Proyecto

Sistema de gestión escolar

Glosario

**INTRODUCCIÓN**

Tipos de instrucciones

**SQL**

se divide en [cuatro sublenguajes](https://medium.com/lucid-plexus/exploring-the-different-sub-languages-in-sql-bf42709c6bba) según su finalidad, DDL, DCL, DML y DQL.

**DDL**: En primer lugar, Data Definition Language, o Lenguaje de Definición de Datos, es el lenguaje con el que se definen las estructuras que almacenan los datos. Es decir, se definen las creaciones, modificaciones y eliminaciones de tablas, bases de datos, vistas…

**DCL:** El Data Control Language, o Lenguaje de Control de Datos permite al administrador controlar los permisos de acceso a los objetos.

**DML:** Data Manipulation Language, o Lenguaje de Manipulación de Datos. Este es el lenguaje que permite introducir, modificar y borrar datos.

**DQL:** Finalmente, el Data Query Language o Lenguaje de Consultas de datos, permite al usuario realizar consultas a la base de datos.

Operadores

Para completar nuestros **comandos SQL**, podemos hacer uso de los siguientes operadores:

Operadores Lógicos

* **AND:**Devuelve true solo si ambas condiciones son ciertas.
* **OR:** Devuelve true si alguna de las dos condiciones es cierta.
* **NOT:** Negación. Devuelve el valor contrario de la expresión.

**Operadores de comparación**

* **<** : Menor que
* **>**: Mayor que
* **<>** : Distinto de
* **<=** : Menor o igual que
* **>=** : Mayor o igual que
* **BETWEEN**: Intervalo
* **LIKE**: Comparación
* **IN**: Especificación
* **DISTINCT**: Eliminar duplicados
* **IS NULL:** Comprueba si tiene valor nulo

**Funciones de agregado**

Estas se usan dentro de una sentencia SELECT, se usan en grupo y devuelven un único valor.

* **AVG**: Para calcular el valor promedio.
* **COUNT**: Cuenta el número de registros.
* **SUM**: Suma los valores.
* **MAX**: Devuelve el máximo de los valores.
* **MIN**: Devuelve el mínimo de los valores.

**Comandos SQL: Listado completo**

Aquí tienes el **listado completo de comandos SQL**:

* **SELECT**: Permite definir qué datos quieres recibir de la consulta.
  + **SELECT \*:** Seleccionar todos los datos.
  + **SELECT INTO**: Copiar valores en otra tabla.
  + **SELECT TOP X**: Selecciona el X porcentaje superior de la tabla. Por ejemplo, SELECT TOP 50 PERCENT \* FROM Empleados.
  + **SELECT DISTINCT**: Selecciona los valores sin duplicados.

SELECT Nombre, Apellido, Salario

FROM Empleados;

SELECT \*

FROM Empleados;

SELECT Nombre, Apellido, Salario

INTO EmpleadosCopia

FROM Empleados;

SELECT TOP 50 PERCENT \*

FROM Empleados;

SELECT DISTINCT DepartamentoID

FROM Empleados;

* **AS**: Renombra las tablas o columnas con un alias.

SELECT Nombre AS PrimerNombre, Apellido AS SegundoNombre

FROM Empleados AS E;

* **FROM**: Especifica la tabla o tablas sobre la que queremos hacer la consulta. En caso de seleccionar más de una tabla, se deben relacionar. Por ejemplo:

SELECT nombre, departamento

FROM empleados, departamentos

WHERE empleados.id\_dept = departamentos.id;

* **WHERE**: Para definir los filtros de la consulta.

SELECT \*

FROM Empleados

WHERE Salario > 60000;

* **CREATE**: Para crear bases de datos, tablas, vistas o índices.

CREATE TABLE Proyectos (

ProyectoID int NOT NULL,

NombreProyecto varchar(255) NOT NULL,

EmpleadoID int,

FechaInicio date,

CONSTRAINT PK\_Proyectos PRIMARY KEY (ProyectoID)

);

* **DROP**: Para eliminar bases de datos, tablas o índices.

DROP TABLE Proyectos;

* **UPDATE**: Para alterar los datos de la tabla.

UPDATE Empleados

SET Salario = Salario \* 1.10

WHERE DepartamentoID = 1;

* **DELETE**: Para eliminar los datos de la tabla.

DELETE FROM Empleados

WHERE EmpleadoID = 5;

* **ALTER TABLE**: Permite añadir o eliminar columnas en una tabla.

ALTER TABLE Empleados

ADD FechaContratacion date;

* **RENAME**: Permite renombrar un objeto existente de la base de datos.

ALTER TABLE Empleados

RENAME TO Personal;

* **LOCK**: Impide consultas y cambios en un objeto.

LOCK TABLES Empleados WRITE;

* **GROUP BY**: Define en base a que se agrupan los datos en las funciones de agregación.

SELECT DepartamentoID, AVG(Salario) AS SalarioPromedio

FROM Empleados

GROUP BY DepartamentoID;

* **HAVING**: Del mismo modo que WHERE, define los filtros de la consulta en funciones de agregación.

SELECT DepartamentoID, AVG(Salario) AS SalarioPromedio

FROM Empleados

GROUP BY DepartamentoID

HAVING AVG(Salario) > 60000;

* **ORDER BY**: Define con base a que se ordenan los datos. Por ejemplo: ORDER BY DATE ASC ordenará los resultados respecto a la fecha en orden ascendente.

SELECT \*

FROM Empleados

ORDER BY Salario DESC;

* **OFFSET:** Funciona con ORDER BY, establece cuantas filas de la respuesta se debe saltar.

SELECT \*

FROM Empleados

ORDER BY Salario DESC

OFFSET 2 ROWS;

* **FETCH**: Establece cuantas filas devuelve después de usar OFFSET.

SELECT \*

FROM Empleados

ORDER BY Salario DESC

OFFSET 2 ROWS

FETCH NEXT 3 ROWS ONLY;

* **CASE**: Para sentencias condicionales, se usa con SELECT. Por ejemplo:

SELECT Nombre, Apellido,

CASE

WHEN Salario > 60000 THEN 'Alta'

WHEN Salario BETWEEN 40000 AND 60000 THEN 'Media'

ELSE 'Baja'

END AS RangoSalario

FROM Empleados;

* **LIMIT**: Así, podemos especificar el límite de registros que queremos obtener.

SELECT \*

FROM Empleados

LIMIT 5;

* **ROUND**: Al especificar el nombre de columna y un número de decimales, redondea los valores de esa columna al número de decimales establecido. Por ejemplo, ROUND(salario,2) redondeará el salario a dos decimales.

SELECT Nombre, Salario, ROUND(Salario, 2) AS SalarioRedondeado

FROM Empleados;

* **JOINS**:
  + **INNER JOIN**: Selecciona los registros que coinciden en las dos tablas.
  + **LEFT JOIN**: Selecciona los registros de la tabla de la izquierda que coinciden con los de la tabla derecha.
  + **RIGHT JOIN:** Al contrario que con LEFT JOIN, este selecciona los registros de la tabla de la derecha que coinciden con los de la tabla izquierda.
  + **FULL JOIN:** Selecciona los registros que tienen alguna coincidencia en alguna de las dos tablas.

SELECT E.Nombre, E.Apellido, D.NombreDepartamento

FROM Empleados E

INNER JOIN Departamentos D ON E.DepartamentoID = D.DepartamentoID;

SELECT E.Nombre, E.Apellido, D.NombreDepartamento

FROM Empleados E

LEFT JOIN Departamentos D ON E.DepartamentoID = D.DepartamentoID;

SELECT E.Nombre, E.Apellido, D.NombreDepartamento

FROM Empleados E

RIGHT JOIN Departamentos D ON E.DepartamentoID = D.DepartamentoID;

SELECT E.Nombre, E.Apellido, D.NombreDepartamento

FROM Empleados E

FULL JOIN Departamentos D ON E.DepartamentoID = D.DepartamentoID;

* **EXISTS:** Para comprobar la existencia de un registro en una subconsulta.

SELECT Nombre, Apellido

FROM Empleados E

WHERE EXISTS (SELECT 1 FROM Departamentos D WHERE D.DepartamentoID = E.DepartamentoID);

* **GRANT**: Para que el administrador pueda dar permisos sobre objetos.

GRANT SELECT, INSERT ON Empleados TO 'usuario';

* **REVOKE**: Al contrario que GRANT, para quitar permisos sobre objetos.

REVOKE SELECT, INSERT ON Empleados FROM 'usuario';

* **WITH**: Permite guardar el resultado de una consulta temporalmente utilizando un alias.

WITH EmpleadosAltos AS (

SELECT Nombre, Apellido, Salario

FROM Empleados

WHERE Salario > 60000

)

SELECT \*

FROM EmpleadosAltos;

* **SAVEPOINT**: Recuerda el punto actual como copia de seguridad para poder volver atrás.

SAVEPOINT MiPunto;

* **COMMIT**: Para confirmar acciones en la base de datos, una vez hecho COMMIT, se pierden los SAVEPOINT.

COMMIT;

* **ROLLBACK**: Para retroceder sobre acciones que aún no están volcadas en la base de datos. Puedes hacer ROLLBACK a un COMMIT o ROLLBACK anteriormente definido, o a un SAVEPOINT.

ROLLBACK TO MiPunto;

* **TRUNCATE**: Elimina todos los registros de una tabla, pero mantiene la tabla y la estructura.

TRUNCATE TABLE Empleados;

* **UNION**: Combina diferentes SELECT y elimina registros duplicados. Por ejemplo esta consulta devuelve las ciudades de los clientes y proveedores, eliminando registros repetidos.

SELECT Ciudad FROM Clientes

UNION

SELECT Ciudad FROM Proveedores;

* **UNION ALL**: Combina diferentes SELECT y mantiene registros duplicados.

SELECT Ciudad FROM Clientes

UNION ALL

SELECT Ciudad FROM Proveedores;

* **AGGREGATOR SYNC AUTO\_INCREMENT**: Establece el contador de autoincremento en los agregadores. Por ejemplo, en este ejemplo establece el valor del contador de autoincremento de la tabla Empleados a 1000.

ALTER TABLE Empleados AUTO\_INCREMENT = 1000;

* **BACKUP DATABASE:** Para crear copias de seguridad de la base de datos.

BACKUP DATABASE NombreBaseDatos

TO DISK = 'C:\backup\NombreBaseDatos.bak';

* **BEGIN**: Hace un COMMIT al estado actual y después comienza una nueva transacción.

BEGIN TRANSACTION;

* **CALL**: Ejecuta un procedimiento almacenado, pero no devuelve nada.

CALL ActualizarSalarios();