

# MySQL

## para Windows y Linux

2ª EDICIÓN



César Pérez

Alfaomega  Ra-Ma®

# SQL PARA MySQL. LENGUAJE DE CONSULTA DE DATOS: SENTENCIA SELECT

---

## INTRODUCCIÓN

Dentro del lenguaje SQL (*Structured Query Language*) que MySQL utiliza para tratar la información, suelen distinguirse varias categorías de sentencias (o comandos). El lenguaje de consulta de datos DQL (*Data Query Language*) permite básicamente obtener datos de tablas y especificar la forma en que se presentan. La sentencia esencial es SELECT. Se trata de la más sencilla de estas categorías y suele ser la primera en abordarse cuando se trata de aprender SQL.

Existen otras categorías de sentencias SQL como el Lenguaje de Modificación de Datos DML (*Data Modification Language*) cuyas sentencias se usan en la interrogación y manipulación de datos en esquemas de bases de datos ya existentes. Como ejemplos característicos tenemos las sentencias INSERT, UPDATE, DELETE y DROP.

Otras categorías de sentencias SQL son el Lenguaje de Definición de Datos DDL (*Data Definition Language*) y Lenguaje de Control de Datos DCL (*Data Control Language*). Las sentencias del lenguaje de definición de datos se utilizan para crear, alterar o borrar objetos de base de datos, tales como tablas, columnas y secuencias. Como ejemplos característicos tenemos las sentencias CREATE, ALTER y DROP. Las sentencias del lenguaje de control de datos se utilizan en el control de acceso a datos en la base de datos. Como ejemplos característicos tenemos los comandos GRANT y REVOKE. Estas últimas categorías de sentencias o comandos suelen utilizarse en tareas típicas de administración de la base de datos.

En este capítulo se tratan las sentencias del lenguaje de modificación de datos y en capítulos posteriores se tratarán las restantes categorías.

## LA SENTENCIA SELECT

En el capítulo 4 se introdujo una versión sencilla de la sintaxis de la sentencia SELECT. Básicamente dicha sintaxis puede resumirse como sigue:

```
SELECT [DISTINCT | ALL | Otras_opciones ] lista_selección  
[INTO OUTFILE 'nombre_fichero' opciones_exportación]  
[FROM tabla_origen [,tabla_origen]...]  
[WHERE condición_de_búsqueda]  
[LIMIT m,n]  
[GROUP BY expresión_de_agrupamiento]  
[HAVING condición_de_búsqueda]  
[ORDER BY expresión_order_by [ ASC | DESC ] ]
```

La sentencia SELECT recupera filas o columnas (las especificadas en *lista\_selección*) de una o más tablas especificadas en la cláusula FROM que satisfagan las condiciones de búsqueda especificadas en WHERE, agrupadas según la expresión de agrupamiento de GROUP BY, cumpliendo las condiciones de búsqueda de HAVING y ordenadas según lo especificado en ORDER BY. ALL significa que se devolverán todas las filas que satisfagan las condiciones de WHERE (es el valor por defecto) y DISTINCT significa que las filas duplicadas sólo se devolverán una vez. DISTINCT es sinónimo de DISTINCTROW.

La sentencia SELECT obtiene filas de la base de datos y permite realizar la selección de una o varias filas o columnas de una o varias tablas. La sintaxis completa de la instrucción SELECT es compleja. Sus cláusulas principales se analizarán en los párrafos siguientes.

### Cláusula SELECT

La cláusula SELECT especifica las columnas que va a devolver la consulta. En cuanto a las opciones de esta cláusula ya sabemos que ALL especifica que pueden aparecer filas duplicadas en el conjunto de resultados (valor predeterminado) y que DISTINCT especifica que sólo pueden aparecer filas exclusivas en el conjunto de resultados. Los valores NULL se consideran iguales a efectos de la palabra clave DISTINCT. Pueden usarse otras opciones menos importantes como HIGH\_PRIORITY, que da a la instrucción SELECT una mayor prioridad sobre otras sentencias como INSERT o UPDATE que pueden estar siendo utilizadas por algún otro cliente sobre las mismas tablas. Otra opción posible es STRAIGHT\_JOIN, que fuerza a las tablas seleccionadas a unirse en el orden citado en la cláusula FROM. Otras dos opciones que suelen usarse son SQL\_BIG\_RESULT y SQL\_SMALL\_RESULT, que especifican que el resultado obtenido en la sentencia SELECT se presentará de modo más extenso (con más información) o más resumido, respectivamente.

En la cláusula **SELECT**, a continuación de las opciones, se sitúa la lista de selección que incluye las columnas que se van a seleccionar para el conjunto de resultados separadas por comas. De modo más general, la lista de selección puede ser una serie de expresiones separadas por comas donde cada expresión puede ser un nombre de columna, constante, función o una combinación de nombres de columnas, constantes y funciones conectados mediante operadores.

La lista de selección también puede ser la expresión **\***, que especifica que se deben devolver todas las columnas de todas las tablas de la cláusula **FROM**. Las columnas se devuelven por tabla, tal como se especifique en la cláusula **FROM**, en el orden en que se encuentran en la tabla. La lista de selección también puede ser la expresión *nombre\_tabla* .**\***, que limita el alcance de **\*** a la tabla especificada. Al especificar las columnas en la lista de selección es conveniente situar los nombres de las tablas antes de los de las columnas separadas por un punto (calificar) para evitar realizar una referencia ambigua, como ocurre cuando dos tablas de la cláusula **FROM** tienen columnas de idéntico nombre.

Por lo tanto, la lista de selección también puede contener los nombres de las columnas expresadas como *nombre\_tabla* .*nombre\_columna*.

Detrás del nombre de cualquier columna de la lista de selección puede situarse un alias para la columna mediante *nombre\_columna* **AS** *alias\_columna*. La sintaxis *alias\_columna* es un nombre alternativo para reemplazar el nombre de la columna en el conjunto de resultados de la consulta. Los alias también se utilizan para especificar nombres para los resultados de expresiones. La sintaxis *alias\_columna* puede utilizarse en una cláusula **ORDER BY**, **GROUP BY** o **HAVING**; sin embargo, no puede utilizarse en una cláusula **WHERE**.

## Cláusula **INTO**

La cláusula **INTO OUTFILE** permite escribir el resultado de la instrucción **SELECT** en el archivo *nombre\_fichero*. Su sintaxis es:

```
[INTO OUTFILE 'nombre_fichero' opciones_exportación]
```

El nombre del archivo (que no debe existir previamente) se interpreta usando las mismas reglas que se aplican cuando se leen archivos con la sentencia **LOAD DATA** vista en el capítulo anterior. El desarrollo de la sintaxis de la cláusula *opciones\_exportación* es igual que para la cláusula *opciones\_importación* de la sentencia **LOAD DATA** vista en el capítulo anterior.

En definitiva, la sentencia **INTO OUTFILE** permite recoger en ficheros, de forma adecuada, los resultados de las consultas.

## Cláusula FROM

Esta cláusula especifica las tablas de donde se van a obtener las filas. La cláusula FROM es necesaria excepto cuando la lista de selección sólo contiene constantes, variables, expresiones aritméticas (no nombres de columna) y expresiones del tipo \*.

Su sintaxis básica es la siguiente:

```
[ FROM lista_tablas ]
```

Lo habitual es que *lista\_tablas* sean varios nombres de tablas separados por comas, en cuyo caso SELECT devuelve todas las combinaciones posibles de filas desde dichas tablas.

Pero los elementos de *lista\_tablas* pueden estar separados por cláusulas de unión (en vez de por comas). Estas cláusulas pueden ser JOIN, CROSS JOIN, INNER JOIN, STRAIGHT\_JOIN y LEFT JOIN. **El argumento JOIN indica que las tablas especificadas en FROM deben combinarse.** También pueden aparecer mezcladas las comas y las cláusulas de unión. Algunas sintaxis particulares más detalladas para *lista\_tablas* podrían ser cualquiera de las siguientes:

```
nombre_tabla [CROSS] JOIN nombre_tabla
nombre_tabla INNER JOIN nombre_tabla
nombre_tabla STRAIGHT_JOIN nombre_tabla
nombre_tabla LEFT [OUTER]JOIN nombre_tabla ON expresión_condicional
nombre_tabla LEFT [OUTER]JOIN nombre_tabla USIGN (lista_columnas)
nombre_tabla NATURAL LEFT [OUTER]JOIN nombre_tabla
nombre_tabla [CROSS] JOIN nombre_tabla
```

LEFT JOIN recupera filas desde tablas unidas, pero obliga a generar una fila por cada fila en la tabla izquierda, incluso si no hay equivalente en la tabla derecha. Cuando no tiene pareja, las columnas de la tabla derecha se devuelven como valores NULL. Las filas emparejadas pueden determinarse de acuerdo con la condición especificada en las cláusulas ON *expresión\_condicional* o USIGN (*lista\_columnas*). La expresión condicional es una expresión que se puede usar en la cláusula WHERE y la lista de columnas contiene nombres de columnas separadas por comas que deben aparecer en las dos tablas unidas.

LEFT OUTER JOIN equivale a LEFT JOIN y NATURAL LEFT JOIN equivale a LEFT JOIN USING (*lista\_columnas*), donde los nombres de la lista de columnas son comunes en ambas tablas.

CROSS JOIN especifica el producto resultante de dos tablas. Devuelve las filas que se devolverían si las tablas que se combinan se indicaran en la cláusula FROM, sin especificar ninguna cláusula WHERE.

INNER JOIN especifica que se devuelvan todos los pares de filas coincidentes y descarta las filas no coincidentes de las dos tablas. Éste es el valor predeterminado si no se especifica ningún tipo de combinación.

STRAIGHT\_JOIN fuerza al optimizador a unir las tablas en el orden en que se citan.

En algunas versiones de SQL se usan LEFT, RIGHT y FULL para especificar las uniones izquierda, derecha y completa de las tablas especificadas. LEFT [ OUTER ] especifica que todas las filas de la tabla de la izquierda que no cumplan la condición especificada se incluyan en el conjunto de resultados, además de todas las filas que devuelva la combinación interna. Las columnas de salida de la tabla de la izquierda se establecen a NULL. RIGHT [ OUTER ] especifica que todas las filas de la tabla de la derecha que no cumplan la condición especificada se incluyan en el conjunto de resultados, además de las que devuelva la combinación interna. Las columnas de salida de la tabla de la derecha se establecen a NULL. FULL [ OUTER ] especifica que si una fila de la tabla de la izquierda o de la derecha no coincide con los criterios de selección, la fila se incluya en el conjunto de resultados y las columnas de resultados que corresponden a la otra tabla se establezcan como NULL. Se trata de una adición a todas las filas que normalmente devuelve la combinación interna.

En un capítulo posterior se tratará la unión de tablas de forma extensa.

## Cláusula WHERE

Especifica una condición de búsqueda para restringir las filas que se van a devolver. Su sintaxis es la siguiente:

```
[ WHERE < condición_búsqueda > | < nombre_columna { * = | = * }  
nombre_columna > ]
```

El argumento *condición de búsqueda* limita las filas devueltas en el conjunto de resultados mediante el uso de predicados. No hay límite en el número de predicados que se pueden incluir en una condición de búsqueda. El argumento *nombre\_columna { \* = | = \* } nombre\_columna* especifica una combinación externa con la sintaxis antigua específica SQL Server y la cláusula WHERE. Se utiliza el operador \*= para especificar una combinación externa izquierda y el operador =\* para especificar una combinación externa derecha.

El conjunto de resultados puede estar limitado por HAVING y LIMIT.

## ADMINISTRACIÓN DE CUENTAS DE USUARIO

El administrador de la base de datos es el encargado de establecer las cuentas de usuario de MySQL, definiendo qué usuarios se pueden conectar al servidor, desde dónde pueden conectarse y qué tareas pueden hacer cuando están conectados.

Existen en SQL de MySQL dos sentencias vitales que regulan la administración de cuentas de usuario. La sentencia **GRANT** crea usuarios y regula sus privilegios y la sentencia **REVOKE** elimina los privilegios. Las sentencias **GRANT** y **REVOKE** afectan a cuatro tablas de privilegios, denominadas *tablas de transferencia*, cuyos nombres son: *user* (contiene los usuarios que pueden conectar con el servidor con cualquier privilegio que tengan), *db* (contiene los privilegios de nivel de base de datos), *tables\_priv* (contiene los privilegios de nivel de tabla) y *columns\_priv* (contiene los privilegios de nivel de columna). Hay otra tabla de transferencia llamada *host*, pero que no es afectada por **GRANT** ni **REVOKE**. Cuando se usa **GRANT** para un usuario, se crea una entrada para ese usuario en la tabla *user*, y si la sentencia especifica cualquier privilegio global (administrativo o aplicable a todas las bases de datos) también se registran en *user*. Si se especifica una base de datos, tabla o privilegios de columna, se registran en las tablas *db*, *tables\_priv* y *columns\_priv*.

### Sentencia GRANT

La sentencia **GRANT** concede privilegios de acceso a uno o más usuarios de MySQL que se crean. Su sintaxis es la siguiente:

```
GRANT tipo_priv [(lista_col)] [, tipo_priv [lista_col]]...
ON {*. * | nombre_db.* | nombre_db. nombre_tabla | nombre_tabla}
TO usuario IDENTIFIED BY "contraseña"
    [, usuario IDENTIFIED BY "contraseña"]...
[WITH GRANT OPTION]
```

El argumento *tipo\_priv* especifica los privilegios que se concederán y *lista\_col* especifica las columnas (separadas por comas) a las que se concederán estos privilegios en su caso. La tabla siguiente resume los tipos de privilegios que se aplican a bases de datos, tablas y columnas.

<i>Especificador de privilegio</i>	<i>Operación permitida por el privilegio</i>
ALTER	Asigna permisos para alterar tablas e índices.
CREATE	Asigna permisos para crear bases de datos y tablas.
DELETE	Asigna permisos para borrar registros de tablas.
DROP	Asigna permisos para eliminar bases de datos y tablas.
INDEX	Asigna permisos para crear o eliminar índices.
INSERT	Asigna permisos para insertar valores en tablas.
SELECT	Asigna permisos para ver la información de las tablas.
UPDATE	Asigna permisos para actualizar tablas.

La tabla siguiente resume los tipos de privilegios administrativos.

<i>Especificador de privilegio</i>	<i>Operación permitida por el privilegio</i>
FILE	Asigna permisos para leer y escribir archivos en el servidor.
PROCESS	Asigna permisos para revisar información sobre los hilos ejecutados o eliminados en el servidor.
RELOAD	Asigna permisos para recargar las tablas cedidas o vaciar los registros, la caché <i>host</i> o la caché de tablas.
SHUTDOWN	Asigna permisos para cerrar el servidor.
ALL (u ALL PRIVILEGES)	Asigna todos los permisos.
USAGE	Privilegio especial “sin privilegios”.

La cláusula ON especifica el nivel (amplitud) a que se aplican los privilegios. Los privilegios pueden ser globales aplicables a todas las bases de datos y a todas las tablas, específicos de una base de datos o específicos de tabla. La siguiente tabla especifica los posibles niveles a que se aplican los privilegios.

<i>Especificador de privilegio</i>	<i>Nivel al que se aplica el privilegio</i>
*.*	Privilegios globales (todas las bases de datos y tablas).
*	Privilegios globales (sin base de datos por defecto).
nombre_bd.*	Privilegios a nivel de la base de datos para todas las tablas nombradas en la base de datos.
nombre_bd.nombre_tabla	Privilegios a nivel de tabla: todas las columnas que se nombran en la tabla.
Nombre_tabla	Privilegios a nivel de tabla: todas las columnas que se nombran en la tabla en la base de datos.

Cuando en una tabla se utiliza la cláusula ON, los privilegios se pueden hacer por columna específica nombrando una o más columnas separadas por comas en la cláusula *lista\_col*.

La cláusula TO especifica el usuario para quien son los privilegios. Cada *usuario* consiste en un nombre de usuario y un nombre de *host* mediante la especificación *nombre\_usuario@nombre\_host* y opcionalmente una cláusula IDENTIFIED BY que asigna una contraseña al usuario y que debe especificarse en texto plano.

En el ejemplo siguiente se crea el usuario *cesar*, que puede acceder a todas las tablas en la base de datos *samp\_db* de cualquier host con la contraseña *secret*.

```
GRANT ALL ON samp_db.* TO cesar IDENTIFIED BY "secret"
```

En el ejemplo siguiente se crea un usuario de nombre *cesar* con privilegios de sólo lectura para las tablas en la base de datos *base* pudiendo conectarse con cualquier host en el dominio *xyz.com* (se usará el símbolo % como comodín para indicar todos los hosts).

```
GRANT SELECTION ON base.* TO cesar@%.xyz.com
```

En el ejemplo siguiente se crea un usuario con todos los privilegios, pero sólo para la tabla *member* en la base datos *sampdb*. El usuario, de nombre *cesar* puede conectarse sólo desde el host *venus.planeta.net* con la contraseña *galaxia*.

```
GRANT ALL ON sam_db.member TO cesar@venus.planeta.net IDENTIFIED BY galaxia
```

En el ejemplo siguiente se crea un usuario de nombre *supercesar* que pueda hacerlo todo, incluyendo garantizar privilegios a otros usuarios, pero que deba conectarse desde el host local con la contraseña *super*.

```
GRANT ALL ON *.* TO supercesar@localhost IDENTIFIED BY super WHIT GRANT OPTION
```

La cláusula **WITH GRANT OPTION** permite al usuario darle a otros usuarios cualquier privilegio transferido por la sentencia **GRANT**.

En el ejemplo siguiente se crea un usuario anónimo de la base de datos *base* que se puede conectar desde el host local sin contraseña.

```
GRANT ALL ON base.* TO localhost
```

En el ejemplo siguiente se conceden privilegios de selección, modificación, borrado y actualización de datos en todas las tablas de la base de datos *samp\_db* al usuario de nombre *cesar* conectado desde cualquier host con la contraseña *secret*.

```
GRANT SELECT, INSERT, DELETE, UPDATE ON samp_db.* TO cesar@% IDENTIFIED BY secret
```

Hay que tener presente que los nombres de usuario, las contraseñas y los nombres de bases de datos y tablas son sensibles a mayúsculas y minúsculas. Los nombres de host y columnas no lo son.

## ***Sentencia REVOKE***

La sentencia **REVOKE** se utiliza para revocar privilegios y eliminar usuarios. Su sintaxis es la siguiente:

```
REVOKE tipo_priv [(lista_col)] [, tipo_priv [lista_col]]...  
ON { *.* | * | nombre_db.* | nombre_db. nombre_tabla | nombre_tabla }  
FROM usuario [, usuario]...
```

Los argumentos de esta sentencia son idénticos a los de la sentencia **GRANT**.

**REVOKE** no elimina al usuario de la tabla, por lo que éste puede seguir conectado a MySQL. Para eliminar totalmente al usuario hay que borrar manualmente todas sus entradas en las tablas concedidas.

En el ejemplo siguiente se revocan todos los privilegios para el usuario *supercesar* en el host local.

```
REVOKE ALL ON *.* FROM supercesar@localhost
```

En el ejemplo siguiente se revocan los privilegios que permiten al usuario de nombre *cesar* modificar la tabla *member* en la base datos *sampdb* cuando se conecta desde el host *venus.planeta.net*.

```
REVOKE INSERT, DELETE, UPDATE ON sam_db.member FROM cesar@venus.planeta.net
```

En el ejemplo siguiente se revocan los privilegios de la tabla de nombre *tabla* de la base de datos de nombre *base* de un usuario anónimo del host local.

```
REVOKE ALL ON base.tabla FROM " " @localhost
```

Ya sabemos que **REVOKE** no elimina al usuario de la tabla, por lo que este puede seguir conectada a MySQL. Para eliminar totalmente al usuario hay que borrar manualmente todas sus entradas en las tablas concedidas de la siguiente forma:

```
shell> mysql -u root mysql
mysql> DELETE FROM user
      -> WHERE User = "nombre_usuario" and Host = "nombre_host";
mysql> FLUSH PRIVILEGES
```

La sentencia **DELETE** elimina la entrada del usuario y la sentencia **FLUSH** indica al servidor que recargue las tablas de concesión.

## Contraseñas mediante **SET PASSWORD**

La sentencia **SET** mediante la opción **PASSWORD** permite establecer contraseñas para usuarios dados. Estos usuarios se especifican mediante *nombre\_usuario@nombre\_host* igual que en las sentencias **GRANT** y **REVOKE**. La sintaxis es la siguiente:

```
SET PASSWORD [FOR usuario] = PASSWORD("contraseña")
```

Si no se especifica la cláusula **FOR**, la contraseña se establece para el usuario actual.

En el ejemplo siguiente se establece la contraseña "cigarros-puros" para el usuario de nombre Hill al acceder a cualquier host del dominio tabacos.com.

```
SET PASSWORD FOR bill@%.tabacos.com = PASSWORD("cigarros-puros")
```

En el ejemplo siguiente se establece la contraseña "secreto" para el usuario actual.

```
SET PASSWORD = PASSWORD("secreto")
```