



1º DAW

U.T. 4

El lenguaje SQL como DDL



Pablo Berciano Posada

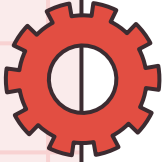


Tabla de contenidos

01

**Lenguaje de
definición de datos**

02

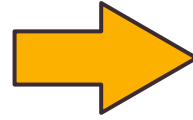
Gestión de tablas

03

Gestión de índices

04

Gestión de vistas



01

Lenguaje de definición de datos





¿Qué es SQL?



El lenguaje de consulta estructurada (**SQL**, siglas en inglés de Structured Query Language) es un **lenguaje de programación** utilizado para trabajar con **bases de datos relacionales**.

Es un estándar **ANSI** y es ampliamente utilizado en aplicaciones de bases de datos de todo tipo, desde sistemas de gestión de bases de datos empresariales hasta aplicaciones más simples como Microsoft Access.





Evolución histórica del lenguaje SQL

En 1974, **IBM** desarrolló el lenguaje **SEQUEL** (Structured English Query Language), utilizado para trabajar con su base de datos de investigación de sistemas de información. Cinco años después, en 1979, el estándar **ANSI** aprobó una versión mejorada de SEQUEL, cambiando su nombre a **SQL**.

En **1986**, ANSI aprobó la **primera versión oficial** de SQL. Posteriormente, en 1992, se aprobó la segunda versión, conocida como SQL-92 o **SQL2**, que incorporó soporte para transacciones, vistas y procedimientos almacenados. En 1999, se aprobó la tercera versión, SQL-99 o **SQL3**, la cual introdujo triggers (disparadores), tipos de datos avanzados y soporte para objetos y métodos.



Evolución histórica del lenguaje SQL

En 2003, ANSI actualizó la versión SQL-99, creando **SQL:2003**, que incluyó soporte para **XML** y mejoras en transacciones y vistas. Tres años después, en 2006, la versión **SQL:2006** agregó **mejoras adicionales** en el soporte para XML, arreglos y tablas temporales.

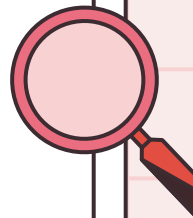
En 2011, la versión **SQL:2011** mejoró el soporte para transacciones, vistas, tablas temporales y arreglos. En 2016, ANSI aprobó **SQL:2016**, con mejoras en el soporte para **JSON** y operaciones de **recopilación de datos en grandes conjuntos** de datos.

La última versión del estándar SQL, **SQL:2023**, añade soporte para consultas de grafos, mejora la manipulación de datos JSON y agrega funciones para facilitar cálculos avanzados, manteniendo a SQL relevante en la **gestión de datos moderna**.



¿Qué es el DDL?

El **lenguaje de definición de datos** (DDL, siglas en inglés de Data Definition Language) es un conjunto de comandos utilizados en SQL para **definir y modificar la estructura de las bases de datos**. El DDL se utiliza para crear, modificar y eliminar objetos del SGBD, como bases de datos, tablas, índices y vistas. Los comandos DDL incluyen instrucciones como:

- **CREATE:** Crea objetos en la base de datos.
 - **ALTER:** Modifica la estructura de objetos ya creados, como puede ser, agregar una columna a una tabla.
 - **DROP:** Elimina objetos de la base de datos.
 - **TRUNCATE:** Borra todos los registros de una tabla de manera rápida, sin afectar su estructura.
- 



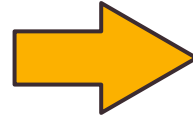
El catálogo de datos



El **catálogo de datos** en SQL es un conjunto de tablas que contienen **información sobre la estructura de la base de datos y los objetos que contiene**. El catálogo de datos se utiliza para almacenar **metadatos**, es decir, información sobre los datos. Para acceder en MySQL:

- **SHOW TABLES;** : Muestra información sobre las tablas de la base de datos.
- **SHOW COLUMNS FROM nombre_tabla;** : Contiene información sobre las columnas de una tabla de la base de datos.
- **SHOW INDEX FROM nombre_tabla;** : Contiene información sobre los índices de las tablas de la base de datos.
- **SHOW CREATE TABLE nombre_tabla;** : Muestra información sobre la creación de la tabla y sus restricciones actuales.

El catálogo de datos se utiliza principalmente para **facilitar la administración** de la base de datos y para proporcionar información sobre la estructura de la base de datos a las aplicaciones que la utilizan.



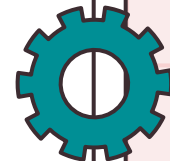
02

Gestión de tablas





Bases de datos



Las sentencias de DDL relacionadas con bases de datos en SQL permiten crear, modificar y eliminar bases de datos en un sistema gestor de bases de datos (SGBD). Estas sentencias principales son:

- **CREATE DATABASE nombre:** Crea una nueva base de datos en el sistema.
- **ALTER DATABASE nombre SET opción:** Modifica las propiedades de una base de datos existente. Las modificaciones permitidas dependen del sistema.
- **DROP DATABASE nombre:** Elimina una base de datos y todos sus datos y objetos de manera irreversible. Esta acción debe usarse con cautela.

Estas sentencias de DDL permiten gestionar el ciclo de vida de las bases de datos en SQL.

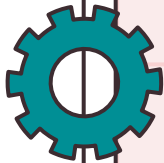
Sin embargo, las opciones específicas de cada sentencia pueden variar según el SGBD, como MySQL, SQL Server, PostgreSQL, u Oracle.



Definición de tablas

En SQL, se puede crear una tabla utilizando el comando **CREATE TABLE**. Este comando toma la siguiente forma general:

```
CREATE TABLE [IF NOT EXISTS] [esquema].nombre_tabla (  
    <nombre_columna> <tipo_dato> <restricciones_columna>,  
    <nombre_columna> <tipo_dato> <restricciones_columna>,  
    ...  
    <restricciones>  
    ...);
```





Tipos de datos

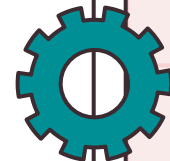


En MySQL, los **tipos de datos** se dividen en varias categorías según el tipo de información que almacenan. A continuación se presentan los tipos de datos principales:

- **Tipos de datos numéricos:** estos pueden ser **enteros** como: SMALLINT, INTEGER, BIGINT, SERIAL y BIGSERIAL; o **decimales** como: REAL, DOUBLE y NUMERIC/DECIMAL. Con UNSIGNED para eliminar los negativos.
- **Tipos de datos de cadenas de texto:** como CHAR, VARCHAR y TEXT. El tipo BLOB es similar a TEXT pero para datos binarios.
- **Tipos de datos de fechas y horas:** incluyendo DATE, TIME, DATETIME, TIMESTAMP e INTERVAL. Cuentan con funciones CURRENT para obtener el dato al momento.
- **Tipos de datos booleanos:** el tipo BOOLEAN se utiliza para representar valores de verdadero o falso.
- **Tipos de datos especiales:** como ENUM y SET que permite almacenar una lista predefinida de valores.



Restricciones en las columnas



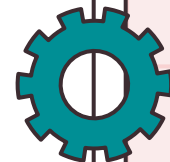
Dentro de la definición de una columna se pueden añadir **restricciones aplicables a la columna** que se está definiendo. Las restricciones son elementos que se utilizan en SQL para garantizar la **integridad de los datos** en una base de datos.

Para comenzar, es posible establecer una **longitud máxima** para los campos de tipo texto y los de tipo numérico decimal. La longitud se especifica entre paréntesis detrás del tipo de dato de la columna con un número. Dependiendo del tipo de dato el número puede implicar algo diferente, por ejemplo:

- nombre CHAR (20) → deja 20 caracteres para el nombre que ocupa siempre
- direccion VARCHAR(100) → deja 100 caracteres máximos para el campo y solo ocupa realmente los necesarios
- salario_mensual NUMERIC (4,2) → define que el campo tendrá una longitud de 4 dígitos significativos, de los cuales, 2 serán posiciones decimales.



Restricciones en las columnas



Después del tipo de dato se pueden indicar las siguientes **restricciones**:

- **NOT NULL**: Indica que la columna no puede tener un valor NULL.
- **DEFAULT**: Especifica un valor predeterminado para la columna. Si se inserta una fila en la tabla y no se especifica un valor para esta columna, se utilizará el valor predeterminado. **AUTO_INCREMENT** permite, para claves primarias numéricas, que por defecto aumente su valor de uno en uno.
- **UNIQUE**: Indica que el valor de la columna debe ser único en toda la tabla.
- **PRIMARY KEY**: Indica que la columna es la clave primaria de la tabla. Solo se puede utilizar para claves compuestas por una sola columna.
- **FOREIGN KEY**: Indica que la columna es una clave externa que se refiere a la clave primaria de otra tabla.
- **CHECK**: Especifica una condición que debe cumplirse para que el valor de la columna sea válido.



Restricciones en la tabla

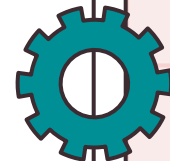


Se pueden añadir restricciones en sentencias fuera de la definición de la columna. Las principales son:

- **PRIMARY KEY:** Indica las columnas que forman la clave primaria de la tabla. Solo se puede utilizar una vez por tabla por lo que en caso de claves múltiples se debe hacer en una sentencia propia. Deben estar definidas las columnas previamente y se indicará como una lista: **PRIMARY KEY(columna1, columna2, ...)**
- **FOREIGN KEY:** Indica que la columna es una clave ajena que se refiere a la clave primaria de otra tabla. Para poder añadir una clave ajena debe estar ya definida la tabla a la que se hace referencia y la columna que es clave ajena. Se define de la siguiente forma:
FOREIGN KEY (columna_esta_tabla)
REFERENCES otra_tabla(columna_otra_tabla)
ON DELETE <Accion1> ON UPDATE <Accion2>



Modificación de tablas



El comando **ALTER TABLE** se utiliza para **modificar la estructura de una tabla existente**. Este comando permite agregar, eliminar o modificar columnas de la tabla, así como agregar o eliminar restricciones de clave y otros elementos de la tabla.

ALTER TABLE nombre_tabla [acciones_a_realizar];

Las acciones a realizar sobre la tabla pueden ser:

- **ADD:** Agrega una nueva columna a la tabla.
- **DROP:** Elimina una columna existente de la tabla.
- **MODIFY:** Modifica la definición de una columna existente.
- **RENAME:** Cambia el nombre de una columna existente.
- **ADD CONSTRAINT:** Agrega una nueva restricción a la tabla.
- **DROP CONSTRAINT:** Elimina una restricción existente de la tabla.



Eliminación de tablas



El comando **DROP TABLE** se utiliza para **eliminar una tabla** de la base de datos. Este comando toma la siguiente forma:

DROP TABLE nombre_tabla;

El comando **TRUNCATE TABLE** se utiliza para **eliminar todos los datos de una tabla**. A diferencia del comando **DELETE**, que elimina fila por fila, el comando **TRUNCATE TABLE** elimina todos los datos de la tabla de manera más rápida y eficiente.

TRUNCATE TABLE nombre_tabla;

Es importante tener en cuenta que estos comandos eliminan permanentemente la tabla y todos los datos contenidos en ella, por lo que se deben utilizar con cautela.

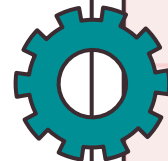


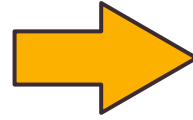
Cambio de nombre de tablas

El comando **RENAME TABLE** se utiliza para **cambiar el nombre de una tabla**. Este comando toma la siguiente forma:

```
RENAME TABLE nombre_tabla_actual TO nuevo_nombre_tabla;
```

Es importante tener en cuenta que el comando **RENAME TABLE** cambia el nombre de la tabla en la base de datos, pero **no modifica la estructura de la tabla ni los datos que contiene**.





03

Gestión de índices





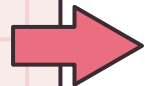
Índices



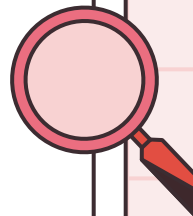
Un **índice** en SQL es una estructura de datos que se utiliza para **mejorar el rendimiento de las búsquedas** en una tabla de una base de datos. Un índice **crea una copia** de los datos de una o varias columnas de una tabla y los **ordena** de manera lógica, lo que permite realizar búsquedas más rápidas y eficientes en los datos de la tabla.

Cuando se realiza una **consulta** en una tabla que tiene un índice, el motor de evaluación de consultas de la base de datos **utiliza el índice** para encontrar rápidamente los datos que se solicitan en la consulta, en lugar de tener que escanear toda la tabla. Esto **puede mejorar significativamente el rendimiento de consulta** y hacerla mucho más rápida.

Hay que tener cautela a la hora de crear índices, ya que al copiar de forma íntegra las columnas de las que está formado se emplea una **gran cantidad de almacenamiento**. Por lo tanto es importante evaluar cuáles son los índices **necesarios**.



Definición y eliminación de índices

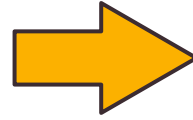


Se pueden **crear índices** en las tablas de una base de datos utilizando el comando **CREATE INDEX**. Este comando permite crear un índice en una tabla de manera que se puedan realizar búsquedas más rápidas y eficientes en los datos de la tabla.

```
CREATE INDEX nombre_indice  
ON nombre_tabla (columna1, columna2, ...);
```

Para eliminar un índice se utiliza el comando **DROP INDEX**. Toma la siguiente forma:

```
DROP INDEX nombre_indice  
ON nombre_tabla;
```



04

Gestión de vistas





Vistas



Una **vista** en SQL es una **tabla virtual** que se crea a partir de una consulta **SELECT**. Las vistas se utilizan para **presentar los datos de una base de datos de manera diferente a como están almacenados** en la base de datos, o para combinar datos de varias tablas de manera más sencilla.

Las vistas son útiles porque permiten a los usuarios **acceder** a los datos de una base de datos de manera más **sencilla** y sin tener que preocuparse por la **estructura** de las tablas subyacentes. Además, las vistas son muy útiles para **proteger la privacidad** de los datos, ya que solo muestran los datos que se han incluido en la vista y ocultan los datos restantes.



Creación y eliminación de vistas

Para crear una vista en SQL, se utiliza el comando **CREATE VIEW**. Este comando toma la siguiente forma:

CREATE VIEW nombre_vista AS consulta;

Para eliminar una vista en SQL, se utiliza el comando **DROP VIEW**.

DROP VIEW nombre_vista;

Es importante tener en cuenta que al eliminar una vista no elimina los datos subyacentes de la base de datos, sino que sólo elimina la forma de acceder a ellos a través de la vista.

