MARIADB NIVEL I

UNEWEB

TABLA DE CONTENIDO

INTRODUCCIÓN

Existe una aplicación de base de datos separada de las principales colecciones de datos de aplicación y almacena. Cada base de datos emplea una o varias API's para la creación, el acceso, la gestión, la búsqueda y la replicación de los datos que contiene.

Las bases de datos también utilizan fuentes de datos no relacionales como objetos o archivos. Sin embargo, las bases de datos prueban la mejor opción para grandes conjuntos de datos, que sufren de recuperación lenta y la escritura con otras fuentes de datos.

Los sistemas relacionales de gestión de base de datos, o RDBMS, almacenar datos en diferentes tablas. Las relaciones entre estas tablas se establecen utilizando las claves principales y claves externas.

RDBMS ofrece las siguientes características -

- Que le permiten implementar un origen de datos con tablas, columnas e índices.
- Se aseguran la integridad de las referencias a través de filas de varias tablas.
- Se actualizan los índices automáticamente.
- Ellos interpretan las consultas y operaciones de SQL en la manipulación de los datos o de aprovisionamiento de las tablas.

RDBMS TERMINOLOGÍA

Antes de comenzar revisemos algunos términos relacionados con las bases de datos.

- **Database - Una base de datos es una fuente de datos que consiste en tablas que contienen datos relacionados.**
- **Table - Una tabla, es decir, una hoja de cálculo, es una matriz que contiene datos.**
- **Column - Una columna, es decir, elemento de datos, es una estructura que sostiene los datos de un tipo;** por ejemplo, las fechas de envío.
- **Row - Una fila es una estructura de agrupación de datos relacionados;** por ejemplo, datos para un cliente. También se conoce como una tupla, entrada o registro.
- **Redundancy - Este término se refiere al almacenamiento de datos dos veces con el fin de acelerar el sistema.**

- **Primary Key - Esto se refiere a un valor de identificación único.** Este valor no puede aparecer dos veces en una mesa, y sólo hay una fila asociada con él.
- **Foreign Key - Una clave externa sirve como un enlace entre dos tablas.**
- **Compound Key - Una clave de compuesto, o clave compuesta, es una clave que se refiere a varias columnas.** Se refiere a varias columnas, debido a una columna que carecen de una calidad única.
- **Index - Un índice es prácticamente idéntico al índice de un libro.**
- **Referential Integrity - Este término se refiere a asegurar que todos los valores de clave externa apuntan a filas existentes.**

BASE DE DATOS MARIADB

MariaDB es un manejador popular de MySQL creado por los desarrolladores originales de MySQL. Se desarrolló a partir de las preocupaciones relacionadas con la adquisición de MySQL por parte de Oracle. Ofrece soporte tanto para pequeñas tareas de procesamiento de datos y necesidades de la empresa. Su objetivo es ser una gota en el reemplazo para MySQL que sólo requiere una desinstalación sencilla de MySQL y una instalación de MariaDB. MariaDB ofrece las mismas características de MySQL y mucho más.

CARACTERÍSTICAS PRINCIPALES DE MARIADB

Las características importantes de MariaDB son -

- Todas MariaDB es bajo licencia GPL, LGPL, o BSD.
- MariaDB incluye una amplia selección de motores de almacenamiento, incluyendo los motores de almacenamiento de alto rendimiento, para trabajar con otras fuentes de datos de RDBMS.
- MariaDB utiliza un lenguaje de consulta estándar y popular.
- MariaDB se ejecuta en un número de sistemas operativos y soporta una amplia variedad de lenguajes de programación.
- MariaDB ofrece soporte para PHP, uno de los lenguajes de desarrollo web más populares.
- MariaDB ofrece la tecnología de clúster Galera.

- MariaDB también ofrece muchas operaciones y comandos disponible en MySQL, y elimina / reemplaza características afectar al rendimiento negativamente.

INSTALACIÓN MARIADB

Todas las descargas de MariaDB se encuentran en la Descarga sección del sitio web oficial de la Fundación MariaDB. Haga clic en el enlace a la versión que le gustaría, y una lista de descargas para varios sistemas operativos, arquitecturas y tipos de archivos de instalación se muestra.

INSTALACIÓN EN LINUX / UNIX

Si usted tiene un conocimiento profundo de los sistemas Linux / Unix, sólo tiene que descargar la fuente para construir su instalación. Nuestro método recomendado para instalar es utilizar paquetes de distribución. MariaDB ofrece paquetes para las siguientes distribuciones de Linux / Unix -

- RedHat / CentOS / Fedora
- Debian / Ubuntu

Las siguientes distribuciones incluyen un paquete MariaDB en sus repositorios -

- openSUSE
- arch Linux
- mageia
- menta
- Slackware

Siga estos pasos para instalar en un entorno Ubuntu -

Paso 1 - Inicio de sesión como usuario raíz.

Paso 2 - Vaya al directorio que contiene el paquete MariaDB.

Paso 3 - Importe la clave GnuPG firma con el siguiente código -

```
sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 0x
cbcb082a1bb943db
```

Paso 4 - Añadir MariaDB al sources.list archivo. Abra el archivo y añada el siguiente código -

```
sudo add-apt-repository 'deb http://ftp.osuosl.org/pub/mariadb/r
epo/5.5/ubuntuprecise main'
```

Paso 5 - Actualizar el sistema con el siguiente -

```
sudo apt-get update
```

Paso 6 - Instalar MariaDB con la siguiente -

```
sudo apt-get install mariadb-server
```

INSTALACIÓN EN WINDOWS

Después de localizar y descargar un archivo de instalación automatizada (MSI) , simplemente haga doble clic en el archivo para iniciar la instalación. El asistente de instalación le guiará a través de cada paso de la instalación y los ajustes necesarios.

Pruebe la instalación iniciándolo desde el símbolo del sistema. Vaya a la ubicación de la instalación, por lo general en el directorio, y escriba lo siguiente en el símbolo -

```
mysqld.exe --console
```

Si la instalación es correcta, verá los mensajes relacionados con la puesta en marcha. Si esto no aparece, es posible que tenga problemas de permisos. Asegúrese de que su cuenta de usuario puede acceder a la aplicación. Los clientes gráficos están disponibles para la administración MariaDB en el entorno Windows.

PRUEBA DE LA INSTALACIÓN

Realizar algunas tareas sencillas para confirmar el funcionamiento e instalación de MariaDB.

Usar la utilidad de Administrador para obtener el estatus del servidor

Ver la versión de servidor con el binario mysqladmin.

```
[root@host]# mysqladmin --version
```

Debe mostrar la versión, la distribución, el sistema operativo y la arquitectura. Si no ve la salida de ese tipo, examine su instalación para cuestiones.

Ejecutar comandos sencillos con un cliente

Abrir el símbolo del sistema para MariaDB. Esto debe conectarse a MariaDB y permitir la ejecución de comandos. Introduzca un simple comando de la siguiente manera -

```
mysql> SHOW DATABASES;
```

POSTERIOR A LA INSTALACIÓN

Después de la instalación exitosa de MariaDB, establecer una contraseña de root. Una nueva instalación tendrá una contraseña en blanco. Introduzca los siguientes datos para establecer la nueva contraseña -

```
mysqladmin -u root password "[enter your password here]";
```

Entre lo siguiente para conectarse al servidor con sus nuevas credenciales -

```
mysql -u root -p
Enter password:*******
```

ACTUALIZACIÓN EN WINDOWS

Si ya tiene instalado MySQL en el sistema Windows, y desea actualizar a MariaDB; no desinstale MySQL e instalar MariaDB. Esto provocará un conflicto con la base de datos existente. En su lugar, debe instalar MariaDB, y luego usar el Asistente de actualización en el archivo de instalación de Windows.

Las opciones de su fichero my.cnf MySQL deben trabajar con MariaDB. Sin embargo, MariaDB tiene muchas características que no se encuentran en MySQL.

Tenga en cuenta los siguientes conflictos en el archivo my.cnf -

- MariaDB utiliza el motor de almacenamiento Aria por defecto para los archivos temporales. Si usted tiene una gran cantidad de archivos temporales, modificar el tamaño del búfer de clave si no utiliza tablas MyISAM.
- Si las aplicaciones de conexión / desconexión con frecuencia, alterar el tamaño de caché hilo.

- Si utiliza más de 100 conexiones, utilice el grupo de subprocesos.

COMPATIBILIDAD

MySQL y MariaDB son esencialmente idénticas. Sin embargo, hay suficientes diferencias para crear problemas en la actualización.

ADMINISTRACIÓN MARIADB

Antes de intentar ejecutar MariaDB, en primer lugar determinar su estado actual, encendido o apagado. Hay tres opciones para iniciar y detener MariaDB:

- Mysqld ejecutar (the MariaDB binary) .
- Ejecutar el script de inicio mysqld_safe.
- Ejecutar el script de arranque mysql.server.

Si ha instalado MariaDB en una ubicación no estándar, puede que tenga que editar la información de ubicación de los archivos de script. Deja de MariaDB simplemente añadiendo un parámetro de “parada” con el guión.

Si desea iniciar automáticamente bajo Linux, añadir scripts de inicio a su **init de sistema.** Cada distribución tiene un procedimiento diferente. Consulte la documentación del sistema.

LA CREACIÓN DE UNA CUENTA DE USUARIO

Crear una nueva cuenta de usuario con el siguiente código -

```
'newusername'@'localhost' IDENTIFIED BY 'userpassword';
```

Este código agrega una fila a la tabla de usuario sin privilegios. También tiene la opción de utilizar un valor hash de la contraseña. Conceder los privilegios de usuario con el siguiente código -

```
GRANT SELECT, INSERT, UPDATE, DELETE ON database1 TO 'newusernam
e'@'localhost';
```

Otros privilegios incluyen casi todos los comandos u operación posible en MariaDB. Después de crear un usuario, ejecutar un comando “privilegios flush” con el fin de actualizar las tablas de permisos. Esto permite que la cuenta de usuario para ser utilizado.

EL ARCHIVO DE CONFIGURACIÓN

Después de una acumulación en Unix / Linux, el archivo de configuración "/etc/my.conf" debe ser editado a aparecer de la siguiente manera:

```
# Example mysql config file.
# You can copy this to one of:
# /etc/my.cnf to set global options,
# /mysql-data-dir/my.cnf to get server specific options or
# ~/my.cnf for user specific options.

#

# One can use all long options that the program supports.
# Run the program with --help to get a list of available options

# This will be passed to all mysql clients
[client]
#password = my_password
#port = 3306
#socket = /tmp/mysql.sock

# Here is entries for some specific programs
# The following values assume you have at least 32M ram

# The MySQL server
[mysqld]
#port = 3306
#socket = /tmp/mysql.sock
temp-pool

# The following three entries caused mysqld 10.0.1-MariaDB (and
possibly other
```

```
   versions) to abort...
# skip-locking
# set-variable = key_buffer = 16M
# set-variable = thread_cache = 4

loose-innodb_data_file_path = ibdata1:1000M
loose-mutex-deadlock-detector
gdb

######### Fix the two following paths

# Where you want to have your database
data = /path/to/data/dir

# Where you have your mysql/MariaDB source + sql/share/english
language = /path/to/src/dir/sql/share/english

[mysqldump]
quick
MariaDB
8
set-variable = max_allowed_packet=16M
[mysql]
no-auto-rehash

[myisamchk]
set-variable = key_buffer = 128M
```

Editar las líneas de datos “=” y “language =” para que coincida con su entorno.

Después de modificación del archivo, navegue hasta el directorio de origen y ejecutar el siguiente -

```
./scripts/mysql_install_db --srcdir = $PWD --datadir = /path/to/
data/dir --
   user = $LOGNAME
```

Omitir la variable "$ PWD" si se ha añadido datadir el archivo de configuración. Garantizar "$ LOGNAME" se utiliza cuando se ejecuta la versión 10.0.1 de MariaDB.

COMANDOS DE ADMINISTRACIÓN

Revisar la siguiente lista de comandos importantes que va a utilizar con regularidad cuando se trabaja con MariaDB:

- **USE [database name] - Establece la base de datos por defecto actual.**
- **SHOW DATABASES - Lista las bases de datos actualmente en el servidor.**
- **SHOW TABLES - Lista todas las tablas no temporales.**
- **SHOW COLUMNS FROM [table name] - Proporciona información de la columna correspondiente a la tabla especificada.**
- **SHOW INDEX FROM TABLENAME [table name] - Proporciona información de índice de la tabla relativa a la tabla especificada.**
- **SHOW TABLE STATUS LIKE [table name]\G – - Proporciona tablas con información acerca de las tablas no temporales, y el patrón que aparece después de la cláusula LIKE es usado para obtener los nombres de tabla.**

MARIADB SINTAXIS DE PHP

El administrador MariaDB se soporta bien con una amplia variedad de lenguajes de programación y los marcos tales como PHP, C #, JavaScript, Ruby on Rails, Django, y mucho más. PHP sigue siendo el más popular de todos los idiomas disponibles, debido a su simplicidad y la huella histórica. Esta guía se centrará en PHP en asociación con MariaDB.

PHP proporciona una selección de funciones para trabajar con la base de datos MySQL. Estas funciones realizan tareas como acceder a ella o la

realización de operaciones, y son totalmente compatibles con MariaDB. Basta con llamar a estas funciones se llamaba a cualquier otra función de PHP.

Las funciones de PHP que va a utilizar para MariaDB se ajustan al siguiente formato:

```
mysql_function(value,value,...) ;
```

La segunda parte de la función especifica su acción. Dos de las funciones utilizadas en esta guía son las siguientes -

```
mysqli_connect($connect) ;
mysqli_query($connect,"SQL statement") ;
```

El siguiente ejemplo muestra la sintaxis general de una llamada a una función PHP MariaDB -

```
<html>
   <head>
      <title>PHP and MariaDB</title>
   </head>

   <body>
      <?php
         $retval = mysql_function(value, [value,...]) ;

         if( !$retval ) {
            die ( "Error: Error message here" ) ;
         }
         // MariaDB or PHP Statements
      ?>
   </body>
</html>
```

En la siguiente sección, vamos a examinar las tareas esenciales MariaDB, utilizando las funciones de PHP.

CONEXIÓN MARIADB

Una forma de establecer una conexión con MariaDB consiste en utilizar el binario de MySQL en el símbolo del sistema.

MYSQL BINARIO

Consultar un ejemplo que figura a continuación.

```
[root@host]# mysql -u root -p


Enter password:******
```

El código proporcionado anteriormente se conecta a MariaDB y proporciona una línea de comandos para ejecutar comandos SQL. Después de introducir el código, un mensaje de bienvenida debería aparecer indicando la correcta conexión, con el número de versión que se muestra.

```
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 122323232
Server version: 5.5.40-MariaDB-log


Type 'help;' or '\h' for help. Type '\c' to clear the current in
put statement.
mysql>
```

El ejemplo utiliza el acceso de root, pero cualquier usuario con privilegios puede, por supuesto, tener acceso a la MariaDB y realizar operaciones.

Desconectarse de MariaDB a través de **exit de la ventana de comandos de la siguiente manera -**

```
mysql> exit
```

SCRIPT DE CONEXIÓN PHP

Otra forma de conectarse y desconectarse de MariaDB consiste en emplear un script PHP. PHP proporciona **mysql_connect(), la función para la apertura de una conexión de base de datos.** Se utiliza cinco parámetros opcionales, y devuelve un identificador de enlace MariaDB

después de una conexión exitosa, o una falsa sobre la conexión sin éxito. También proporciona **mysql_close(); la función para cerrar conexiones de base de datos, que utiliza un solo parámetro.**

Revisar la sintaxis siguiente secuencia de comandos de conexión de PHP -

```
connection mysql_connect(server,user,passwd,new_link,client_flag
) ;
```

La descripción de los parámetros es la siguiente –

S.NO	PARÁMETRO Y DESCRIPCIÓN
1	**Server** Este parámetro opcional especifica el nombre de host que ejecuta el servidor de base de datos. Su valor por defecto es "localhost: 0,3036."
2	**User** Este parámetro opcional especifica el nombre de usuario para acceder a la base de datos. Su valor por defecto es el propietario del servidor.
3	**Passwd** Este parámetro opcional especifica la contraseña del usuario. Su valor por defecto está en blanco.
4	**new_link** Este parámetro opcional especifica que en una segunda llamada a **mysql_connect() con idénticos argumentos, en lugar de una nueva conexión, se devuelve el identificador de la conexión actual.**
5	**client flags** Este parámetro opcional utiliza una combinación de los siguientes valores constantes - • MYSQL_CLIENT_SSL - Se utiliza el cifrado SSL. • MYSQL_CLIENT_COMPRESS - Utiliza el protocolo de compresión. • MYSQL_CLIENT_IGNORE_SPACE - Permite espacio después de los nombres de función. • MYSQL_CLIENT_INTERACTIVE - Permite segundos de tiempo de espera interactivos de inactividad antes de cerrar la conexión.

Revisar la sintaxis de script PHP desconexión se indica a continuación -

```
bool mysql_close ( resource $link_identifier ) ;
```

Si se omite el recurso, la conexión más reciente se cerrará. Se devuelve un valor de verdad en un cierre exitoso, o falsa.

Pruebe el siguiente código de ejemplo para conectar con un servidor MariaDB -

```
<html>
   <head>
      <title>Connect to MariaDB Server</title>
   </head>

   <body>
      <?php
         $dbhost = 'localhost:3036';
         $dbuser = 'guest1';
         $dbpass = 'guest1a';
         $conn = mysql_connect($dbhost, $dbuser, $dbpass) ;

         if(! $conn ) {
            die('Could not connect: ' . mysql_error() );
         }

         echo 'Connected successfully';
         mysql_close($conn) ;
      ?>
   </body>
</html>
```

En una conexión con éxito, se verá la siguiente salida -

```
mysql> Connected successfully
```

CREAR BASE DE DATOS MARIADB

Creación o eliminación de bases de datos en MariaDB requiere privilegios los cuales normalmente sólo se les da a los usuarios root o administradores. En estas cuentas, usted tiene dos opciones para crear una base de datos - el binario mysqladmin y un script PHP.

BINARIO MYSQLADMIN

El siguiente ejemplo muestra el uso del binario mysqladmin en la creación de una base de datos con los nombres de **Products -**

```
[root@host]# mysqladmin -u root -p create PRODUCTS
Enter password:******
```

PHP CREAR BASE DE DATOS DE SECUENCIAS DE COMANDOS

PHP emplea el **mysql_query función en la creación de una base de datos MariaDB.** La función utiliza dos parámetros, uno opcionales, y devuelve un valor de “true” cuando tiene éxito, o “falso” cuando no.

Revisar la siguiente script de sintaxis **create database:**

```
bool mysql_query( sql, connection ) ;
```

La descripción de los parámetros es la siguiente:

S.No	Parámetro y Descripción
1	**Sql** Este parámetro obligatorio consiste en la consulta SQL necesario para realizar la operación.
2	**connection** Cuando no se especifica, este parámetro opcional utiliza la conexión más reciente utilizado.

Pruebe el siguiente código de ejemplo para la creación de una base de datos:

```
<html>
   <head>
      <title>Create a MariaDB Database</title>
   </head>

   <body>
      <?php
         $dbhost = 'localhost:3036';
         $dbuser = 'root';
         $dbpass = 'rootpassword';
         $conn = mysql_connect($dbhost, $dbuser, $dbpass) ;

         if(! $conn ) {
            die('Could not connect: ' . mysql_error() );
         }

         echo 'Connected successfully<br />';
         $sql = 'CREATE DATABASE PRODUCTS';
         $retval = mysql_query( $sql, $conn ) ;

         if(! $retval ) {
            die('Could not create database: ' . mysql_error() );
         }

         echo "Database PRODUCTS created successfully\n";
         mysql_close($conn) ;
      ?>
   </body>
</html>
```

En la eliminación con éxito, verá la siguiente salida -

```
mysql> Database PRODUCTS created successfully
mysql> SHOW DATABASES;
+-----------------------+
| Database              |
+-----------------------+
| PRODUCTS              |
+-----------------------+
```

MARIADB SELECCIONAR BASE DE DATOS

Después de conectarse a MariaDB, debe seleccionar una base de datos de trabajar porque pueden existir muchas bases de datos. Hay dos formas de realizar esta tarea: desde la línea de comandos, o a través de un script PHP.

EL SÍMBOLO DEL SISTEMA

En la elección de una base de datos en el símbolo del sistema, basta con utilizar el comando SQL **'use' -**

```
[root@host]# mysql -u root -p

Enter password:******

mysql> use PRODUCTS;

Database changed

mysql> SELECT database() ;
+------------------------+
| Database               |
+------------------------+
| PRODUCTS               |
+------------------------+
```

Una vez que seleccione una base de datos, todos los comandos posteriores operarán en la base de datos elegida.

SELECCIONE LA BASE DE DATOS DE SECUENCIAS DE COMANDOS PHP

PHP proporciona la **mysql_select_db función para la selección de la base de datos.** La función utiliza dos parámetros, uno opcional, y devuelve un valor de "true" en la selección exitosa, en caso de fallo.

Revisar la sintaxis siguiente secuencia de comandos de selección de base de datos.

```
bool mysql_select_db( db_name, connection ) ;
```

La descripción de los parámetros es la siguiente -

S.No	Parámetro y Descripción
1	**db_name** Este parámetro obligatorio especifica el nombre de la base de datos para su uso.
2	**connection** Cuando no se especifica, este parámetro opcional utiliza la conexión más reciente utilizado.

Pruebe el siguiente código de ejemplo para seleccionar una base de datos -

```
<html>
   <head>
      <title>Select a MariaDB Database</title>
   </head>

   <body>
      <?php
         $dbhost = 'localhost:3036';
```

```
         $dbuser = 'guest1';
         $dbpass = 'guest1a';
         $conn = mysql_connect($dbhost, $dbuser, $dbpass) ;

         if(! $conn ) {
            die('Could not connect: ' . mysql_error() );
         }
         echo 'Connected successfully';

         mysql_select_db( 'PRODUCTS' ) ;
         mysql_close($conn) ;
      ?>
   </body>
</html>
```

En la selección exitosa, verá la siguiente salida -

```
mysql> Connected successfully
```

TIPOS DE DATOS MARIADB

Buenas definiciones de campo son esenciales para la optimización de su base de datos. El enfoque ideal requiere que se utilice exclusivamente un campo del tipo y tamaño necesario. Por ejemplo, si sólo va a utilizar un campo de cinco caracteres de ancho, no defina un campo, a 20 caracteres de ancho. El tipo de campo también se conoce como tipo de dato, dados los datos almacenados en el campo.

Los tipos de datos de MariaDB se pueden categorizar como valores de cadena numérica, fecha y hora, y.

TIPOS DE DATOS NUMÉRICOS

Los tipos de datos numéricos soportados por MariaDB son los siguientes -

- **TINYINT - Este tipo de dato representa enteros pequeños que caen dentro del rango firmado -128 y 127, y el rango sin signo de 0 a 255.**
- **BOOLEAN - Este tipo de datos asocia un valor 0 con "falsa", y un valor de 1 con "verdadero".**
- **SMALLINT - Este tipo de dato representa números enteros dentro de la gama firmado -32768 a 32768 y el rango sin signo de 0 a 65535.**
- **MEDIUMINT - Este tipo de dato representa enteros en el rango firmada de -8.388.608-8388607, y el rango de 0 a firmar 16777215.**
- **INT(also INTEGER) - Este tipo de datos representa un número entero de tamaño normal.** Cuando marcado como sin firmar, el alcance se extiende por 0 a 4294967295. Cuando firmado (the default setting) , el alcance se extiende por -2147483648 a 2147483647. Cuando una columna se establece en ZEROFILL(an unsigned state) , todos sus valores están precedidas por ceros a cabo M dígitos en el valor INT.
- **BIGINT - Este tipo de dato representa números enteros dentro de la gama firmada de 92233720368547758O8-9223372036854775807, y el rango de 0 a firmar 18446744073709551615.**
- **DECIMAL (también DEC, NUMERIC, FIJO) - Este tipo de datos representa números de punto fijo precisas, con M especificando sus dígitos y D especificando los dígitos después de la coma decimal.** Al valor M no añadir "-" o la coma decimal. Si D se establece en 0, no decimal o fracción parte aparece y el valor se redondea al decimal más cercano en INSERT. Los dígitos máxima permitida es de 65 y el máximo para los decimales es de 30. El valor por defecto para M en omisión es 10, y 0 para D en omisión.
- **FLOAT - Este tipo de datos representa un pequeño número, de punto flotante del valor 0 o un número dentro de los siguientes rangos -**
    - -3.402823466E + 38 a -1.175494351E-38
    - 1.175494351E-38 a 3.402823466E + 38
- **DOUBLE (also REAL and DOUBLE PRECISION) - Este tipo de datos representa de tamaño normal, números de punto flotante del valor 0 o dentro de las siguientes gamas -**
    - -1.7976931348623157E + 308 a -2.2250738585072014E-308
    - 2.2250738585072014E-308 a 1.7976931348623157E + 308

- **BIT - Este tipo de datos representa los campos de bits con M que especifica el número de bits por valor.** Por omisión de M, el valor predeterminado es 1. Los valores de bit se pueden aplicar con "b '[valor]'" en el que el valor representa un valor de bit en 0 y 1. Zero-padding se produce de forma automática desde la izquierda con longitud completa; por ejemplo, "10" se convierte en "0010."

FECHA Y HORA DE TIPOS DE DATOS

Los tipos de datos de fecha y hora con el apoyo de MariaDB son los siguientes -

- **DATE - Este tipo de dato representa un intervalo de fechas de "01/01/1000" a "9999-12-31", y utiliza el formato de fecha "AAAA-MM-DD".**
- **TIME - Este tipo de datos representa un intervalo de tiempo de "-838: 59: 59.999999" a "838: 59: 59.999999."**
- **DATETIME - Este tipo de datos representa el rango "01/01/1000 00: 00,000000: 00" a "9999-12-31 23: 59: 59.999999." Se utiliza el "AAAA-MM-DD HH: SS: MM" formato.**
- **TIMESTAMP - Este tipo de datos representa una marca de tiempo de la "AAAA-MM-DD HH: MM: DD" formato.** Se encuentra principalmente utilizar en detallar el momento de modificaciones de bases de datos, por ejemplo, inserción o actualización.
- **YEAR - Este tipo de dato representa un año en formato de 4 dígitos.** El formato de cuatro dígitos permite que los valores en el rango de 1901 a 2155 y 0000.

TIPOS DE DATOS CADENA

Los valores de tipo cadena apoyados por MariaDB son los siguientes -

- **String literals - Este tipo de dato representa secuencias de caracteres entre comillas.**
- **CHAR - Este tipo de datos representa una cadena derecha acolchado, de longitud fija que contiene espacios de longitud especificada.** M representa longitud de la columna de caracteres en un rango de 0 a 255, el valor predeterminado es 1.
- **VARCHAR - Este tipo de datos representa una cadena de longitud variable, con un rango de M (maximum column length) de 0 a 65535.**

- **BINARY - Este tipo de datos representa cadenas de bytes binarios, con M como la longitud de la columna en bytes.**
- **VARBINARY - Este tipo de datos representa cadenas de bytes binarios de longitud variable, con M como longitud de la columna.**
- **TINYBLOB - Este tipo de datos representa una columna blob con una longitud máxima de 255 (28 - 1) bytes.** En el almacenamiento, cada uno utiliza una longitud de prefijo de un byte que indica la cantidad de bytes en el valor.
- **BLOB - Este tipo de datos representa una columna blob con una longitud máxima de 65 535 (216 - 1) bytes.** En el almacenamiento, cada uno utiliza una longitud de prefijo de dos bytes que indica la cantidad de bytes en el valor.
- **MEDIUMBLOB - Este tipo de datos representa una columna blob con una longitud máxima de 16.777.215 (22^4 - 1) bytes.** En el almacenamiento, cada uno utiliza una longitud de prefijo de tres bytes que indica la cantidad de bytes en el valor.
- **LONGBLOB - Este tipo de datos representa una columna blob con una longitud máxima de 4,294,967,295(2^{32} - 1) bytes.** En el almacenamiento, cada uno utiliza una longitud de prefijo de cuatro bytes que indica la cantidad de bytes en el valor.
- **TINYTEXT - Este tipo de datos representa una columna de texto con una longitud máxima de 255 (2^8 - 1) caracteres.** En el almacenamiento, cada uno utiliza una longitud de prefijo de un byte que indica la cantidad de bytes en el valor.
- **TEXT - Este tipo de datos representa una columna de texto con una longitud máxima de 65 535 (2^{16} - 1) caracteres.** En el almacenamiento, cada uno utiliza una longitud de prefijo de dos bytes que indica la cantidad de bytes en el valor.
- **MEDIUMTEXT - Este tipo de datos representa una columna de texto con una longitud máxima de 16.777.215 (2^{24} - 1) caracteres.** En el almacenamiento, cada uno utiliza una longitud de prefijo de tres bytes que indica la cantidad de bytes en el valor.
- **LONGTEXT - Este tipo de datos representa una columna de texto con una longitud máxima de 4294967295 o 4 GB (2^{32} - 1) caracteres.** En el almacenamiento, cada uno utiliza una longitud de prefijo de cuatro bytes que indica la cantidad de bytes en el valor.
- **ENUM - Este tipo de dato representa un objeto de cadena que tiene un único valor de una lista.**

- **SET - Este tipo de datos representa un objeto de cadena que tiene cero o más valores de una lista, con un máximo de 64 miembros.** valores de SET presentes internamente como valores enteros.

MARIADB CREAR TABLAS

En esta sección vamos a aprender cómo crear tablas. Antes de crear una tabla, determinar primero su nombre, nombres de los campos, y las definiciones de campo.

La siguiente es la sintaxis general para la creación de la tabl:

```
CREATE TABLE table_name (column_name column_type);
```

Revisar el comando se aplica a la creación de una tabla en la base de datos PRODUCTOS -

```
databaseproducts_ tbl(
   product_id INT NOT NULL AUTO_INCREMENT,
   product_name VARCHAR(100) NOT NULL,
   product_manufacturer VARCHAR(40) NOT NULL,
   submission_date DATE,
   PRIMARY KEY ( product_id )
);
```

El ejemplo anterior utiliza “NOT NULL” como un atributo de campo para evitar los errores causados por un valor nulo. El atributo “AUTO_INCREMENT” instruye a MariaDB para agregar el siguiente valor disponible para el campo ID. La clave principal de palabras clave define una columna como la **primary key.** Múltiples columnas separadas por comas pueden definir una clave principal.

Los dos métodos principales para la creación de tablas están utilizando la línea de comandos y un script PHP.

EL SÍMBOLO DEL SISTEMA

Utilizar el comando CREATE TABLE para realizar la tarea, como se muestra a continuación -

```
root@host# mysql -u root -p
```

```
Enter password:*******
mysql> use PRODUCTS;
Database changed
mysql> CREATE TABLE products_tbl(
   -> product_id INT NOT NULL AUTO_INCREMENT,
   -> product_name VARCHAR(100) NOT NULL,
   -> product_manufacturer VARCHAR(40) NOT NULL,
   -> submission_date DATE,
   -> PRIMARY KEY ( product_id )
   -> );
mysql> SHOW TABLES;
+------------------------+
| PRODUCTS               |
+------------------------+
| products_tbl           |
+------------------------+
```

Asegurarse que todos los comandos se terminan con un punto y coma.

PHP CREAR TABLA DE SECUENCIAS DE COMANDOS

PHP proporciona **mysql_query() para la creación de la tabla.** Su segundo argumento contiene el comando necesario SQL -

```
<html>
   <head>
      <title>Create a MariaDB Table</title>
   </head>

   <body>
      <?php
         $dbhost = 'localhost:3036';
         $dbuser = 'root';
```

```
         $dbpass = 'rootpassword';
         $conn = mysql_connect($dbhost, $dbuser, $dbpass) ;

         if(! $conn ) {
            die('Could not connect: ' . mysql_error() );
         }
         echo 'Connected successfully<br />';

         $sql = "CREATE TABLE products_tbl( " .
            "product_id INT NOT NULL AUTO_INCREMENT, " .
            "product_name VARCHAR(100) NOT NULL, ".
            "product_manufacturer VARCHAR(40) NOT NULL, ".
            "submission_date DATE, " .
            "PRIMARY KEY ( product_id ) ); ";

         mysql_select_db( 'PRODUCTS' ) ;
         $retval = mysql_query( $sql, $conn ) ;

         if(! $retval ) {
            die('Could not create table: ' . mysql_error() );
         }
         echo "Table created successfully\n";

         mysql_close($conn) ;
      ?>
   </body>
</html>
```

Si la creación creación de la tabla es exitosa, verá la siguiente salida –

```
mysql> Table created successfully
```

TABLAS MARIADB ELIMINAR TABLAS

En esta sección, vamos a aprender a eliminar tablas. La eliminación de tablas es muy fácil, pero recuerde que todas las tablas borradas son irrecuperables. La sintaxis general para eliminación tabla es la siguiente -

```
DROP TABLE table_name ;
```

Existen dos opciones para realizar una eliminación de tabla: utilizar la línea de comandos o un script PHP.

EL SÍMBOLO DEL SISTEMA

En el símbolo del sistema, basta con utilizar el comando SQL **DROP TABLE:**

```
root@host# mysql -u root -p
Enter password:*******
mysql> use PRODUCTS;
Database changed
mysql> DROP TABLE products_tbl

mysql> SELECT * from products_tbl
ERROR 1146 (42S02) : Table 'products_tbl' doesn't exist
```

SCRIPT PHP TABLA GOTA

PHP proporciona **mysql_query() para eliminar tablas.** Basta con pasar su segundo argumento en el comando adecuado de SQL:

```
<html>
   <head>
      <title>Create a MariaDB Table</title>
   </head>

   <body>
      <?php
         $dbhost = 'localhost:3036';
         $dbuser = 'root';
         $dbpass = 'rootpassword';
         $conn = mysql_connect($dbhost, $dbuser, $dbpass) ;

         if(! $conn ) {
            die('Could not connect: ' . mysql_error() );
         }
         echo 'Connected successfully<br />';

         $sql = "DROP TABLE products_tbl";
         mysql_select_db( 'PRODUCTS' ) ;
         $retval = mysql_query( $sql, $conn ) ;

         if(! $retval ) {
            die('Could not delete table: ' . mysql_error() );
         }
         echo "Table deleted successfully\n";

         mysql_close($conn) ;
      ?>
   </body>
```

```
</html>
```

En la tabla de eliminada con éxito, verá la siguiente salida -

```
mysql> Table deleted successfully
```

MARIADB INSERTAR CONSULTA

En este capítulo, vamos a aprender cómo insertar datos en una tabla. Insertar datos en una tabla requiere el comando INSERT. La sintaxis general del comando INSERT es seguido por el nombre de tabla, campos y valores.

Revisar su sintaxis general a continuación:

```
INSERT INTO tablename (field,field2,...) VALUES (value, value2,.
..) ;
```

La declaración requiere el uso de comillas simples o dobles para los valores de cadena. Otras opciones para el comunicado incluyen declaraciones "INSERT ... SET", "INSERT ... SELECT" declaraciones, y varias otras opciones.

Note - Los VALUES() la función que aparece en la declaración, sólo se aplica a las declaraciones INSERT y devuelve NULL si se utiliza en otros lugares.

Existen dos opciones para realizar la operación: utilizar la línea de comandos o utilizar un script PHP.

EL SÍMBOLO DEL SISTEMA

En el indicador, hay muchas formas de realizar una operación de selección. Una declaración de rutina se da a continuación –

```
belowmysql>
INSERT INTO products_tbl (ID_number, Nomenclature) VALUES (12345
,
"Orbitron 4000");
mysql> SHOW COLUMNS FROM products_tbl;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
```

```
+-------------+-------------+------+-----+---------+-------+
| ID_number   | int(5)      |      |     |         |       |
| Nomenclature| char(13)    |      |     |         |       |
+-------------+-------------+------+-----+---------+-------+
```

Puede insertar varias filas -

```
INSERT INTO products VALUES (1, “first row”) , (2, “second row”)
 ;
```

También se puede emplear la cláusula SET -

```
INSERT INTO products SELECT * FROM inventory WHERE status = 'ava
ilable';
```

SCRIPT PHP INSERCIÓN

Emplear el mismo “INSERT INTO …” dentro de una función PHP para realizar la operación. Utilizar el **mysql_query() la función una vez más.**

Revisar el ejemplo dado a continuación -

```
<?php
   if(isset($_POST['add']) ) {
      $dbhost = 'localhost:3036';
      $dbuser = 'root';
      $dbpass = 'rootpassword';
      $conn = mysql_connect($dbhost, $dbuser, $dbpass) ;

      if(! $conn ) {
         die('Could not connect: ' . mysql_error() );
      }

      if(! get_magic_quotes_gpc() ) {
         $product_name = addslashes ($_POST['product_name']) ;
```

```
         $product_manufacturer = addslashes ($_POST['product_nam
e']) ;
      } else {
         $product_name = $_POST['product_name'];
         $product_manufacturer = $_POST['product_manufacturer'];
      }
      $ship_date = $_POST['ship_date'];
      $sql = "INSERT INTO products_tbl " .
         "(product_name,product_manufacturer, ship_date) ".
         "VALUES"."('$product_name','$product_manufacturer','$sh
ip_date') ";

      mysql_select_db('PRODUCTS') ;
      $retval = mysql_query( $sql, $conn ) ;

      if(! $retval ) {
         die('Could not enter data: ' . mysql_error() );
      }

      echo "Entered data successfully\n";
      mysql_close($conn) ;
   }
?>
```

En la inserción exitosa de datos, verá la siguiente salida -

```
mysql> Entered data successfully
```

MARIADB CONSULTA DE SELECCIÓN

En este capítulo, vamos a aprender cómo seleccionar datos de una tabla.

Las instrucciones SELECT para recuperar las filas seleccionadas pueden incluir declaraciones de unión, una cláusula de ordenación, una cláusula LIMIT, una cláusula WHERE, GROUP BY ... cláusula HAVING, y subconsultas.

Revisar la siguiente sintaxis general -

```
SELECT field, field2,... FROM table_name, table_name2,... WHERE...
```

Una instrucción SELECT ofrece múltiples opciones para especificar la tabla utilizada -

- database_name.table_name
- nombre_tabla.nombre_columna
- database_name.table_name.column_name

Todas las declaraciones SELECT debe contener uno o más **expresiones select.** Las expresiones SELECT consisten en lo siguiente:

- Un nombre de columna.
- Una expresión que emplea operadores y funciones.
- La especificación "nombre_tabla. *" Para seleccionar todas las columnas de la tabla dada.
- El carácter "*" para seleccionar todas las columnas de todas las tablas especificadas en la cláusula FROM.

La línea de comandos o un script PHP se pueden emplear en la ejecución de una instrucción de selección.

EL SÍMBOLO DEL SISTEMA

En el símbolo del sistema, ejecutar sentencias de la siguiente manera -

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use PRODUCTS;
Database changed
mysql> SELECT * from products_tbl
+-------------+---------------+
| ID_number   | Nomenclature  |
+-------------+---------------+
| 12345       | Orbitron 4000 |
+-------------+---------------+
```

SELECCIONAR PHP SCRIPT

Emplear la misma sentencia SELECT dentro de una función PHP para realizar la operación. Utilizar el **mysql_query()** una vez más**.** Revisar el ejemplo dado a continuación -

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass) ;

   if(! $conn ) {
      die('Could not connect: ' . mysql_error() );
   }

   $sql = 'SELECT product_id, product_name,product_manufacturer,
 ship_date
      FROM products_tbl';

   mysql_select_db('PRODUCTS') ;
   $retval = mysql_query( $sql, $conn ) ;

   if(! $retval ) {
      die('Could not get data: ' . mysql_error() );
   }

   while($row = mysql_fetch_array($retval, MYSQL_ASSOC) ) {
      echo "Product ID :{$row['product_id']} <br> ".
         "Name: {$row['product_name']} <br> ".
         "Manufacturer: {$row['product_manufacturer']} <br> ".
         "Ship Date : {$row['ship_date']} <br>".
         "--------------------------------<br>";
   }
```

```
   echo "Fetched data successfully\n";
   mysql_close($conn) ;
?>
```

En el éxito de recuperación de datos, verá la siguiente salida -

```
Product ID: 12345
Nomenclature: Orbitron 4000
Manufacturer: XYZ Corp
Ship Date: 01/01/17
------------------------------------------------
Product ID: 12346
Nomenclature: Orbitron 3000
Manufacturer: XYZ Corp
Ship Date: 01/02/17
------------------------------------------------
mysql> Fetched data successfully
```

Las mejores prácticas sugieren que liberar la memorias del cursor después de cada instrucción SELECT. PHP proporciona la **mysql_free_result().** Revisar su uso como se muestra a continuación:

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass) ;

   if(! $conn ) {
      die('Could not connect: ' . mysql_error() );
   }

   $sql = 'SELECT product_id, product_name, product_manufacturer
, ship_date
```

```
      FROM products_tbl';

   mysql_select_db('PRODUCTS') ;
   $retval = mysql_query( $sql, $conn ) ;

   if(! $retval ) {
      die('Could not get data: ' . mysql_error() );
   }

   while($row = mysql_fetch_array($retval, MYSQL_NUM) ) {
      echo "Product ID :{$row[0]} <br> ".
         "Name: {$row[1]} <br> ".
         "Manufacturer: {$row[2]} <br> ".
         "Ship Date : {$row[3]} <br> ".
         "--------------------------------<br>";
   }

   mysql_free_result($retval) ;
   echo "Fetched data successfully\n";
   mysql_close($conn) ;
?>
```

CLÁUSULA WHERE MARIADB

La cláusula WHERE filtra varias declaraciones como SELECT, UPDATE, DELETE e INSERT. Presentan criterios utilizados para especificar la acción. Por lo general aparecen después de un nombre de tabla en un comunicado, y su estado sigue. La cláusula WHERE funciona esencialmente como una sentencia if.

Revisar la sintaxis general de la cláusula WHERE se indican a continuación -

```
[COMMAND] field,field2,... FROM table_name,table_name2,... WHERE
 [CONDITION]
```

Tenga en cuenta las siguientes cualidades de la cláusula WHERE -

- Es opcional.
- Permite a cualquier condición que se determine.
- Permite para la especificación de condiciones múltiples mediante el uso de un operador AND u OR.
- mayúsculas y minúsculas sólo se aplica a los estados usando LIKE comparaciones.

La cláusula WHERE permite el uso de los siguientes operadores:

Operador
=! =
> <
> = <=

La cláusula WHERE puede ser utilizada en el símbolo del sistema o dentro de un script PHP.

EL SÍMBOLO DEL SISTEMA

En el símbolo del sistema, basta con utilizar un comando estándar -

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use PRODUCTS;
Database changed
mysql> SELECT * from products_tbl WHERE product_manufacturer = '
XYZ Corp';
+-------------+----------------+----------------------+
| ID_number   | Nomenclature   | product_manufacturer |
+-------------+----------------+----------------------+
| 12345       | Orbitron 4000  | XYZ Corp             |
+-------------+----------------+----------------------+
| 12346       | Orbitron 3000  | XYZ Corp             |
+-------------+----------------+----------------------+
| 12347       | Orbitron 1000  | XYZ Corp             |
```

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

Consultar un ejemplo utilizando el estado **AND:**

```
SELECT *
FROM products_tbl
WHERE product_name = 'Bun Janshu 3000';
AND product_id <= 344;
```

Este ejemplo combina tanto AND y OR

```
SELECT *
FROM products_tbl
WHERE (product_name = 'Bun Janshu 3000' AND product_id < 344)
OR (product_name = 'Bun Janshu 3000') ;
```

LOS SCRIPTS PHP USANDO CLÁSULA WHERE

Emplear el **mysql_query() la función en las operaciones utilizando una cláusula WHERE -**

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass) ;

   if(! $conn ) {
      die('Could not connect: ' . mysql_error() );
   }

   $sql = 'SELECT product_id, product_name, product_manufacturer
, ship_date
      FROM products_tbl
      WHERE product_manufacturer = "XYZ Corp"';
```

```
   mysql_select_db('PRODUCTS') ;
   $retval = mysql_query( $sql, $conn ) ;

   if(! $retval ) {
      die('Could not get data: ' . mysql_error() );
   }

   while($row = mysql_fetch_array($retval, MYSQL_ASSOC) ) {
      echo "Product ID :{$row['product_id']} <br> ".
         "Name: {$row['product_name']} <br> ".
         "Manufacturer: {$row['product_manufacturer']} <br> ".
         "Ship Date: {$row['ship_date']} <br> ".
         "--------------------------------<br>";
   }

   echo "Fetched data successfully\n";
   mysql_close($conn) ;
?>
```

En el éxito de recuperación de datos, verá la siguiente salida -

```
Identificación del producto: 12345
Nomenclatura: Orbitron 4000
Fabricante: XYZ Corp
Fecha de envío: 01/01/17
------------------------------------------------
Identificación del producto: 12346
Nomenclatura: Orbitron 3000
Fabricante: XYZ Corp
Fecha de envío: 02/01/17
------------------------------------------------
```

```
Identificación del producto: 12347
Nomenclatura: Orbitron 1000
Fabricante: XYZ Corp
Fecha de envío: 02/01/17
-----------------------------------------------
mysql> datos obtenidos con éxito
```

MARIADB CONSULTA DE ACTUALIZACIÓN

El comando **UPDATE modifica campos existente cambiando los valores.** Se utiliza la cláusula SET para especificar columnas de modificación, y para especificar los nuevos valores asignados. Estos valores pueden ser o bien una expresión o el valor por defecto del campo. Establecer un valor por defecto requiere el uso de la palabra clave DEFAULT. El comando también se puede emplear una cláusula WHERE para especificar las condiciones para una actualización y / o una cláusula ORDER BY para actualizar en un cierto orden.

Revisar la siguiente sintaxis general –

```
UPDATE table_name SET field=new_value, field2=new_value2,...
[WHERE ...]
```

Ejecutar un comando UPDATE, ya sea del símbolo del sistema o utilizando un script PHP.

EL SÍMBOLO DEL SISTEMA

En el símbolo del sistema, basta con utilizar un estándar commandroot -

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use PRODUCTS;
Database changed
mysql> UPDATE products_tbl
   SET nomenclature = 'Fiber Blaster 300Z'
      WHERE ID_number = 112;
mysql> SELECT * from products_tbl WHERE ID_number='112';
```

```
+-------------+---------------------+----------------------+
| ID_number   | Nomenclature        | product_manufacturer |
+-------------+---------------------+----------------------+
| 112         | Fiber Blaster 300Z  | XYZ Corp             |
+-------------+---------------------+----------------------+
```

LA SECUENCIA DE ACTUALIZACIÓN DE CONSULTAS DE PHP

Emplear el **mysql_query()en las sentencias de comandos ACTUALIZACIÓN -**

```
<?php
   $dbhost = ‘localhost:3036’;
   $dbuser = ‘root’;
   $dbpass = ‘rootpassword’;
   $conn = mysql_connect($dbhost, $dbuser, $dbpass) ;

   if(! $conn ) {
      die(‘Could not connect: ‘ . mysql_error() );
   }

   $sql = ‘UPDATE products_tbl
      SET product_name = ”Fiber Blaster 300z”
      WHERE product_id = 112’;

   mysql_select_db(‘PRODUCTS’) ;
   $retval = mysql_query( $sql, $conn ) ;

   if(! $retval ) {
      die(‘Could not update data: ‘ . mysql_error() );
   }
```

```
   echo “Updated data successfully\n”;
   mysql_close($conn) ;
?>
```

En la actualización de datos con éxito, verá la siguiente salida -

```
mysql> Updated data successfully
```

MARIADB CONSULTA DE ELIMINACIÓN

El comando DELETE elimina filas de la tabla de la tabla especificada, y devuelve la cantidad eliminada. Acceder a la cantidad que ha de suprimirse la ROW_COUNT()función. Una cláusula WHERE especifica las filas, y en su ausencia, se eliminan todas las filas. Una cláusula LIMIT controla el número de filas suprimidas.

En una instrucción DELETE para múltiples filas, se elimina sólo las filas que satisfacen una condición; y las cláusulas LIMIT y donde no se permiten. DELETE permiten eliminar filas de las tablas en diferentes bases de datos, pero no permiten la supresión de una tabla y luego seleccionar de la misma tabla en una subconsulta.

Revisar la siguiente sintaxis Eliminar -

```
DELETE FROM table_name [WHERE …]
```

Ejecutar un comando elimina de la línea de comandos o mediante un script PHP.

EL SÍMBOLO DEL SISTEMA

En el símbolo del sistema, basta con utilizar un comando estándar -

```
root@host# mysql –u root –p password;
Enter password:*******
mysql> use PRODUCTS;
Database changed
mysql> DELETE FROM products_tbl WHERE product_id=133;
mysql> SELECT * from products_tbl WHERE ID_number='133';
```

```
ERROR 1032 (HY000) : Can't find record in 'products_tbl'
```

ELIMINAR SCRIPT DE CONSULTA PHP

Utilice el **mysql_query() en declaraciones comando de eliminación -**

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass) ;

   if(! $conn ) {
      die('Could not connect: ' . mysql_error() );
   }

   $sql = 'DELETE FROM products_tbl WHERE product_id = 261';
   mysql_select_db('PRODUCTS') ;
   $retval = mysql_query( $sql, $conn ) ;

   if(! $retval ) {
      die('Could not delete data: ' . mysql_error() );
   }

   echo "Deleted data successfully\n";
   mysql_close($conn) ;
?>
```

En los datos de eliminación con éxito, verá la siguiente salida -

```
mysql> Deleted data successfully
mysql> SELECT * from products_tbl WHERE ID_number='261';
ERROR 1032 (HY000) : Can't find record in 'products_tbl'
```

CLÁUSULA LIKE MARIADB

La cláusula WHERE proporciona una manera de recuperar los datos cuando una operación se utilizan una coincidencia exacta. En situaciones que requieren múltiples resultados con características compartidas, la cláusula **LIKE acomoda una amplia coincidencia de patrones.**

Al probar la cláusula LIKE para una coincidencia de patrón, devuelve un verdadero o falso. Los patrones utilizados para la comparación aceptan los siguientes caracteres comodín: "%", que coincide con el número de caracteres (0 o más) ; y "_", que coincide con un carácter único. El carácter comodín "_" sólo coincide con caracteres en su conjunto, lo que significa que no hará caso de caracteres latinos cuando se utiliza otro conjunto. El resto es sensible a mayúsculas; por defecto, requiere ajustes adicionales para mayúsculas y minúsculas.

Una cláusula NOT LIKE permite probar la condición opuesta, al igual que el operador **not.**

Si la expresión declaración o patrón se evalúan como NULL, el resultado es NULL.

Revisar la sintaxis de la cláusula general, como se indica a continuación -

```
SELECT field, field2,... FROM table_name, table_name2,...
WHERE field LIKE condition
```

Emplear una cláusula LIKE ya sea en el símbolo del sistema o dentro de un script PHP.

EL SÍMBOLO DEL SISTEMA

En el símbolo del sistema, basta con utilizar un comando estándar -

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT * from products_tbl
   WHERE product_manufacturer LIKE 'XYZ%';
+-------------+----------------+----------------------+
| ID_number   | Nomenclature   | product_manufacturer |
```

```
+-------------+----------------+----------------------+
| 12345       | Orbitron 4000  | XYZ Corp             |
+-------------+----------------+----------------------+
| 12346       | Orbitron 3000  | XYZ Corp             |
+-------------+----------------+----------------------+
| 12347       | Orbitron 1000  | XYZ Corp             |
+-------------+----------------+----------------------+
```

SCRIPT PHP UTILIZANDO COMO CLÁUSULA

Utilice **mysql_query() en los estados que emplean la cláusula LIKE**

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass) ;

   if(! $conn ) {
      die('Could not connect: ' . mysql_error() );
   }

   $sql = 'SELECT product_id, product_name, product_manufacturer
, ship_date
      FROM products_tbl WHERE product_manufacturer LIKE "xyz%"';

   mysql_select_db('PRODUCTS') ;
   $retval = mysql_query( $sql, $conn ) ;

   if(! $retval ) {
      die('Could not get data: ' . mysql_error() );
   }
```

```
   while($row = mysql_fetch_array($retval, MYSQL_ASSOC) ) {
      echo "Product ID:{$row['product_id']} <br> ".
         "Name: {$row['product_name']} <br> ".
         "Manufacturer: {$row['product_manufacturer']} <br> ".
         "Ship Date: {$row['ship_date']} <br> ".
         "--------------------------------<br>";
   }

   echo "Fetched data successfully\n";
   mysql_close($conn) ;
?>
```

En el éxito de recuperación de datos, verá la siguiente salida -

```
Product ID: 12345
Nomenclature: Orbitron 4000
Manufacturer: XYZ Corp
Ship Date: 01/01/17
--------------------------------------------
Product ID: 12346
Nomenclature: Orbitron 3000
Manufacturer: XYZ Corp
Ship Date: 01/02/17
--------------------------------------------
Product ID: 12347
Nomenclature: Orbitron 1000
Manufacturer: XYZ Corp
Ship Date: 01/02/17
--------------------------------------------
mysql> Fetched data successfully
```

MARIADB CLÁUSULA ORDER BY

El **ORDER BY cláusula, como se ha mencionado en las discusiones anteriores, ordena los resultados de una instrucción.** Se especifica el orden de los datos operados en, e incluye la opción para ordenar en orden ascendente (ASC) ó orden descendente (DESC). Por omisión de especificación de la orden, el orden predeterminado es ascendente.

Cláusulas ORDER BY aparece en una amplia variedad de estados tales como borrar y actualizar. Siempre aparecen al final de una declaración, no en una subconsulta o antes de una función de conjunto, debido a que operan en la mesa final resultante. Tampoco se puede utilizar un número entero para identificar una columna.

Revisar la sintaxis general de la cláusula ORDER BY que se indica a continuación -

```
SELECT field, field2,... [or column] FROM table_name, table_name
2,...
ORDER BY field, field2,... ASC[or DESC]
```

Utilizar una cláusula ORDER BY, ya sea en el símbolo del sistema o dentro de un script PHP.

EL SÍMBOLO DEL SISTEMA

En el símbolo del sistema, basta con utilizar un comando estándar -

```
root@ host# mysql -u root -p password;
Enter password:*******
mysql> use PRODUCTS;
Database changed

mysql> SELECT * from products_tbl ORDER BY product_manufacturer
ASC
+-------------+----------------+----------------------+
| ID_number   | Nomenclature   | product_manufacturer |
+-------------+----------------+----------------------+
| 56789       | SuperBlast 400 | LMN Corp             |
+-------------+----------------+----------------------+
```

```
| 67891       | Zoomzoom 5000  | QFT Corp             |
+-------------+----------------+----------------------+
| 12347       | Orbitron 1000  | XYZ Corp             |
+-------------+----------------+----------------------+
```

SCRIPT PHP EL USO DE LA CLÁUSULA ORDER BY

Utilizar el **mysql_query(), una vez más, en los estados que emplean la cláusula ORDER BY –**

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass) ;

   if(! $conn ) {
      die('Could not connect: ' . mysql_error() );
   }

   $sql = 'SELECT product_id, product_name, product_manufacturer
, ship_date
      FROM products_tbl ORDER BY product_manufacturer DESC';

   mysql_select_db('PRODUCTS') ;
   $retval = mysql_query( $sql, $conn ) ;

   if(! $retval ) {
      die('Could not get data: ' . mysql_error() );
   }

   while($row = mysql_fetch_array($retval, MYSQL_ASSOC) ) {
```

```
      echo "Product ID :{$row['product_id']} <br> ".
         "Name: {$row['product_name']} <br> ".
         "Manufacturer: {$row['product_manufacturer']} <br> ".
         "Ship Date : {$row['ship_date']} <br> ".
         "--------------------------------<br>";
   }

   echo "Fetched data successfully\n";
   mysql_close($conn) ;
?>
```

En el éxito de recuperación de datos, verá la siguiente salida -

```
Product ID: 12347
Nomenclature: Orbitron 1000
Manufacturer: XYZ Corp
Ship Date: 01/01/17
--------------------------------------------
Product ID: 67891
Nomenclature: Zoomzoom 5000
Manufacturer: QFT Corp
Ship Date: 01/01/17
--------------------------------------------
Product ID: 56789
Nomenclature: SuperBlast 400
Manufacturer: LMN Corp
Ship Date: 01/04/17
--------------------------------------------
mysql> Fetched data successfully
```

MARIADB JOIN

En los ejemplos anteriores, se analizó la recuperación de una sola tabla, o recuperar múltiples valores de múltiples fuentes. La mayoría de las operaciones de datos del mundo real son mucho más complejo, que requiere la agregación, la comparación y la recuperación de varias tablas.

Los JOINs permiten la fusión de dos o más tablas en un solo objeto. Son empleados a través de SELECT, UPDATE y DELETE.

Revisar la sintaxis general de una declaración que emplea un JOIN como se muestra a continuación -

```
SELECT column
FROM table_name1
INNER JOIN table_name2
ON table_name1.column = table_name2.column;
```

Es posible utilizar una cláusula WHERE para lograr una unión, pero las palabras clave funcionan mejor para facilitar la lectura, el mantenimiento y las mejores prácticas.

Se puede unir en muchas formas, tales como una combinación izquierda, a la derecha, o combinación interna. Varios tipos de combinación ofrecen diferentes tipos de agregación basadas en valores o características compartidas.

Emplear un JOIN ya sea en el símbolo del sistema o con un script PHP.

EL SÍMBOLO DEL SISTEMA

En el símbolo del sistema, basta con utilizar una declaración estándar -

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use PRODUCTS;
Database changed

mysql> SELECT products.ID_number, products.Nomenclature, invento
ry.inventory_ct
```

```
   FROM products
   INNER JOIN inventory
   ON products.ID_numbeer = inventory.ID_number;
+-------------+----------------+-----------------+
| ID_number   | Nomenclature   | Inventory Count |
+-------------+----------------+-----------------+
| 12345       | Orbitron 4000  | 150             |
+-------------+----------------+-----------------+
| 12346       | Orbitron 3000  | 200             |
+-------------+----------------+-----------------+
| 12347       | Orbitron 1000  | 0               |
+-------------+----------------+-----------------+
```

SCRIPT PHP EL USO REGISTRARSE

Utilice el **mysql_query() para realizar una operación de combinación -**

```
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'root';
   $dbpass = 'rootpassword';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass) ;

   if(! $conn ) {
      die('Could not connect: ' . mysql_error() );
   }

   $sql = 'SELECT a.product_id, a.product_manufacturer, b.produc
t_count
      FROM products_tbl a, pcount_tbl b
      WHERE a.product_manufacturer = b.product_manufacturer';
```

```
   mysql_select_db('PRODUCTS') ;
   $retval = mysql_query( $sql, $conn ) ;

   if(! $retval ) {
      die('Could not get data: ' . mysql_error() );
   }

   while($row = mysql_fetch_array($retval, MYSQL_ASSOC) ) {
      echo "Manufacturer:{$row['product_manufacturer']} <br> ".
         "Count: {$row['product_count']} <br> ".
         "Product ID: {$row['product_id']} <br> ".
         "--------------------------------<br>";
   }

   echo "Fetched data successfully\n";
   mysql_close($conn) ;
?>
```

En el éxito de recuperación de datos, verá la siguiente salida -

```
ID Number: 12345
Nomenclature: Orbitron 4000
Inventory Count: 150
---------------------------------------
ID Number: 12346
Nomenclature: Orbitron 3000
Inventory Count: 200
---------------------------------------
ID Number: 12347
Nomenclature: Orbitron 1000
Inventory Count: 0
```

```
---------------------------------------

mysql> Fetched data successfully
```

MARIADB VALORES NULOS

Cuando se trabaja con valores NULL, recuerde que son valores desconocidos. No son cadenas vacías o cero, que son valores válidos. En la creación de tablas, las especificaciones de columna permiten ajustarlos a aceptar valores nulos, o rechazarlos. Basta con utilizar una cláusula NULL o NOT NULL. Esto tiene aplicaciones en casos de información de registro que falta como un número de identificación.

Las variables definidas por el usuario tienen un valor de NULL hasta que se realice la asignación explícita. Los parámetros de rutina almacenados y variables locales permiten establecer un valor NULL. Cuando una variable local no tiene ningún valor por defecto, que tiene un valor de NULL.

El NULL es sensible a las mayúsculas, y tiene los siguientes alias -

- UNKNOWN (a boolean value)
- \NORTH

LOS OPERADORES NULL

Operadores de comparación estándar no se pueden utilizar con NULL (ejemplo: =, >, >=, <=, <, or !=) Porque todas las comparaciones con un nulo retorna valor NULL, no verdadera o falsa. Las comparaciones con NULL debe utilizar el operador “<=>” (NULL-SAFE).

Otros operadores disponibles son -

- ISNULL - Pone a prueba por un valor NULL.
- NOTNULL - Se confirma la ausencia de un valor NULL.
- ISNULL - Se devuelve un valor de 1 en el descubrimiento de un valor NULL, y 0 en su ausencia.
- COALESCE - Se devuelve el primer valor no nulo de una lista, o se devuelve un valor NULL en la ausencia de uno.

CLASIFICANDO LOS VALORES NULL

En operaciones de clasificación, los valores NULL tienen el valor más bajo, por lo que DESC va ordenar los resultados en valores NULL en la parte

inferior. MariaDB permite establecer un valor superior para los valores NULL.

Hay dos maneras de hacer esto, como se muestra a continuación -

```
SELECT column1 FROM product_tbl ORDER BY ISNULL(column1) , colum
n1;
```

La otra forma -

```
SELECT column1 FROM product_tbl ORDER BY IF(column1 IS NULL, 0,
1) , column1 DESC;
```

FUNCIONES NULL

Hay funciones diseñadas específicamente para la gestión de valores NULL. Son -

- **IFNULL() - Si la primera expresión no es NULL se devuelve.** Cuando se evalúa a NULL, devuelve la segunda expresión.
- **NULLIF() - Devuelve NULL cuando las expresiones comparados son iguales, si no, se devuelve la primera expresión.**

Funciones como SUM y AVG ignoran los valores nulos.

LA INSERCIÓN DE VALORES NULL

En la inserción de un valor NULL en una columna NO declaró NULL, se produce un error. En el modo SQL por defecto, una columna NOT NULL en lugar insertar un valor predeterminado basado en el tipo de datos.

Cuando un campo es un TIMESTAMP, AUTO_INCREMENT o columna virtual, MariaDB gestiona los valores NULL de manera diferente. La inserción en una columna AUTO_INCREMENT hace que el siguiente número de la secuencia para insertar en su lugar. En un campo de timestamp, MariaDB asigna la fecha y hora actual en su lugar.

Índices de tipo unique pueden contener muchos valores NULL, sin embargo, las claves primarias no pueden ser NULL.

LOS VALORES NULL Y EL COMANDO ALTER

Cuando se utiliza el comando ALTER para modificar una columna, en ausencia de especificaciones NULL, MariaDB asigna automáticamente valores.

MARIADB DE EXPRESIONES REGULARES

Más allá de la coincidencia de patrones disponibles de cláusulas WHERE, MariaDB ofrece concordancia basada en expresiones regulares a través del operador REGEXP. El operador realiza la coincidencia de patrones de expresión de cadena basada en un patrón determinado.

MariaDB 10.0.5 introdujo expresiones regulares PCRE, lo que aumenta considerablemente el alcance de juego en áreas como patrones recursivos, look-ahead afirmaciones, y más.

Revisar el uso de la sintaxis del operador REGEXP norma dada a continuación -

```
SELECT column FROM table_name WHERE column REGEXP '[PATTERN]';
```

REGEXP devuelve 1 para una coincidencia de patrón o 0 en ausencia de uno.

Una opción para todo lo contrario existe en forma de NO REGEXP. MariaDB también ofrece sinónimos de REGEXP y NO REGEXP, RLIKE y NO RLIKE, que fueron creados por razones de compatibilidad.

El patrón de comparación puede ser una cadena literal o alguna otra cosa, como una columna de tabla. En las cadenas, que utiliza la sintaxis de escape de C "\". REGEXP también es sensible a las mayúsculas, con la excepción de cadenas binarias.

Una tabla de posibles patrones, que pueden utilizarse se da a continuación -

S.No	Patrón y Descripción
1	^ Que coincide con el inicio de la cadena.
2	**$** Que coincide con el final de la cadena.
3	**.** Coincide con un carácter único.
4	**[...]** Coincide con cualquier carácter en los soportes.
5	**[^...]** Coincide con cualquier carácter que no figura en los soportes.
6	**p1|p2|p3**

	Coincide con cualquiera de los patrones.
7	***** Se ajusta a 0 o más instancias del elemento precedente.
8	**+** Coincide con 1 o más instancias del elemento precedente.
9	**{n}** Coincide con n instancias del elemento precedente.
10	**{m,n}** Coincide con m a n instancias del elemento precedente.

Revisar el patrón de búsqueda de ejemplos dados a continuación -

Productos que comienzan con “PR” -

```
SELECT name FROM product_tbl WHERE name REGEXP '^pr';
```

Productos que terminan con “na” -

```
SELECT name FROM product_tbl WHERE name REGEXP 'na$';
```

Productos a partir de una vocal -

```
SELECT name FROM product_tbl WHERE name REGEXP '^[aeiou]';
```