¿Qué es SQL?

SQL es un lenguaje utilizado en bases de datos relacionales para consultar u obtener datos. Su nombre proviene de las siglas en inglés *Structured Query Language*, que significa "lenguaje de consulta estructurado". En otras palabras, SQL es un lenguaje que se emplea para interactuar con los datos almacenados en una base de datos.

Pero, ¿qué son los datos? ¿Y qué es una base de datos?

Los datos son una colección de hechos que pueden presentarse en forma de palabras, números o incluso imágenes. Son uno de los activos más valiosos para cualquier organización, ya que se recopilan y utilizan en casi todos los ámbitos. Por ejemplo, tu banco almacena datos sobre ti, como tu nombre, dirección, número de teléfono y números de cuenta. Lo mismo ocurre con las compañías de tarjetas de crédito y plataformas como PayPal.
Debido a su importancia, los datos deben mantenerse seguros y ser accesibles rápidamente. La solución para esto es una base de datos.

Las bases de datos están en todas partes y se utilizan a diario, aunque muchas veces pasen desapercibidas.
Una **base de datos** es un repositorio de datos, es decir, un programa que almacena información. Además, permite agregar, modificar y consultar esos datos.

Existen diferentes tipos de bases de datos, según las necesidades específicas. Los datos pueden almacenarse de distintas formas, pero cuando se organizan en tablas (similares a hojas de cálculo con filas y columnas), hablamos de una **base de datos relacional**.

Las **columnas** de una tabla contienen propiedades del elemento registrado, como el apellido, nombre, dirección de correo electrónico, ciudad, etc.
Una **tabla** es un conjunto de elementos relacionados, como una lista de empleados o de autores de libros.

En una base de datos relacional, es posible establecer relaciones entre diferentes tablas.
Un conjunto de herramientas de software que permite gestionar una base de datos se llama **sistema de gestión de bases de datos**, o **DBMS** por sus siglas en inglés.

Términos como *base de datos*, *servidor de base de datos*, *sistema de base de datos*, *servidor de datos* y *sistema de gestión de bases de datos* suelen usarse indistintamente. Cuando nos referimos específicamente a bases de datos relacionales, el término adecuado es **sistema de gestión de bases de datos relacionales**, o **RDBMS** (*Relational Database Management System*).

Un **RDBMS** es un conjunto de herramientas de software que controla el acceso, la organización y el almacenamiento de los datos. Sirve como la columna vertebral de muchas aplicaciones en sectores como la banca, el transporte y la salud.

Algunos ejemplos de sistemas RDBMS son:

- MySQL
- Oracle Database
- DB2 Warehouse
- DB2 en la nube

Para la mayoría de los usuarios, trabajar con bases de datos se resume en cinco comandos fundamentales:

1. **Crear** una tabla
2. **Insertar** datos en la tabla
3. **Seleccionar** datos desde la tabla
4. **Actualizar** datos existentes
5. **Eliminar** datos de la tabla

Declaración SELECT

El propósito principal de un sistema de gestión de bases de datos no es solo almacenar los datos, sino también facilitar su recuperación.
Una vez que se ha creado una tabla en una base de datos relacional y se han insertado datos en ella, es común querer visualizar esa información. Para hacerlo, utilizamos la instrucción **SELECT**.

La instrucción SELECT pertenece al **lenguaje de manipulación de datos** (DML, por sus siglas en inglés). Las declaraciones DML se utilizan para **leer** y **modificar** datos.
Cuando ejecutamos una instrucción SELECT, estamos realizando una **consulta**, y el resultado de dicha consulta se conoce como **conjunto de resultados** o **tabla de resultados**.

En su forma más simple, una sentencia SELECT se escribe así:

*SELECT * FROM nombre_de_tabla;*

El asterisco (`*`) indica que deseamos recuperar **todas las columnas** de la tabla especificada.

--- ---

Supongamos que tenemos una entidad llamada **libro**. Creamos la tabla utilizando el nombre de la entidad (`libro`) y sus atributos (como `id`, `título`, `autor`, etc.) como columnas.
Luego, insertamos datos en la tabla agregando filas mediante la instrucción `INSERT`.

Cuando usamos la instrucción:

*SELECT * FROM libro;*

el conjunto de resultados mostrará **todas las filas** y **todas las columnas** de la tabla `libro`.

También es posible **especificar columnas individuales** en la instrucción SELECT si no deseamos recuperar toda la información. Por ejemplo, si solo nos interesa ver el ID del libro y su título, escribimos:

SELECT book_ID, title FROM libro;

Este comando devolverá únicamente esas dos columnas para cada fila de la tabla.
Es importante destacar que **el orden de las columnas** en el resultado será el mismo que se indique en la instrucción SELECT.

Ahora bien, ¿qué ocurre si queremos ver solo el libro cuyo ID es `'B1'`?
Para eso utilizamos la cláusula **WHERE**, que nos permite **restringir el conjunto de resultados** aplicando una condición.

La cláusula WHERE siempre requiere un **predicado**, es decir, una condición que se evalúa como **verdadera**, **falsa** o **desconocida**.
Por ejemplo, si queremos obtener el título del libro cuyo ID es `'B1'`, usamos la siguiente consulta:

SELECT book_ID, title FROM libro WHERE book_ID = 'B1';

El resultado será una sola fila que cumpla con esa condición.

Se utilizó el operador de comparación `=` (igual a). Sin embargo, un sistema de gestión de bases de datos relacional (RDBMS) admite varios operadores de comparación, tales como:

- `=` : igual a
- `>` : mayor que
- `<` : menor que
- `>=` : mayor o igual que
- `<=` : menor o igual que
- `<>` o `!=` : distinto de

Estos operadores permiten construir condiciones más complejas y precisas al consultar los datos.

Funciones: COUNT, DISTINCT y LIMIT

Al trabajar con bases de datos, es común necesitar funciones que nos permitan contar registros, eliminar duplicados o limitar la cantidad de resultados. SQL proporciona funciones integradas como `COUNT`, `DISTINCT` y `LIMIT` para estos propósitos.

`COUNT`: *Recuento de filas*

La función `COUNT` permite contar la cantidad de filas que cumplen con ciertos criterios.
Por ejemplo, si queremos obtener el número total de filas en una tabla:

SELECT COUNT(*) FROM nombre_de_tabla;

Ejemplo:

Supongamos que tenemos una tabla llamada `Metales`, con una columna llamada `País`. Si deseamos saber cuántas veces **Canadá** aparece como país receptor de metales:

```
SELECT COUNT(País) FROM Metales WHERE País = 'Canadá';
```

Este comando contará cuántas filas tienen el valor `'Canadá'` en la columna `País`.

DISTINCT: Eliminar duplicados

La cláusula `DISTINCT` se utiliza para obtener **valores únicos** de una columna, es decir, eliminar duplicados del conjunto de resultados.

```
SELECT DISTINCT nombre_columna FROM nombre_de_tabla;
```

Ejemplo:

Si queremos obtener una lista de países únicos que han recibido medallas de **oro** en la tabla `Metales`, podemos usar:

```
SELECT DISTINCT País FROM Metales WHERE TipoMetal = 'Oro';
```

Esto devolverá una lista sin países repetidos, aunque un mismo país haya recibido medallas de oro varias veces.

LIMIT: Restringir la cantidad de resultados

La cláusula `LIMIT` nos permite especificar cuántas filas queremos recuperar. Esto es útil cuando se trabaja con conjuntos de datos muy grandes y solo se quiere inspeccionar una parte.

```
SELECT * FROM nombre_de_tabla LIMIT cantidad;
```

Ejemplo:

Si queremos ver solo las primeras **10** filas de la tabla `Metales`:

```
SELECT * FROM Metales LIMIT 10;
```

También podemos combinar `LIMIT` con una condición. Por ejemplo, para ver solo las primeras **5** filas del año **2018**:

```
SELECT * FROM Metales WHERE Año = 2018 LIMIT 5;
```

--- ---

Ejemplos prácticos con otra tabla (FilmLocations)

```
-- Cuenta todas las filas de la tabla FilmLocations
SELECT COUNT(*) FROM FilmLocations;

-- Cuenta cuántas veces aparece una ubicación asociada al guionista James
Cameron
SELECT COUNT(Locations) FROM FilmLocations WHERE Writer = 'James
Cameron';

-- Recupera todos los títulos únicos de películas en la tabla
SELECT DISTINCT Title FROM FilmLocations;

-- Muestra las primeras 25 filas de la tabla
SELECT * FROM FilmLocations LIMIT 25;
```

Declaración INSERT

Después de crear una tabla en una base de datos, es necesario llenarla con datos. Para esto, utilizamos la declaración `INSERT`, la cual se emplea para **agregar nuevas filas** a una tabla.

La sentencia `INSERT` pertenece al **Lenguaje de Manipulación de Datos** (DML, por sus siglas en inglés), que incluye aquellas instrucciones utilizadas para leer y modificar datos en una base de datos.

Sintaxis básica

```
INSERT INTO nombre_de_tabla (columna1, columna2, ..., columnaN)
VALUES (valor1, valor2, ..., valorN);

--nombre_de_tabla: es el nombre de la tabla donde se insertarán los
datos.
--La lista de columnas indica el orden en que se insertarán los valores.
--La cláusula VALUES contiene los datos que se insertarán en cada
columna.
```

--- ---

Ejemplo

Supongamos que tenemos una entidad llamada `Autor`, y ya hemos creado una tabla con las siguientes columnas:

- `Autor_ID`

- Apellido
- Nombre
- Correo
- Ciudad
- País

Para insertar una nueva fila con los datos de **Raoul Chong**, la sentencia sería:

```
INSERT INTO Autor (Autor_ID, Apellido, Nombre, Correo, Ciudad, País)
VALUES ('A1', 'Chong', 'Raoul', 'rfc@ibm.com', 'Toronto', 'CA');
```

Es importante que el número de columnas coincida exactamente con el número de valores. Esto asegura que cada columna reciba un dato.

Inserción múltiple

No es necesario insertar una sola fila por sentencia. También se pueden insertar **varias filas al mismo tiempo**, separando cada conjunto de valores por comas:

```
INSERT INTO Autor (Autor_ID, Apellido, Nombre, Correo, Ciudad, País)
VALUES
  ('A1', 'Chong', 'Raoul', 'rfc@ibm.com', 'Toronto', 'CA'),
  ('A2', 'Ahuja', 'Rav', 'rav@ibm.com', 'Ottawa', 'CA');
```

Esto permite ahorrar tiempo y simplificar el código cuando necesitamos añadir múltiples registros a una tabla.

Sentencias UPDATE y DELETE

Después de crear una tabla y llenarla con datos, es posible modificarlos utilizando la sentencia **`UPDATE`**. Esta sentencia hace parte del Lenguaje de Manipulación de Datos (DML), que se emplea para consultar y modificar la información almacenada en las tablas.

Por ejemplo, supongamos que ya se creó una tabla llamada `autor`, con sus respectivos atributos como columnas, y se insertaron varias filas.

En algún momento, podrías necesitar cambiar la información de uno o varios registros. Para ello, se utiliza la sintaxis de la sentencia `UPDATE`:

Sintaxis básica

```
UPDATE nombre_de_tabla
SET columna1 = valor1, columna2 = valor2
WHERE condición;

--nombre_de_tabla: Indica la tabla que se va a actualizar.
-- columna1, columna2: representan las columnas cuyos valores deseas
cambiar.
```

-- condición: especifica qué filas deben actualizarse. Si no se incluye una cláusula WHERE, todas las filas de la tabla serán modificadas.

Ejemplo:

Supón que queremos cambiar el nombre y apellido del autor con ID `A2`, reemplazando "Rav Ahuja" por "Lakshmi Kata". La sentencia sería:

```
UPDATE autor
SET lastname = 'Kata', firstname = 'Lakshmi'
WHERE author_id = 'A2';
```

Luego, para verificar el cambio con:

```
SELECT * FROM autor;
```

Sentencia `DELETE`

También puede ser necesario eliminar uno o más registros de una tabla. Para eso, se utiliza la sentencia `DELETE`, cuya sintaxis es:

```
DELETE FROM nombre_de_tabla
WHERE condición;
```

La condición indica qué filas deben eliminarse. Si omites la cláusula WHERE, se eliminarán todas las filas de la tabla.

Ejemplo:

Si deseas eliminar las filas correspondientes a los autores con ID `A2` y `A3`, puedes hacerlo con:

```
DELETE FROM autor
WHERE author_id IN ('A2', 'A3');
```

Ejemplo completo en otra tabla (`Instructor`):

```
-- Insertar un nuevo instructor
INSERT INTO Instructor(ins_id, lastname, firstname, city, country)
VALUES (7, 'Cangiano', 'Antonio', 'Vancouver', 'CA');

-- Ver todos los registros
SELECT * FROM Instructor;


-- Actualizar la ciudad del instructor con ID 1
UPDATE Instructor
SET city = 'Markham'
```

```
WHERE ins_id = 1;

-- Verificar cambios
SELECT * FROM Instructor;


-- Eliminar al instructor llamado Hima
DELETE FROM Instructor
WHERE firstname = 'Hima';

-- Verificar cambios
SELECT * FROM Instructor;
```