Introducción al lenguaje SQL

Tras la exposición del modelo relacional por parte de E.E Codd en 1970, dos compañeros del IBM Research Center, Donald D. Chamberlin y Raymond E Boyce, definieron cuatro años más tarde un lenguaje de programación gracias al que interactuar con la información contenida en una base de datos. Dada la similitud de dicho lenguaje con la lengua cotidiana, lo llamaron SEQUEL (Structured English Query Language, lenguaje de consulta estructurado en inglés). Debido a problemas de copyright, el nombre se cambió a SQL.

SQL es un lenguaje de alto nivel con el que comunicarse con un SGBD. No solamente sirve para realizar consultas, altas, modificaciones y borrado de datos, sino también para definir la estructura de la base de datos, declarar sus objetos y elementos (usuarios, perfiles, backups, temporizaciones) y establecer elementos de seguridad.

Es importante ubicar SQL como lenguaje de programación:

- Es un lenguaje no procedimental, ya que se le especifica qué es lo que se quiere obtener sin entrar en el cómo (no se escriben instrucciones secuenciales y precisas, tan solo se indica qué acción se quiere realizar, y es tarea del propio lenguaje encontrar el camino adecuado para ejecutarla).
- Solo sirve para ejecutar consultas sobre bases de datos. Carece de estructuras de control (if, while, for), por lo que no es un lenguaje Turing completo.

Las compañías que desarrollan SGBDR han ido ampliado notablemente el ámbito de SQL, acercándose y alejándose del estándar según la evolución de este y del mercado. En este sentido, han introducido una sintaxis propietaria que, si bien simplifica ciertas instrucciones u ofrece nuevas funcionalidades, dificulta la migración de bases de datos entre distintos sistemas gestores y crea dependencia por parte de sus clientes.

Las funcionalidades de SQL se suelen dividir en tres sublenguajes:

- DDL (data definition language, lenguaje de definición de datos). Permite crear la estructura de los elementos de la base de datos (tablas, índices, la propia base de datos).

- DML (data manipulation language, lenguaje de manipulación de datos). Opera sobre la in formación contenida en la base de datos (valores de los campos y registros) Al DML corresponde la inserción, modificación, borrado y consulta de los datos.
- DCL (data control language, lenguaje de control de datos). Se encarga de gestionar los permisos de los usuarios y los perfiles, así como la integridad de las transacciones sobre información de la base de datos.

Tipos de datos MySql

Dada la gran diversidad de SQBDR en el mercado, cada uno de ellos se ha acercado o alejado al estándar en determinados aspectos. A continuación veremos los tipos de datos que emplea MySQL y que, en algunos casos, amplian la funcionalidad de los tipos definidos inicialmente en el lenguaje SQL. Estos tipos de datos están además, sujetos a cambios dependiendo de las modificaciones de las distintas versiones de MySQL existentes.

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 a16777215.

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

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 está 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-dia

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-dia 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	AñoMesDia 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

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 Test 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 test 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
LONGBLOB, LONGTEXT	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'	4 bytes
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes

Creación de Bases de datos en MySql. Sintaxis

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

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

El comando "CREATE DATABASE" o "CREATE SCHEMA" se utiliza para crear una nueva base de datos o esquema en un sistema de gestión de bases de datos.

La sintaxis del comando es la siguiente:

CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] nombre_de_la_base_de_datos [opciones_de_creación] ...

La opción "IF NOT EXISTS" es opcional y se utiliza para evitar que se cree la base de datos si ya existe una con el mismo nombre.

Las opciones de creación también son opcionales y se utilizan para personalizar la base de datos. Puedes utilizar las siguientes opciones:

- "CHARACTER SET [=] nombre_del_juego_de_caracteres": Define el conjunto de caracteres que se utilizará para almacenar y mostrar los datos en la base de datos.
- "COLLATE [=] nombre_del_juego_de_ordenamiento": Define el juego de ordenamiento que se utilizará para clasificar los datos en la base de datos.
- "ENCRYPTION [=] {'Y' | 'N'}": Habilita o deshabilita la encriptación de la base de datos. Si se habilita, los datos almacenados estarán encriptados y protegidos.

Esta sintaxis te permitirá crear bases de datos o esquemas personalizados en tu sistema de gestión de bases de datos.

Ejemplo:

```
Create database if not exists prueba1
Character_set latin1
Collate latin1_spanish_ci
Encryption ‘N’;
```

Para acceder a una base de datos y poder realizar operaciones en ella utilizaremos la orden:

```
Use {nombre_de_la_basededatos}
```

Para eliminar una base de datos emplearemos la orden

```
Drop database {nombre de la base de datos}
```