

MySQL

1. Entorno MySQL
2. PhpMyAdmin
3. Workbench.
4. Acceso a un servidor de Base de datos de prueba.
5. Crear, eliminar y seleccionar una base de datos.
6. Crear, eliminar y seleccionar una tabla.
7. Concepto de Entidad, atributo y tipo de datos.
8. Estructura de una consulta sql y Cláusula SELECT
9. Alias de Tablas
10. Cláusula FROM y Cláusula WHERE
11. Cláusula Insert / Delete y Update

¿Qué es MySQL?

MySQL, el sistema de administración de bases de datos SQL de código abierto más popular, está desarrollado, distribuido y respaldado por Oracle Corporation.

MySQL es un sistema de gestión de bases de datos.

Una base de datos es una colección estructurada de datos. Puede ser cualquier cosa, desde una simple lista de compras hasta una galería de imágenes o la gran cantidad de información en una red corporativa. Para agregar, acceder y procesar datos almacenados en una base de datos informática, se necesita un sistema de administración de bases de datos como MySQL Server. Dado que las computadoras son muy buenas para manejar grandes cantidades de datos, los sistemas de administración de bases de datos juegan un papel central en los sistemas actuales, como utilidades independientes o como parte de otras aplicaciones.

Las bases de datos MySQL son relacionales.

Una base de datos relacional almacena datos en tablas separadas en lugar de poner todos los datos en un gran archivo. Las estructuras de la base de datos están organizadas en archivos físicos optimizados para la velocidad. El modelo lógico, con objetos como bases de datos, tablas, vistas, filas y columnas, ofrece un entorno de programación flexible.

Esto configura reglas que gobiernan las relaciones entre diferentes campos de datos, como uno a uno, uno a muchos, único, obligatorio u opcional, y " punteros " entre diferentes tablas. La base de datos hace cumplir estas reglas, de modo que con una base de datos bien diseñada, tu aplicación nunca ve datos inconsistentes, duplicados, huérfanos, desactualizados o faltantes.

Las letras SQL de " MySQL " significan " Lenguaje de consulta estructurado " . SQL es el lenguaje estandarizado más común utilizado para acceder a bases de datos. Dependiendo de tu entorno de programación, podés ingresar SQL directamente (por ejemplo, para generar informes), incrustar declaraciones SQL en código escrito en otro idioma o usar una API específica del idioma que oculta la sintaxis SQL.

SQL está definido por el estándar ANSI / ISO SQL. El estándar SQL ha evolucionado desde 1986 y existen varias versiones. Por ejemplo, " SQL-92 " se refiere al estándar publicado en 1992, " SQL: 1999 " se refiere al estándar publicado en 1999 y " SQL: 2003 " se refiere a la versión actual del estándar. Usamos la frase " el estándar SQL " para referirnos a la versión actual del estándar SQL en cualquier momento.

El software MySQL es de código abierto.

Código abierto significa que cualquiera puede utilizar y modificar el software. Cualquiera puede descargar el software MySQL de Internet y usarlo sin pagar nada. Si lo deseas, podés estudiar el código fuente y modificarlo para adaptarlo a tus necesidades. El software MySQL utiliza la GPL (Licencia Pública General GNU), <http://www.fsf.org/licenses/> , para definir lo que podés y no podés hacer con el software en diferentes situaciones. Si no te sentís cómodo con la GPL o necesitás incrustar código MySQL en una aplicación comercial, se puede adquirir una versión con licencia comercial. Para ello hay una descripción general de licencias de MySQL en <http://www.mysql.com/company/legal/licensing/>.

MySQL es un servidor de base de datos muy rápido, confiable, escalable y fácil de usar.

También puede ejecutarse cómodamente en una computadora de escritorio o portátil, junto con tus otras aplicaciones, servidores web, etc., requiriendo poca o ninguna atención. Si dedicás una máquina completa a MySQL, podés ajustar la configuración para aprovechar toda la memoria, la potencia de la CPU y la capacidad de E / S disponible. MySQL también puede escalar a grupos de máquinas conectadas en red.

MySQL Server se desarrolló originalmente para manejar grandes bases de datos mucho más rápido que las soluciones existentes y se ha utilizado con éxito en entornos de producción muy exigentes durante varios años. Aunque en constante desarrollo, MySQL Server ofrece hoy en día un conjunto de funciones útiles y variadas. Su conectividad, velocidad y seguridad hacen que MySQL Server sea muy adecuado para acceder a bases de datos en Internet.

MySQL Server funciona en sistemas cliente / servidor o integrados.

El software de base de datos MySQL es un sistema cliente / servidor que consta de un servidor SQL multiproceso que admite diferentes backends, varios programas cliente y bibliotecas diferentes, herramientas administrativas y una amplia gama de interfaces de programación de aplicaciones (API).

MySQL Server, es también, como una biblioteca multiproceso integrada que puede vincular a su aplicación para obtener un producto independiente más pequeño, más rápido y más fácil de administrar.

Hay disponible una gran cantidad de software MySQL contribuido.

MySQL Server tiene un conjunto práctico de características desarrolladas en estrecha colaboración con usuarios de la comunidad. Es muy probable que tu aplicación o lenguaje de programación favorito sea compatible con el servidor de base de datos MySQL.

Concepto de Entidad

Se define una entidad (o instancia) como una unidad de una base de datos que contiene información. Esta unidad es una representación dentro de la base de datos de un objeto, persona, empresa... etc, del mundo real, y como tal posee ciertos atributos que la diferencian del resto de entidades. Así por ejemplo, en una base de datos de una oficina, una entidad podría ser el material de oficina, otra los empleados, otra los clientes, incluso el ambiente laboral, la empatía y cosas más abstractas. Cada una de estas entidades tendría ciertos atributos propios. Así, los empleados tendrían atributos como nombre, edad, estatura... las computadoras otros como un nro de serie, procesador, año de compra... y así para cada una de ellas.

En una base de datos compleja pueden existir entidades relacionadas entre si por diversos parámetros o atributos, de tal modo que la existencia de una puede ir ligada a la existencia de otra. Así, las entidades pueden ser fuertes (existen por si mismas) o débiles (su existencia depende de que exista otra entidad). Las relaciones entre entidades suelen describirse en el esquema de la estructura de la base de datos e incluso pueden agruparse entre si para formar conjuntos de entidades, también llamados clases.

El modelo Entidad-Relación es de hecho uno de los más importantes a la hora de diseñar e implementar una base de datos con éxito. Mediante este modelo se relacionan una o varias entidades por sus atributos, que pueden ser comunes o no a varias de ellas.

Invocar programas MySQL

Para invocar un programa MySQL desde la línea de comando (es decir, desde su shell o símbolo del sistema CMD en Windows), ingresá a la carpeta en donde se instaló MySQL y escribí el nombre del programa seguido de cualquier opción u otros argumentos necesarios para indicarle al programa lo que desea que haga.

Los siguientes comandos muestran algunas invocaciones de programas de muestra.

C:\> representa la solicitud de su intérprete de comandos (CMD en Windows); no es parte de lo que se escribe. El indicador particular que ve depende de su intérprete de comandos. Las solicitudes típicas son: Linux/ Unix / Mac \$para sh , ksh o bash , %para csh o tcsh y C:\>para los intérpretes de comandos Windows DOS o cmd .

Por ejemplo:

```
C:\> mysql --user=root test
```

```
C:\> mysqladmin extended-status variables
```

```
C:\> mysqlshow --help
```

```
C:\> mysqldump -u root personnel
```

Los argumentos que comienzan con un guión simple o doble (-, --) especifican las opciones del programa. Las opciones suelen indicar el tipo de conexión que un programa debe realizar con el servidor o afectar su modo operativo.

Los argumentos que no son de opción (argumentos sin guiones iniciales) proporcionan información adicional al programa. Por ejemplo, el programa mysql interpreta el primer argumento que no es de opción como un nombre de base de datos, por lo que el comando mysql --user=root test indica que desea utilizar la test base de datos.

Las secciones posteriores que describen programas individuales indican qué opciones admite un programa y describen el significado de cualquier argumento adicional que no sea de opción.

Algunas opciones son comunes a varios programas.

Las más utilizadas son las opciones

--host(o -h),

--user(o -u) y --password(o -p)

que especifican los parámetros de conexión. Indican el host donde se ejecuta el servidor MySQL y el nombre de usuario y contraseña de su cuenta MySQL.

Todos los programas cliente de MySQL comprenden estas opciones; te permiten especificar a qué servidor conectarse y la cuenta a utilizar en ese servidor.

Otras opciones de conexión son:

--port(o -P) especificar un número de puerto TCP / IP y

--socket(o-S) para especificar un archivo de socket Unix en Unix (o un nombre de canalización con nombre en Windows).

Puede resultar necesario invocar programas MySQL utilizando el nombre de la ruta al directorio bin en el que están instalados.

Es probable que este sea el caso si obtiene un error de " programa no encontrado " cada vez que intenta ejecutar un programa MySQL desde cualquier directorio que no sea el directorio bin. Para que sea más conveniente usar MySQL, puede agregar el nombre de la ruta del directorio bin a la PATH configuración de la variable de entorno. Eso le permite ejecutar un programa escribiendo solo su nombre, no su ruta completa. Por ejemplo, si mysql está instalado c:\Archivos\mysql\bin, podés ejecutar el programa invocándolo como mysql, y no es necesario invocarlo como /c:\Archivos\mysql\bin .

Consultá en la web la documentación de los comandos que utiliza tu sistema operativo para obtener instrucciones sobre cómo configurar tu variable PATH.

Después de modificar su configuración PATH, abrí una nueva ventana de consola en Windows (CMD) o iniciá sesión nuevamente en Unix para que la configuración entre en vigor.

Uso de opciones en la línea de comandos

Las opciones del programa especificadas en la línea de comando siguen estas reglas:

- Las opciones se dan después del nombre del comando.
- Un argumento de opción comienza con un guión o dos guiones, dependiendo de si es una forma corta o larga del nombre de la opción. Muchas opciones tienen formas cortas y largas.

Por ejemplo, `-?` y `--help` son las formas cortas y largas de la opción que le indica a un programa MySQL que muestre su mensaje de ayuda.

- Los nombres de las opciones distinguen entre mayúsculas y minúsculas. `-v` y `-V` son legales y tienen diferentes significados. (Son las formas breves correspondientes de las opciones `--verbose` y `--version`).

- Algunas opciones toman un valor después del nombre de la opción. Por ejemplo, `-h localhost` o `--host=localhost` indican el host del servidor MySQL a un programa cliente. El valor de la opción le dice al programa el nombre del host donde se ejecuta el servidor MySQL.

- Para una opción larga que toma un valor, separa el nombre de la opción y el valor con un signo `=`.

Para una opción corta que toma un valor, el valor de la opción puede seguir inmediatamente a la letra de la opción, o puede haber un espacio entre: `-hlocalhost` y `-h localhost` son equivalentes. Una excepción a esta regla es la opción para especificar tu contraseña de MySQL. Esta opción se puede dar en forma larga como `--password=pass_val` o como `--p pass_val`. En el último caso (sin ningún valor de contraseña dado), el programa le solicita interactivamente la contraseña. La opción de contraseña también se puede dar en forma abreviada como `--password=pass_val` o como `--p pass_val`.

Sin embargo, para la forma abreviada, si se proporciona el valor de la contraseña, debe seguir la letra de la opción sin espacios intermedios: si un espacio sigue a la letra de la opción, el programa no tiene forma de saber si se supone que el siguiente argumento es la contraseña valor o algún otro tipo de argumento. En consecuencia, los dos comandos siguientes tienen dos significados completamente diferentes:

```
mysql -ptest
```

```
mysql -p test
```

El primer comando le indica a mysql que use un valor de contraseña de test, pero no especifica una base de datos predeterminada. El segundo indica a mysql que solicite el valor de la contraseña y que la utilice test como base de datos predeterminada.

Dentro de los nombres de las opciones, el guión (`-`) y el subrayado (`_`) se pueden usar indistintamente. Por ejemplo, `--skip-grant-tables` y `--skip_grant_tables` son equivalentes. (Sin embargo, los guiones iniciales no se pueden dar como guiones bajos).

El servidor MySQL tiene ciertas opciones de comando que pueden especificarse solo al inicio, y un conjunto de variables del sistema, algunas de las cuales pueden configurarse al inicio, en tiempo de ejecución o ambos. Los nombres de las variables del sistema usan guiones bajos en lugar de guiones, y cuando se hace referencia a ellos en tiempo de ejecución (por ejemplo, usando instrucciones `SET` o `SELECT`), deben escribirse usando guiones bajos:

```
SET GLOBAL general_log = ON;  
SELECT @@GLOBAL.general_log;
```

Al iniciar el servidor, la sintaxis de las variables del sistema es la misma que la de las opciones de comando, por lo que dentro de los nombres de las variables, los guiones y los guiones bajos se pueden usar indistintamente. Por ejemplo, `--general_log=ON` y `--general-log=ON` son equivalentes. (Esto también es cierto para las variables del sistema establecidas dentro de los archivos de opciones).

Por ejemplo, el siguiente comando le dice a `mysqladmin` que haga ping al servidor 1024 veces, durmiendo 10 segundos entre cada ping:

```
mysqladmin --count=1K --sleep=10 ping
```

Cuando especifique nombres de archivo como valores de opción, evite el uso del carácter `~` meta de shell. Puede que no se interprete como esperaba.

Los valores de opción que contienen espacios deben estar entre comillas cuando se dan en la línea de comando. Por ejemplo, la opción `--execute_`(ó `-e`) se puede utilizar con `mysql` para pasar una o más sentencias SQL separadas por punto y coma al servidor. Cuando se usa esta opción, `mysql` ejecuta las declaraciones en el valor de la opción y sale. Las declaraciones deben ir entre comillas. Por ejemplo:

```
shell> mysql -u root -p -e "SELECT VERSION();SELECT NOW()"  
Enter password: *****  
  
+-----+  
| VERSION() |  
+-----+  
| 8.0.19    |  
+-----+  
  
+-----+  
| NOW()      |  
+-----+  
| 2019-09-03 10:36:48 |  
+-----+
```

```
shell>
```

Nota: La forma larga (--execute) va seguida de un signo igual (=). Para usar valores entre comillas dentro de una declaración, debe escapar de las comillas internas o usar un tipo de comillas diferente dentro de la declaración de las que se usan para citar la declaración en sí. Las capacidades de su procesador de comandos dictan sus opciones sobre si puede usar comillas simples o dobles y la sintaxis para escapar de los caracteres de comillas. Por ejemplo, si su procesador de comandos admite comillas con comillas simples o dobles, puede utilizar comillas dobles alrededor de la declaración y comillas simples para cualquier valor entre comillas dentro de la declaración. Por ej: “ ’ Hola Mundo ‘ “

Conectarse al servidor MySQL mediante opciones de comando

Esta sección describe el uso de las opciones de la línea de comandos para especificar cómo establecer conexiones con el servidor MySQL, para clientes como `mysql` o `mysqldump`.

Para que un programa cliente se conecte al servidor MySQL, debes utilizar los parámetros de conexión adecuados, como el nombre del host donde se ejecuta el servidor y el nombre de usuario y contraseña de tu cuenta MySQL. Cada parámetro de conexión tiene un valor predeterminado, pero puede anular los valores predeterminados según sea necesario utilizando las opciones del programa especificadas en la línea de comandos o en un archivo de opciones.

Los siguiente ejemplos usan el programa cliente `mysql`, pero los principios se aplican a otros clientes como `mysqldump`, `mysqladmin` o `mysqlshow`.

Este comando invoca a `mysql` sin especificar ningún parámetro de conexión explícito:

```
mysql
```

Como no hay opciones de parámetros, se aplican los valores predeterminados:

- El nombre de host predeterminado es `localhost`.
- El nombre de usuario predeterminado está en el ODBC (estándar de acceso a las bases de datos) en Windows o su nombre de inicio de sesión de Unix en Unix.
- No se envía contraseña porque no se da `--password` ni `-p`.
- Para `mysql`, el primer argumento que no es de opción se toma como el nombre de la base de datos predeterminada. Como no existe tal argumento, `mysql` no selecciona una base de datos predeterminada.

Para especificar el nombre de host y el nombre de usuario de forma explícita, así como una contraseña, proporcioná las opciones adecuadas en la línea de comandos. Para seleccionar una base de datos predeterminada, incorporá un argumento de nombre de base de datos.

Ejemplos:

```
mysql --host=localhost --user=myname --password=password mydb
```

```
mysql -h localhost -u myname -ppassword mydb
```

Para las opciones de contraseña, el valor de la contraseña es opcional:

- Si utiliza una opción `--password` o `-p` y especifica un valor de contraseña, no debe haber ningún espacio entre `--password=` o `-p` y la contraseña que le sigue.
- Si usa `--password` o `-p` pero no especifica un valor de contraseña, el programa cliente le solicita que ingrese la contraseña. La contraseña no se muestra cuando la ingresa. Esto es más seguro que dar la contraseña en la línea de comando, lo que podría permitir a otros usuarios de su sistema ver la línea anterior de comando simplemente apretando la flecha arriba del teclado y accediendo al historial de órdenes ingresadas.
- Para especificar explícitamente que no hay contraseña y que el programa cliente no debe solicitar una, use la opción `--skip-password`.

Como se acaba de mencionar, incluir el valor de la contraseña en la línea de comando es un riesgo para la seguridad. Para evitar este riesgo, especifique la opción `--password` o `-p` sin ningún valor de contraseña siguiente:

```
mysql --host=localhost --user=myname --password mydb
```

```
mysql -h localhost -u myname -p mydb
```

Cuando se da la opción `--password` o `-p` sin valor de contraseña, el programa cliente imprime un mensaje y espera a que ingrese la contraseña. (En estos ejemplos, `mydb` no se ha interpretado como una contraseña, ya que está separada de la precedente opción de contraseña por un espacio.)

En algunos sistemas, la rutina de la biblioteca que utiliza MySQL para solicitar una contraseña limita automáticamente la contraseña a ocho caracteres. Esa limitación es una propiedad de la biblioteca del sistema, no MySQL. Internamente, MySQL no tiene ningún límite para la longitud de la contraseña. Otra solución es cambiar tu contraseña de MySQL a un valor que tenga ocho o menos caracteres, pero que tiene la desventaja de que las contraseñas más cortas tienden a ser menos seguras.

Los programas cliente determinan qué tipo de conexión realizar de la siguiente manera:

- Si el host no se especifica o lo está `localhost`, se produce una conexión con el host local:
 - En Windows, el cliente se conecta mediante memoria compartida, si el servidor se inició con la variable `shared_memory` del sistema habilitada para admitir conexiones de memoria compartida.
 - En Unix, los programas MySQL tratan el nombre de host `localhost` de manera especial, de una manera que probablemente sea diferente de lo que espera en comparación con otros programas basados en la red: el cliente se conecta mediante un archivo de socket Linux.
- En Windows, si host es `.` (punto), o TCP / IP no está habilitado y `--socket` no se especifica o el host está vacío, el cliente se conecta mediante con nombre, si el servidor se inició con la variable `named_pipe` del sistema habilitada para admitir conexiones con nombre `.`. Si no se admiten las conexiones de canalización con nombre o si el usuario que realiza la conexión no es miembro del grupo de Windows especificado por la `named_pipe_full_access_group` variable del sistema, se produce un error.
- De lo contrario, la conexión utiliza TCP / IP.

Solo se utilizan o verifican las opciones de conexión que son relevantes para el protocolo de transporte seleccionado. Se ignoran otras opciones de conexión. Por ejemplo, `--host=localhost` en Unix, el cliente intenta conectarse al servidor local utilizando un archivo de socket Unix, incluso si se da una opción `--p` o `-P` para especificar un número de puerto TCP / IP.

Para asegurarse de que el cliente realiza una conexión TCP / IP con el servidor local, utilice `--host` o `-h` para especificar un valor de nombre de host de `127.0.0.1` (en lugar de `localhost`), o la dirección IP o el nombre del servidor local. También puede especificar el protocolo de transporte explícitamente, incluso para `localhost`, utilizando la opción `--protocol=TCP`. Ejemplos:

```
mysql --host=127.0.0.1
```

```
mysql --protocol=TCP
```

Las conexiones a servidores remotos utilizan TCP / IP. Este comando se conecta al servidor en ejecución remote.example.com utilizando el número de puerto predeterminado (3306):

```
mysql --host=remote.example.com
```

Para especificar un número de puerto explícitamente, use la opción --port o -P:

```
mysql --host=remote.example.com --port=13306
```

También puede especificar un número de puerto para las conexiones a un servidor local. Sin embargo, como se indicó anteriormente, las conexiones localhost en Unix usan un archivo de socket de forma predeterminada, por lo que, a menos que fuerce una conexión TCP / IP como se describió anteriormente, se ignorará cualquier opción que especifique un número de puerto.

Para este comando, el programa usa un archivo de socket en Unix y la --port opción se ignora:

```
mysql --port=13306 --host=localhost
```

Para hacer que se utilice el número de puerto, fuerce una conexión TCP / IP. Por ejemplo, invoque el programa de cualquiera de estas formas:

```
mysql --port=13306 --host=127.0.0.1
```

```
mysql --port=13306 --protocol=TCP
```

mysqld : el servidor MySQL

mysqld , también conocido como MySQL Server, es un único programa multiproceso que hace la mayor parte del trabajo en una instalación de MySQL. No genera procesos adicionales. MySQL Server administra el acceso al directorio de datos MySQL que contiene bases de datos y tablas. El directorio de datos también es la ubicación predeterminada para otra información, como archivos de registro y archivos de estado.

Cuando se inicia el servidor MySQL, escucha las conexiones de red de los programas cliente y administra el acceso a las bases de datos en nombre de esos clientes.

El programa mysqld tiene muchas opciones que se pueden especificar al inicio.

MySQL Server también tiene un conjunto de variables del sistema que afectan su funcionamiento mientras se ejecuta. Las variables del sistema se pueden configurar al iniciar el servidor y muchas de ellas se pueden cambiar en tiempo de ejecución para efectuar la reconfiguración dinámica del servidor.

MySQL Server también tiene un conjunto de variables de estado que brindan información sobre su funcionamiento. Podés supervisar estas variables de estado para acceder a las características de rendimiento en tiempo de ejecución.

Configuración del servidor

El servidor MySQL, mysqld , tiene muchas opciones de comando y variables del sistema que se pueden configurar al inicio para configurar su funcionamiento. Para determinar la opción de comando predeterminada y los valores de las variables del sistema que utiliza el servidor, ejecutá este comando:

```
c:\> mysqld --verbose --help
```

El comando produce una lista de todas las opciones de mysqld y variables de sistema configurables. Su salida incluye la opción predeterminada y los valores de las variables y se parece a esto:

abort-slave-event-count	0
allow-suspicious-udfs	FALSE
archive	ON
auto-increment-increment	1
auto-increment-offset	1
autocommit	TRUE
automatic-sp-privileges	TRUE
avoid-temporal-upgrade	FALSE
back-log	80

basedir	/home/jon/bin/mysql-8.0/
tmpdir	/tmp
transaction-alloc-block-size	8192
transaction-isolation	REPEATABLE-READ
transaction-prealloc-size	4096
transaction-read-only	FALSE
transaction-write-set-extraction	OFF
updatable-views-with-limit	YES
validate-user-plugins	TRUE
verbose	TRUE
wait-timeout	28800

Para ver los valores actuales de las variables del sistema que realmente utiliza el servidor mientras se ejecuta, conectate y ejecutá esta declaración:

```
mysql> SHOW VARIABLES;
```

Para ver algunos indicadores estadísticos y de estado de un servidor en ejecución, ejecutá esta declaración:

```
mysql> SHOW STATUS;
```

La variable del sistema y la información de estado también están disponibles usando el comando `mysqladmin` :

```
shell> mysqladmin variables
```

```
shell> mysqladmin extended-status
```

Creando una Base de Datos

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_option] ...

create_option: [DEFAULT] {
    CHARACTER SET [=] charset_name
    | COLLATE [=] collation_name
    | ENCRYPTION [=] {'Y' | 'N'}
}
```

CREATE DATABASE crea una base de datos con el nombre de indicado en esa orden. Para utilizar esta declaración, se necesita el privilegio o permiso del sistema de la base de datos. CREATE SCHEMA es sinónimo de CREATE DATABASE.

Se produce un error si la base de datos ya existe y no le especificaste IF NOT EXISTS.

CREATE DATABASE no está permitido dentro de una sesión que tiene una declaración activa LOCK TABLES.

Cada **create_option** especifica una característica de la base de datos. Las características de la base de datos se almacenan en el diccionario de datos.

- La opción CHARACTER SET especifica el juego de caracteres predeterminado de la base de datos. La COLLATE opción especifica la clasificación de la base de datos predeterminada.

Para ver los conjuntos de caracteres y las intercalaciones disponibles, utiliza las instrucciones SHOW CHARACTER SET y SHOW COLLATION, respectivamente.

- La opción ENCRYPTION, introducida en MySQL 8.0.16, define el cifrado de base de datos predeterminado, que es heredado por las tablas creadas en la base de datos. Los valores permitidos son 'Y' (cifrado activado) y 'N' (cifrado desactivado). Si la opción ENCRYPTION no se especifica, el valor de la variable del sistema default_table_encryption define el cifrado predeterminado de la base de datos.

Una base de datos en MySQL se implementa como un directorio que contiene archivos que corresponden a tablas en la base de datos. Debido a que no hay tablas en una base de datos cuando se crea inicialmente, la declaración CREATE DATABASE crea solo un directorio bajo el directorio

de datos MySQL. Si el nombre de una base de datos contiene caracteres especiales, el nombre del directorio de la base de datos contiene versiones codificadas de esos caracteres.

La creación de un directorio de base de datos mediante la creación manual de un directorio en el directorio de datos (por ejemplo, con `mkdir`) no es compatible con MySQL 8.0.

Al crear una base de datos, dejá que el servidor administre el directorio y los archivos que contiene. La manipulación de archivos y directorios de bases de datos directamente puede provocar inconsistencias y resultados inesperados.

MySQL no tiene límite en la cantidad de bases de datos. El sistema de archivos subyacente puede tener un límite en el número de directorios.

También podés utilizar el programa `mysqladmin` para crear bases de datos.

Sentencia CREATE TABLE

CREATE TABLE crea una tabla con el nombre especificado. Debes tener el privilegio / permiso CREATE para crear una tabla.

De forma predeterminada, las tablas se crean en la base de datos predeterminada, utilizando el motor InnoDB de almacenamiento. Se produce un error si la tabla existe, si no hay una base de datos predeterminada o si la base de datos no existe.

MySQL no tiene límite en el número de tablas. El sistema de archivos subyacente puede tener un límite en la cantidad de archivos que representan tablas. Los motores de almacenamiento individuales pueden imponer restricciones específicas del motor. InnoDB permite hasta 4 mil millones de tablas.

Hay varios aspectos importantes de CREATE TABLE, descritos en los siguientes temas de esta sección:

Nombre de la tabla

- ***tbl_name***

El nombre de la tabla se puede especificar *db_name.tbl_name* para crear la tabla en una base de datos específica. Esto funciona independientemente de si existe una base de datos predeterminada, asumiendo que la base de datos existe. Si utiliza identificadores entre comillas, cite los nombres de la base de datos y de la tabla por separado. Por ejemplo, escribe ``mydb`.`mytbl``, no ``mydb.mytbl``.

- **IF NOT EXISTS**

Evita que se produzca un error si la tabla existe. Sin embargo, no se verifica que la tabla existente tenga una estructura idéntica a la indicada por el CREATE TABLE.

Tablas temporales

Puede utilizar la palabra clave **TEMPORARY** al crear una tabla. Una tabla **TEMPORARY** es visible solo dentro de la sesión actual y se elimina automáticamente cuando se cierra la sesión.

Clonación y copia de tablas

- **LIKE**

Usá **CREATE TABLE ... LIKE** para crear una tabla vacía basada en la definición de otra tabla, incluidos los atributos e índices de columna definidos en la tabla original:

```
CREATE TABLE destino_tbl LIKE origen_tbl;
```

- **[AS] *query_expression***

Para crear una tabla a partir de otra, agregá una declaración **SELECT** al final de la **CREATE TABLE** declaración:

```
CREATE TABLE destino_tbl AS SELECT * FROM origen_tbl;
```

- **IGNORE | REPLACE**

Las opciones **IGNORE** y **REPLACE** indican cómo manejar filas que duplican valores clave únicos al copiar una tabla usando una declaración **SELECT**.

Atributos y tipos de datos de columna

Hay un límite estricto de 4096 columnas por tabla, pero el máximo efectivo puede ser menor para una tabla dada y depende de los factores de soporte del sistema operativo y hardware.

- ***data_type***

data_type representa el tipo de datos en una definición de columna.

- Algunos atributos no se aplican a todos los tipos de datos. **AUTO_INCREMENT** se aplica solo a los tipos enteros y de coma flotante. Antes de MySQL 8.0.13, **DEFAULT** no se aplicaba a los **BLOB**, **TEXT**, **GEOMETRY**, y tipos **JSON**.

- Tipos de datos de caracteres (**CHAR**, **VARCHAR**, los tipos **TEXT**, **ENUM**, **SET**, y cualquier sinónimo) pueden incluir **CHARACTER SET** para especificar el conjunto de caracteres para la columna. **CHARSET** es sinónimo de **CHARACTER SET**. Se puede especificar una colación para el juego de caracteres con el atributo **COLLATE**, junto con cualquier otro atributo. Ejemplo:

```
CREATE TABLE mitabla (micolumna CHAR(20) CHARACTER SET utf8 COLLATE utf8_bin);
```

MySQL 8.0 interpreta las especificaciones de longitud en las definiciones de columna de caracteres en caracteres.

- Para columnas CHAR, VARCHAR, BINARY, y VARBINARY, los índices pueden ser creados para el uso de sólo la parte principal de los valores de columna, utilizando la sintaxis para especificar un índice de longitud de prefijo, y aunque las columnas también se pueden indexar, se *debe* dar una longitud de prefijo . Por ejemplo:

```
CREATE TABLE mitabla (micolumna BLOB, INDEX(blob_col(10)));
```

Si un prefijo de índice especificado excede el tamaño máximo de tipo de datos de columna, CREATE TABLE maneja el índice de la siguiente manera:

- Para un índice no único, se produce un error (si el modo SQL estricto está habilitado) o la longitud del índice se reduce para que se encuentre dentro del tamaño máximo de tipo de datos de columna y se genera una advertencia (si el modo SQL estricto no está habilitado).
- Para un índice exclusivo, se produce un error independientemente del modo SQL porque la reducción de la longitud del índice puede permitir la inserción de entradas no únicas que no cumplen con el requisito de unicidad especificado.
- JSON las columnas no se pueden indexar. Podés solucionar esta restricción creando un índice en una columna generada que extraiga un valor de la columna JSON.
- NOT NULL | NULL

Si no se especifica NULL ni NOT NULL, la columna se trata como si se hubiera especificado NULL.

En MySQL 8.0, sólo los motores de almacenamiento InnoDB, MyISAM y MEMORY soportan índices en columnas que pueden tener NULL valores. En otros casos, deberás declarar las columnas indexadas como NOT NULL o tendrás como resultado de un error.

- DEFAULT

Especifica un valor predeterminado para una columna.

Si el modo NO_ZERO_DATE o NO_ZERO_IN_DATE SQL está habilitado y un valor predeterminado de fecha no es correcto de acuerdo con ese modo, CREATE TABLE produce una advertencia si el modo SQL estricto no está habilitado y un error si el modo estricto está habilitado. Por ejemplo, con NO_ZERO_IN_DATE habilitado, produce una advertencia.

- AUTO_INCREMENT

Una columna de número entero o de punto flotante puede tener el atributo adicional `AUTO_INCREMENT`. Cuando insertás un valor de `NULL`(recomendado) o `0` en una columna indexada `AUTO_INCREMENT`, la columna se establece en el siguiente valor de secuencia. Por lo general, esto es ***value***+1, donde ***value*** está el valor más grande para la columna actualmente en la tabla. Las secuencias `AUTO_INCREMENT` comienzan con 1.

Para recuperar un valor `AUTO_INCREMENT` después de insertar una fila, usá la función SQL `LAST_INSERT_ID()` o la función API de C `mysql_insert_id()`.

Si el modo SQL `NO_AUTO_VALUE_ON_ZERO` está habilitado, podés almacenar `0` en columnas `AUTO_INCREMENT` como `0` sin generar un nuevo valor de secuencia.

Solo puede haber una columna `AUTO_INCREMENT` por tabla, debe estar indexada y no puede tener un valor `DEFAULT`. Una columna `AUTO_INCREMENT` funciona correctamente si contiene solo valores positivos. Se considera que insertar un número negativo es insertar un número positivo muy grande. Esto se hace para evitar problemas de precisión cuando los números " pasan " de positivo a negativo y también para garantizar que no obtengas accidentalmente una columna `AUTO_INCREMENT` que contenga `0`.

Por ej:

```
SELECT * FROM mitabla WHERE auto_columna IS NULL
```

Este método requiere que la variable `sql_auto_is_null` no se establezca en `0`.

- **COMMENT**

Se puede especificar un comentario para una columna con la opción `COMMENT`, hasta 1024 caracteres. El comentario se muestra mediante las declaraciones `SHOW CREATE TABLE` y `SHOW FULL COLUMNS`.

- **PRIMARY KEY**

Un índice único donde todas las columnas clave deben definirse como `NOT NULL`. Si no se declaran explícitamente como `NOT NULL`, MySQL los declara implícitamente (y silenciosamente). Una tabla solo puede tener una `PRIMARY KEY`. El nombre de a `PRIMARY KEY` es siempre `PRIMARY`, por lo que no se puede utilizar como nombre para ningún otro tipo de índice. Si no tiene un `PRIMARY KEY` y una aplicación solicita el `PRIMARY KEY` en sus tablas, MySQL devuelve el primer `UNIQUE` índice que no tiene columnas `NULL` como `PRIMARY KEY`.

En la tabla creada, primero se coloca a **PRIMARY KEY**, seguido de todos los índices **UNIQUE** y luego los índices no únicos. Esto ayuda al optimizador de MySQL a priorizar qué índice usar y también más rápidamente y a detectar claves **UNIQUE** duplicadas .

PRIMARY KEY puede ser un índice de varias columnas. Sin embargo, no puede crear un índice de varias columnas utilizando el atributo clave **PRIMARY KEY** en una especificación de columna. Si lo hace, solo marcará esa columna como principal. Debe utilizar una cláusula separada . **PRIMARY KEY(key_part, ...)**

- **KEY | INDEX**

KEY es normalmente un sinónimo de **INDEX**. El atributo de clave **PRIMARY KEY** también se puede especificar **KEY** cuando se proporciona en una definición de columna. Esto se implementó por compatibilidad con otros sistemas de bases de datos.

- **UNIQUE**

Un índice **UNIQUE** crea una restricción tal que todos los valores del índice deben ser distintos. Se produce un error si intenta agregar una nueva fila con un valor clave que coincide con una fila existente. Para todos los motores, un índice **UNIQUE** permite múltiples valores **NULL** para las columnas que pueden contener **NULL**. Si especifica un valor de prefijo para una columna en un índice **UNIQUE**, los valores de columna deben ser únicos dentro de la longitud del prefijo.

- **FULLTEXT**

Un **FULLTEXT** índice es un tipo especial de índice que se utiliza para búsquedas de texto completo. Solo los motores de almacenamiento InnoDB y MyISAM admiten índices **FULLTEXT**. Ellos sólo se pueden crear a partir de columnas CHAR, VARCHAR y TEXT. La indexación siempre ocurre en toda la columna; la indexación de prefijo de columna no es compatible y cualquier longitud de prefijo se ignora si se especifica.

- **FOREIGN KEY**

MySQL admite claves externas, que le permiten realizar referencias cruzadas de datos relacionados entre tablas y restricciones de claves externas, que ayudan a mantener la coherencia de estos datos dispersos.

Las tablas particionadas que emplean el motor InnoDB de almacenamiento no admiten claves externas.

- ***key_part***
- Una especificación ***key_part*** puede terminar con **ASC** o **DESC** para especificar si los valores de índice se almacenan en orden ascendente o descendente. El valor predeterminado es ascendente si no se proporciona un especificador de orden.

Tipos de datos

Tabla de contenido

MySQL admite tipos de datos [SQL](#) en varias categorías: tipos numéricos, tipos de fecha y hora, tipos de cadenas (caracteres y bytes), tipos espaciales y el **JSON** tipo de datos. Este capítulo proporciona una descripción general y más detallada de las propiedades de los tipos en cada categoría, y un resumen de los requisitos de almacenamiento de tipos de datos. Las descripciones iniciales son intencionalmente breves. Consulte las descripciones más detalladas para obtener información adicional sobre tipos de datos particulares, como los formatos permitidos en los que puede especificar valores.

Las descripciones de los tipos de datos utilizan estas convenciones:

- Para los tipos enteros, ***M*** indica el ancho máximo de visualización. Para los tipos de coma flotante y de coma fija, ***M*** es el número total de dígitos que se pueden almacenar (la precisión). Para los tipos de cadena, ***M*** es la longitud máxima. El valor máximo permitido de ***M*** depende del tipo de datos.
- ***D*** se aplica a los tipos de coma flotante y de coma fija e indica el número de dígitos que siguen al punto decimal (la escala). El valor máximo posible es 30, pero no debe ser mayor que ***M***-2.
- ***Fsp*** se aplica a la **TIME**, **DATETIME** y **TIMESTAMP** los tipos y representa la precisión de fracciones de segundo; es decir, el número de dígitos que siguen al punto decimal en fracciones de segundos. El ***fsp*** valor, si se da, debe estar en el rango de 0 a 6. Un valor de 0 significa que no hay una parte fraccionaria. Si se omite, la precisión predeterminada es 0 (esto difiere del estándar SQL predeterminado de 6, por compatibilidad con versiones anteriores de MySQL).
- Los corchetes ([y]) indican partes opcionales de las definiciones de tipo.

Elegir el tipo correcto para una columna

Para un almacenamiento óptimo, debe intentar utilizar el tipo más preciso en todos los casos. Por ejemplo, si se usa una columna de números enteros para valores en el rango

de 1a 99999, **MEDIUMINT UNSIGNED** es el mejor tipo. De los tipos que representan todos los valores requeridos, este tipo utiliza la menor cantidad de almacenamiento.

Todos los cálculos básicos (+, -, *, y /) con **DECIMAL** columnas se realizan con precisión de 65 decimales (base 10) dígitos.

Si la precisión no es demasiado importante o si la velocidad es la máxima prioridad, el tipo **DOUBLE** puede ser lo suficientemente bueno. Para una alta precisión, siempre puede convertir a un tipo de punto fijo almacenado en un **BIGINT**. Esto le permite hacer todos los cálculos con números enteros de 64 bits y luego convertir los resultados a valores de punto flotante según sea necesario.

Tipos numéricos:

Existen tipos de datos numéricos, que se pueden dividir en dos grandes grupos, los que están en coma flotante (con decimales) y los que no.

TinyInt:

Es un número entero con o sin signo. Con signo el rango de valores válidos va desde -128 a 127. Sin signo, el rango de valores es de 0 a 255

Bit ó Bool:

Un número entero que puede ser 0 ó 1

SmallInt:

Número entero con o sin signo. Con signo el rango de valores va desde -32768 a 32767. Sin signo, el rango de valores es de 0 a 65535.

MediumInt:

Número entero con o sin signo. Con signo el rango de valores va desde -8.388.608 a 8.388.607. Sin signo el rango va desde 0 a 16777215.

Integer, Int:

Número entero con o sin signo. Con signo el rango de valores va desde -2147483648 a 2147483647. Sin signo el rango va desde 0 a 429.4967.295

BigInt:

Número entero con o sin signo. Con signo el rango de valores va desde -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807. Sin signo el rango va desde 0 a 18.446.744.073.709.551.615.

Float:

Número pequeño en coma flotante de precisión simple. Los valores válidos van desde -3.402823466E+38 a -1.175494351E-38, 0 y desde 1.175494351E-38 a 3.402823466E+38.

xReal, Double:

Número en coma flotante de precisión doble. Los valores permitidos van desde -1.7976931348623157E+308 a -2.2250738585072014E-308, 0 y desde 2.2250738585072014E-308 a 1.7976931348623157E+308

Decimal, Dec, Numeric:

Número en coma flotante desempaquetado. El número se almacena como una cadena

Tipo de Campo	Tamaño de Almacenamiento
TINYINT	1 byte
SMALLINT	2 bytes
MEDIUMINT	3 bytes
INT	4 bytes
INTEGER	4 bytes
BIGINT	8 bytes
FLOAT(X)	4 ú 8 bytes
FLOAT	4 bytes
DOUBLE	8 bytes
DOUBLE PRECISION	8 bytes
REAL	8 bytes
DECIMAL(M,D)	M+2 bytes sí D > 0, M+1 bytes sí D = 0
NUMERIC(M,D)	M+2 bytes if D > 0, M+1 bytes if D = 0

2 Tipos fecha:

A la hora de almacenar fechas, hay que tener en cuenta que Mysql no comprueba de una manera estricta si una fecha es válida o no. Simplemente comprueba que el mes esta comprendido entre 0 y 12 y que el día esta comprendido entre 0 y 31.

Date:

Tipo fecha, almacena una fecha. El rango de valores va desde el 1 de enero del 1001 al 31 de diciembre de 9999. El formato de almacenamiento es de año-mes-día

DateTime:

Combinación de fecha y hora. El rango de valores va desde el 1 de enero del 1001 a las 0 horas, 0 minutos y 0 segundos al 31 de diciembre del 9999 a las 23 horas, 59 minutos y 59 segundos. El formato de almacenamiento es de año-mes-día horas:minutos:segundos

TimeStamp:

Combinación de fecha y hora. El rango va desde el 1 de enero de 1970 al año 2037. El formato de almacenamiento depende del tamaño del campo:

Tamaño	Formato
14	AñoMesDiaHoraMinutoSegundo

	aaaammddhhmmss
12	AñoMesDiaHoraMinutoSegundo aammddhhmmss
8	ñoMesDia aaaammdd
6	AñoMesDia aammdd
4	AñoMes aamm
2	Año aa

Time:

Almacena una hora. El rango de horas va desde -838 horas, 59 minutos y 59 segundos a 838, 59 minutos y 59 segundos. El formato de almacenamiento es de 'HH:MM:SS'

Year:

Almacena un año. El rango de valores permitidos va desde el año 1901 al año 2155. El campo puede tener tamaño dos o tamaño 4 dependiendo de si queremos almacenar el año con dos o cuatro dígitos.

Tipo de Campo	Tamaño de Almacenamiento
DATE	3 bytes
DATETIME	8 bytes
TIMESTAMP	4 bytes
TIME	3 bytes
YEAR	1 byte

3 Tipos de cadena:

Char(n):

Almacena una cadena de longitud fija. La cadena podrá contener desde 0 a 255 caracteres.

VarChar(n):

Almacena una cadena de longitud variable. La cadena podrá contener desde 0 a 255 caracteres.

Dentro de los tipos de cadena se pueden distinguir otros dos subtipos, los tipo Text y los tipo BLOB (Binary large Object)

La diferencia entre un tipo y otro es el tratamiento que reciben a la hora de realizar ordenamientos y comparaciones. Mientras que el tipo text se ordena sin tener en cuenta las Mayúsculas y las minúsculas, el tipo BLOB se ordena teniéndolas en cuenta.

Los tipos BLOB se utilizan para almacenar datos binarios como pueden ser ficheros.

TinyText y TinyBlob:

Columna con una longitud máxima de 255 caracteres.

Blob y Text:

Un texto con un máximo de 65535 caracteres.

MediumBlob y MediumText:

Un texto con un máximo de 16.777.215 caracteres.

LongBlob y LongText:

Un texto con un máximo de caracteres 4.294.967.295. Hay que tener en cuenta que debido a los protocolos de comunicación los paquetes pueden tener un máximo de 16 Mb.

Enum:

Campo que puede tener un único valor de una lista que se especifica. El tipo Enum acepta hasta 65535 valores distintos

Set:

Un campo que puede contener ninguno, uno ó varios valores de una lista. La lista puede tener un máximo de 64 valores.

Tipo de campo	Tamaño de Almacenamiento
CHAR(n)	n bytes
VARCHAR(n)	n +1 bytes
TINYBLOB, TINYTEXT	Longitud+1 bytes
BLOB, TEXT	Longitud +2 bytes
MEDIUMBLOB, MEDIUMTEXT	Longitud +3 bytes
LOBLOB, LOBTEXT	Longitud +4 bytes
ENUM('value1','value2',...)	1 ó dos bytes dependiendo del número de valores
SET('value1','value2',...)	1, 2, 3, 4 ó 8 bytes, dependiendo del número de valores

Diferencia de almacenamiento entre los tipos Char y VarChar

Valor	CHAR(4)	Almacenamiento	VARCHAR(4)	Almacenamiento
"	"	4 bytes	"	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes