos aún no conocidos, como fecha `end_date` , `rating`
- Datos opcionales, como `middle_initial` (aunque eso podría ser mejor que la cadena vacía)
- 0/0 - El resultado de ciertos cálculos, como cero dividido por cero.
- NULL no es igual a "" (cadena en blanco) o 0 (en el caso de un entero).
- ¿otros?

Prueba de valores nulos

- `IS NULL` / `IS NOT NULL` - `= NULL` no funciona como esperaba.
- `x <=> y` es una comparación "segura nula".

En un `LEFT JOIN` pruebas de filas de `a` para la cual *no* hay una fila correspondiente en `b` .

```
SELECT ...
    FROM a
    LEFT JOIN b ON ...
    WHERE b.id IS NULL
```

Lea NULO en línea: https://riptutorial.com/es/mysql/topic/6757/nulo

Capítulo 49: Operaciones de cuerdas

Parámetros

Nombre	Descripción
ASCII ()	Devuelve el valor numérico del carácter más a la izquierda
COMPARTIMIENTO()	Devuelve una cadena que contiene la representación binaria de un número.
BIT_LENGTH ()	Devuelve la longitud del argumento en bits
CARBONIZARSE()	Devuelve el caracter para cada entero pasado
CHAR_LENGTH ()	Devuelve el número de caracteres en el argumento
CHARACTER_LENGTH ()	Sinónimo para CHAR_LENGTH ()
CONCAT ()	Regresar cadena concatenada
CONCAT_WS ()	Volver concatenar con separador
ELT ()	Cadena de retorno en el número de índice
EXPORT_SET ()	Devuelve una cadena tal que por cada bit establecido en los bits de valor, obtienes una cadena activa y por cada bit no establecido, obtienes una cadena desactivada
CAMPO()	Devuelve el índice (posición) del primer argumento en los argumentos subsiguientes
FIND_IN_SET ()	Devuelve la posición del índice del primer argumento dentro del segundo argumento
FORMATO()	Devuelve un número formateado a un número especificado de lugares decimales
FROM_BASE64 ()	Decodificar a una cadena de base 64 y devolver el resultado
MALEFICIO()	Devuelve una representación hexadecimal de un valor decimal o cadena
INSERTAR()	Inserte una subcadena en la posición especificada hasta el número especificado de caracteres
INSTR ()	Devuelve el índice de la primera aparición de subcadenas.

Nombre	Descripción
LCASE ()	Sinónimo para LOWER ()
IZQUIERDA()	Devuelve el número de caracteres más a la izquierda como se especifica
LONGITUD()	Devuelve la longitud de una cadena en bytes
ME GUSTA	Simple patrón de coincidencia
CARGAR ARCHIVO()	Cargar el archivo nombrado
LOCALIZAR()	Devuelve la posición de la primera aparición de subcadena.
INFERIOR()	Devuelve el argumento en minúscula
LPAD ()	Devuelve el argumento de cadena, rellenado a la izquierda con la cadena especificada
LTRIM ()	Eliminar espacios iniciales
MAKE_SET ()	Devuelve un conjunto de cadenas separadas por comas que tienen el bit correspondiente en bits establecido
PARTIDO	Realizar búsqueda de texto completo
MEDIO()	Devuelve una subcadena a partir de la posición especificada
DIFERENTE A	Negación de la coincidencia de patrón simple
NO REGEXP	Negación de REGEXP
OCT()	Devuelve una cadena que contiene la representación octal de un número
OCTET_LENGTH ()	Sinónimo para LONGITUD ()
ORD ()	Código de carácter de retorno para el carácter más a la izquierda del argumento
POSICIÓN()	Sinónimo para LOCATE ()
CITAR()	Escape del argumento para usar en una declaración SQL
REGEXP	Coincidencia de patrones usando expresiones regulares
REPETIR()	Repite una cadena el número de veces especificado
REEMPLAZAR()	Reemplazar las ocurrencias de una cadena especificada
MARCHA ATRÁS()	Invertir los caracteres en una cadena

Nombre	Descripción
CORRECTO()	Devuelve el número de caracteres más a la derecha especificado
RLIKE	Sinónimo para REGEXP
RPAD ()	Añadir cadena el número especificado de veces
RTRIM ()	Eliminar espacios finales
SOUNDEX ()	Devuelve una cadena soundex
SUENA COMO	Comparar sonidos
ESPACIO()	Devuelve una cadena del número especificado de espacios
STRCMP ()	Compara dos cuerdas
SUBSTR ()	Devuelve la subcadena como se especifica
SUBSTRING ()	Devuelve la subcadena como se especifica
SUBSTRING_INDEX ()	Devuelve una subcadena de una cadena antes del número especificado de apariciones del delimitador
TO_BASE64 ()	Devuelve el argumento convertido a una cadena base-64
RECORTAR()	Eliminar espacios iniciales y finales
UCASE ()	Sinónimo para SUPERIOR ()
UNHEX ()	Devuelve una cadena que contiene la representación hexadecimal de un número.
SUPERIOR()	Convertir a mayúsculas
WEIGHT_STRING ()	Devuelve la cadena de peso para una cadena

Examples

Encontrar elemento en la lista separada por comas

```
SELECT FIND_IN_SET('b','a,b,c');
```

Valor de retorno:

2

```
SELECT FIND_IN_SET('d','a,b,c');
```

Valor de retorno:

0

STR_TO_DATE - Convertir cadena a la fecha

Con una columna de uno de los tipos de cadena, denominada `my_date_field` con un valor como [la cadena] `07/25/2016` , la siguiente declaración demuestra el uso de la función `STR_TO_DATE` :

```
SELECT STR_TO_DATE(my_date_field, '%m/%d/%Y') FROM my_table;
```

Podría usar esta función como parte de la cláusula `WHERE` también.

LOWER () / LCASE ()

Convertir en minúsculas el argumento de cadena

Sintaxis: LOWER (str)

```
LOWER('fOoBar') -- 'foobar'
LCASE('fOoBar') -- 'foobar'
```

REEMPLAZAR()

Convertir en minúsculas el argumento de cadena

Sintaxis: REPLACE (str, from_str, to_str)

```
REPLACE('foobarbaz', 'bar', 'BAR') -- 'fooBARbaz'
REPLACE('foobarbaz', 'zzz', 'ZZZ') -- 'foobarbaz'
```

SUBSTRING ()

SUBSTRING (o equivalente: SUBSTR) devuelve la subcadena a partir de la posición especificada y, opcionalmente, con la longitud especificada

Sintaxis: `SUBSTRING(str, start_position)`

```
SELECT SUBSTRING('foobarbaz', 4); -- 'barbaz'
SELECT SUBSTRING('foobarbaz' FROM 4); -- 'barbaz'

-- using negative indexing
SELECT SUBSTRING('foobarbaz', -6); -- 'barbaz'
SELECT SUBSTRING('foobarbaz' FROM -6); -- 'barbaz'
```

Sintaxis: `SUBSTRING(str, start_position, length)`

```
SELECT SUBSTRING('foobarbaz', 4, 3); -- 'bar'
SELECT SUBSTRING('foobarbaz', FROM 4 FOR 3); -- 'bar'
```

```
-- using negative indexing
SELECT SUBSTRING('foobarbaz', -6, 3); -- 'bar'
SELECT SUBSTRING('foobarbaz' FROM -6 FOR 3); -- 'bar'
```

SUPERIOR () / UCASE ()

Convertir en mayúsculas el argumento de cadena

Sintaxis: SUPERIOR (str)

```
UPPER('fOoBar') -- 'FOOBAR'
UCASE('fOoBar') -- 'FOOBAR'
```

LONGITUD()

Devuelve la longitud de la cadena en bytes. Dado que algunos caracteres pueden codificarse utilizando más de un byte, si desea la longitud en caracteres, consulte CHAR_LENGTH ()

Sintaxis: LONGITUD (str)

```
LENGTH('foobar') -- 6
LENGTH('fööbar') -- 8 -- contrast with CHAR_LENGTH(...) = 6
```

CHAR_LENGTH ()

Devuelve el número de caracteres en la cadena

Sintaxis: CHAR_LENGTH (str)

```
CHAR_LENGTH('foobar') -- 6
CHAR_LENGTH('fööbar') -- 6 -- contrast with LENGTH(...) = 8
```

HEX (str)

Convertir el argumento a hexadecimal. Esto se usa para cuerdas.

```
HEX('fööbar') -- 66F6F6626172 -- in "CHARACTER SET latin1" because "F6" is hex for ö
HEX('fööbar') -- 66C3B6C3B6626172  -- in "CHARACTER SET utf8 or utf8mb4" because "C3B6" is hex
for ö
```

Lea Operaciones de cuerdas en línea: https://riptutorial.com/es/mysql/topic/1399/operaciones-de-cuerdas

Capítulo 50: Operaciones de fecha y hora

Examples

Ahora()

```
Select Now();
```

Muestra la fecha y hora actual del servidor.

```
Update `footable` set mydatefield = Now();
```

Esto actualizará el campo `mydatefield` con la fecha y hora actuales del servidor en la zona horaria configurada del servidor, por ejemplo

```
'2016-07-21 12:00:00'
```

Aritmética de fecha

```
NOW() + INTERVAL 1 DAY  -- This time tomorrow

CURDATE() - INTERVAL 4 DAY -- Midnight 4 mornings ago
```

Muestre las preguntas de mysql almacenadas que se formularon de 3 a 10 horas (hace 180 a 600 minutos):

```
SELECT qId,askDate,minuteDiff
FROM
(   SELECT qId,askDate,
    TIMESTAMPDIFF(MINUTE,askDate,now()) as minuteDiff
    FROM questions_mysql
) xDerived
WHERE minuteDiff BETWEEN 180 AND 600
ORDER BY qId DESC
LIMIT 50;

+----------+---------------------+------------+
| qId      | askDate             | minuteDiff |
+----------+---------------------+------------+
| 38546828 | 2016-07-23 22:06:50 |        182 |
| 38546733 | 2016-07-23 21:53:26 |        195 |
| 38546707 | 2016-07-23 21:48:46 |        200 |
| 38546687 | 2016-07-23 21:45:26 |        203 |
| ...      |                     |            |
+----------+---------------------+------------+
```

Páginas de manual de MySQL para `TIMESTAMPDIFF()` .

Cuidado No intente usar expresiones como `CURDATE() + 1` para la aritmética de fechas en MySQL.

No devuelven lo que usted espera, especialmente si está acostumbrado al producto de base de datos Oracle. Use `CURDATE() + INTERVAL 1 DAY` lugar.

Pruebas contra un rango de fechas

Aunque es muy tentador usar `BETWEEN` ... `AND` ... para un intervalo de fechas, es problemático. En su lugar, este patrón evita la mayoría de los problemas:

```
WHERE x >= '2016-02-25'
  AND x  < '2016-02-25' + INTERVAL 5 DAY
```

Ventajas:

- `BETWEEN` es "inclusivo", incluida la fecha final o la segunda.
- `23:59:59` es torpe e incorrecto si tiene una resolución de microsegundos en un `DATETIME` .
- Este patrón evita tratar con años bisiestos y otros cálculos de datos.
- Funciona si `x` es `DATE` , `DATETIME` o `TIMESTAMP` .

SYSDATE (), NOW (), CURDATE ()

```
  SELECT SYSDATE();
```

Esta función devuelve la fecha y la hora actuales como un valor en `'YYYY-MM-DD HH:MM:SS'` o `YYYYMMDDHHMMSS` , dependiendo de si la función se usa en una cadena o contexto numérico. Devuelve la fecha y la hora en la zona horaria actual.

```
  SELECT NOW();
```

Esta función es un sinónimo de `SYSDATE()` .

```
  SELECT CURDATE();
```

Esta función devuelve la fecha actual, sin ningún tiempo, como un valor en `'YYYY-MM-DD'` o `YYYYMMDD` , dependiendo de si la función se usa en una cadena o contexto numérico. Devuelve la fecha en la zona horaria actual.

Extraer la fecha de la fecha dada o la expresión de fecha y hora

```
SELECT DATE('2003-12-31 01:02:03');
```

La salida será:

```
2003-12-31
```

Uso de un índice para una búsqueda de fecha y hora

Muchas tablas de bases de datos del mundo real tienen muchas filas con valores de columna `DATETIME` OR `TIMESTAMP` que abarcan mucho tiempo, incluso años o incluso décadas. A menudo es necesario usar una cláusula `WHERE` para recuperar un subconjunto de ese intervalo de tiempo. Por ejemplo, podríamos querer recuperar filas para la fecha 1 de septiembre de 2016 de una tabla.

Una forma ineficiente de hacerlo es esta:

```
WHERE DATE(x) = '2016-09-01'   /* slow! */
```

Es ineficiente porque aplica una función, `DATE()` , a los valores de una columna. Eso significa que MySQL debe examinar cada valor de `x` , y no se puede usar un índice.

Una mejor manera de hacer la operación es esta

```
WHERE x >= '2016-09-01'
  AND x <  '2016-09-01' + INTERVAL 1 DAY
```

Esto selecciona un rango de valores de `x` encuentran en cualquier lugar del día en cuestión, hasta pero *sin incluir* (por lo tanto, `<`) la medianoche del día siguiente.

Si la tabla tiene un índice en la columna `x` , entonces el servidor de la base de datos puede realizar una exploración de rango en el índice. Eso significa que puede encontrar rápidamente el primer valor relevante de x, y luego escanear el índice de forma secuencial hasta que encuentre el último valor relevante. Una exploración de rango de índice es mucho más eficiente que la exploración de tabla completa requerida por `DATE(x) = '2016-09-01` .

No se sienta tentado a usar esto, aunque parezca más eficiente.

```
 WHERE x BETWEEN '2016-09-01' AND '2016-09-01' + INTERVAL 1 DAY  /*  wrong! */
```

Tiene la misma eficiencia que el escaneo de rango, pero seleccionará filas con valores de `x` caen exactamente a la medianoche del 2 de septiembre de 2016, lo que no es lo que desea.

Lea Operaciones de fecha y hora en línea:
https://riptutorial.com/es/mysql/topic/1882/operaciones-de-fecha-y-hora

Capítulo 51: ORDEN POR

Examples

Contextos

Las cláusulas en un `SELECT` tienen un orden específico:

```
SELECT ... FROM ... WHERE ... GROUP BY ... HAVING ...
    ORDER BY ...  -- goes here
    LIMIT ... OFFSET ...;

( SELECT ... ) UNION ( SELECT ... ) ORDER BY ...  -- for ordering the result of the UNION.

SELECT ... GROUP_CONCAT(DISTINCT x ORDER BY ... SEPARATOR ...) ...

ALTER TABLE ... ORDER BY ... -- probably useful only for MyISAM; not for InnoDB
```

BASIC

ORDEN POR x

`x` puede ser cualquier tipo de datos.

- `NULLs` preceden a los que no son NULL.
- El valor predeterminado es `ASC` (de menor a mayor)
- Las cadenas (`VARCHAR` , etc.) se ordenan de acuerdo con la `COLLATION` de la declaración
- `ENUMs` son ordenados por la orden de declaración de sus cadenas.

Ascendiendo descendiendo

```
ORDER BY x ASC  -- same as default
ORDER BY x DESC  -- highest to lowest
ORDER BY lastname, firstname  -- typical name sorting; using two columns
ORDER BY submit_date DESC  -- latest first
ORDER BY submit_date DESC, id ASC  -- latest first, but fully specifying order.
```

- `ASC` = `ASCENDING` , `DESC` = `DESCENDING`
- `NULLs` son los primeros incluso para `DESC` .
- En los ejemplos anteriores, `INDEX(x)` , `INDEX(lastname, firstname)` , `INDEX(submit_date)` pueden mejorar significativamente el rendimiento.

Pero ... Mezclar `ASC` y `DESC` , como en el ejemplo anterior, no puede usar un índice compuesto para beneficiarse. Tampoco lo ayudará `INDEX(submit_date DESC, id ASC)` - " `DESC` " se reconoce sintácticamente en la declaración `INDEX` , pero se ignora.

Algunos trucos

```
ORDER BY FIND_IN_SET(card_type, "MASTER-CARD,VISA,DISCOVER") -- sort 'MASTER-CARD' first.
ORDER BY x IS NULL, x  -- order by `x`, but put `NULLs` last.
```

Pedidos personalizados

```
SELECT * FROM some_table WHERE id IN (118, 17, 113, 23, 72)
ORDER BY FIELD(id, 118, 17, 113, 23, 72);
```

Devuelve el resultado en el orden especificado de ID.

carné de identidad	...
118	...
17	...
113	...
23	...
72	...

Es útil si los identificadores ya están ordenados y solo necesita recuperar las filas.

Lea ORDEN POR en línea: https://riptutorial.com/es/mysql/topic/5469/orden-por

Capítulo 52: Palabras reservadas

Introducción

MySQL tiene algunos nombres especiales llamados *palabras reservadas* . Una palabra reservada se puede usar como un identificador para una tabla, columna, etc. solo si está envuelta en backticks (`), de lo contrario dará lugar a un error.

Para evitar tales errores, no use palabras reservadas como identificadores o envuelva el identificador ofensivo en backticks.

Observaciones

A continuación se enumeran todas las palabras reservadas (de la documentación oficial):

- ACCESIBLE
- AÑADIR
- TODOS
- ALTERAR
- ANALIZAR
- Y
- COMO
- ASC
- Asensivo
- ANTES DE
- ENTRE
- BIGINT
- BINARIO
- GOTA
- AMBOS
- POR
- LLAMADA
- CASCADA
- CASO
- CAMBIO
- CARBONIZARSE
- PERSONAJE
- COMPROBAR
- COTEJAR
- COLUMNA
- CONDICIÓN
- RESTRICCIÓN
- CONTINUAR
- CONVERTIR
- CREAR

- CRUZAR
- FECHA ACTUAL
- TIEMPO ACTUAL
- FECHA Y HORA ACTUAL
- USUARIO ACTUAL
- CURSOR
- BASE DE DATOS
- BASES DE DATOS
- DAY_HOUR
- DAY_MICROSECOND
- DAY_MINUTE
- DAY_SECOND
- DIC
- DECIMAL
- DECLARAR
- DEFECTO
- RETRASADO
- BORRAR
- DESC
- DESCRIBIR
- Determinante
- DISTINTO
- DISTINTO
- Div
- DOBLE
- SOLTAR
- DOBLE
- CADA
- MÁS
- Elseif
- ADJUNTO
- ESCAPADO
- EXISTE
- SALIDA
- EXPLIQUE
- FALSO
- HA PODIDO RECUPERAR
- FLOTADOR
- FLOTADOR 4
- FLOTA8
- PARA
- FUERZA
- EXTERIOR
- DESDE
- TEXTO COMPLETO
- GENERADO

- OBTENER
- CONCEDER
- GRUPO
- TENIENDO
- ALTA PRIORIDAD
- HOUR_MICROSECOND
- HOUR_MINUTE
- HOUR_SECOND
- SI
- IGNORAR
- EN
- ÍNDICE
- EN ARCHIVO
- INTERIOR
- EN FUERA
- INSENSIBLE
- INSERTAR
- EN T
- INT1
- INT2
- INT3
- INT4
- INT8
- ENTERO
- INTERVALO
- DENTRO
- IO_AFTER_GTIDS
- IO_BEFORE_GTIDS
- ES
- ITERAR
- UNIRSE
- LLAVE
- LLAVES
- MATAR
- LÍDER
- SALIR
- IZQUIERDA
- ME GUSTA
- LÍMITE
- LINEAL
- LÍNEAS
- CARGA
- HORA LOCAL
- LOCALTIMESTAMP
- BLOQUEAR
- LARGO

- LONGBLOB
- TEXTO LARGO
- LAZO
- BAJA PRIORIDAD
- MASTER_BIND
- MASTER_SSL_VERIFY_SERVER_CERT
- PARTIDO
- VALOR MÁXIMO
- MEDIUMBLOB
- MEDIUMINT
- MEDIUMTEXTO
- MIDDLEINT
- MINUTE_MICROSECOND
- MINUTE_SECOND
- MOD
- Modifica
- NATURAL
- NO
- NO_WRITE_TO_BINLOG
- NULO
- NUMÉRICO
- EN
- OPTIMIZAR
- OPTIMIZER_COSTS
- OPCIÓN
- Opcionalmente
- O
- ORDEN
- AFUERA
- EXTERIOR
- PERFIL
- DIVIDIR
- PRECISIÓN
- PRIMARIO
- PROCEDIMIENTO
- PURGA
- DISTANCIA
- LEER
- Leer
- LEER ESCRIBIR
- REAL
- Referencias
- REGEXP
- LANZAMIENTO
- REBAUTIZAR
- REPETIR

- REEMPLAZAR
- EXIGIR
- RESIGNAL
- RESTRINGIR
- REGRESO
- REVOCAR
- CORRECTO
- RLIKE
- ESQUEMA
- Esquemas
- SECOND_MICROSECOND
- SELECCIONAR
- SENSIBLE
- SEPARADOR
- CONJUNTO
- ESPECTÁCULO
- SEÑAL
- Pequeño
- ESPACIAL
- ESPECÍFICO
- SQL
- SQLEXCEPTION
- SQLSTATE
- SQLWARNING
- SQL_BIG_RESULT
- SQL_CALC_FOUND_ROWS
- SQL_SMALL_RESULT
- SSL
- COMENZANDO
- ALMACENADO
- STRAIGHT_JOIN
- MESA
- TERMINADO
- ENTONCES
- TINYBLOB
- TINYINT
- TINTEXTO
- A
- TRAILING
- DESENCADENAR
- CIERTO
- DESHACER
- UNIÓN
- ÚNICO
- DESBLOQUEAR
- NO FIRMADO

- ACTUALIZAR
- USO
- UTILIZAR
- UTILIZANDO
- UTC_DATE
- UTC_TIME
- UTC_TIMESTAMP
- VALORES
- VARBINARIO
- VARCHAR
- VARCARADOR
- VARIAR
- VIRTUAL
- CUANDO
- DÓNDE
- MIENTRAS
- CON
- ESCRIBIR
- XOR
- AÑO MES
- Rellenar
- GENERADO
- OPTIMIZER_COSTS
- ALMACENADO
- VIRTUAL

Examples

Errores debidos a palabras reservadas

Al intentar seleccionar de una tabla llamada `order` como esta

```
select * from order
```

el error sube:

> Código de error: 1064. Usted tiene un error en su sintaxis SQL; verifique el manual que corresponde a la versión de su servidor MySQL para conocer la sintaxis correcta para usar cerca de 'orden' en la línea 1

Las palabras clave reservadas en MySQL deben escaparse con comillas invertidas (`)

```
select * from `order`
```

para distinguir entre una palabra clave y un nombre de tabla o columna.

Vea también: Error de sintaxis debido al uso de una palabra reservada como nombre de tabla o

columna en MySQL .

Lea Palabras reservadas en línea: https://riptutorial.com/es/mysql/topic/1398/palabras-reservadas

Capítulo 53: Particionamiento

Observaciones

- **RANGO de particionamiento** . Este tipo de partición asigna filas a particiones basadas en valores de columna que se encuentran dentro de un rango determinado.
- **Partición de la lista** . Similar a la partición por RANGO, excepto que la partición se selecciona en base a columnas que coinciden con uno de un conjunto de valores discretos.
- **Particionamiento HASH** . Con este tipo de partición, se selecciona una partición en función del valor devuelto por una expresión definida por el usuario que opera en valores de columna en filas para insertarse en la tabla. La función puede consistir en cualquier expresión válida en MySQL que produzca un valor entero no negativo. Una extensión de este tipo, `LINEAR HASH` , también está disponible.
- **Partición clave** . Este tipo de partición es similar a la partición por HASH, excepto que solo se suministran una o más columnas para evaluar, y el servidor MySQL proporciona su propia función de hashing. Estas columnas pueden contener valores distintos a los enteros, ya que la función de hashing proporcionada por MySQL garantiza un resultado entero independientemente del tipo de datos de la columna. Una extensión de este tipo, `LINEAR KEY` , también está disponible.

Examples

RANGO de particionamiento

Una tabla que está particionada por rango se particiona de tal manera que cada partición contiene filas para las cuales el valor de la expresión de partición se encuentra dentro de un rango dado. Los rangos deben ser contiguos, pero no superpuestos, y se definen utilizando el operador `VALUES LESS THAN` operador. Para los siguientes ejemplos, suponga que está creando una tabla como la siguiente para almacenar registros de personal para una cadena de 20 tiendas de video, numeradas del 1 al 20:

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT NOT NULL,
    store_id INT NOT NULL
);
```

Esta tabla se puede dividir por rango de varias maneras, según sus necesidades. Una forma sería usar la columna `store_id` . Por ejemplo, puede decidir dividir la tabla en 4 formas agregando una cláusula `PARTITION BY RANGE` como se muestra aquí:

```
ALTER TABLE employees PARTITION BY RANGE (store_id) (
    PARTITION p0 VALUES LESS THAN (6),
    PARTITION p1 VALUES LESS THAN (11),
    PARTITION p2 VALUES LESS THAN (16),
    PARTITION p3 VALUES LESS THAN MAXVALUE
);
```

`MAXVALUE` representa un valor entero que siempre es mayor que el mayor valor entero posible (en lenguaje matemático, sirve como un límite mínimo superior).

Basado en el documento oficial de MySQL .

Partición de la lista

La partición de listas es similar a la partición de rango de muchas maneras. Como en la partición por RANGO, cada partición debe definirse explícitamente. La principal diferencia entre los dos tipos de partición es que, en la partición de lista, cada partición se define y selecciona en función de la pertenencia de un valor de columna en una de un conjunto de listas de valores, en lugar de en uno de un conjunto de rangos contiguos de valores. Esto se hace usando `PARTITION BY LIST(expr)` donde `expr` es un valor de columna o una expresión basada en un valor de columna y devolviendo un valor entero, y luego definiendo cada partición por medio de `VALUES IN (value_list)` , donde `value_list` es un Lista de enteros separados por comas.

Para los ejemplos que siguen, asumimos que la definición básica de la tabla a particionar es proporcionada por la `CREATE TABLE` que se muestra aquí:

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
);
```

Supongamos que hay 20 tiendas de video distribuidas entre 4 franquicias, como se muestra en la siguiente tabla.

Región	Números de identificación de la tienda
norte	3, 5, 6, 9, 17
Este	1, 2, 10, 11, 19, 20
Oeste	4, 12, 13, 14, 18
Central	7, 8, 15, 16

Para particionar esta tabla de tal manera que las filas de las tiendas que pertenecen a la misma

región se almacenen en la misma partición

```
ALTER TABLE employees PARTITION BY LIST(store_id) (
    PARTITION pNorth VALUES IN (3,5,6,9,17),
    PARTITION pEast VALUES IN (1,2,10,11,19,20),
    PARTITION pWest VALUES IN (4,12,13,14,18),
    PARTITION pCentral VALUES IN (7,8,15,16)
);
```

Basado en el documento oficial de MySQL .

Particionamiento HASH

La partición mediante HASH se utiliza principalmente para garantizar una distribución uniforme de los datos entre un número predeterminado de particiones. Con el rango o la partición de la lista, debe especificar explícitamente en qué partición se almacenará un determinado valor de columna o un conjunto de valores de columna; Con la partición hash, MySQL se encarga de esto, y solo necesita especificar un valor de columna o una expresión basada en un valor de columna que se va a hash y el número de particiones en las que se dividirá la tabla particionada.

La siguiente declaración crea una tabla que usa hashing en la columna store_id y se divide en 4 particiones:

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
)
PARTITION BY HASH(store_id)
PARTITIONS 4;
```

Si no incluye una cláusula de `PARTITIONS` , el número de particiones por defecto es 1.

Basado en el documento oficial de MySQL .

Lea Particionamiento en línea: https://riptutorial.com/es/mysql/topic/5128/particionamiento

Capítulo 54: Personalizar PS1

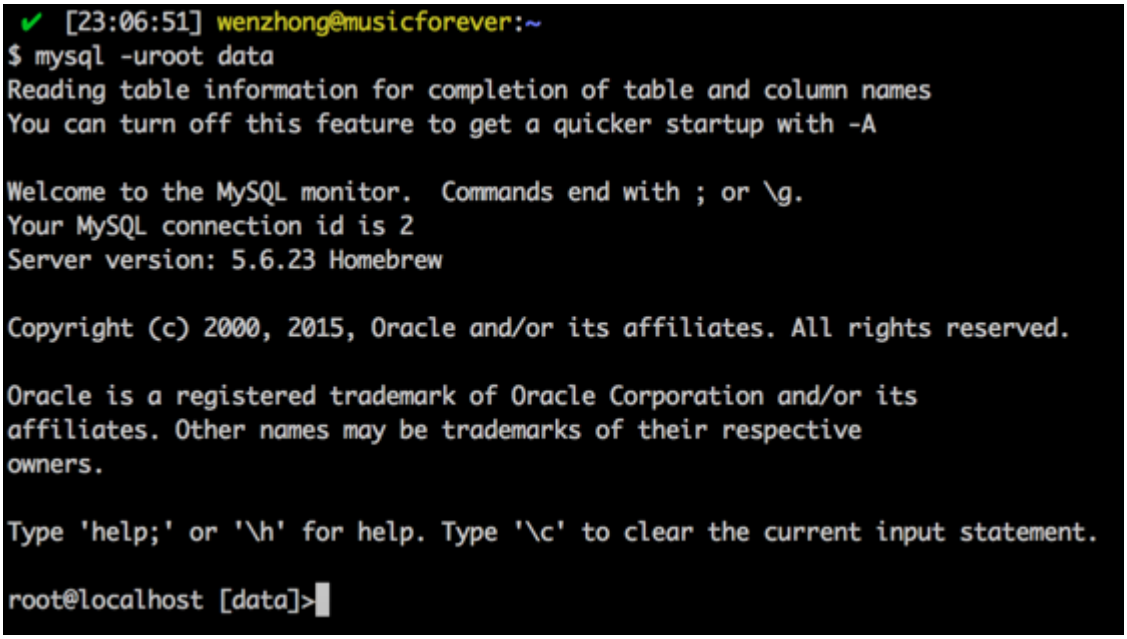
Examples

Personaliza el MySQL PS1 con la base de datos actual.

En el .bashrc o .bash_profile, agregando:

```
export MYSQL_PS1="\u@\h [\d]>"
```

haga que el cliente MySQL PROMPT muestre al usuario actual @ host [base de datos].



PS1 personalizado a través del archivo de configuración de MySQL

En `mysqld.cnf` o equivalente:

```
[mysql]
prompt = '\u@\h [\d]> '
```

Esto logra un efecto similar, sin tener que lidiar con `.bashrc` 's.

Lea Personalizar PS1 en línea: https://riptutorial.com/es/mysql/topic/5795/personalizar-ps1

Capítulo 55: Preparar declaraciones

Sintaxis

- PREPARAR stmt_name FROM preparable_stmt
- EJECUTAR stmt_name [USING @var_name [, @var_name] ...]
- {DEALLOCATE | DROP} PREPARAR stmt_name

Examples

PREPARAR, EJECUTAR y DESALARCAR las declaraciones de PREPARACIÓN

PREPARAR prepara una declaración para su ejecución.

EJECUTAR ejecuta una sentencia preparada.

DEALLOCATE PREPARE lanza una declaración preparada

```
SET @s = 'SELECT SQRT(POW(?,2) + POW(?,2)) AS hypotenuse';
PREPARE stmt2 FROM @s;
SET @a = 6;
SET @b = 8;
EXECUTE stmt2 USING @a, @b;
```

Resultado:

```
+------------+
| hypotenuse |
+------------+
|         10 |
+------------+
```

Finalmente,

```
DEALLOCATE PREPARE stmt2;
```

Notas:

- Debe usar @variables, no variables DECLARADAS para `FROM @s`
- Un uso principal para Preparar, etc., es "construir" una consulta para situaciones donde el enlace no funcionará, como insertar el nombre de la tabla.

Construir y ejecutar

(Esta es una solicitud de un buen ejemplo que muestra cómo *construir* un `SELECT` usando `CONCAT` , luego prepararlo y ejecutarlo. Por favor, enfatice el uso de las variables @variables contra las

variables DECLAREd: hace una gran diferencia, y es algo que los novatos (incluyéndome) tropezando.)

Alterar tabla con añadir columna

```
SET v_column_definition := CONCAT(
  v_column_name
  ,' ',v_column_type
  ,' ',v_column_options
);

SET @stmt := CONCAT('ALTER TABLE ADD COLUMN ', v_column_definition);

PREPARE stmt FROM @stmt;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
```

Lea Preparar declaraciones en línea: https://riptutorial.com/es/mysql/topic/2603/preparar-declaraciones

Capítulo 56: Recuperar de la contraseña de root perdida

Examples

Establecer contraseña de root, habilitar usuario root para socket y acceso http

Resuelve el problema de: acceso denegado para la raíz del usuario con la contraseña SÍ Detener mySQL:

```
sudo systemctl stop mysql
```

Reinicie mySQL, saltándose tablas de permisos:

```
sudo mysqld_safe --skip-grant-tables
```

Iniciar sesión:

```
mysql -u root
```

En el shell SQL, mira si existen usuarios:

```
select User, password,plugin FROM mysql.user ;
```

Actualizar los usuarios (el complemento nulo se habilita para todos los complementos):

```
update mysql.user set password=PASSWORD('mypassword'), plugin = NULL WHERE User = 'root';
exit;
```

En el shell de Unix, detenga mySQL sin otorgar tablas, luego reinicie con las tablas de otorgamiento:

```
sudo service mysql stop
sudo service mysql start
```

Lea Recuperar de la contraseña de root perdida en línea: https://riptutorial.com/es/mysql/topic/9973/recuperar-de-la-contrasena-de-root-perdida

Capítulo 57: Recuperar y restablecer la contraseña de root predeterminada para MySQL 5.7+

Introducción

Después de MySQL 5.7, cuando instalamos MySQL a veces no necesitamos crear una cuenta de root o dar una contraseña de root. De forma predeterminada, cuando iniciamos el servidor, la contraseña predeterminada se almacena en el archivo `mysqld.log` . Necesitamos iniciar sesión en el sistema usando esa contraseña y necesitamos cambiarla.

Observaciones

Recuperar y restablecer la contraseña de root predeterminada usando este método solo es aplicable para MySQL 5.7+

Examples

¿Qué sucede cuando se inicia el servidor por primera vez?

Dado que el directorio de datos del servidor está vacío:

- El servidor está inicializado.
- El certificado SSL y los archivos de claves se generan en el directorio de datos.
- El complemento validate_password está instalado y habilitado.
- Se crea la cuenta de superusuario 'root' @ 'localhost'. La contraseña para el superusuario se establece y almacena en el archivo de registro de errores.

Cómo cambiar la contraseña de root usando la contraseña predeterminada

Para revelar la contraseña "root" predeterminada:

```
shell> sudo grep 'temporary password' /var/log/mysqld.log
```

Cambie la contraseña de root lo antes posible iniciando sesión con la contraseña temporal generada y establezca una contraseña personalizada para la cuenta de superusuario:

```
shell> mysql -uroot -p

mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'MyNewPass5!';
```

Nota: el complemento validate_password de MySQL se instala de forma predeterminada. Esto requerirá que las contraseñas contengan al menos una letra mayúscula, una letra minúscula, un

dígito y un carácter especial, y que la longitud total de la contraseña sea de al menos 8 caracteres.

restablecer la contraseña de root cuando "/ var / run / mysqld 'para el archivo socket UNIX no existe"

Si olvido la contraseña, obtendré un error.

```
$ mysql -u root -p
```

Introducir la contraseña:

ERROR 1045 (28000): Acceso denegado para el usuario 'root' @ 'localhost' (usando la contraseña: YES)

Intenté resolver el problema conociendo primero el estado:

```
$ systemctl status mysql.service
```

mysql.service - Servidor de comunidad MySQL cargado: cargado (/lib/systemd/system/mysql.service; habilitado; proveedor preestablecido: en Activo: activo (en ejecución) desde Jue 2017-06-08 14:31:33 IST; 38s ago

Luego utilicé el código `mysqld_safe --skip-grant-tables &` pero `mysqld_safe --skip-grant-tables &` el error:

> mysqld_safe El directorio '/ var / run / mysqld' para el archivo de socket UNIX no existe.

```
$ systemctl stop  mysql.service
$ ps -eaf|grep mysql
$ mysqld_safe --skip-grant-tables &
```

> Lo resolví:

```
$ mkdir -p /var/run/mysqld
$ chown mysql:mysql /var/run/mysqld
```

Ahora uso el mismo código `mysqld_safe --skip-grant-tables &` y obtengo

> mysqld_safe Iniciar el demonio mysqld con bases de datos desde / var / lib / mysql

Si uso `$ mysql -u root` obtendré:

Versión del servidor: 5.7.18-0ubuntu0.16.04.1 (Ubuntu)

ser marcas registradas de sus respectivos dueños.

Escriba 'ayuda'; o '\ h' para ayuda. Escriba '\ c' para borrar la declaración de entrada actual.

mysql>

Ahora es el momento de cambiar la contraseña:

```
mysql> use mysql
mysql> describe user;
```

Lectura de la información de la tabla para completar los nombres de tablas y columnas. Puede desactivar esta función para obtener un inicio más rápido con -A

Base de datos cambiada

```
mysql> FLUSH PRIVILEGES;
mysql> SET PASSWORD FOR root@'localhost' = PASSWORD('newpwd');
```

o Si tiene una cuenta raíz de mysql que puede conectarse desde cualquier lugar, también debe hacer:

```
UPDATE mysql.user SET Password=PASSWORD('newpwd') WHERE User='root';
```

Metodo alternativo:

```
   USE mysql
   UPDATE user SET Password = PASSWORD('newpwd')
   WHERE Host = 'localhost' AND User = 'root';
```

Y si tiene una cuenta de root que puede acceder desde cualquier lugar:

```
 USE mysql
 UPDATE user SET Password = PASSWORD('newpwd')
 WHERE Host = '%' AND User = 'root';`enter code here
```

Ahora necesito `quit` de mysql y detener / iniciar

```
FLUSH PRIVILEGES;
sudo /etc/init.d/mysql stop
sudo /etc/init.d/mysql start
```

ahora de nuevo `mysql -u root -p 'y use la nueva contraseña para obtener

mysql>

Lea Recuperar y restablecer la contraseña de root predeterminada para MySQL 5.7+ en línea: https://riptutorial.com/es/mysql/topic/9563/recuperar-y-restablecer-la-contrasena-de-root-predeterminada-para-mysql-5-7plus

Capítulo 58: Replicación

Observaciones

La replicación se utiliza para copiar datos de [Copia de seguridad] de un servidor de base de datos MySQL a uno o más servidores de base de datos MySQL.

Maestro : el servidor de base de datos MySQL, que sirve para copiar los datos.

Esclavo : el servidor de la base de datos MySQL, copia los datos que sirve Master.

Con MySQL, la replicación es asíncrona por defecto. Esto significa que los esclavos no necesitan estar conectados permanentemente para recibir actualizaciones del maestro. Por ejemplo, si su esclavo está apagado o no está conectado con el maestro y usted lo está conectando o se conecta con el maestro más tarde, entonces se sincronizará automáticamente con el maestro.

Dependiendo de la configuración, puede replicar todas las bases de datos, bases de datos seleccionadas o incluso tablas seleccionadas dentro de una base de datos.

Formatos de replicación

Hay dos tipos principales de formatos de replicación

Replicación basada en instrucciones *(SBR)* , que replica declaraciones SQL completas. En esto, el maestro escribe sentencias de SQL en el registro binario. La replicación del maestro al esclavo funciona ejecutando las sentencias de SQL en el esclavo.

Replicación basada en filas (RBR) , que replica solo las filas modificadas. En esto, el maestro escribe eventos en el registro binario que indican cómo se cambian las filas de la tabla individual. La replicación del maestro al esclavo funciona copiando los eventos que representan los cambios en las filas de la tabla al esclavo.

También puede utilizar una tercera variedad, la *replicación basada mixta (MBR)* . En esto, se utiliza tanto el registro basado en sentencias como el basado en filas. El registro se creará según cuál sea el más apropiado para el cambio.

El formato basado en declaraciones fue el predeterminado en las versiones de MySQL anteriores a 5.7.7. En MySQL 5.7.7 y versiones posteriores, el formato basado en filas es el predeterminado.

Examples

Maestro - Configuración de replicación de esclavos

Considere 2 servidores MySQL para la configuración de la replicación, uno es un maestro y el otro es un esclavo.

Vamos a configurar el Maestro para que mantenga un registro de cada acción realizada en él.

Vamos a configurar el servidor esclavo para que mire el registro en el maestro y cada vez que ocurran cambios en el registro en el maestro, debe hacer lo mismo.

Configuración maestra

En primer lugar, necesitamos crear un usuario en el Master. Este usuario será utilizado por Slave para crear una conexión con el Maestro.

```
CREATE USER 'user_name'@'%' IDENTIFIED BY 'user_password';
GRANT REPLICATION SLAVE ON *.* TO 'user_name'@'%';
FLUSH PRIVILEGES;
```

Cambio `user_name` y `user_password` acuerdo con su nombre de usuario y contraseña.

Ahora el `my.inf` (my.cnf en Linux) debe ser editado. Incluya las siguientes líneas en la sección [mysqld].

```
server-id = 1
log-bin = mysql-bin.log
binlog-do-db = your_database
```

La primera línea se utiliza para asignar una ID a este servidor MySQL.

La segunda línea le dice a MySQL que comience a escribir un registro en el archivo de registro especificado. En Linux, esto puede configurarse como `log-bin = /home/mysql/logs/mysql-bin.log` . Si está iniciando la replicación en un servidor MySQL en el que ya se ha utilizado la replicación, asegúrese de que este directorio esté vacío de todos los registros de replicación.

La tercera línea se utiliza para configurar la base de datos para la que vamos a escribir el registro. Debe reemplazar `your_database` con su nombre de base de datos.

Asegúrese de que `skip-networking` no haya sido habilitado y reinicie el servidor MySQL (Master)

Configuración de esclavo

`my.inf` archivo `my.inf` debe editarse en Slave. Incluya las siguientes líneas en la sección [mysqld].

```
server-id = 2
master-host = master_ip_address
master-connect-retry = 60

master-user = user_name
master-password = user_password
replicate-do-db = your_database

relay-log = slave-relay.log
relay-log-index = slave-relay-log.index
```

La primera línea se utiliza para asignar una ID a este servidor MySQL. Esta identificación debe ser única.

La segunda línea es la dirección IP del servidor maestro. Cambia esto de acuerdo a tu sistema

maestro IP

La tercera línea se utiliza para establecer un límite de reintento en segundos.

Las siguientes dos líneas indican el nombre de usuario y la contraseña al esclavo, mediante el cual se conecta al maestro.

La siguiente línea establece la base de datos que necesita para replicar.

Las dos últimas líneas utilizadas para asignar `relay-log` nombres de los archivos de `relay-log` y `relay-log-index` .

Asegúrese de que `skip-networking` no se haya habilitado y reinicie el servidor MySQL (esclavo)

Copiar datos a esclavo

Si constantemente se agregan datos al Maestro, tendremos que evitar todos los accesos a la base de datos en el Maestro para que no se pueda agregar nada. Esto se puede lograr ejecutando la siguiente declaración en Master.

```
FLUSH TABLES WITH READ LOCK;
```

Si no se agregan datos al servidor, puede omitir el paso anterior.

Vamos a realizar una copia de seguridad de los datos del Maestro usando `mysqldump`

```
mysqldump your_database –u root –p > D://Backup/backup.sql;
```

Cambie `your_database` directorio de base de `your_database` y copia de seguridad según su configuración. Ahora tendrás un archivo llamado `backup.sql` en la ubicación dada.

Si su base de datos no existe en su Esclavo, cree eso ejecutando lo siguiente

```
CREATE DATABASE `your_database`;
```

Ahora tenemos que importar la copia de seguridad en el servidor MySQL esclavo.

```
mysql –u root –p your_database  <D://Backup/backup.sql
--->Change `your_database` and backup directory according to your setup
```

Iniciar la replicación

Para iniciar la replicación, debemos encontrar el nombre del archivo de registro y la posición del registro en el Maestro. Entonces, ejecuta lo siguiente en Master

```
SHOW MASTER STATUS;
```

Esto le dará una salida como abajo

```
+--------------------+----------+------------------------------+------------------+
| File               | Position | Binlog_Do_DB                 | Binlog_Ignore_DB |
+--------------------+----------+------------------------------+------------------+
| mysql-bin.000001   | 130      | your_database                |                  |
+--------------------+----------+------------------------------+------------------+
```

A continuación, ejecute el siguiente en esclavo

```
SLAVE STOP;
CHANGE MASTER TO MASTER_HOST='master_ip_address', MASTER_USER='user_name',
   MASTER_PASSWORD='user_password', MASTER_LOG_FILE='mysql-bin.000001', MASTER_LOG_POS=130;
SLAVE START;
```

Primero paramos al esclavo. Luego le decimos exactamente dónde buscar en el archivo de registro maestro. Para el nombre de `MASTER_LOG_FILE` y `MASTER_LOG_POS` , use los valores que obtuvimos al ejecutar el comando `SHOW MASTER STATUS` en el Maestro.

Debe cambiar la IP del maestro en `MASTER_HOST` , y cambiar el usuario y la contraseña en consecuencia.

El esclavo ahora estará esperando. El estado del esclavo se puede ver ejecutando lo siguiente

```
SHOW SLAVE STATUS;
```

Si anteriormente ejecutó `FLUSH TABLES WITH READ LOCK` en Master, libere las tablas del bloqueo ejecutando lo siguiente

```
UNLOCK TABLES;
```

Ahora el Maestro mantiene un registro para cada acción que se realiza en él y el servidor Esclavo mira el registro en el Maestro. Cada vez que se producen cambios en el inicio de sesión en el maestro, Slave lo replica.

Errores de replicación

Cuando se produce un error al ejecutar una consulta en el esclavo, MySQL detiene la replicación automáticamente para identificar el problema y solucionarlo. Esto se debe principalmente a que un evento causó una clave duplicada o no se encontró una fila y no se puede actualizar o eliminar. Puede omitir tales errores, incluso si no se recomienda

Para omitir una sola consulta que cuelga el esclavo, use la siguiente sintaxis

```
SET GLOBAL sql_slave_skip_counter = N;
```

Esta declaración omite los siguientes N eventos del maestro. Esta declaración es válida solo cuando los subprocesos esclavos no se están ejecutando. De lo contrario, se produce un error.

```
STOP SLAVE;
SET GLOBAL sql_slave_skip_counter=1;
```

```
START SLAVE;
```

En algunos casos esto está bien. Pero si la declaración es parte de una transacción de múltiples declaraciones, se vuelve más compleja, porque omitir la declaración que produce el error hará que se omita toda la transacción.

Si desea omitir más consultas que producen el mismo código de error y está seguro de que omitir esos errores no hará que su esclavo sea inconsistente y desea omitirlos todos, agregará una línea para omitir ese código de error en su `my.cnf` .

Por ejemplo, es posible que desee omitir todos los errores duplicados que pueda estar recibiendo

```
1062 | Error 'Duplicate entry 'xyz' for key 1' on query
```

Luego agrega lo siguiente a tu `my.cnf`

```
slave-skip-errors = 1062
```

También puede omitir otro tipo de errores o todos los códigos de error, pero asegúrese de que omitir esos errores no hará que su esclavo sea inconsistente. Los siguientes son la sintaxis y ejemplos.

```
slave-skip-errors=[err_code1,err_code2,...|all]

slave-skip-errors=1062,1053
slave-skip-errors=all
slave-skip-errors=ddl_exist_errors
```

Lea Replicación en línea: https://riptutorial.com/es/mysql/topic/7218/replicacion

Capítulo 59: Rutinas almacenadas (procedimientos y funciones)

Parámetros

Parámetro	Detalles
DEVOLUCIONES	Especifica el tipo de datos que se pueden devolver desde una función.
REGRESO	La variable real o el valor que sigue a la sintaxis de `RETURN` es lo que se devuelve a donde se llamó la función.

Observaciones

Una rutina almacenada es un procedimiento o una función.

Se invoca un procedimiento utilizando una instrucción CALL y solo puede devolver valores utilizando variables de salida.

Se puede llamar a una función desde dentro de una declaración como cualquier otra función y puede devolver un valor escalar.

Examples

Crear una función

La siguiente función de ejemplo (trivial) simplemente devuelve el valor constante de `INT 12` .

```
DELIMITER ||
CREATE FUNCTION functionname()
RETURNS INT
BEGIN
    RETURN 12;
END;
||
DELIMITER ;
```

La primera línea define a qué se `DELIMITER ||` carácter delimitador (`DELIMITER ||`), esto se debe configurar antes de que se cree una función, de lo contrario, si se deja en su valor predeterminado `;` entonces el primero `;` que se encuentra en el cuerpo de la función se tomará como el final de la `CREATE` , que generalmente no es lo que se desea.

Una vez que se haya ejecutado `CREATE FUNCTION` , debe configurar el delimitador a su valor predeterminado de `;` como se ve después del código de función en el ejemplo anterior (`DELIMITER ;`).

La ejecución de esta función es la siguiente:

```
SELECT functionname();
+----------------+
| functionname() |
+----------------+
|             12 |
+----------------+
```

Un ejemplo ligeramente más complejo (pero aún trivial) toma un parámetro y le agrega una constante:

```
DELIMITER $$
CREATE FUNCTION add_2 ( my_arg INT )
  RETURNS INT
BEGIN
  RETURN (my_arg + 2);
END;
$$
DELIMITER ;

SELECT add_2(12);
+-----------+
| add_2(12) |
+-----------+
|        14 |
+-----------+
```

Tenga en cuenta el uso de un argumento diferente a la directiva `DELIMITER` . En realidad, puede usar cualquier secuencia de caracteres que no aparezca en el cuerpo de la sentencia `CREATE` , pero la práctica habitual es usar un carácter no alfanumérico duplicado, como `\\` , `||` o `$$` .

Es una buena práctica cambiar siempre el parámetro antes y después de la creación, actualización o activación de una función, procedimiento o disparador, ya que algunas GUI no requieren que el delimitador cambie, mientras que las consultas en ejecución a través de la línea de comandos siempre requieren que se establezca el delimitador.

Crear procedimiento con una preparación construida

```
DROP PROCEDURE if exists displayNext100WithName;
DELIMITER $$
CREATE PROCEDURE displayNext100WithName
(    nStart int,
    tblName varchar(100)
)
BEGIN
    DECLARE thesql varchar(500); -- holds the constructed sql string to execute

    -- expands the sizing of the output buffer to accomodate the output (Max value is at least
4GB)
    SET session group_concat_max_len = 4096; -- prevents group_concat from barfing with error
1160 or whatever it is

    SET @thesql=CONCAT("select group_concat(qid order by qid SEPARATOR '%3B') as nums ","from
(    select qid from ");
```

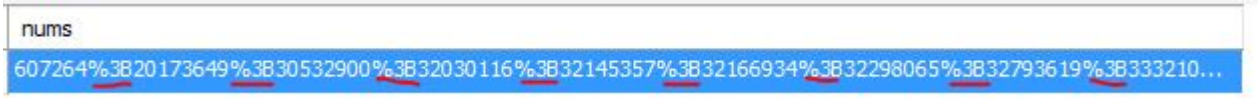
```
    SET @thesql=CONCAT(@thesql,tblName," where qid>? order by qid limit 100 )xDerived");
    PREPARE stmt1 FROM @thesql; -- create a statement object from the construct sql string to
execute
    SET @p1 = nStart; -- transfers parameter passed into a User Variable compatible with the
below EXECUTE
    EXECUTE stmt1 USING @p1;

    DEALLOCATE PREPARE stmt1; -- deallocate the statement object when finished
END$$
DELIMITER ;
```

La creación del procedimiento almacenado muestra el ajuste con un DELIMITER necesario en muchas herramientas del cliente.

Ejemplo de llamada:

```
call displayNext100WithName(1,"questions_mysql");
```

Salida de muestra con separador `%3B` (punto y coma):



Procedimiento almacenado con parámetros IN, OUT, INOUT.

```
DELIMITER $$

DROP PROCEDURE IF EXISTS sp_nested_loop$$
CREATE PROCEDURE sp_nested_loop(IN i INT, IN j INT, OUT x INT, OUT y INT, INOUT z INT)
BEGIN
    DECLARE a INTEGER DEFAULT 0;
    DECLARE b INTEGER DEFAULT 0;
    DECLARE c INTEGER DEFAULT 0;
    WHILE a < i DO
        WHILE b < j DO
            SET c = c + 1;
            SET b = b + 1;
        END WHILE;
        SET a = a + 1;
        SET b = 0;
    END WHILE;
    SET x = a, y = c;
    SET z = x + y + z;
END $$
DELIMITER ;
```

Invoca (LLAMAR) el procedimiento almacenado:

```
SET @z = 30;
call sp_nested_loop(10, 20, @x, @y, @z);
SELECT @x, @y, @z;
```

Resultado:

```
+------+------+------+
|  @x  |  @y  |  @z  |
+------+------+------+
|  10  |  200 |  240 |
+------+------+------+
```

Un parámetro `IN` pasa un valor a un procedimiento. El procedimiento puede modificar el valor, pero la modificación no es visible para la persona que llama cuando el procedimiento vuelve.

Un parámetro `OUT` pasa un valor del procedimiento a la persona que llama. Su valor inicial es NULL dentro del procedimiento, y su valor es visible para la persona que llama cuando el procedimiento vuelve.

Un parámetro `INOUT` es inicializado por la persona que llama, puede ser modificado por el procedimiento, y cualquier cambio realizado por el procedimiento es visible para la persona que llama cuando el procedimiento regresa.

Ref: http://dev.mysql.com/doc/refman/5.7/en/create-procedure.html

Cursores

Los cursores le permiten iterar los resultados de la consulta uno por línea. `DECLARE` comando `DECLARE` se usa para iniciar el cursor y asociarlo con una consulta SQL específica:

```
DECLARE student CURSOR FOR SELECT name FROM studend;
```

Digamos que vendemos productos de algunos tipos. Queremos contar cuántos productos de cada tipo existen.

Nuestros datos:

```
CREATE TABLE product
(
  id   INT(10) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  type VARCHAR(50)      NOT NULL,
  name VARCHAR(255)     NOT NULL

);
CREATE TABLE product_type
(
  name VARCHAR(50) NOT NULL PRIMARY KEY
);
CREATE TABLE product_type_count
(
  type  VARCHAR(50)      NOT NULL PRIMARY KEY,
  count INT(10) UNSIGNED NOT NULL DEFAULT 0
);

INSERT INTO product_type (name) VALUES
  ('dress'),
  ('food');

INSERT INTO product (type, name) VALUES
  ('dress', 'T-shirt'),
```

```
  ('dress', 'Trousers'),
  ('food', 'Apple'),
  ('food', 'Tomatoes'),
  ('food', 'Meat');
```

Podemos lograr el objetivo usando el procedimiento almacenado usando el cursor:

```
DELIMITER //
DROP PROCEDURE IF EXISTS product_count;
CREATE PROCEDURE product_count()
  BEGIN
    DECLARE p_type VARCHAR(255);
    DECLARE p_count INT(10) UNSIGNED;
    DECLARE done INT DEFAULT 0;
    DECLARE product CURSOR FOR
      SELECT
        type,
        COUNT(*)
      FROM product
      GROUP BY type;
    DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done = 1;

    TRUNCATE product_type;

    OPEN product;

    REPEAT
      FETCH product
      INTO p_type, p_count;
      IF NOT done
      THEN
        INSERT INTO product_type_count
        SET
          type  = p_type,
          count = p_count;
      END IF;
    UNTIL done
    END REPEAT;

    CLOSE product;
  END //
DELIMITER ;
```

Cuando puede llamar al procedimiento con:

```
CALL product_count();
```

El resultado estaría en la tabla `product_type_count` :

```
type   | count
---------------
dress  |   2
food   |   3
```

Si bien ese es un buen ejemplo de un `CURSOR` , observe cómo todo el cuerpo del procedimiento puede ser reemplazado por solo

```
INSERT INTO product_type_count
        (type, count)
    SELECT type, COUNT(*)
        FROM product
        GROUP BY type;
```

Esto se ejecutará mucho más rápido.

Conjuntos de resultados múltiples

A diferencia de una instrucción `SELECT` , un `Stored Procedure` devuelve varios conjuntos de resultados. El código requiere que se use un código diferente para recopilar los resultados de una `CALL` en Perl, PHP, etc.

(¡Necesitas un código específico aquí o en otra parte!)

Crear una función

```
DELIMITER $$
CREATE
    DEFINER=`db_username`@`hostname_or_IP`
    FUNCTION `function_name`(optional_param data_type(length_if_applicable))
    RETURNS data_type
BEGIN
    /*
    SQL Statements goes here
    */
END$$
DELIMITER ;
```

El RETURNS data_type es cualquier tipo de datos MySQL.

Lea Rutinas almacenadas (procedimientos y funciones) en línea:
https://riptutorial.com/es/mysql/topic/1351/rutinas-almacenadas--procedimientos-y-funciones-

Capítulo 60: Se une

Sintaxis

- `INNER` y `OUTER` son ignorados.
- `FULL` no está implementado en MySQL.
- "commajoin" (`FROM a,b WHERE ax=by`) está mal visto; use `FROM a JOIN b ON ax=by` lugar.
- FROM a JOIN b ON ax = by incluye filas que coinciden en ambas tablas.
- DESDE una UNIÓN IZQUIERDA b ON ax = by incluye todas las filas de `a` , más los datos coincidentes de `b` , o `NULLs` si no hay una fila coincidente.

Examples

Ejemplos de unión

Consulta para crear tabla en db

```
CREATE TABLE `user` (
`id` smallint(5) unsigned NOT NULL AUTO_INCREMENT,
`name` varchar(30) NOT NULL,
`course` smallint(5) unsigned DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB;

CREATE TABLE `course` (
`id` smallint(5) unsigned NOT NULL AUTO_INCREMENT,
`name` varchar(50) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB;
```

Ya que estamos usando tablas InnoDB y sabemos que user.course y course.id están relacionados, podemos especificar una relación de clave externa:

```
ALTER TABLE `user`
ADD CONSTRAINT `FK_course`
FOREIGN KEY (`course`) REFERENCES `course` (`id`)
ON UPDATE CASCADE;
```

Consulta de unión (unión interna)

```
SELECT user.name, course.name
FROM `user`
INNER JOIN `course` on user.course = course.id;
```

ÚNETE con la subconsulta (tabla "Derivado")

```
SELECT x, ...
    FROM ( SELECT y, ... FROM ... ) AS a
    JOIN tbl  ON tbl.x = a.y
    WHERE ...
```

Esto evaluará la subconsulta en una tabla temporal, luego `JOIN` a `tbl` .

Antes de 5.6, no podía haber un índice en la tabla temporal. Por lo tanto, esto fue potencialmente muy ineficiente:

```
SELECT ...
    FROM ( SELECT y, ... FROM ... ) AS a
    JOIN ( SELECT x, ... FROM ... ) AS b  ON b.x = a.y
    WHERE ...
```

Con 5.6, el optimizador calcula el mejor índice y lo crea sobre la marcha. (Esto tiene algo de sobrecarga, por lo que aún no es "perfecto").

Otro paradigma común es tener una subconsulta para inicializar algo:

```
SELECT
        @n := @n + 1,
        ...
    FROM ( SELECT @n := 0 ) AS initialize
    JOIN the_real_table
    ORDER BY ...
```

(Nota: esto es técnicamente un `CROSS JOIN` (producto cartesiano), como lo indica la falta de `ON` . Sin embargo, es eficiente porque la subconsulta devuelve solo una fila que debe coincidir con las n filas en `the_real_table`).

Recuperar clientes con pedidos - variaciones en un tema

Esto obtendrá todos los pedidos para todos los clientes:

```
SELECT c.CustomerName, o.OrderID
    FROM Customers AS c
    INNER JOIN Orders AS o
        ON c.CustomerID = o.CustomerID
    ORDER BY c.CustomerName, o.OrderID;
```

Esto contará el número de pedidos para cada cliente:

```
SELECT c.CustomerName, COUNT(*) AS 'Order Count'
    FROM Customers AS c
    INNER JOIN Orders AS o
        ON c.CustomerID = o.CustomerID
    GROUP BY c.CustomerID;
    ORDER BY c.CustomerName;
```

Además, cuenta, pero probablemente más rápido:

```
SELECT  c.CustomerName,
        ( SELECT COUNT(*) FROM Orders WHERE CustomerID = c.CustomerID ) AS 'Order Count'
    FROM Customers AS c
    ORDER BY c.CustomerName;
```

Listar solo el cliente con pedidos.

```
SELECT  c.CustomerName,
    FROM Customers AS c
    WHERE EXISTS ( SELECT * FROM Orders WHERE CustomerID = c.CustomerID )
    ORDER BY c.CustomerName;
```

Unión externa completa

MySQL no es compatible con `FULL OUTER JOIN` , pero hay formas de emular una.

Configurando los datos

```
-- ----------------------------
-- Table structure for `owners`
-- ----------------------------
DROP TABLE IF EXISTS `owners`;
CREATE TABLE `owners` (
`owner_id` int(11) NOT NULL AUTO_INCREMENT,
`owner` varchar(30) DEFAULT NULL,
PRIMARY KEY (`owner_id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=latin1;

-- ----------------------------
-- Records of owners
-- ----------------------------
INSERT INTO `owners` VALUES ('1', 'Ben');
INSERT INTO `owners` VALUES ('2', 'Jim');
INSERT INTO `owners` VALUES ('3', 'Harry');
INSERT INTO `owners` VALUES ('6', 'John');
INSERT INTO `owners` VALUES ('9', 'Ellie');
-- ----------------------------
-- Table structure for `tools`
-- ----------------------------
DROP TABLE IF EXISTS `tools`;
CREATE TABLE `tools` (
`tool_id` int(11) NOT NULL AUTO_INCREMENT,
`tool` varchar(30) DEFAULT NULL,
`owner_id` int(11) DEFAULT NULL,
PRIMARY KEY (`tool_id`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=latin1;
-- ----------------------------
-- Records of tools
-- ----------------------------
INSERT INTO `tools` VALUES ('1', 'Hammer', '9');
INSERT INTO `tools` VALUES ('2', 'Pliers', '1');
INSERT INTO `tools` VALUES ('3', 'Knife', '1');
INSERT INTO `tools` VALUES ('4', 'Chisel', '2');
INSERT INTO `tools` VALUES ('5', 'Hacksaw', '1');
INSERT INTO `tools` VALUES ('6', 'Level', null);
INSERT INTO `tools` VALUES ('7', 'Wrench', null);
INSERT INTO `tools` VALUES ('8', 'Tape Measure', '9');
```

```
INSERT INTO `tools` VALUES ('9', 'Screwdriver', null);
INSERT INTO `tools` VALUES ('10', 'Clamp', null);
```

¿Qué queremos ver?

Queremos obtener una lista, en la que veamos quién es el propietario de qué herramientas y qué herramientas podrían no tener un propietario.

Las consultas

Para lograr esto, podemos combinar dos consultas utilizando `UNION` . En esta primera consulta, uniremos las herramientas de los propietarios mediante el uso de un `LEFT JOIN` . Esto agregará a todos nuestros propietarios a nuestro conjunto de resultados, sin importar si realmente poseen herramientas.

En la segunda consulta, estamos utilizando un `RIGHT JOIN` para unir las herramientas a los propietarios. De esta manera, conseguimos obtener todas las herramientas en nuestro conjunto de resultados, si no son propiedad de nadie, su columna de propietario simplemente contendrá `NULL` . Al agregar la CLASE- `WHERE` que está filtrando por `owners.owner_id IS NULL` , estamos definiendo el resultado como esos conjuntos de datos, que la primera consulta aún no ha devuelto, ya que solo buscamos los datos en la tabla de la derecha.

Dado que estamos utilizando `UNION ALL` el conjunto de resultados de la segunda consulta se adjuntará al primer conjunto de resultados de consultas.

```
SELECT `owners`.`owner`, tools.tool
FROM `owners`
LEFT JOIN `tools` ON `owners`.`owner_id` = `tools`.`owner_id`
UNION ALL
SELECT `owners`.`owner`, tools.tool
FROM `owners`
RIGHT JOIN `tools` ON `owners`.`owner_id` = `tools`.`owner_id`
WHERE `owners`.`owner_id` IS NULL;

+-------+--------------+
| owner | tool         |
+-------+--------------+
| Ben   | Pliers       |
| Ben   | Knife        |
| Ben   | Hacksaw      |
| Jim   | Chisel       |
| Harry | NULL         |
| John  | NULL         |
| Ellie | Hammer       |
| Ellie | Tape Measure |
| NULL  | Level        |
| NULL  | Wrench       |
| NULL  | Screwdriver  |
| NULL  | Clamp        |
+-------+--------------+
12 rows in set (0.00 sec)
```

Unión interna para 3 mesas

Supongamos que tenemos tres tablas que se pueden usar para sitios web simples con etiquetas.

- La primera mesa es para postes.
- Segundo para las etiquetas
- Tercero para Tags y Post relación

primera mesa "videojuego"

carné de identidad	título	fecha de registro	Contenido
1	BioShock Infinite	2016-08-08	....

tabla de "etiquetas"

carné de identidad	nombre
1	yenne
2	elizabeth

tabla "tags_meta"

ID del mensaje	tag_id
1	2

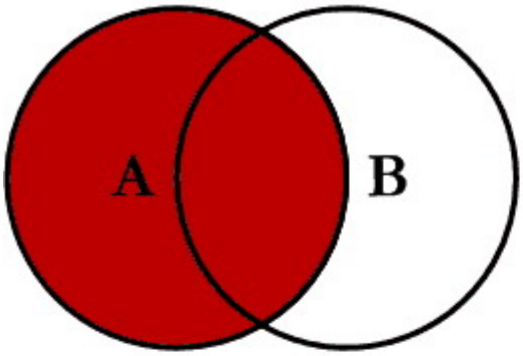
```
SELECT videogame.id,
    videogame.title,
    videogame.reg_date,
    tags.name,
    tags_meta.post_id
FROM tags_meta
INNER JOIN videogame ON videogame.id = tags_meta.post_id
INNER JOIN tags ON tags.id = tags_meta.tag_id
WHERE tags.name = "elizabeth"
ORDER BY videogame.reg_date
```

este código puede devolver todas las publicaciones relacionadas con esa etiqueta "#elizabeth"

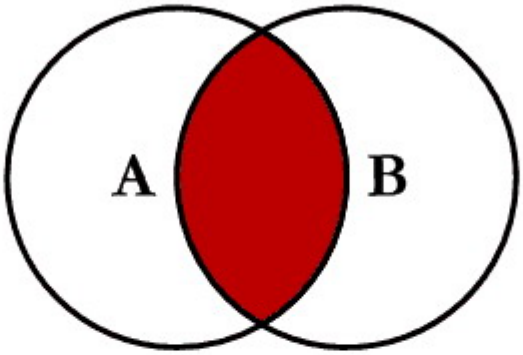
Uniones visualizadas

Si eres una persona orientada visualmente, este diagrama de Venn puede ayudarte a comprender los diferentes tipos de `JOIN` que existen dentro de MySQL.

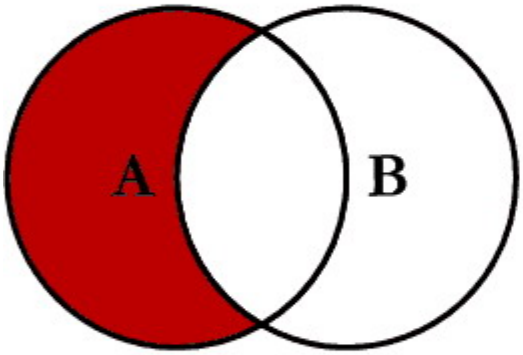
SQL JOINS



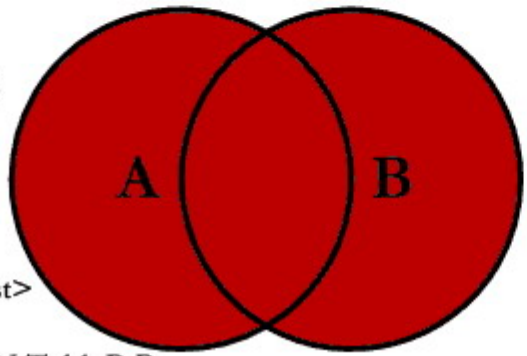

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

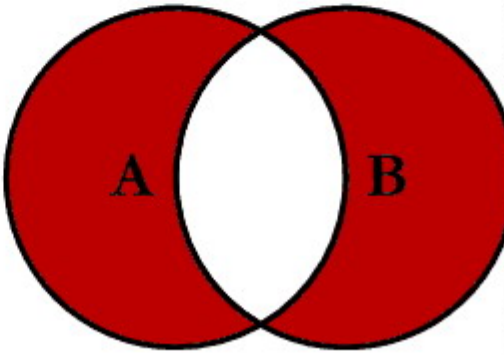



```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```






```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```


Lea Se une en línea: https://riptutorial.com/es/mysql/topic/2736/se-une

Capítulo 61: Seguridad a través de GRANTs

Examples

Mejores prácticas

Limite la raíz (y cualquier otro usuario con privilegios SUPER) a

```
GRANT ... TO root@localhost ...
```

Eso impide el acceso desde otros servidores. Debe entregar SUPER a muy pocas personas y ellos deben ser conscientes de su responsabilidad. La aplicación no debe tener SUPER.

Limite los inicios de sesión de la aplicación a la base de datos que usa:

```
GRANT ... ON dbname.* ...
```

De esa manera, alguien que hackea el código de la aplicación no puede pasar el dbname. Esto puede ser refinado aún más a través de cualquiera de estos:

```
GRANT SELECT ON dname.* ...    -- "read only"
GRANT ... ON dname.tblname ... -- "just one table"
```

La lectura solo puede necesitar cosas "seguras" como

```
GRANT SELECT, CREATE TEMPORARY TABLE ON dname.* ...    -- "read only"
```

Como dices, no hay seguridad absoluta. Mi punto aquí es que puedes hacer algunas cosas para frenar a los hackers. (Lo mismo ocurre con las personas honestas que hacen el tonto).

En raras ocasiones, es posible que necesite la aplicación para hacer algo disponible solo para root. Esto se puede hacer a través de un "Procedimiento almacenado" que tiene `SECURITY DEFINER` (y la raíz lo define). Eso expondrá solo lo que hace el SP, que podría ser, por ejemplo, una acción particular en una tabla en particular.

Host (del usuario @ host)

El "host" puede ser un nombre de host o una dirección IP. Además, puede implicar comodines.

```
GRANT SELECT ON db.* TO sam@'my.domain.com' IDENTIFIED BY 'foo';
```

Ejemplos: Nota: estos usualmente necesitan ser citados

```
localhost -- the same machine as mysqld
'my.domain.com' -- a specific domain; this involves a lookup
'11.22.33.44' -- a specific IP address
```

```
'192.168.1.%' -- wild card for trailing part of IP address.  (192.168.% and 10.% and 11.% are
"internal" ip addresses.)
```

El uso de `localhost` basa en la seguridad del servidor. Para las mejores prácticas, la `root` solo debe permitirse a través de localhost. En algunos casos, estos significan lo mismo: `0.0.0.1` y `::1` .

Lea Seguridad a través de GRANTs en línea:
https://riptutorial.com/es/mysql/topic/5131/seguridad-a-traves-de-grants

Capítulo 62: SELECCIONAR

Introducción

`SELECT` se utiliza para recuperar filas seleccionadas de una o más tablas.

Sintaxis

- SELECCIONAR DISTINTO [expresiones] DE TableName [DÓNDE condiciones]; /// Selección simple
- SELECT DISTINCT (a), b ... es lo mismo que SELECT DISTINCT a, b ...
- SELECCIONAR [TODO | DISTINCIÓN | DISTINCTROW] [HIGH_PRIORITY] [STRAIGHT_JOIN] [SQL_SMALL_RESULT | SQL_BIG_RESULT] [SQL_BUFFER_RESULT] [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS] expresiones DE las tablas [DÓNDE condiciones] [GRUPO POR expresiones] [QUE HAYA condición] [ORDEN POR expresión [ASC | DESC]] [LIMIT [offset_value] number_rows | LIMIT number_rows OFFSET offset_value] [PROCEDIMIENTO procedure_name] [INTO [OUTFILE 'file_name' opciones | DUMPFILE 'nombre_archivo' | @ variable1, @ variable2, ... @variable_n] [PARA ACTUALIZAR | BLOQUEO EN MODO COMPARTIDO]; /// Sintaxis de selección completa

Observaciones

Para obtener más información sobre la instrucción `SELECT` de MySQL, consulte MySQL Docs .

Examples

SELECCIONAR por nombre de columna

```
CREATE TABLE stack(
    id INT,
    username VARCHAR(30) NOT NULL,
    password VARCHAR(30) NOT NULL
);

INSERT INTO stack (`id`, `username`, `password`) VALUES (1, 'Foo', 'hiddenGem');
INSERT INTO stack (`id`, `username`, `password`) VALUES (2, 'Baa', 'verySecret');
```

Consulta

```
SELECT id FROM stack;
```

Resultado

```
+------+
| id   |
+------+
|    1 |
|    2 |
+------+
```

SELECCIONAR todas las columnas (*)

Consulta

```
SELECT * FROM stack;
```

Resultado

```
+------+----------+----------+
| id   | username | password |
+------+----------+----------+
|    1 | admin    | admin    |
|    2 | stack    | stack    |
+------+----------+----------+
2 rows in set (0.00 sec)
```

Puede seleccionar todas las columnas de una tabla en una combinación haciendo:

```
SELECT stack.* FROM stack JOIN Overflow ON stack.id = Overflow.id;
```

Práctica recomendada No utilice `*` menos que esté depurando o recuperando la (s) fila (s) en matrices asociativas, de lo contrario los cambios de esquema (AGREGAR / DROPAR / reorganizar columnas) pueden provocar errores de aplicación desagradables. Además, si proporciona la lista de columnas que necesita en su conjunto de resultados, el planificador de consultas de MySQL a menudo puede optimizar la consulta.

Pros:

1. Cuando agrega / elimina columnas, no tiene que hacer cambios donde usó `SELECT *`
2. Es mas corto para escribir
3. También ve las respuestas, ¿entonces el uso de `SELECT *` puede justificarse alguna vez?

Contras:

1. Usted está devolviendo más datos de los que necesita. Supongamos que agrega una columna VARBINARY que contiene 200k por fila. Solo necesita estos datos en un solo lugar para un solo registro: con `SELECT *` puede terminar devolviendo 2MB por 10 filas que no necesita
2. Explícito sobre qué datos se utilizan
3. Especificar columnas significa que recibe un error cuando se elimina una columna
4. El procesador de consultas tiene que hacer un poco más de trabajo: averiguar qué columnas existen en la tabla (gracias @vinodadhikary)
5. Puedes encontrar donde se usa una columna más fácilmente

6. Obtiene todas las columnas en uniones si usa SELECT *
7. No se pueden usar referencias ordinales de manera segura (aunque usar referencias ordinales para columnas es una mala práctica en sí misma)
8. En consultas complejas con campos de `TEXT` , la consulta puede verse ralentizada por un procesamiento de la tabla temporal menos óptimo

SELECCIONA con DONDE

Consulta

```
SELECT * FROM stack WHERE username = "admin" AND password = "admin";
```

Resultado

```
+------+----------+----------+
| id   | username | password |
+------+----------+----------+
|    1 | admin    | admin    |
+------+----------+----------+
1 row in set (0.00 sec)
```

Consulta con un SELECT anidado en la cláusula WHERE

La cláusula `WHERE` puede contener cualquier instrucción `SELECT` válida para escribir consultas más complejas. Esta es una consulta 'anidada'

Consulta

Las consultas anidadas se utilizan generalmente para devolver valores atómicos únicos de las consultas para las comparaciones.

```
SELECT title FROM books WHERE author_id = (SELECT id FROM authors WHERE last_name = 'Bar' AND
first_name = 'Foo');
```

Selecciona todos los nombres de usuario sin dirección de correo electrónico

```
SELECT * FROM stack WHERE username IN (SELECT username FROM signups WHERE email IS NULL);
```

Descargo de responsabilidad: considere la posibilidad de utilizar uniones para mejorar el rendimiento al comparar un conjunto de resultados completo.

SELECCIONAR con LIKE (%)

```
CREATE TABLE stack
(  id int AUTO_INCREMENT PRIMARY KEY,
```

```
   username VARCHAR(100) NOT NULL
);

INSERT stack(username) VALUES
('admin'),('k admin'),('adm'),('a adm b'),('b XadmY c'), ('adm now'), ('not here');
```

"adm" en cualquier lugar:

```
SELECT * FROM stack WHERE username LIKE "%adm%";
+----+-----------+
| id | username  |
+----+-----------+
|  1 | admin     |
|  2 | k admin   |
|  3 | adm       |
|  4 | a adm b   |
|  5 | b XadmY c |
|  6 | adm now   |
+----+-----------+
```

Comienza con "adm":

```
SELECT * FROM stack WHERE username LIKE "adm%";
+----+----------+
| id | username |
+----+----------+
|  1 | admin    |
|  3 | adm      |
|  6 | adm now  |
+----+----------+
```

Termina con "adm":

```
SELECT * FROM stack WHERE username LIKE "%adm";
+----+----------+
| id | username |
+----+----------+
|  3 | adm      |
+----+----------+
```

Al igual que el carácter `%` en una cláusula `LIKE` coincide con cualquier número de caracteres, el carácter _ solo coincide con un carácter. Por ejemplo,

```
SELECT * FROM stack WHERE username LIKE "adm_n";
+----+----------+
| id | username |
+----+----------+
|  1 | admin    |
+----+----------+
```

Notas de rendimiento Si hay un índice en el `username` de `username` , entonces

- `LIKE 'adm'` realiza lo mismo que `= 'adm'
- `LIKE 'adm%` es un "rango", similar a `BETWEEN..AND..` Puede hacer un buen uso de un índice en

la columna.
- `LIKE '%adm'` (o cualquier variante con un comodín *líder)* no pueden utilizar cualquier índice. Por lo tanto será lento. En tablas con muchas filas, es probable que sea tan lento que sea inútil.
- `RLIKE` (`REGEXP`) tiende a ser más lento que `LIKE` , pero tiene más capacidades.
- Mientras que MySQL ofrece la indexación `FULLTEXT` en muchos tipos de tablas y columnas, esos índices `FULLTEXT` *no se* usan para cumplir las consultas que usan `LIKE` .

SELECCIONAR con Alias (AS)

Los alias de SQL se utilizan para cambiar temporalmente el nombre de una tabla o una columna. Generalmente se utilizan para mejorar la legibilidad.

Consulta

```
SELECT username AS val FROM stack;
SELECT username val FROM stack;
```

(Nota: `AS` es sintácticamente opcional.)

Resultado

```
+-------+
| val   |
+-------+
| admin |
| stack |
+-------+
2 rows in set (0.00 sec)
```

SELECT con una cláusula LIMIT

Consulta:

```
SELECT *
  FROM Customers
 ORDER BY CustomerID
 LIMIT 3;
```

Resultado:

Identificación del cliente	Nombre del cliente	Nombre de contacto	Dirección	Ciudad	Código postal	País
1	Alfreds Futterkiste	Maria anders	Obere Str. 57	Berlina	12209	Alemania
2	Ana Trujillo Emparedados	Ana trujillo	Avda. de la Constitución	México DF	05021	Méjico

	Y Helados		2222			
3	Taquería antonio moreno	Antonio moreno	Mataderos 2312	México DF	05023	Méjico

Mejores Prácticas Siempre use `ORDER BY` al usar `LIMIT` ; De lo contrario, las filas que obtendrá serán impredecibles.

Consulta:

```
SELECT *
  FROM Customers
 ORDER BY CustomerID
 LIMIT 2,1;
```

Explicación:

Cuando una cláusula `LIMIT` contiene dos números, se interpreta como `LIMIT offset,count` . Entonces, en este ejemplo, la consulta salta dos registros y devuelve uno.

Resultado:

Identificación del cliente	Nombre del cliente	Nombre de contacto	Dirección	Ciudad	Código postal	País
3	Taquería antonio moreno	Antonio moreno	Mataderos 2312	México DF	05023	Méjico

Nota:

Los valores en las cláusulas `LIMIT` deben ser constantes; pueden no ser valores de columna.

SELECCIONAR con DISTINTO

La cláusula `DISTINCT` después de `SELECT` elimina filas duplicadas del conjunto de resultados.

```
CREATE TABLE `car`
(   `car_id` INT UNSIGNED NOT NULL PRIMARY KEY,
    `name` VARCHAR(20),
    `price` DECIMAL(8,2)
);

INSERT INTO CAR (`car_id`, `name`, `price`) VALUES (1, 'Audi A1', '20000');
INSERT INTO CAR (`car_id`, `name`, `price`) VALUES (2, 'Audi A1', '15000');
INSERT INTO CAR (`car_id`, `name`, `price`) VALUES (3, 'Audi A2', '40000');
INSERT INTO CAR (`car_id`, `name`, `price`) VALUES (4, 'Audi A2', '40000');

SELECT DISTINCT `name`, `price` FROM CAR;
+---------+----------+
```

```
| name    | price    |
+---------+----------+
| Audi A1 | 20000.00 |
| Audi A1 | 15000.00 |
| Audi A2 | 40000.00 |
+---------+----------+
```

`DISTINCT` funciona en todas las columnas para entregar los resultados, no columnas individuales. Este último es a menudo una idea falsa de los nuevos desarrolladores de SQL. En resumen, lo que importa es la distinción en el nivel de fila del conjunto de resultados, no la distinción en el nivel de columna. Para visualizar esto, mire "Audi A1" en el conjunto de resultados anterior.

Para versiones posteriores de MySQL, `DISTINCT` tiene implicaciones con su uso junto con `ORDER BY` . La configuración para `ONLY_FULL_GROUP_BY` entra en juego como se ve en la siguiente página del Manual de MySQL titulada Manejo de MySQL de GROUP BY .

SELECCIONAR con LIKE (_)

Un carácter _ en un patrón de cláusula `LIKE` coincide con un solo carácter.

Consulta

```
SELECT username FROM users WHERE users LIKE 'admin_';
```

Resultado

```
+----------+
| username |
+----------+
| admin1   |
| admin2   |
| admin-   |
| adminA   |
+----------+
```

SELECCIONAR con CASO o SI

Consulta

```
SELECT st.name,
       st.percentage,
       CASE WHEN st.percentage >= 35 THEN 'Pass' ELSE 'Fail' END AS `Remark`
FROM student AS st ;
```

Resultado

```
+---------------------------------+
|   name   | percentage | Remark  |
+---------------------------------+
|   Isha   |     67     |  Pass   |
|   Rucha  |     28     |  Fail   |
|   Het    |     35     |  Pass   |
```

```
|   Ansh   |     92     |  Pass  |
+--------------------------------+
```

O con si

```
SELECT st.name,
       st.percentage,
       IF(st.percentage >= 35, 'Pass', 'Fail') AS `Remark`
FROM student AS st ;
```

nótese bien

```
IF(st.percentage >= 35, 'Pass', 'Fail')
```

> Esto significa que: SI st.percentage> = 35 es **TRUE** luego devuelva `'Pass'` ELSE return `'Fail'`

SELECCIONAR CON ENTRE

Puede usar la cláusula BETWEEN para reemplazar una combinación de condiciones "mayores que iguales Y menores que iguales".

Datos

```
+----+-----------+
| id | username  |
+----+-----------+
|  1 | admin     |
|  2 | root      |
|  3 | toor      |
|  4 | mysql     |
|  5 | thanks    |
|  6 | java      |
+----+-----------+
```

Consulta con operadores.

```
SELECT * FROM stack WHERE id >= 2 and id <= 5;
```

Consulta similar con BETWEEN

```
SELECT * FROM stack WHERE id BETWEEN 2 and 5;
```

Resultado

```
+----+-----------+
| id | username  |
+----+-----------+
|  2 | root      |
|  3 | toor      |
|  4 | mysql     |
```

```
|  5 | thanks    |
+----+-----------+
4 rows in set (0.00 sec)
```

Nota

> **BETWEEN usa `>=` y `<=` , no `>` y `<` .**

Usando NO ENTRE

Si quieres usar el negativo puedes usar `NOT` . Por ejemplo :

```
SELECT * FROM stack WHERE id NOT BETWEEN 2 and 5;
```

Resultado

```
+----+-----------+
| id | username  |
+----+-----------+
|  1 | admin     |
|  6 | java      |
+----+-----------+
2 rows in set (0.00 sec)
```

Nota

> **NO ENTRE utiliza `>` y `<` y no `>=` y `<=`** Es decir, `WHERE id NOT BETWEEN 2 and 5` es lo mismo que `WHERE (id < 2 OR id > 5)` .

Si tiene un índice en una columna que usa en una `BETWEEN` search, MySQL puede usar ese índice para una exploración de rango.

SELECCIONAR con rango de fechas

```
SELECT ... WHERE dt >= '2017-02-01'
             AND dt  < '2017-02-01' + INTERVAL 1 MONTH
```

Claro, esto podría hacerse con `BETWEEN` e inclusión de `23:59:59` . Pero, el patrón tiene estos beneficios:

- No tiene un cálculo previo de la fecha de finalización (que a menudo es una longitud exacta desde el inicio)
- No incluye ambos puntos finales (como lo hace `BETWEEN`), ni escriba '23: 59: 59 'para evitarlo.
- Funciona para `DATE` , `TIMESTAMP` , `DATETIME` e incluso el `DATETIME(6)` incluido en el microsegundo `DATETIME(6)` .
- Se ocupa de los días bisiesto, fin de año, etc.
- Es fácil de indexar (también lo es `BETWEEN`).

Lea SELECCIONAR en línea: https://riptutorial.com/es/mysql/topic/3307/seleccionar

Capítulo 63: Tabla de mapeo de muchos a muchos

Observaciones

- Falta de un ID de `AUTO_INCREMENT` para esta tabla: la PK dada es la PK 'natural'; No hay una buena razón para un sustituto.
- `MEDIUMINT` : este es un recordatorio de que todos los `INTs` deben hacerse tan pequeños como seguros (más pequeños ⇒ más rápidos). Por supuesto, la declaración aquí debe coincidir con la definición en la tabla a la que está vinculado.
- `UNSIGNED` : Casi todos los INT también pueden ser declarados no negativos
- `NOT NULL` - Bueno, eso es verdad, ¿no es así?
- `InnoDB` : más eficiente que MyISAM debido a la forma en que la `PRIMARY KEY` está agrupada con los datos en InnoDB.
- `INDEX(y_id, x_id)` - La `PRIMARY KEY` hace que sea eficiente ir en una dirección; el hace la otra dirección eficiente. No hace falta decir `UNIQUE` ; eso sería un esfuerzo extra en `INSERTs` .
- En el índice secundario, decir que solo `INDEX(y_id)` funcionaría porque incluiría implícitamente `x_id` . Pero preferiría que sea más obvio que espero un índice de "cobertura".

Es *posible que* desee agregar más columnas a la tabla; esto es raro. Las columnas adicionales podrían proporcionar información sobre la *relación* que representa la tabla.

Es *posible que* desee agregar restricciones `FOREIGN KEY` .

Examples

Esquema tipico

```
CREATE TABLE XtoY (
    # No surrogate id for this table
    x_id MEDIUMINT UNSIGNED NOT NULL,   -- For JOINing to one table
    y_id MEDIUMINT UNSIGNED NOT NULL,   -- For JOINing to the other table
    # Include other fields specific to the 'relation'
    PRIMARY KEY(x_id, y_id),            -- When starting with X
    INDEX      (y_id, x_id)             -- When starting with Y
) ENGINE=InnoDB;
```

(Ver Comentarios, a continuación, para la justificación.)

Lea Tabla de mapeo de muchos a muchos en línea:
https://riptutorial.com/es/mysql/topic/4857/tabla-de-mapeo-de-muchos-a-muchos

Capítulo 64: Tabla dinámica de Un-Pivot usando una declaración preparada

Examples

Des-pivote un conjunto dinámico de columnas basado en condición

El siguiente ejemplo es una base muy útil cuando intenta convertir datos de transacciones a datos no pivotados por motivos de BI / informes, donde las dimensiones que deben no pivotarse pueden tener un conjunto dinámico de columnas.

Para nuestro ejemplo, suponemos que la tabla de datos sin procesar contiene datos de evaluación de empleados en forma de preguntas marcadas.

La tabla de datos en bruto es la siguiente:

```
create table rawdata

(
 PersonId VARCHAR(255)
,Question1Id INT(11)
,Question2Id INT(11)
,Question3Id INT(11)
)
```

La tabla rawdata es una tabla temporal como parte del procedimiento ETL y puede tener un número variable de preguntas. El objetivo es utilizar el mismo procedimiento de no giro para un número arbitrario de preguntas, es decir, columnas que no serán pivotadas.

A continuación se muestra un ejemplo de juguete de la tabla rawdata:

PersonId	Question1Id	Question2Id	Question3Id
Giannaros	1	3	1
Patra	2	4	3

La forma bien conocida y estática de no particionar los datos, en MYSQL, es mediante el uso de UNION ALL:

```
create table unpivoteddata

(

 PersonId VARCHAR(255)
,QuestionId VARCHAR(255)
,QuestionValue INT(11)

);

INSERT INTO unpivoteddata SELECT PersonId, 'Question1Id' col, Question1Id
```

```
FROM rawdata
UNION ALL
SELECT PersonId, 'Question2Id' col, Question2Id
FROM rawdata
UNION ALL
SELECT PersonId, 'Question3Id' col, Question3Id
FROM rawdata;
```

En nuestro caso, queremos definir una manera de no dividir un número arbitrario de columnas QuestionId. Para eso necesitamos ejecutar una declaración preparada que es una selección dinámica de las columnas deseadas. Para poder elegir qué columnas deben ser no pivotadas, usaremos una declaración GROUP_CONCAT y elegiremos las columnas para las cuales el tipo de datos se establece en 'int'. En el GROUP_CONCAT también incluimos todos los elementos adicionales de nuestra instrucción SELECT que se ejecutará.

```
set @temp2 = null;

SELECT GROUP_CONCAT(' SELECT ', 'PersonId',',','''',COLUMN_NAME,'''', ' col
',',',COLUMN_NAME,' FROM rawdata' separator ' UNION ALL' ) FROM     INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'rawdata' AND DATA_TYPE = 'Int' INTO     @temp2;

select @temp2;
```

En otra ocasión podríamos haber elegido columnas con las que el nombre de la columna coincide con un patrón, por ejemplo, en lugar de

```
DATA_TYPE = 'Int'
```

utilizar

```
COLUMN_NAME LIKE 'Question%'
```

o algo adecuado que puede ser controlado a través de la fase ETL.

La declaración preparada se finaliza de la siguiente manera:

```
set @temp3 = null;

select concat('INSERT INTO unpivoteddata',@temp2) INTO @temp3;

select @temp3;

prepare stmt FROM @temp3;
execute stmt;
deallocate prepare stmt;
```

La tabla de datos no divididos es la siguiente:

```
SELECT * FROM unpivoteddata
```

PersonId	QuestionId	QuestionValue
Giannaros	Question1Id	1
Patra	Question1Id	2
Giannaros	Question2Id	3
Patra	Question2Id	4
Giannaros	Question3Id	1
Patra	Question3Id	3

La selección de columnas de acuerdo con una condición y luego la elaboración de una declaración preparada es una forma eficiente de desviar dinámicamente los datos.

Lea Tabla dinámica de Un-Pivot usando una declaración preparada en línea: https://riptutorial.com/es/mysql/topic/6491/tabla-dinamica-de-un-pivot-usando-una-declaracion-preparada

Capítulo 65: Tiempo con precisión subsecundaria.

Observaciones

Debe estar en la versión 5.6.4 o posterior de MySQL para declarar columnas con tipos de datos de fracciones de segundo.

Por ejemplo, `DATETIME(3)` le dará una resolución de milisegundos en sus marcas de tiempo, y `TIMESTAMP(6)` le dará una resolución de microsegundos en una marca de tiempo de estilo * nix.

Lea esto: http://dev.mysql.com/doc/refman/5.7/en/fractional-seconds.html

`NOW(3)` le dará la actualidad del sistema operativo de su servidor MySQL con una precisión de milisegundos.

(Tenga en cuenta que la aritmética fraccional interna de MySQL, como * 0.001, siempre se maneja como punto flotante de doble precisión IEEE754, por lo que es poco probable que pierda precisión antes de que el Sol se convierta en una estrella enana blanca).

Examples

Obtén la hora actual con milisegundos de precisión.

```
SELECT NOW(3)
```

Hace el truco.

Obtenga la hora actual en un formulario que se parece a una marca de tiempo de Javascript.

Las marcas de tiempo de Javascript se basan en el venerable UNIX `time_t` data type y muestran el número de milisegundos desde `1970-01-01 00:00:00` UTC.

Esta expresión obtiene la hora actual como un entero de marca de tiempo de Javascript. (Lo hace correctamente, independientemente de la configuración actual de time_zone.)

```
ROUND(UNIX_TIMESTAMP(NOW(3)) * 1000.0, 0)
```

Si tiene valores de `TIMESTAMP` almacenados en una columna, puede recuperarlos como marcas de tiempo de Javascript enteras utilizando la función UNIX_TIMESTAMP ().

```
 SELECT ROUND(UNIX_TIMESTAMP(column) * 1000.0, 0)
```

Si su columna contiene columnas `DATETIME` y las recupera como marcas de tiempo de Javascript, esas marcas de tiempo se compensarán con el desplazamiento de zona horaria de la zona horaria en la que están almacenados.

Crear una tabla con columnas para almacenar sub-segundo tiempo.

```
CREATE TABLE times (
     dt DATETIME(3),
     ts TIMESTAMP(3)
 );
```

crea una tabla con campos de fecha / hora de precisión de milisegundos.

```
 INSERT INTO times VALUES (NOW(3), NOW(3));
```

inserta una fila que contiene valores de `NOW()` con una precisión de milisegundos en la tabla.

```
 INSERT INTO times VALUES ('2015-01-01 16:34:00.123','2015-01-01 16:34:00.128');
```

Inserta valores específicos de milisegundos de precisión.

Tenga en cuenta que debe usar `NOW(3)` lugar de `NOW()` si usa esa función para insertar valores de tiempo de alta precisión.

Convertir un valor de fecha / hora de precisión de milisegundos en texto.

`%f` es el especificador de formato de precisión fraccional para la función DATE_FORMAT () .

```
SELECT DATE_FORMAT(NOW(3), '%Y-%m-%d %H:%i:%s.%f')
```

muestra un valor como `2016-11-19 09:52:53.248000` con microsegundos fraccionarios. Debido a que usamos `NOW(3)` , los tres dígitos finales en la fracción son 0.

Almacenar una marca de tiempo de Javascript en una columna TIMESTAMP

Si tiene un valor de marca de tiempo de Javascript, por ejemplo `1478960868932` , puede convertirlo en un valor de tiempo fraccional de MySQL como este:

```
FROM_UNIXTIME(1478960868932 * 0.001)
```

Es fácil de usar ese tipo de expresión para almacenar su marca de tiempo de Javascript en una tabla MySQL. Hacer esto:

```
INSERT INTO table (col) VALUES (FROM_UNIXTIME(1478960868932 * 0.001))
```

(Obviamente, querrás insertar otras columnas).

Lea Tiempo con precisión subsecundaria. en línea:
https://riptutorial.com/es/mysql/topic/7850/tiempo-con-precision-subsecundaria-

Capítulo 66: Tipos de datos

Examples

Fundición implícita / automática

```
select '123' * 2;
```

Para hacer la **multiplicación** con `2` MySQL convierte automáticamente la cadena `123` en un número.

Valor de retorno:

> 246

La conversión a un número comienza de izquierda a derecha. Si la conversión no es posible el resultado es `0`

```
select '123ABC' * 2
```

Valor de retorno:

> 246

```
select 'ABC123' * 2
```

Valor de retorno:

> 0

VARCHAR (255) - o no

Sugerido max len

Primero, mencionaré algunas cadenas comunes que siempre son hexadecimales o, de lo contrario, limitadas a ASCII. Para estos, debe especificar `CHARACTER SET ascii` (`latin1` está bien) para que no desperdicie espacio:

```
UUID CHAR(36) CHARACTER SET ascii -- or pack into BINARY(16)
country_code CHAR(2) CHARACTER SET ascii
ip_address CHAR(39) CHARACTER SET ascii -- or pack into BINARY(16)
phone VARCHAR(20) CHARACTER SET ascii -- probably enough to handle extension
postal_code VARCHAR(20) CHARACTER SET ascii -- (not 'zip_code') (don't know the max

city VARCHAR(100) -- This Russian town needs 91:
    Poselok Uchebnogo Khozyaystva Srednego Professionalno-Tekhnicheskoye Uchilishche Nomer
Odin
country VARCHAR(50) -- probably enough
```

```
name VARCHAR(64) -- probably adequate; more than some government agencies allow
```

¿Por qué no simplemente 255? Hay dos razones para evitar la práctica común de usar (255) para todo.

- Cuando un `SELECT` complejo necesita crear una tabla temporal (para una subconsulta, `UNION` , `GROUP BY` , etc.), la opción preferida es usar el motor `MEMORY` , que coloca los datos en la RAM. Pero los `VARCHARs` se convierten en `CHAR` en el proceso. Esto hace que `VARCHAR(255) CHARACTER SET utf8mb4` tome 1020 bytes. Eso puede llevar a la necesidad de derramar en el disco, que es más lento.
- En ciertas situaciones, InnoDB verá el tamaño potencial de las columnas en una tabla y decidirá que será demasiado grande, abortando una `CREATE TABLE` .

VARCHAR versus **TEXTO**

Consejos de uso para `*TEXT` , `CHAR` y `VARCHAR` , además de algunas de las mejores prácticas:

- Nunca uses `TINYTEXT` .
- Casi nunca se usa `CHAR` - es de longitud fija; cada carácter es la longitud máxima del conjunto de `CHARACTER SET` (por ejemplo, 4 bytes / carácter para utf8mb4).
- Con `CHAR` , use `CHARACTER SET ascii` menos que sepa lo contrario.
- `VARCHAR(n)` truncará en n *caracteres* ; `TEXT` se truncará en algún número de *bytes* . (Pero, ¿quieres truncamiento?)
- `*TEXT` *puede* ralentizar los `SELECTs` complejos debido a cómo se manejan las tablas temporales.

INT como AUTO_INCREMENT

Cualquier tamaño de `INT` se puede utilizar para `AUTO_INCREMENT` . `UNSIGNED` siempre es apropiado.

Tenga en cuenta que ciertas operaciones "queman" los `AUTO_INCREMENT` . Esto podría llevar a una brecha inesperada. Ejemplos: `INSERT IGNORE` y `REPLACE` . *Pueden* asignar previamente una identificación *antes de* darse cuenta de que no será necesaria. Este es el comportamiento esperado y por diseño en el motor InnoDB y no debe desalentar su uso.

Otros

Ya hay una entrada por separado para "FLOAT, DOUBLE, DECIMAL" y "ENUM". Es probable que una sola página sobre tipos de datos sea difícil de manejar; sugiero que los "tipos de campo" (¿o debería llamarse "tipos de datos"?) Sean una descripción general, luego se dividan en estas páginas de temas:

- INTs
- FLOTACIÓN, DOBLE, Y DECIMAL
- Cuerdas (CHARs, texto, etc.)
- BINARIO y BLOB
- DATETIME, TIMESTAMP, y amigos
- ENUM y SET

- Datos espaciales
- Tipo JSON (MySQL 5.7.8+)
- Cómo representar el dinero y otros 'tipos' comunes que necesitan adaptarse a los tipos de datos existentes

Cuando sea apropiado, cada página del tema debe incluir, además de la sintaxis y los ejemplos:

- Consideraciones al alterar
- Tamaño (bytes)
- Contraste con los motores que no son MySQL (baja prioridad)
- Consideraciones al utilizar el tipo de datos en una CLAVE PRIMARIA o clave secundaria
- otra mejor práctica
- otros problemas de rendimiento

(Supongo que este "ejemplo" se auto distraccionará cuando mis sugerencias hayan sido satisfechas o vetadas).

Introducción (numérica)

MySQL ofrece una serie de diferentes tipos numéricos. Estos se pueden dividir en

Grupo	Los tipos
Tipos enteros	`INTEGER` , `INT` , `SMALLINT` , `TINYINT` , `MEDIUMINT` , `BIGINT`
Tipos de puntos fijos	`DECIMAL` , `NUMERIC`
Tipos de punto flotante	`FLOAT` , `DOUBLE`
Tipo de valor de bit	`BIT`

Tipos enteros

El valor mínimo sin firmar es siempre 0.

Tipo	Almacenamiento (Bytes)	Valor mínimo (Firmado)	Valor máximo (Firmado)	Valor máxi (No firmado
`TINYINT`	1	-2^{7} -128	$2^{7}-1$ 127	$2^{8}-1$ 255
`SMALLINT`	2	-2^{15} -32,768	$2^{15}-1$ 32.767	$2^{16}-1$ 65,535
`MEDIUMINT`	3	-2^{23} -8,388,608	$2^{23}-1$ 8,388,607	$2^{24}-1$ 16,777,215

Tipo	Almacenamiento (Bytes)	Valor mínimo (Firmado)	Valor máximo (Firmado)	Valor máxi (No firmad
`INT`	4	-2^{31} -2,147,483,648	$2^{31}-1$ 2,147,483,647	$2^{32}-1$ 4,294,967,2
`BIGINT`	8	-2^{63} - 9,223,372,036,854,775,808	$2^{63}-1$ 9.223.372.036.854.775.807	$2^{64}-1$ 18,446,744,

Tipos de puntos fijos

Los tipos `DECIMAL` y `NUMERIC` MySQL almacenan valores de datos numéricos exactos. Se recomienda utilizar estos tipos para preservar la precisión exacta, como por ejemplo, por dinero.

Decimal

Estos valores se almacenan en formato binario. En una declaración de columna, se debe especificar la precisión y la escala.

La precisión representa el número de dígitos significativos que se almacenan para los valores.

La escala representa el número de dígitos almacenados *después* del decimal

```
salary DECIMAL(5,2)
```

5 representa la `precision` y 2 representa la `scale` . Para este ejemplo, el rango de valores que se pueden almacenar en esta columna es de `-999.99 to 999.99`

Si se omite el parámetro de escala, el valor predeterminado es 0

Este tipo de datos puede almacenar hasta 65 dígitos.

El número de bytes tomados por `DECIMAL(M,N)` es *aproximadamente* `M/2` .

Tipos de punto flotante

`FLOAT` y `DOUBLE` representan tipos de datos *aproximados* .

Tipo	Almacenamiento	Precisión	Distancia
FLOTADOR	4 bytes	23 bits significativos / ~ 7 dígitos decimales	10 ^ + / - 38
DOBLE	8 bytes	53 bits significativos / ~ 16 dígitos decimales	10 ^ + / - 308

`REAL` es un sinónimo de `FLOAT` . `DOUBLE PRECISION` es sinónimo de `DOUBLE` .

Aunque MySQL también permite el calificador (M, D), no *lo* use. (M, D) significa que los valores se pueden almacenar con hasta M dígitos totales, donde D puede estar después del decimal. *Los números se redondearán dos veces o se truncarán; Esto causará más problemas que beneficios.*

Debido a que los valores de punto flotante son aproximados y no se almacenan como valores exactos, los intentos de tratarlos como exactos en las comparaciones pueden llevar a problemas. Tenga en cuenta en particular que un valor `FLOAT` rara vez es igual a un valor `DOUBLE` .

Tipo de valor de bit

El tipo `BIT` es útil para almacenar valores de campo de bits. `BIT(M)` permite el almacenamiento de hasta valores de M-bit donde M está en el rango de `1 to 64`

También puede especificar valores con notación de `bit value` .

```
b'111'      -> 7
b'10000000' -> 128
```

A veces es útil usar 'shift' para construir un valor de un solo bit, por ejemplo `(1 << 7)` para 128.

El tamaño máximo combinado de todas las columnas BIT en una tabla `NDB` es 4096.

CHAR (n)

`CHAR(n)` es una cadena de una longitud *fija* de `n` *caracteres* . Si es un conjunto de `CHARACTER SET utf8mb4` , eso significa que ocupa exactamente `4*n` *bytes* , independientemente del texto que `CHARACTER SET utf8mb4` .

La mayoría de los casos de uso para `CHAR(n)` involucran cadenas que contienen caracteres en inglés, por lo tanto, debe ser `CHARACTER SET ascii` . (`latin1` hará tan bien).

```
country_code CHAR(2) CHARACTER SET ascii,
postal_code  CHAR(6) CHARACTER SET ascii,
uuid    CHAR(39) CHARACTER SET ascii,  -- more discussion elsewhere
```

FECHA, DATETIME, TIMESTAMP, AÑO, Y HORA

El tipo de datos `DATE` comprende la fecha pero no el componente de hora. Su formato es `'YYYY-MM-DD'` con un rango de '1000-01-01' a '9999-12-31'.

El tipo `DATETIME` incluye la hora con el formato 'YYYY-MM-DD HH: MM: SS'. Tiene un rango desde '1000-01-01 00:00:00' a '9999-12-31 23:59:59'.

El tipo `TIMESTAMP` es un tipo entero que comprende la fecha y la hora con un rango efectivo desde '1970-01-01 00:00:01' UTC hasta '2038-01-19 03:14:07' UTC.

El tipo `YEAR` representa un año y tiene un rango de 1901 a 2155.

El tipo de `TIME` representa una hora con un formato de 'HH: MM: SS' y tiene un rango de '-838: 59: 59' a '838: 59: 59'.

Requisitos de almacenamiento:

```
|-----------|--------------------|----------------------------------------|
| Data Type | Before MySQL 5.6.4 | as of MySQL 5.6.4                      |
|-----------|--------------------|----------------------------------------|
| YEAR      |      1 byte        |  1 byte                                |
| DATE      |      3 bytes       |  3 bytes                               |
| TIME      |      3 bytes       |  3 bytes + fractional seconds storage  |
| DATETIME  |      8 bytes       |  5 bytes + fractional seconds storage  |
| TIMESTAMP |      4 bytes       |  4 bytes + fractional seconds storage  |
|-----------|--------------------|----------------------------------------|
```

Segundos fraccionarios (a partir de la versión 5.6.4):

```
|------------------------------|------------------|
| Fractional Seconds Precision | Storage Required |
|------------------------------|------------------|
|              0               |      0 bytes     |
|              1,2             |      1 byte      |
|              3,4             |      2 byte      |
|              5,6             |      3 byte      |
|------------------------------|------------------|
```

Consulte los tipos de FECHA, FECHA, FECHA y TIMESTAMP de las páginas del manual de MySQL, los requisitos de almacenamiento del tipo de datos y los segundos fraccionarios en los valores de tiempo .

Lea Tipos de datos en línea: https://riptutorial.com/es/mysql/topic/4137/tipos-de-datos

Capítulo 67: Transacción

Examples

Iniciar Transacción

Una transacción es un grupo secuencial de sentencias SQL, como seleccionar, insertar, actualizar o eliminar, que se realiza como una sola unidad de trabajo.

En otras palabras, una transacción nunca se completará a menos que cada operación individual dentro del grupo tenga éxito. Si falla alguna operación dentro de la transacción, la transacción completa fallará.

La transacción bancaria será el mejor ejemplo para explicar esto. Considere una transferencia entre dos cuentas. Para lograr esto tienes que escribir sentencias SQL que hagan lo siguiente

1. Verifique la disponibilidad del monto solicitado en la primera cuenta.
2. Deducir cantidad solicitada de la primera cuenta
3. Depositarlo en segunda cuenta

Si alguien falla este proceso, el conjunto debe revertirse a su estado anterior.

ACID: Propiedades de las transacciones

Las transacciones tienen las siguientes cuatro propiedades estándar

- **Atomicidad:** asegura que todas las operaciones dentro de la unidad de trabajo se completen con éxito; de lo contrario, la transacción se cancela en el punto de falla y las operaciones anteriores se devuelven a su estado anterior.
- **Coherencia:** garantiza que la base de datos cambie correctamente los estados en una transacción confirmada con éxito.
- **Aislamiento:** permite que las transacciones operen independientemente y transparentes entre sí.
- **Durabilidad:** asegura que el resultado o efecto de una transacción confirmada persista en caso de una falla del sistema.

Las transacciones comienzan con la instrucción `START TRANSACTION` o `BEGIN WORK` y terminan con una `ROLLBACK COMMIT` o `ROLLBACK` . Los comandos SQL entre las instrucciones de inicio y finalización forman la mayor parte de la transacción.

```
START TRANSACTION;
SET @transAmt = '500';
SELECT @availableAmt:=ledgerAmt FROM accTable WHERE customerId=1 FOR UPDATE;
UPDATE accTable SET ledgerAmt=ledgerAmt-@transAmt WHERE customerId=1;
UPDATE accTable SET ledgerAmt=ledgerAmt+@transAmt WHERE customerId=2;
COMMIT;
```

Con `START TRANSACTION` , la confirmación automática permanece deshabilitada hasta que finalice la

transacción con `COMMIT` o `ROLLBACK` . El modo de confirmación automática vuelve a su estado anterior.

La `FOR UPDATE` indica (y bloquea) la (s) fila (s) durante la transacción.

Si bien la transacción permanece sin confirmar, esta transacción no estará disponible para otros usuarios.

Procedimientos generales involucrados en la transacción.

- Comience la transacción emitiendo el comando SQL `BEGIN WORK` o `START TRANSACTION` .
- Ejecute todas sus declaraciones SQL.
- Compruebe si todo se ejecuta de acuerdo a sus requerimientos.
- Si es así, emita el comando `COMMIT` ; de lo contrario, emita un comando `ROLLBACK` para revertir todo al estado anterior.
- Compruebe si hay errores incluso después de `COMMIT` si está utilizando, o podría usar, Galera / PXC.

COMPROMISO, ROLLBACK y AUTOCOMMIT

AUTOCOMIT

MySQL automáticamente confirma declaraciones que no son parte de una transacción. Los resultados de cualquier `UPDATE` , `DELETE` o `INSERT` no precedida por `BEGIN` o `START TRANSACTION` serán inmediatamente visibles para todas las conexiones.

La variable `AUTOCOMMIT` se establece como *verdadera* por defecto. Esto se puede cambiar de la siguiente manera,

```
--->To make autcommit false
SET AUTOCOMMIT=false;
--or
SET AUTOCOMMIT=0;

--->To make autcommit true
SET AUTOCOMMIT=true;
--or
SET AUTOCOMMIT=1;
```

Para ver `AUTOCOMMIT` estado de `AUTOCOMMIT`

```
SELECT @@autocommit;
```

COMETER

Si `AUTOCOMMIT` establece en falso y la transacción no se confirma, los cambios serán visibles solo para la conexión actual.

Después de que la instrucción `COMMIT` confirme los cambios en la tabla, el resultado será visible para todas las conexiones.

Consideramos dos conexiones para explicar esto.

Conexión 1

```
--->Before making autocommit false one row added in a new table
mysql> INSERT INTO testTable VALUES (1);

--->Making autocommit = false
mysql> SET autocommit=0;

mysql> INSERT INTO testTable VALUES (2), (3);
mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|   1 |
|   2 |
|   3 |
+-----+
```

Conexión 2

```
mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|   1 |
+-----+
---> Row inserted before autocommit=false only visible here
```

Conexión 1

```
mysql> COMMIT;
--->Now COMMIT is executed in connection 1
mysql> SELECT * FROM testTable;
    +-----+
    | tId |
    +-----+
    |   1 |
    |   2 |
    |   3 |
    +-----+
```

Conexión 2

```
mysql> SELECT * FROM testTable;
    +-----+
    | tId |
    +-----+
    |   1 |
    |   2 |
    |   3 |
    +-----+
--->Now all the three rows are visible here
```

RETROCEDER

Si algo salió mal en la ejecución de su consulta, `ROLLBACK` utiliza para revertir los cambios. Vea la explicación a continuación.

```
--->Before making autocommit false one row added in a new table
mysql> INSERT INTO testTable VALUES (1);

--->Making autocommit = false
mysql> SET autocommit=0;

mysql> INSERT INTO testTable VALUES (2), (3);
mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|   1 |
|   2 |
|   3 |
+-----+
```

Ahora estamos ejecutando `ROLLBACK`

```
--->Rollback executed now
mysql> ROLLBACk;

mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|   1 |
+-----+
--->Rollback removed all rows which all are not committed
```

Una vez que se ejecuta `COMMIT` , `ROLLBACK` no causará nada

```
mysql> INSERT INTO testTable VALUES (2), (3);
mysql> SELECT * FROM testTable;
mysql> COMMIT;
+-----+
| tId |
+-----+
|   1 |
|   2 |
|   3 |
+-----+

--->Rollback executed now
mysql> ROLLBACk;

mysql> SELECT * FROM testTable;
+-----+
| tId |
+-----+
|   1 |
|   2 |
|   3 |
+-----+
--->Rollback not removed any rows
```

Si `AUTOCOMMIT` se establece en *verdadero* , `COMMIT` y `ROLLBACK` son inútiles

Transacción utilizando el controlador JDBC

La transacción que utiliza el controlador JDBC se utiliza para controlar cómo y cuándo se debe confirmar y deshacer una transacción. La conexión al servidor MySQL se crea utilizando el controlador JDBC

El controlador JDBC para MySQL se puede descargar aquí

Comencemos por obtener una conexión a la base de datos utilizando el controlador JDBC

```
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection(DB_CONNECTION_URL,DB_USER,USER_PASSWORD);
--->Example for connection url "jdbc:mysql://localhost:3306/testDB");
```

Conjuntos de caracteres : Esto indica qué conjunto de caracteres utilizará el cliente para enviar declaraciones SQL al servidor. También especifica el conjunto de caracteres que el servidor debe usar para enviar los resultados al cliente.

Esto debe mencionarse al crear la conexión al servidor. Así que la cadena de conexión debe ser como,

```
jdbc:mysql://localhost:3306/testDB?useUnicode=true&amp;characterEncoding=utf8
```

Vea esto para obtener más detalles sobre conjuntos de caracteres y colaciones.

Cuando abre la conexión, el modo `AUTOCOMMIT` se establece en *verdadero* de forma predeterminada, que debe cambiarse como *falso* para iniciar la transacción.

```
con.setAutoCommit(false);
```

Siempre debe llamar `setAutoCommit()` método `setAutoCommit()` justo después de abrir una conexión.

De lo contrario, use `START TRANSACTION` o `BEGIN WORK` para iniciar una nueva transacción. Al utilizar `START TRANSACTION` o `BEGIN WORK` , no es necesario cambiar `AUTOCOMMIT` *false* . Eso se desactivará automáticamente.

Ahora puedes iniciar la transacción. Vea un ejemplo completo de transacción JDBC a continuación.

```
package jdbcTest;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;


public class accTrans {
```

```
    public static void doTransfer(double transAmount,int customerIdFrom,int customerIdTo) {

        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;

        try {
            String DB_CONNECTION_URL =
"jdbc:mysql://localhost:3306/testDB?useUnicode=true&amp;characterEncoding=utf8";

            Class.forName("com.mysql.jdbc.Driver");
            con = DriverManager.getConnection(DB_CONNECTION_URL,DB_USER,USER_PASSWORD);

            --->set auto commit to false
            con.setAutoCommit(false);
            ---> or use con.START TRANSACTION / con.BEGIN WORK

            --->Start SQL Statements for transaction
            --->Checking availability of amount
            double availableAmt    = 0;
            pstmt = con.prepareStatement("SELECT ledgerAmt FROM accTable WHERE customerId=?
FOR UPDATE");
            pstmt.setInt(1, customerIdFrom);
            rs = pstmt.executeQuery();
            if(rs.next())
                availableAmt    = rs.getDouble(1);

            if(availableAmt >= transAmount)
            {
                ---> Do Transfer
                ---> taking amount from cutomerIdFrom
                pstmt = con.prepareStatement("UPDATE accTable SET ledgerAmt=ledgerAmt-? WHERE
customerId=?");
                pstmt.setDouble(1, transAmount);
                pstmt.setInt(2, customerIdFrom);
                pstmt.executeUpdate();

                ---> depositing amount in cutomerIdTo
                pstmt = con.prepareStatement("UPDATE accTable SET ledgerAmt=ledgerAmt+? WHERE
customerId=?");
                pstmt.setDouble(1, transAmount);
                pstmt.setInt(2, customerIdTo);
                pstmt.executeUpdate();

                con.commit();
            }
            --->If you performed any insert,update or delete operations before
            ----> this availability check, then include this else part
            /*else { --->Rollback the transaction if availability is less than required
                con.rollback();
            }*/

        } catch (SQLException ex) {
            ---> Rollback the transaction in case of any error
            con.rollback();
        } finally {
            try {
                if(rs != null)  rs.close();
                if(pstmt != null) pstmt.close();
                if(con != null) con.close();
            }
```

```
        }
    }

    public static void main(String[] args) {
        doTransfer(500, 1020, 1021);
        -->doTransfer(transAmount, customerIdFrom, customerIdTo);
    }
}
```

La transacción JDBC se asegura de que todas las sentencias de SQL dentro de un bloque de transacción se ejecuten correctamente, si una de las sentencias de SQL dentro del bloque de transacción falla, anule y deshaga todo dentro del bloque de transacción.

Lea Transacción en línea: https://riptutorial.com/es/mysql/topic/5771/transaccion

Capítulo 68: Tratar con datos escasos o faltantes

Examples

Trabajar con columnas que contienen valores NULL

En MySQL y otros dialectos de SQL, los valores `NULL` tienen propiedades especiales.

Considere la siguiente tabla que contiene los solicitantes de empleo, las empresas para las que trabajaron y la fecha en que abandonaron la empresa. `NULL` indica que un solicitante todavía trabaja en la empresa:

```
CREATE TABLE example
(`applicant_id` INT, `company_name` VARCHAR(255), `end_date` DATE);

+--------------+-----------------+------------+
| applicant_id | company_name    | end_date   |
+--------------+-----------------+------------+
|            1 | Google          | NULL       |
|            1 | Initech         | 2013-01-31 |
|            2 | Woodworking.com | 2016-08-25 |
|            2 | NY Times        | 2013-11-10 |
|            3 | NFL.com         | 2014-04-13 |
+--------------+-----------------+------------+
```

Su tarea es la de generar una consulta que devuelve todas las filas después de `2016-01-01` , incluyendo cualquier empleado que todavía están trabajando en una empresa (los que tienen `NULL` fechas de finalización). Esta declaración de selección:

```
SELECT * FROM example WHERE end_date > '2016-01-01';
```

no incluye ninguna fila con valores `NULL` :

```
+--------------+-----------------+------------+
| applicant_id | company_name    | end_date   |
+--------------+-----------------+------------+
|            2 | Woodworking.com | 2016-08-25 |
+--------------+-----------------+------------+
```

Según la documentación de MySQL , las comparaciones con los operadores aritméticos <,>, = y <> devuelven `NULL` lugar de un valor booleano `TRUE` o `FALSE` . Por lo tanto, una fila con una fecha de `NULL` NULA no es mayor que 2016-01-01 ni menor que 2016-01-01.

Esto se puede resolver utilizando las palabras clave IS NULL:

```
SELECT * FROM example WHERE end_date > '2016-01-01' OR end_date IS NULL;
```

```
+--------------+-----------------+------------+
| applicant_id | company_name    | end_date   |
+--------------+-----------------+------------+
|            1 | Google          | NULL       |
|            2 | Woodworking.com | 2016-08-25 |
+--------------+-----------------+------------+
```

Trabajar con NULL se vuelve más complejo cuando la tarea involucra funciones de agregación como `MAX()` y una cláusula `GROUP BY` . Si su tarea fuera seleccionar la fecha de empleo más reciente para cada ID de solicitante, la siguiente consulta parecería un primer intento lógico:

```
SELECT applicant_id, MAX(end_date) FROM example GROUP BY applicant_id;

+--------------+---------------+
| applicant_id | MAX(end_date) |
+--------------+---------------+
|            1 | 2013-01-31    |
|            2 | 2016-08-25    |
|            3 | 2014-04-13    |
+--------------+---------------+
```

Sin embargo, sabiendo que `NULL` indica que un solicitante aún está empleado en una empresa, la primera fila del resultado es inexacta. El uso de `CASE WHEN` proporciona una solución para el problema `NULL` :

```
 SELECT
    applicant_id,
    CASE WHEN MAX(end_date is null) = 1 THEN 'present' ELSE MAX(end_date) END
    max_date
FROM example
GROUP BY applicant_id;

+--------------+------------+
| applicant_id | max_date   |
+--------------+------------+
|            1 | present    |
|            2 | 2016-08-25 |
|            3 | 2014-04-13 |
+--------------+------------+
```

Este resultado se puede unir de nuevo a la tabla de `example` original para determinar la compañía en la que trabajó el solicitante por última vez:

```
SELECT
  data.applicant_id,
  data.company_name,
  data.max_date
FROM (
  SELECT
    *,
    CASE WHEN end_date is null THEN 'present' ELSE end_date END max_date
  FROM example
) data
INNER JOIN (
 SELECT
   applicant_id,
```

```
   CASE WHEN MAX(end_date is null) = 1 THEN 'present' ELSE MAX(end_date) END max_date
 FROM
   example
 GROUP BY applicant_id
) j
ON data.applicant_id = j.applicant_id AND data.max_date = j.max_date;

+--------------+-----------------+------------+
| applicant_id | company_name    | max_date   |
+--------------+-----------------+------------+
|            1 | Google          | present    |
|            2 | Woodworking.com | 2016-08-25 |
|            3 | NFL.com         | 2014-04-13 |
+--------------+-----------------+------------+
```

Estos son solo algunos ejemplos de `NULL` trabajar con valores `NULL` en MySQL.

Lea Tratar con datos escasos o faltantes en línea:
https://riptutorial.com/es/mysql/topic/5866/tratar-con-datos-escasos-o-faltantes

Capítulo 69: UNE: Únete a la tabla 3 con el mismo nombre de ID.

Examples

Unir 3 tablas en una columna con el mismo nombre

```
CREATE TABLE Table1 (
    id INT UNSIGNED NOT NULL,
    created_on DATE NOT NULL,
    PRIMARY KEY (id)
)
CREATE TABLE Table2 (
    id INT UNSIGNED NOT NULL,
    personName VARCHAR(255) NOT NULL,
    PRIMARY KEY (id)
)
CREATE TABLE Table3 (
    id INT UNSIGNED NOT NULL,
    accountName VARCHAR(255) NOT NULL,
    PRIMARY KEY (id)
)
```

después de crear las tablas, podría hacer una consulta de selección para obtener el ID de las tres tablas que son iguales

```
SELECT
    t1.id AS table1Id,
    t2.id AS table2Id,
    t3.id AS table3Id
FROM Table1 t1
LEFT JOIN Table2 t2 ON t2.id = t1.id
LEFT JOIN Table3 t3 ON t3.id = t1.id
```

Lea UNE: Únete a la tabla 3 con el mismo nombre de ID. en línea: https://riptutorial.com/es/mysql/topic/9921/une--unete-a-la-tabla-3-con-el-mismo-nombre-de-id-

Capítulo 70: UNIÓN

Sintaxis

- UNION DISTINCT - Deducciones después de combinar los SELECTs
- UNION ALL - no dedup (más rápido)
- UNION - el valor predeterminado es DISTINCT
- SELECT ... UNION SELECT ... - está bien, pero es ambiguo en `ORDER BY`
- (SELECCIONAR ...) UNIÓN (SELECCIONAR ...) ORDENAR POR ... - resuelve la ambigüedad

Observaciones

UNION no usa múltiples CPUs.

UNION siempre * implica una tabla temporal para recopilar los resultados. * A partir de 5.7.3 / MariaDB 10.1, algunas formas de UNION entregan los resultados sin usar una tabla tmp (por lo tanto, más rápido).

Examples

Combinando sentencias SELECT con UNION

Puede combinar los resultados de dos consultas idénticamente estructuradas con la palabra clave `UNION` .

Por ejemplo, si desea una lista de toda la información de contacto de dos tablas separadas, `authors` y `editors` , por ejemplo, puede usar la palabra clave `UNION` manera:

```
select name, email, phone_number
from authors

union

select name, email, phone_number
from editors
```

Usar la `union` por sí misma eliminará los duplicados. Si necesita mantener duplicados en su consulta, puede usar la palabra clave `ALL` así: `UNION ALL` .

ORDEN POR

Si necesita ordenar los resultados de una UNION, use este patrón:

```
( SELECT ... )
UNION
```

```
( SELECT ... )
ORDER BY
```

Sin los paréntesis, el último ORDEN POR pertenecería al último SELECCIONAR.

Paginación via OFFSET

Al agregar un LÍMITE a una UNIÓN, este es el patrón a utilizar:

```
( SELECT ... ORDER BY x  LIMIT 10 )
UNION
( SELECT ... ORDER BY x  LIMIT 10 )
ORDER BY x  LIMIT 10
```

Como no puede predecir de qué SELECT (s) vendrá el "10", necesita obtener 10 de cada uno, luego reduzca la lista, repitiendo `ORDER BY` y `LIMIT` .

Para la 4ta página de 10 artículos, este patrón es necesario:

```
( SELECT ... ORDER BY x  LIMIT 40 )
UNION
( SELECT ... ORDER BY x  LIMIT 40 )
ORDER BY x  LIMIT 30, 10
```

Es decir, recopile el valor de 4 páginas en cada `SELECT` , luego haga la `OFFSET` en la `UNION` .

Combinando datos con diferentes columnas.

```
SELECT name, caption as title, year, pages FROM books
UNION
SELECT name, title, year, 0 as pages FROM movies
```

Al combinar 2 conjuntos de registros con diferentes columnas, emule los que faltan con los valores predeterminados.

UNION ALL Y UNION

SELECCIONA 1,22,44 UNION SELECCIONA 2,33,55

1	22	44
1	22	44
2	33	55

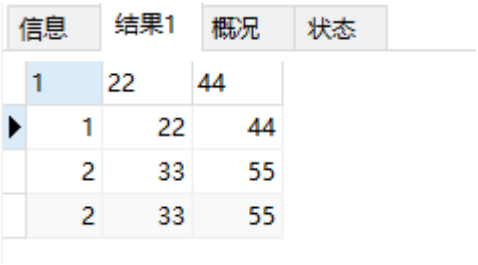
SELECCIONA 1,22,44 UNION SELECCIONA 2,33,55 UNION SELECCIONA 2,33,55

El resultado es el mismo que el de arriba.

usa UNION ALL

cuando

SELECCIONE 1,22,44 UNION SELECCIONE 2,33,55 UNION TODO SELECCIONE 2,33,55



1	22	44
1	22	44
2	33	55
2	33	55

Combinar y combinar datos en diferentes tablas de MySQL con las mismas columnas en filas únicas y ejecutar la consulta

Este ***UNION ALL*** combina datos de varias tablas y sirve como un alias de nombre de tabla para usar en sus consultas:

```
SELECT YEAR(date_time_column), MONTH(date_time_column), MIN(DATE(date_time_column)),
MAX(DATE(date_time_column)), COUNT(DISTINCT (ip)), COUNT(ip), (COUNT(ip) / COUNT(DISTINCT
(ip))) AS Ratio
FROM (
    (SELECT date_time_column, ip FROM server_log_1 WHERE state = 'action' AND log_id = 150)
UNION ALL
    (SELECT date_time_column, ip FROM server_log_2 WHERE state = 'action' AND log_id = 150)
UNION ALL
    (SELECT date_time_column, ip FROM server_log_3 WHERE state = 'action' AND log_id = 150)
UNION ALL
    (SELECT date_time_column, ip FROM server_log WHERE state = 'action' AND log_id = 150)
) AS table_all
GROUP BY YEAR(date_time_column), MONTH(date_time_column);
```

Lea UNIÓN en línea: https://riptutorial.com/es/mysql/topic/3847/union

Capítulo 71: Uno a muchos

Introducción

La idea de uno a muchos (1: M) se refiere a la unión de filas entre sí, específicamente los casos en que una sola fila en una tabla corresponde a muchas filas en otra.

1: M es unidireccional, es decir, cada vez que consulta una relación 1: M, puede usar la fila 'one' para seleccionar 'muchas' filas en otra tabla, pero no puede usar una sola fila 'many' para seleccione más de una sola fila 'uno'.

Observaciones

Para la mayoría de los casos, trabajar con una relación 1: M requiere que comprendamos *las claves principales* y *las claves externas* .

Una clave principal es una columna en una tabla donde cualquier fila de esa columna representa una sola entidad, o, al seleccionar un valor en una columna de clave primaria, se obtiene exactamente una fila. Usando los ejemplos anteriores, un EMP_ID representa a un solo empleado. Si consulta por un solo EMP_ID, verá una sola fila que representa al empleado correspondiente.

Una clave externa es una columna en una tabla que corresponde a la clave principal de otra tabla diferente. De nuestro ejemplo anterior, el MGR_ID en la tabla EMPLEADOS es una clave externa. En general, para unir dos tablas, las unirá en función de la clave principal de una tabla y la clave externa en otra.

Examples

Ejemplo de tablas de empresas

Considere una compañía en la que cada empleado que sea gerente, maneje 1 o más empleados y cada empleado tenga solo 1 gerente.

Esto da como resultado dos tablas:

EMPLEADOS

EMP_ID	NOMBRE DE PILA	APELLIDO	MGR_ID
E01	Johnny	Appleseed	M02
E02	Irlanda	Macklemore	M01
E03	Colby	Papeleo	M03

EMP_ID	NOMBRE DE PILA	APELLIDO	MGR_ID
E04	Ron	Sonswan	M01

Los gerentes

MGR_ID	NOMBRE DE PILA	APELLIDO
M01	Ruidoso	McQueen
M02	Mandón	Pantalones
M03	Barril	Jones

Haga que los empleados sean administrados por un solo gerente

```
SELECT e.emp_id , e.first_name , e.last_name FROM employees e INNER JOIN managers m ON m.mgr_id
= e.mgr_id WHERE m.mgr_id = 'M01' ;
```

Resultados en:

EMP_ID	NOMBRE DE PILA	APELLIDO
E02	Irlanda	Macklemore
E04	Ron	Sonswan

En última instancia, para cada administrador que consultemos, veremos que se devuelve 1 o más empleados.

Obtener el gerente para un solo empleado

Consulte las tablas de ejemplo anteriores al mirar este ejemplo.

```
SELECT m.mgr_id , m.first_name , m.last_name FROM managers m INNER JOIN employees e ON e.mgr_id
= m.mgr_id WHERE e.emp_id = 'E03' ;
```

MGR_ID	NOMBRE DE PILA	APELLIDO
M03	Barril	Jones

Como esto es lo inverso al ejemplo anterior, sabemos que para cada empleado que consultamos, solo veremos un gerente correspondiente.

Lea Uno a muchos en línea: https://riptutorial.com/es/mysql/topic/9600/uno-a-muchos

Capítulo 72: Usando variables

Examples

Variables de configuración

Aquí hay algunas maneras de establecer variables:

1. Puede configurar una variable para una cadena, número, fecha específica usando SET

   EX: SET @var_string = 'my_var'; SET @var_num = '2' SET @var_date = '2015-07-20';

2. puede configurar una variable para que sea el resultado de una instrucción de selección usando: =

   EX: Seleccione @var: = '123'; (Nota: debe usar: = cuando asigne una variable que no use la sintaxis SET, porque en otras declaraciones, (seleccione, actualice ...) se usa el "=" para comparar, por lo que cuando agregue dos puntos antes de " = ", estás diciendo" Esto no es una comparación, esto es un SET ".)

3. Puede configurar una variable para que sea el resultado de una instrucción de selección utilizando INTO

   (Esto fue particularmente útil cuando necesitaba elegir dinámicamente de qué particiones consultar)

EX: SET @start_date = '2015-07-20'; SET @end_date = '2016-01-31';

```
#this gets the year month value to use as the partition names
SET @start_yearmonth = (SELECT EXTRACT(YEAR_MONTH FROM @start_date));
SET @end_yearmonth = (SELECT EXTRACT(YEAR_MONTH FROM @end_date));

#put the partitions into a variable
SELECT GROUP_CONCAT(partition_name)
FROM information_schema.partitions p
WHERE table_name = 'partitioned_table'
AND SUBSTRING_INDEX(partition_name,'P',-1) BETWEEN @start_yearmonth AND @end_yearmonth
INTO @partitions;

#put the query in a variable. You need to do this, because mysql did not recognize my variable
as a variable in that position. You need to concat the value of the variable together with the
rest of the query and then execute it as a stmt.
SET @query =
CONCAT('CREATE TABLE part_of_partitioned_table (PRIMARY KEY(id))
SELECT partitioned_table.*
FROM partitioned_table PARTITION(', @partitions,')
JOIN users u USING(user_id)
WHERE date(partitioned_table.date) BETWEEN ', @start_date,' AND ', @end_date);

#prepare the statement from @query
PREPARE stmt FROM @query;
#drop table
```

```
DROP TABLE IF EXISTS tech.part_of_partitioned_table;
#create table using statement
EXECUTE stmt;
```

Número de fila y grupo utilizando variables en la instrucción Select

Digamos que tenemos una mesa `team_person` como abajo:

```
+======+===========+
| team |    person |
+======+===========+
|   A  |      John |
+------+-----------+
|   B  |     Smith |
+------+-----------+
|   A  |    Walter |
+------+-----------+
|   A  |     Louis |
+------+-----------+
|   C  | Elizabeth |
+------+-----------+
|   B  |     Wayne |
+------+-----------+


CREATE TABLE team_person AS SELECT 'A' team, 'John' person
UNION ALL  SELECT 'B' team,  'Smith' person
UNION ALL  SELECT 'A' team,  'Walter' person
UNION ALL  SELECT 'A' team,  'Louis' person
UNION ALL  SELECT 'C' team,  'Elizabeth' person
UNION ALL  SELECT 'B' team,  'Wayne' person;
```

Para seleccionar la tabla `team_person` con columna adicional `row_number` , ya sea

```
SELECT @row_no := @row_no+1 AS row_number, team, person
FROM team_person, (SELECT @row_no := 0) t;
```

O

```
SET @row_no := 0;
SELECT  @row_no := @row_no + 1 AS row_number, team, person
FROM team_person;
```

Saldrá el resultado a continuación:

```
+============+======+===========+
| row_number | team |    person |
+============+======+===========+
|          1 |   A  |      John |
+------------+------+-----------+
|          2 |   B  |     Smith |
+------------+------+-----------+
|          3 |   A  |    Walter |
+------------+------+-----------+
|          4 |   A  |     Louis |
```

```
+------------+------+-----------+
|          5 |   C  | Elizabeth |
+------------+------+-----------+
|          6 |   B  |     Wayne |
+------------+------+-----------+
```

Finalmente, si queremos obtener el grupo `row_number` por `team` columna

```
SELECT @row_no := IF(@prev_val = t.team, @row_no + 1, 1) AS row_number
   ,@prev_val := t.team AS team
   ,t.person
FROM team_person t,
  (SELECT @row_no := 0) x,
  (SELECT @prev_val := '') y
ORDER BY t.team ASC,t.person DESC;

+============+======+===========+
| row_number | team |    person |
+============+======+===========+
|          1 |   A  |    Walter |
+------------+------+-----------+
|          2 |   A  |     Louis |
+------------+------+-----------+
|          3 |   A  |      John |
+------------+------+-----------+
|          1 |   B  |     Wayne |
+------------+------+-----------+
|          2 |   B  |     Smith |
+------------+------+-----------+
|          1 |   C  | Elizabeth |
+------------+------+-----------+
```

Lea Usando variables en línea: https://riptutorial.com/es/mysql/topic/5013/usando-variables

Capítulo 73: VER

Sintaxis

- CREATE VIEW view_name AS SELECT column_name (s) FROM table_name WHERE condición; /// Sintaxis de crear simple vista
- CREAR [O REEMPLAZAR] [ALGORITMO = {NO DEFINIDO | MERGE | TEMPTABLE}] [DEFINER = {usuario | CURRENT_USER}] [SQL SECURITY {DEFINER | INVOKER}] VIEW view_name [(column_list)] AS select_statement [WITH [CASCADED | LOCAL] OPCIÓN DE VERIFICACIÓN]; /// Sintaxis de la vista de crear completa
- DROP VIEW [IF EXISTS] [db_name.] View_name; /// Sintaxis de vista de caída

Parámetros

Parámetros	Detalles
nombre_vista	Nombre de la vista
Instrucción SELECT	Sentencias SQL para ser empaquetadas en las vistas. Puede ser una instrucción SELECT para obtener datos de una o más tablas.

Observaciones

Las vistas son tablas virtuales y no contienen los datos que se devuelven. Pueden evitar que escriba consultas complejas una y otra vez.

- **Antes de realizar una vista,** su especificación consiste completamente en una instrucción `SELECT` . La instrucción `SELECT` no puede contener una subconsulta en la cláusula FROM.
- **Una vez que se hace una vista** , se usa en gran parte como una tabla y se puede `SELECT` desde una tabla.

Debe crear vistas, cuando desee restringir algunas columnas de su tabla, desde el otro usuario.

- Por ejemplo: en su organización, desea que sus gerentes vean poca información de una tabla llamada "Ventas", pero no desea que sus ingenieros de software puedan ver todos los campos de la tabla "Ventas". Aquí, puede crear dos vistas diferentes para sus gerentes y sus ingenieros de software.

Rendimiento `VIEWs` son azúcar sintáctica. Sin embargo, el rendimiento puede o no ser peor que la consulta equivalente con la selección de la vista plegada. El Optimizador intenta hacer esta "plegada" por usted, pero no siempre es exitoso. MySQL 5.7.6 proporciona algunas mejoras más en el Optimizador. Pero, independientemente, el uso de `VIEW` no generará una consulta *más rápida* .

Examples

Crear una vista

Privilegios

La sentencia CREATE VIEW requiere el privilegio CREATE VIEW para la vista y algún privilegio para cada columna seleccionada por la sentencia SELECT. Para las columnas utilizadas en otros lugares de la instrucción SELECT, debe tener el privilegio SELECT. Si la cláusula OR REPLACE está presente, también debe tener el privilegio DROP para la vista. CREATE VIEW también puede requerir el privilegio SUPER, según el valor DEFINER, como se describe más adelante en esta sección.

Cuando se hace referencia a una vista, se produce la comprobación de privilegios.

Una vista pertenece a una base de datos. De forma predeterminada, se crea una nueva vista en la base de datos predeterminada. Para crear la vista explícitamente en una base de datos determinada, use un nombre completo

Por ejemplo:

db_name.view_name

```
mysql> CREATE VIEW test.v AS SELECT * FROM t;
```

Nota: dentro de una base de datos, las tablas base y las vistas comparten el mismo espacio de nombres, por lo que una tabla base y una vista no pueden tener el mismo nombre.

Una vista puede:

- Ser creado a partir de muchos tipos de declaraciones SELECT
- Consulte las tablas base u otras vistas.
- usar uniones, UNION, y subconsultas
- SELECT no necesita referirse a ninguna tabla

Otro ejemplo

El siguiente ejemplo define una vista que selecciona dos columnas de otra tabla, así como una expresión calculada a partir de esas columnas:

```
mysql> CREATE TABLE t (qty INT, price INT);
mysql> INSERT INTO t VALUES(3, 50);
mysql> CREATE VIEW v AS SELECT qty, price, qty*price AS value FROM t;
mysql> SELECT * FROM v;
+------+-------+-------+
| qty  | price | value |
+------+-------+-------+
|    3 |    50 |   150 |
+------+-------+-------+
```

Restricciones

- Antes de MySQL 5.7.7, la instrucción SELECT no puede contener una subconsulta en la cláusula FROM.
- La instrucción SELECT no puede referirse a variables del sistema o variables definidas por el usuario.
- Dentro de un programa almacenado, la instrucción SELECT no puede referirse a parámetros de programa o variables locales.
- La instrucción SELECT no puede referirse a parámetros de declaración preparados.
- Cualquier tabla o vista referida en la definición debe existir. Una vez creada la vista, es posible eliminar una tabla o vista que
  la definición se refiere a En este caso, el uso de la vista produce un error. Para verificar una definición de vista para problemas de este tipo, use la sentencia CHECK TABLE.
- La definición no puede referirse a una tabla TEMPORAL, y usted no puede
  crear una vista temporal.
- No puede asociar un disparador con una vista.
- Los alias para los nombres de columna en la instrucción SELECT se comparan con la longitud máxima de columna de 64 caracteres (no el alias máximo)
  longitud de 256 caracteres).
- Una `VIEW` puede o no optimizarse, así como el `SELECT` equivalente. Es poco probable que se optimice mejor.

Una vista desde dos mesas.

Una vista es más útil cuando se puede utilizar para extraer datos de más de una tabla.

```
CREATE VIEW myview AS
SELECT a.*, b.extra_data FROM main_table a
LEFT OUTER JOIN other_table b
ON a.id = b.id
```

En mysql las vistas no se materializan. Si ahora realiza la consulta simple `SELECT * FROM myview` , mysql realmente realizará la `SELECT * FROM myview` IZQUIERDA detrás de la escena.

Una vista una vez creada se puede unir a otras vistas o tablas

Actualización de una tabla a través de una vista

A `VIEW` parece mucho a una mesa. Aunque puede `UPDATE` una tabla, es posible que no pueda actualizar una vista en esa tabla. En general, si el `SELECT` en la vista es lo suficientemente complejo como para requerir una tabla temporal, no se permite `UPDATE` .

Cosas como `GROUP BY` , `UNION` , `HAVING` , `DISTINCT` y algunas subconsultas impiden que la vista sea actualizable. Detalles en el *manual de referencia* .

DROPPING A VIEW

- Crear y soltar una vista en la base de datos actual.

```
CREATE VIEW few_rows_from_t1 AS SELECT * FROM t1 LIMIT 10;
DROP VIEW few_rows_from_t1;
```

- Cree y suelte una vista que haga referencia a una tabla en una base de datos diferente.

```
CREATE VIEW table_from_other_db AS SELECT x FROM db1.foo WHERE x IS NOT NULL;
DROP VIEW table_from_other_db;
```

Lea VER en línea: https://riptutorial.com/es/mysql/topic/1489/ver

Creditos

S. No	Capítulos	Contributors
1	Empezando con MySQL	A. Raza, Aman Dhanda, Andy, Athafoud, CodeWarrior, Community, Confiqure, Dipen Shah, e4c5, Epodax, Giacomo Garabello, greatwolf, inetphantom, JayRizzo, juergen d, Lahiru Ashan, Lambda Ninja, Magisch, Marek Skiba, Md. Nahiduzzaman Rose, moopet, msohng, Noah van der Aa, O. Jones, OverCoder, Panda, Parth Patel, rap-2-h, rhavendc, Romain Vincent, YCF_L
2	ACTUALIZAR	4thfloorstudios, Chris, Drew, Khurram, Ponnarasu, Rick James, Sevle
3	Administrador de MySQL	Florian Genser, Matas Vaitkevicius, RationalDev, Rick James
4	Agrupación	Drew, Rick James
5	Agrupar por	Adam, Filipe Martins, Lijo, Rick James, Thuta Aung, WAF, whrrgarbl
6	ALTERAR MESA	e4c5, JohnLBevan, kolunar, LiuYan , Matas Vaitkevicius, mayojava, Rick James, Steve Chambers, Thuta Aung, WAF, YCF_L
7	Archivos de registro	Drew, Rick James
8	Aritmética	Barranka, Dinidu, Drew, JonMark Perry, O. Jones, RamenChef, Richard Hamilton, Rick James
9	Backticks	Drew, SuperDJ
10	BORRAR	Batsu, Drew, e4c5, ForguesR, gabe3886, Khurram, Parth Patel, Ponnarasu, Rick James, strangeqargo, WAF, whrrgarbl, ypercube, Илья Плотников
11	Búsqueda de texto completo	O. Jones
12	Cambia la contraseña	e4c5, Hardik Kanjariya ツ, Rick James, Viktor, ydaetskcoR
13	Cliente MySQL	Batsu, Nathaniel Ford, Rick James

14	Códigos de error	Drew, e4c5, juergen d, Lucas Paolillo, O. Jones, Ponnarasu, Rick James, WAF, Wojciech Kazior
15	Comentar Mysql	Franck Dernoncourt, Rick James, WAF, YCF_L
16	Conectando con UTF-8 usando varios lenguajes de programación.	Epodax, Rick James
17	Configuración de la conexión SSL	4444, a coder, Eugene
18	Configuración y puesta a punto.	ChintaMoney, CodeWarrior, Epodax, Eugene, jan_kiran, Rick James
19	Conjuntos de caracteres y colaciones	frlan, Rick, Rick James
20	Consejos de rendimiento Mysql	arushi, RamenChef, Rick James, Rodrigo Darti da Costa
21	Consultas de pivote	Barranka
22	Conversión de MyISAM a InnoDB	Ponnarasu, Rick James, yukoff
23	Copia de seguridad utilizando mysqldump	agold, Asaph, Barranka, Batsu, KalenGi, Mark Amery, Matthew, mnoronha, Ponnarasu, RamenChef, Rick James , still_learning, strangeqargo, Sumit Gupta, Timothy, WAF
24	Creación de tablas	4444, Alex Shesterov, alex9311, andygeers, Aryo, Asaph, Barranka, Benvorth, Brad Larson, CPHPython, Darwin von Corax, Dinidu, Drew, fedorqui, HCarrasko, Jean Vitor, John M, Matt, Misa Lazovic, Panda, Parth Patel, Paulo Freitas, Přemysl Šťastný, Rick, Rick James, Ronnie Wang , Saroj Sasmal, Sebastian Brosch, skytreader, Stefan Rogin, Strawberry, Timothy, ultrajohn, user6655061, vijaykumar, Vini.g.fer, Vladimir Kovpak, WAF, YCF_L, Yury Fedorov
25	Creando bases de datos	Daniel Käfer, Drew, Ponnarasu, R.K123, Rick James, still_learning
26	Crear nuevo usuario	Aminadav, Batsu, Hardik Kanjariya ツ, josinalvo, Rick James, WAF
27	Datos de carga infile	aries12, Asaph, bhrached, CGritton, e4c5, RamenChef, Rick James, WAF
28	ENUM	Philipp, Rick James

29	Error 1055: ONLY_FULL_GROUP_BY: algo no está en la cláusula GROUP BY ...	Damian Yerrick, O. Jones
30	Eventos	Drew, rene
31	Expresiones regulares	user2314737, YCF_L
32	Extraer valores de tipo JSON	MohaMad
33	Gatillos	Blag, e4c5, Matas Vaitkevicius, ratchet, WAF, YCF_L
34	Índices y claves	Alex Recarey, Barranka, Ben Visness, Drew, kolunar, Rick James, Sanjeev kumar
35	información del servidor	FMashiro
36	INSERTAR	0x49D1, AbcAeffchen, Abubakkar, Aukhan, CGritton, Dinidu, Dreamer, Drew, e4c5, fnkr, gabe3886, Horen, Hugo Buff, Ian Kenney, Johan, Magisch, NEER, Parth Patel, Philipp, Rick James, Riho, strangeqargo, Thuta Aung, zeppelin
37	Instalar el contenedor Mysql con Docker-Compose	Marc Alff, molavec
38	JSON	A. Raza, Ben, Drew, e4c5, Manatax, Mark Amery, MohaMad, phatfingers, Rick James, sunkuet02
39	La optimización del rendimiento	e4c5, RamenChef, Rick James
40	Límite y compensación	Alvaro Flaño Larrondo, Ani Menon, animuson, ChaoticTwist, Chris Rasys, CPHPython, Ian Gregory, Matt S, Rick James, Sumit Gupta, WAF
41	Manejo de zonas horarias	O. Jones
42	Mesa plegable	Noah van der Aa, Parth Patel, Ponnarasu, R.K123, Rick James, trf, Tushar patel, YCF_L
43	Mesas temporales	Ponnarasu, Rick James
44	Motor myisam	Rick James
45	MySQL LOCK TABLE	Ponnarasu, Rick James, vijeeshin

46	MySQL Unions	Ani Menon, Rick James
47	mysqlimport	Batsu
48	NULO	Rick James, Sumit Gupta
49	Operaciones de cuerdas	Abubakkar, Batsu, juergen d, kolunar, Rick James, uruloke, WAF
50	Operaciones de fecha y hora	Abhishek Aggrawal, Drew, Matt S, O. Jones, Rick James, Sumit Gupta
51	ORDEN POR	Florian Genser, Rick James
52	Palabras reservadas	juergen d, user2314737
53	Particionamiento	Majid, Rick James
54	Personalizar PS1	Eugene, Wenzhong
55	Preparar declaraciones	kolunar, Rick James, winter
56	Recuperar de la contraseña de root perdida	BacLuc, Jen R
57	Recuperar y restablecer la contraseña de root predeterminada para MySQL 5.7+	Lahiru, ParthaSen
58	Replicación	Ponnarasu
59	Rutinas almacenadas (procedimientos y funciones)	Abhishek Aggrawal, Abubakkar, Darwin von Corax, Dinidu , Drew, e4c5, juergen d, kolunar, llanato, Rick James, userlond
60	Se une	Artisan72, Batsu, Benvorth, Bikash P, Drew, Matt, Philipp, Rick, Rick James, user3617558
61	Seguridad a través de GRANTs	Rick James
62	SELECCIONAR	Ani Menon, Asjad Athick, Benvorth, Bhavin Solanki, Chip, Drew, greatwolf, Inzimam Tariq IT, julienc, KartikKannapur , Kruti Patel, Matthis Kohli, O. Jones, Ponnarasu, Rick James, SeeuD1, ThisIsImpossible, timmyRS, YCF_L, ypercube
63	Tabla de mapeo de muchos a muchos	Rick James

64	Tabla dinámica de Un-Pivot usando una declaración preparada	rpd
65	Tiempo con precisión subsecundaria.	O. Jones
66	Tipos de datos	Batsu, dakab, Drew, Dylan Vander Berg, e4c5, juergen d, MohaMad, Richard Hamilton, Rick James
67	Transacción	Ponnarasu, Rick James
68	Tratar con datos escasos o faltantes	Batsu, Nate Vaughan
69	UNE: Únete a la tabla 3 con el mismo nombre de ID.	FMashiro
70	UNIÓN	Mattew Whitt, Rick James, Riho, Tarik, wangengzheng
71	Uno a muchos	falsefive
72	Usando variables	kolunar, user6655061
73	VER	Abhishek Aggrawal, Divya, e4c5, Marina K., Nikita Kurtin, Ponnarasu, R.K123, ratchet, Rick James, WAF, Yury Fedorov, Илья Плотников