SQL

Índice

Tipos de consultas

En SQL existen 2 tipos de consultas:

DDL *(Data Definition Language)*

Son las consultas que se utilizan para definir la estructura de la base de datos. Por ejemplo, crear tablas, modificarlas, eliminarlas, etc. Algunos ejemplos son:

- `CREATE TABLE`: Crea una tabla.
- `ALTER TABLE`: Modifica una tabla.
- `DROP TABLE`: Elimina una tabla.
- `TRUNCATE TABLE`: Elimina todos los registros de una tabla.
- `CREATE DATABASE`: Crea una base de datos.

DML *(Data Manipulation Language)*

Son las consultas que se utilizan para manipular los datos de la base de datos. Por ejemplo, insertar, modificar, eliminar, etc. Algunos ejemplos son:

- `SELECT`: Selecciona datos de una tabla.
- `INSERT`: Inserta datos en una tabla.
- `UPDATE`: Actualiza datos de una tabla.
- `DELETE`: Elimina datos de una tabla.

Consultas

SELECT

Esta consulta se utiliza para seleccionar datos de una tabla. Es posible seleccionar todos los campos de una tabla o solo algunos. Además, es posible filtrar las filas que se desean seleccionar.

```
SELECT column1, column2, ...
FROM table_name;
```

* (*wildcard operator*)

Si se desea seleccionar todos los campos de una tabla, se puede utilizar el símbolo `*`:

```
SELECT *
FROM table_name;
```

WHERE

Si se desea filtrar las filas que se desean seleccionar, se puede utilizar la cláusula `WHERE`:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Operadores de comparación

La condición puede ser una comparación entre dos campos, por ejemplo:

```
SELECT *
FROM table_name
WHERE column1 = column2;
```

También puede ser una comparación entre un campo y un valor, por ejemplo:

```
SELECT *
FROM table_name
WHERE column1 = 'value';
```

También es posible utilizar operadores lógicos, por ejemplo:

```
SELECT *
FROM table_name
WHERE column1 > 'value' AND column2 < 'value2';
```

Operadores de comparación complejos

Existen operadores de comparación específicos para cadenas de texto, por ejemplo `LIKE`:

```
SELECT *
FROM table_name
WHERE column1 LIKE 'value%';
```

El símbolo `%` indica que puede haber cualquier cadena de texto. En este caso, se seleccionarían todas las filas cuyo valor del campo `column1` empiece por `value` seguido de cualquier cadena de texto.

Otro ejemplo de `LIKE`:

```
SELECT *
FROM table_name
WHERE column1 LIKE '%value%';
```

En este caso, se seleccionarían todas las filas cuyo valor del campo `column1` contenga la cadena de texto `value`. Ya que puede haber cualquier cadena de texto antes y después de `value`, incluyendo que no haya nada.

También es posible utilizar operadores de comparación específicos para fechas, por ejemplo `BETWEEN`:

```
SELECT *
FROM table_name
```

```
WHERE column1 BETWEEN '2020-01-01' AND '2020-12-31';
```

En este caso, se seleccionarían todas las filas cuyo valor del campo `column1` esté entre las fechas `2020-01-01` y `2020-12-31`.

`IN`

Otro operador de comparación muy útil es `IN`:

```
SELECT *
FROM table_name
WHERE column1 IN ('value1', 'value2', 'value3');
```

En este caso, se seleccionarían todas las filas cuyo valor del campo `column1` sea `value1`, `value2` o `value3`.

Este operador es especialmente potente cuando se utiliza junto con otra consulta dentro de los paréntesis:

```
SELECT *
FROM table_name
WHERE column1 IN (SELECT column1 FROM table_name2);
```

En este caso, se seleccionarían todas las filas cuyo valor del campo `column1` esté dentro de los resultados de la consulta `SELECT column1 FROM table_name2`.

ORDER BY

Si se desea ordenar los resultados de una consulta, se puede utilizar la cláusula `ORDER BY`:

```
SELECT *
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

El orden de los campos es importante, ya que se ordenarán por el primero, y en caso de empate, por el segundo, y así sucesivamente. Además, se puede especificar si se desea ordenar de forma ascendente o descendente.

LIMIT

Si se desea limitar el número de resultados de una consulta, se puede utilizar la cláusula `LIMIT`:

```
SELECT *
FROM table_name
LIMIT number;
```

En este caso, se seleccionarían los primeros `number` resultados de la consulta.

OFFSET

Si se desea saltarse los primeros resultados de una consulta, se puede utilizar la cláusula `OFFSET`:

```
SELECT *
FROM table_name
OFFSET number;
```

En este caso, se seleccionarían todos los resultados de la consulta, excepto los primeros `number` resultados.

GROUP BY

Si se desea agrupar los resultados de una consulta, se puede utilizar la cláusula `GROUP BY`:

```
SELECT column1, column2, ...
FROM table_name
GROUP BY column1;
```

En este caso, se agruparán los resultados por el valor de los campos `column1`. Esto quiere decir que si hay varias filas con el mismo valor de `column1`, se agruparán en una sola fila.

COUNT

Si se desea contar el número de resultados de una consulta, se puede utilizar la función `COUNT`:

```
SELECT COUNT(*)
FROM table_name;
```

En este caso, se devolvería el número de filas de la tabla `table_name`.

SUM

Si se desea sumar los valores de una columna de una consulta, se puede utilizar la función `SUM`:

```
SELECT SUM(column1)
FROM table_name;
```

En este caso, se devolvería la suma de los valores de la columna `column1` de la tabla `table_name`.

INSERT

Para insertar una nueva fila en una tabla, se puede utilizar la sentencia `INSERT`:

```
INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);
```

En este caso, se insertaría una nueva fila en la tabla `table_name` con los valores `value1`, `value2`, `...` en los campos `column1`, `column2`, `...` Es importante que el orden de los campos y los valores coincida. Además, si no se especifican los campos, se insertarán los valores en el mismo orden en el que se creó la tabla.

UPDATE

Para actualizar una fila de una tabla, se puede utilizar la sentencia `UPDATE`:

```
UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition;
```

En este caso, se actualizarían los valores de los campos `column1`, `column2`, `...` de la tabla `table_name` para aquellas filas que cumplan la condición `condition`.

DELETE

Para eliminar una fila de una tabla, se puede utilizar la sentencia `DELETE`:

```
DELETE FROM table_name WHERE condition;
```

En este caso, se eliminarían las filas de la tabla `table_name` que cumplan la condición `condition`. No se especifican los campos, ya que se eliminarían las filas completas que cumplan la condición.

JOIN

Para unir dos tablas, se puede utilizar la sentencia `JOIN`:

```
SELECT *
FROM table_name1
JOIN table_name2 ON table_name1.column1 = table_name2.column2;
```

En este caso, se unirían las tablas `table_name1` y `table_name2` por el campo `column1` de la primera tabla y el campo `column2` de la segunda tabla. Esto quiere decir que se seleccionarían todas las filas de la primera tabla, y para cada una de ellas, se seleccionarían todas las filas de la segunda tabla que cumplan la condición `table_name1.column1 = table_name2.column2`. Por lo tanto, si hay varias filas en la segunda tabla que cumplan la condición, se repetirán las filas de la primera tabla tantas veces como filas cumplan la condición en la segunda tabla.

Por ejemplo, si tenemos las tablas:

id	empresa

id	empresa
1	Google
2	Apple
3	Amazon

id	nombre	empresa
1	Juan	1
2	Pedro	1
3	María	2
4	Marta	3
5	Carlos	3

Si ejecutamos la consulta:

```
SELECT empresas.*, empleados.nombre, empleados.id AS id_empleado
FROM empresas
JOIN empleados ON empresas.id = empleados.empresa;
```

Obtendríamos los resultados:

id	empresa	nombre	id_empleado
1	Google	Juan	1
1	Google	Pedro	2
2	Apple	María	3
3	Amazon	Marta	4
3	Amazon	Carlos	5

Notar que la columna `id` de la tabla `empresas` se ha repetido tantas veces como filas de la tabla `empleados` cumplan la condición `empresas.id = empleados.empresa`. Además, fue necesario especificar el alias `id_empleado` para la columna `id` de la tabla `empleados`, ya que ambas tablas tienen una columna `id`, y si no se especifica el alias, se produciría un error de ambigüedad.

Tipos de JOIN

Existen varios tipos de `JOIN`:

- `INNER JOIN`: Devuelve las filas que tienen valores comunes en ambas tablas.
- `LEFT JOIN`: Devuelve todas las filas de la tabla de la izquierda, y las filas de la tabla de la derecha que tienen valores comunes. Si no hay valores comunes, se rellenan los campos de la tabla de la derecha con `NULL`.
- `RIGHT JOIN`: Devuelve todas las filas de la tabla de la derecha, y las filas de la tabla de la izquierda que tienen valores comunes. Es equivalente a un `LEFT JOIN` pero intercambiando las tablas.
- `FULL JOIN`: Devuelve todas las filas de ambas tablas, y las filas que tienen valores comunes. **No es soportado por algunos motores de base de datos.**

Aclaraciones

Este es un resumen de los comandos más utilizados en SQL. No es una guía completa, ya que no se incluyen todos los comandos disponibles, ni se explican todos los detalles de cada uno de ellos. Para obtener más información, se puede consultar la documentación oficial.

No todos los motores de base de datos soportan todos los comandos y si bien la sintaxis es estándar en la mayoría de los casos, puede variar en algunos detalles. Por ejemplo, algunos motores de base de datos no soportan el comando `FULL JOIN`, otros no soportan el comando `UPDATE` sin especificar la condición `WHERE`, otros soportan `TOP` en lugar de `LIMIT`, etc.

En este resumen se hizo hincapié en las consultas de `DML`, pero también existen las consultas de `DDL` (Data Definition Language) para crear, modificar y eliminar tablas, campos, índices, etc. Estas consultas no se explican en este resumen, pero se pueden consultar en la documentación oficial. También existen las consultas de `DCL` (Data Control Language) para dar permisos a usuarios, etc. Estas consultas no se explican en este resumen, pero se pueden consultar en la documentación oficial.