SQL

OPERADORES BASICOS

- aritméticos: +, -, *, /, %
- comparación: =, !=, <>, <, >, <=, >=
- lógicos: and, or, not
- otros: between, in, like, is null

CREAR TABLAS

```
CREATE TABLE NombreTabla (
    Columna1 TipoDato [CONSTRAINT],
    Columna2 TipoDato,
```

INSERT INTO

Se utiliza para insertar registros dentro de una tabla.

```
INSERT INTO NombreTabla (Columna1, ...)
VALUES (Valor1, ...);
```

METADATA (INFORMATION_SCHEMA)

Se utiliza para consultar información de la base de datos.

- Para tablas ----->

```
SELECT TABLE_NAME
FROM INFORMATION_SCHEMA.TABLES;
```

- Para columnas ---->

```
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'NombreTabla';
```

WHERE

Filtra registros según una condición.

```
SELECT Columna1, Columna2
FROM NombreTabla
WHERE Condicion;
```

ORDER BY

Ordena los resultados de una consulta

```
ELECT Columna1, Columna2
ROM NombreTabla
RDER BY Columna1 [ASC|DESC
```

OPERADORES LOGICOS

OPERADOR	EJEMPLO
IS NULL	verifica si un valor es nulo.

```
SELECT * FROM Empleados WHERE FechaIngreso IS NULL;
```

LIKE	Busca coincidencias por patrón

```
SELECT * FROM NombreTabla WHERE Columna LIKE 'Patron%';
```

IN	Verifica si un valor está dentro de un conjunto.

```
SELECT * FROM NombreTabla WHERE Columna IN (Valor1, Valor2,
```

BETWEEN	Verifica si un valor está dentro de un rango

JOIN

Se utiliza para combinar registros de dos o más tablas

INNER JOIN: devuelve solo coincidencias.

```
SELECT t1.Columna, t2.Columna
FROM Tabla1 t1
INNER JOIN Tabla2 t2 ON t1.ColumnaClave = t2.ColumnaClave;
```

LEFT JOIN: devuelve todos los registros de la izquierda (aunque no haya coincidencias)

```
SELECT t1.Columna, t2.Columna
FROM Tabla1 t1
LEFT JOIN Tabla2 t2 ON t1.ColumnaClave = t2.ColumnaClave;
```

RIGHT JOIN: Devuelve todos los registros de la derecha (aunque no haya coincidencia)

```
SELECT t1.Columna, t2.Columna
FROM Tabla1 t1
RIGHT JOIN Tabla2 t2 ON t1.ColumnaClave = t2.ColumnaClave;
```

FULL JOIN: Devuelve todos los registros cuando hay coincidencia en una u otra tabla

```
SELECT t1.Columna, t2.Columna
FROM Tabla1 t1
FULL JOIN Tabla2 t2 ON t1.ColumnaClave = t2.ColumnaClave;
```

CROSS JOIN: Devuelve el producto cartesiano de ambas tablas (cada fila de la primera con todas las de la segunda).

```
SELECT t1.Columna, t2.Columna
FROM Tabla1 t1
CROSS JOIN Tabla2 t2;
```

SQL

LÓGICA DE 3 VALORES

En SQL, una condición puede ser:

- TRUE (verdadero)
- FALSE (falso)
- UNKNOWN (cuando hay NULL)

FUNCIONES DE GRUPO

TIPO	EJEMPLO
COUNT()	Cuenta el número de filas (registros).
SUM()	Suma los valores de una columna numérica.
AVG()	Calcula el promedio de los valores de una columna numérica.
MAX()	Devuelve el valor máximo de una columna.
MIN()	Devuelve el valor mínimo de una columna.

```
SELECT
    COUNT(*) AS Total,
    SUM(ColumnaNumerica),
    AVG(ColumnaNumerica),
    MAX(ColumnaNumerica),
    MIN(ColumnaNumerica)
FROM NombreTabla;
```

HAVING

Filtra grupos después de un GROUP BY.

```
SELECT Columna, COUNT(*)
FROM NombreTabla
GROUP BY Columna
HAVING COUNT(*) > Valor;
```

GROUP BY

Agrupa registros en base a una o varias columnas

```
ELECT Columna, COUNT('
ROM NombreTabla
ROUP BY Columna;
```

CREAR TABLAS

Devuelve el valor mínimo de una columna.

```
SELECT Columna,
       CASE
           WHEN Condicion1 THEN 'Resultado1'
           WHEN Condicion2 THEN 'Resultado2'
           ELSE 'OtroResultado'
       END AS NuevaColumna
FROM NombreTabla;
```

RESTRICCIONES (CONSTRAINT)

- PRIMARY KEY → identifica de manera única a cada fila.
- FOREIGN KEY → establece relación con otra tabla.
- NOT NULL → evita valores nulos.
- UNIQUE → asegura que no haya valores repetidos.
- CHECK → obliga a cumplir una condición (ej: Edad ≥ 18).
- DEFAULT → asigna un valor por defecto.

UNION

Devuelve los registros combinados eliminando duplicados.

```
SELECT Columna1, Columna2
FROM Tabla1
UNION
SELECT Columna1, Columna2
FROM Tabla2;
```

SQL

UNION ALL

UNION ALL: Devuelve todos los registros combinados, incluyendo duplicados.

```
SELECT Columna1, Columna2
FROM Tabla1
UNION ALL
SELECT Columna1, Columna2
FROM Tabla2;
```

SUBCONSULTAS CON OPERADORES

IN: Sirve para comprobar si un valor está en el conjunto devuelto por la subconsulta.

```
SELECT columna
FROM tabla1
WHERE columna IN (
    SELECT columna
    FROM tabla2
    WHERE condición
```

EXISTS: Evalúa si la subconsulta devuelve al menos una fila (TRUE/FALSE).

```
SELECT columna
FROM tabla1 t1
WHERE EXISTS (
    SELECT 1
    FROM tabla2 t2
    WHERE t1.columna = t2.columna
```

NOT EXISTS: Evalúa si la subconsulta no devuelve filas.

```
SELECT columna
FROM tabla1 t1
WHERE NOT EXISTS (
    SELECT 1
    FROM tabla2 t2
    WHERE t1.columna = t2.columna
```

PIVOT (DE FILAS A COLUMNAS)

El PIVOT permite convertir valores de filas en encabezados de columnas y aplicar una función de agregación (SUM, COUNT, AVG, etc.).

```
SELECT <columnas_fijas>, [valor1], [valor2], [valor3], ...
FROM (
    SELECT <columna_a_fijar>, <columna_a_pivotear>, <valor_a_agregar>
    FROM <tabla>
) AS src
PIVOT (
    <función_agregación>(<valor_a_agregar>)
    FOR <columna_a_pivotear> IN ([valor1], [valor2], [valor3], ...)
) AS p;
```

UPDATE (MODIFICAR DATOS)

- Sin WHERE, se actualizan todas las filas de la tabla.

```
UPDATE tabla
SET columna1 = valor1, columna2 = valor2
WHERE condicion;
```

DELETE (ELIMINAR DATOS)

Sin WHERE, elimina todas las filas (la tabla queda vacía).

```
DELETE FROM tabla
WHERE condicion;
```

TRANSACCIONES Y MANEJO DE ERRORES

BEGIN TRANSACTION: Inicia una nueva transacción.

```
BEGIN TRANSACTION;
```

COMMIT TRANSACTION
Confirma la transacción, haciendo persistentes todos los cambios.

```
COMMIT TRANSACTION;
```

ROLLBACK TRANSACTION
Revierte todas las operaciones de la transacción si ocurre un error.

```
ROLLBACK TRANSACTION;
```

BEGIN TRY / BEGIN CATCH
Maneja errores dentro de transacciones.

- BEGIN TRY: Contiene el código que se intentará ejecutar.
- BEGIN CATCH: Captura y maneja los errores.

```
BEGIN TRY
    BEGIN TRANSACTION;
    -- Código de la transacción
    COMMIT TRANSACTION;
END TRY
BEGIN CATCH
    ROLLBACK TRANSACTION;
    PRINT 'Ocurrió un error';
END CATCH;
```

- **Las transacciones** (BEGIN, COMMIT, ROLLBACK) se usan para asegurar que un conjunto de operaciones SQL se ejecute completo o se deshaga por error.
- **Los procedimientos almacenados** sí pueden usar transacciones, porque pueden ejecutar cambios en tablas.
- **Las funciones** no manejan transacciones, solo calculan y devuelven valores que luego pueden usarse dentro de un procedimiento o consulta.