Tipos de DATOS

NUMBER(n,d) → *Números con **N** Digitos y **D** Decimales*

```
salario NUMBER(10,2),
```

VARCHAR2(n) → *Almacena texto de longitud variable hasta **N** caracteres.*

```
nombre VARCHAR2(50),
```

DATE → *Almacena fechas con el siguiente formato por defecto: **'DD/MM/YYYY'** → 01/01/2025*

```
fecha_nacimiento DATE;

SELECT EXTRACT(YEAR FROM fecha_nacimiento) FROM empleados;  → Obtiene el año
SELECT EXTRACT(MONTH FROM fecha_nacimiento) FROM empleados;  → Obtiene el mes
SELECT EXTRACT(DAY FROM fecha_nacimiento) FROM empleados;  → Obtiene el día
```

Crear Tabla - CREATE TABLE

Restricciones:

- **PRIMARY KEY** → *Identifica de forma única cada fila (no puede ser **NULL**).*
- **FOREIGN KEY** → *Relaciona una tabla con otra (puede repetirse y ser **NULL**).*
- **NOT NULL** → *Establece un valor por defecto si no se proporciona uno.*
- **DEFAULT** → *Obliga a que una columna tenga un valor (no permite **NULL**).*
- **UNIQUE** → *Garantiza que los valores de una columna no se repitan. (permite **NULL**).*
- **CHECK** → *Restringe los valores de una columna según una condición.*

```
CREATE TABLE empleados (
    id NUMBER(5) PRIMARY KEY,  → Clave primaria (única y obligatoria)
    nombre VARCHAR2(50) NOT NULL,  → No puede ser NULL
    salario NUMBER(10,2) DEFAULT 1000,  → Valor por defecto "1000"
    edad NUMBER(3) CHECK (edad >= 18),  → La edad debe ser 18 o más
    email VARCHAR2(100) UNIQUE,  → No se pueden repetir correos
    departamento_id NUMBER(5),  → Columna para la clave foránea
    FOREIGN KEY (departamento_id) REFERENCES departamentos(id)
);
```

Borrar Tabla

TRUNCATE TABLE → *Borra todas las filas de una tabla de manera rápida y eficiente, pero mantiene la estructura.*

```
TRUNCATE TABLE empleados;
```

DROP TABLE → *Borra por completo una tabla, incluyendo datos y estructura.*

```
DROP TABLE empleados;
```

DESCRIBE → *Muestra la estructura de una tabla*

```
DESCRIBE tabla;
```

SELECT → *Consulta datos, Se usa para obtener información de una o más tablas.*

```
SELECT columna1, columna2 FROM empleados WHERE departamento = 'IT';
```

DISTINCT → *Elimina valores duplicados en una consulta.*

```
SELECT DISTINCT columna FROM tabla;
```

INSERT → *Añade una nueva fila a una tabla.*

```
INSERT INTO empleados (id, nombre, departamento) VALUES (1, 'Juan', 'Ventas');
```

UPDATE → *Cambia valores en filas existentes.*

```
UPDATE empleados SET salario = 3000 WHERE id = 1;
```

DELETE FROM → *Elimina filas específicas de una tabla, pero mantiene la estructura de la tabla.*

```
DELETE FROM empleados WHERE departamento = 'Ventas';
```

WHERE

WHERE → *Filtra los resultados de una consulta. Se usa para seleccionar solo las filas que cumplen una condición.*

SELECT * **FROM** *empleados* **WHERE** *departamento* = 'IT';

- ***AND*** → *Ambas condiciones deben cumplirse.*
- ***OR*** → *Al menos una condición debe cumplirse.*
- ***NOT*** → *Niega una condición.*

- *IGUAL A* → **=**
- *DISTINTO DE* → **<>**
- *MENOR QUE* → **<**
- *MAYOR QUE* → **>**
- *MENOR O IGUAL QUE* → **<=**
- *MAYOR O IGUAL QUE* → **>=**

IN LIKE BETWEEN

IN → *Verifica si un valor está dentro de un conjunto de valores específicos.*

SELECT * **FROM** *empleados* **WHERE** *departamento* **IN** ('Ventas', 'Marketing', 'IT');

LIKE → *Se usa para buscar patrones en cadenas de texto con comodines (% y _).*

SELECT * **FROM** *clientes* **WHERE** *nombre* **LIKE** 'A%'; → *Nombres que empiezan con "A"*

SELECT * **FROM** *clientes* **WHERE** *nombre* **LIKE** '%lopez%'; → *Contiene "lopez"*

SELECT * **FROM** *clientes* **WHERE** *nombre* **LIKE** '_a%'; → *Segunda letra es "a"*

SELECT * **FROM** *clientes* **WHERE** *nombre* **NOT LIKE** 'A%'; → *Nombres que **NO** empiezan con "A"*

BETWEEN → *Selecciona valores dentro de un rango (numérico o de fechas).*

SELECT * **FROM** *pedidos* **WHERE** *precio* **BETWEEN 100 AND 500**;

SELECT * **FROM** *empleados* **WHERE** *fecha_contratacion* **BETWEEN** '2023-01-01' **AND** '2023-12-31';

Incluye los valores del límite inferior y superior.

ORDER BY y GROUP BY

ORDER BY → *Se usa para ordenar los resultados de una consulta*

SELECT *nombre, salario* **FROM** *empleados* **ORDER BY** *salario* **ASC**; → *Ordena de menor a mayor*

SELECT *nombre, salario* **FROM** *empleados* **ORDER BY** *salario* **DESC**; → *Ordena de mayor a menor*

GROUP BY → *Agrupa filas con valores iguales en una columna.*

SELECT *departamento*, ***COUNT****(*)*
FROM *empleados*
GROUP BY *departamento;*

→ *Agrupa por departamento y cuenta cuántos empleados hay en cada uno.*

***GROUP BY** casi siempre necesita una función de agregación.*

*Si todas las columnas del **SELECT** están en el **GROUP BY**, no es obligatorio usar agregación.*

COUNT( ) → *Cuenta el número de filas.*

HAVING → *Filtra los grupos después de **GROUP BY**.*

SELECT *columna*, *función_agregación()* **FROM** *tabla*
GROUP BY *columna*
HAVING ***condición***;

Funciones

SUM( ) → *Suma los valores de una columna numérica.*

MAX( ) → *Devuelve el valor máximo de una columna.*

MIN( ) → *Devuelve el valor mínimo de una columna.*

AVG( ) → *Calcula el promedio de los valores en una columna numérica.*

ABS( ) → *Devuelve el valor absoluto de un número.*

CEIL(x, d) → *Redondea un número **X** a **D** decimales (hacia arriba).*

FLOOR(x, d) → *Redondea un número **X** a **D** decimales (hacia abajo).*

TRUNC(x, d) → *Trunca un número **X** a **D** decimales sin redondear.*

MOD(a, b) → *Devuelve el resto de la división de **A** entre **B**.*

POWER(x, y) → *Eleva **X** a la potencia **Y**.*

SQRT( ) → *Devuelve la raíz cuadrada de un número.*

SIGN( ) → *Indica el signo de un número (1 positivo, -1 negativo, 0 si es cero).*

Modificar tabla - ALTER TABLE

ADD → *Agrega una nueva columna a la tabla.*

```
ALTER TABLE empleados ADD salario NUMBER(10,2);
```

MODIFY → *Modifica una columna existente (tipo de dato, tamaño, restricciones).*

```
ALTER TABLE empleados MODIFY salario NUMBER(6,2) DEFAULT 1200;
```

DROP COLUMN → *Elimina una columna de la tabla.*

```
ALTER TABLE empleados DROP COLUMN salario;
```

RENAME COLUMN → *Cambia el nombre de una columna.*

```
ALTER TABLE empleados RENAME COLUMN salario TO sueldo;
```

ADD CONSTRAINT → *Agrega una restricción (PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, etc.).*

```
ALTER TABLE empleados ADD CONSTRAINT pk_empleados PRIMARY KEY (id);

ALTER TABLE empleados ADD CONSTRAINT fk_departamento
FOREIGN KEY (departamento_id) REFERENCES departamentos(id);
```

DROP CONSTRAINT → *Elimina una restricción.*

```
ALTER TABLE empleados DROP CONSTRAINT pk_empleados;
ALTER TABLE empleados DROP CONSTRAINT fk_departamento;
```

Concatenación y Alias

|| → *Se utiliza para concatenar texto y columnas en una consulta*

AS → *Asigna un alias a una columna o tabla en SQL.*

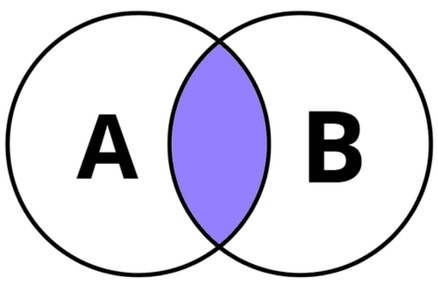
```
SELECT 'El empleado ' || nombre || ' gana: ' || salario || ' euros' AS "información"
FROM empleados;
```

JOIN

JOIN → *Es una operación que combina filas de dos o más tablas relacionadas mediante una condición común.*

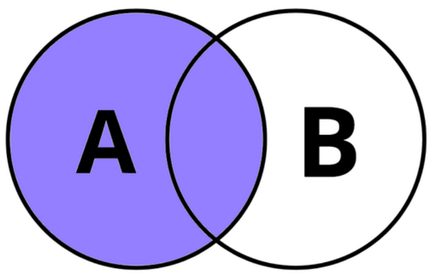
```
SELECT E.nombre, D.nombre_departamento
FROM Empleados E
INNER JOIN Departamentos D
ON E.id_departamento = D.id;
```

↑ *Clave Foranea* ↑ *Clave Primaria*



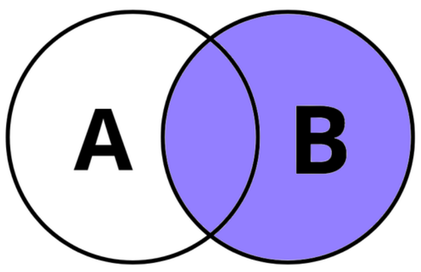
INNER JOIN → *Devuelve solo las filas que tienen coincidencias en ambas tablas.*

```
SELECT A.atributo, B.atributo
FROM Tabla1 A
INNER JOIN Tabla2 B
ON A.claveForanea = B.clavePrimaria;
```



LEFT JOIN → *Devuelve todos los registros de **Tabla1** y las coincidencias con **Tabla2**. Si no hay coincidencia en **Tabla2**, los valores serán **NULL**.*

```
SELECT A.atributo, B.atributo
FROM Tabla1 A
LEFT JOIN Tabla2 B
ON A.claveForanea = B.clavePrimaria;
```

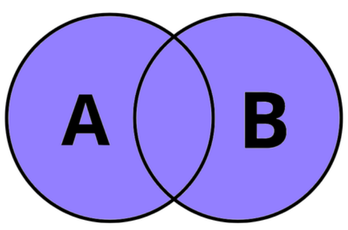


RIGHT JOIN → *Devuelve todos los registros de **Tabla2** y las coincidencias con **Tabla1**. Si no hay coincidencia en **Tabla1**, los valores serán **NULL**.*

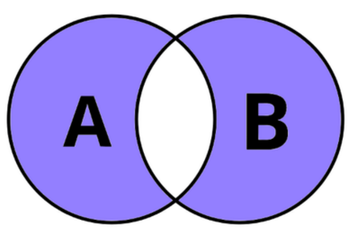
```
SELECT A.atributo, B.atributo
FROM Tabla1 A
RIGHT JOIN Tabla2 B
ON A.claveForanea = B.clavePrimaria;
```

OTROS JOIN

FULL OUTER JOIN

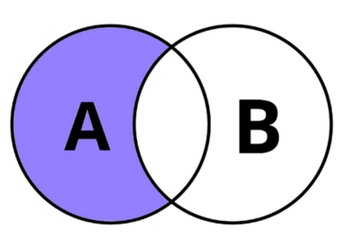



```
SELECT A.atributo, B.atributo
FROM Tabla1 A
FULL OUTER JOIN Tabla2 B
ON A.claveForanea = B.clavePrimaria;
```



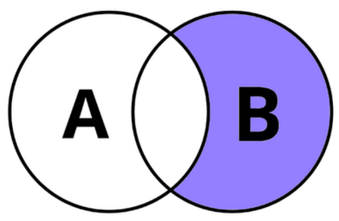

```
SELECT A.atributo, B.atributo
FROM Tabla1 A
FULL OUTER JOIN Tabla2 B
ON A.claveForanea = B.clavePrimaria
WHERE A.claveForanea IS NULL OR B.clavePrimaria IS NULL;
```

LEFT JOIN




```
SELECT A.atributo, B.atributo
FROM Tabla1 A
LEFT JOIN Tabla2 B
ON A.claveForanea = B.clavePrimaria
WHERE B.clavePrimaria IS NULL;
```

RIGHT JOIN




```
SELECT A.atributo, B.atributo
FROM Tabla1 A
RIGHT JOIN Tabla2 B
ON A.claveForanea = B.clavePrimaria
WHERE A.claveForanea IS NULL;
```

SET SERVEROUTPUT ON; *Activar consola*

```
DECLARE ← Declaración de variables
v_num NUMBER := 10;
v_nombre VARCHAR2(20) := 'Juan';
v_flag BOOLEAN := TRUE;
BEGIN ← Código y lógica
  DBMS_OUTPUT.PUT_LINE('id: '||v_num);
END;
```

```
DECLARE
  nota NUMBER;
BEGIN
  IF nota < 5 THEN
    DBMS_OUTPUT.PUT_LINE('Reprobado');
  ELSIF nota < 7 THEN
    DBMS_OUTPUT.PUT_LINE('Aprobado');
  ELSIF nota < 9 THEN
    DBMS_OUTPUT.PUT_LINE('Notable');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Sobresaliente');
  END IF;
END;
```

```
BEGIN
  FOR i IN 1..10 LOOP
    DBMS_OUTPUT.PUT_LINE(i);
  END LOOP;

  FOR i IN REVERSE 1..10 LOOP
    DBMS_OUTPUT.PUT_LINE(i);
  END LOOP;
END;
```

```
-- Ecuación 2° grado (ax² + bx + c = 0)
 DECLARE
   a NUMBER;
   b NUMBER;
   c NUMBER;
   discriminante NUMBER;
   x1 NUMBER;
   x2 NUMBER;
BEGIN
   a := 1;
   b := -3;
   c := 2;
   discriminante := b*b - 4*a*c;
   IF discriminante > 0 THEN
      x1 := (-b + SQRT(discriminante)) / (2*a);
      x2 := (-b - SQRT(discriminante)) / (2*a);
      DBMS_OUTPUT.PUT_LINE('Las soluciones son: ' || x1 || ' y ' || x2);
   ELSIF discriminante = 0 THEN
      x1 := -b / (2*a);
      DBMS_OUTPUT.PUT_LINE('La única solución es: ' || x1);
   ELSE
      DBMS_OUTPUT.PUT_LINE('La ecuación no tiene soluciones reales.');
   END IF;
END;
```

Funciones

SUM( ) → *Suma los valores de una columna numérica.*

MAX( ) → *Devuelve el valor máximo de una columna.*

MIN( ) → *Devuelve el valor mínimo de una columna.*

AVG( ) → *Calcula el promedio de los valores en una columna numérica.*

ABS( ) → *Devuelve el valor absoluto de un número.*

CEIL(x, d) → *Redondea un número **X** a **D** decimales (hacia arriba).*

FLOOR(x, d) → *Redondea un número **X** a **D** decimales (hacia abajo).*

TRUNC(x, d) → *Trunca un número **X** a **D** decimales sin redondear.*

MOD(a, b) → *Devuelve el resto de la división de **A** entre **B**.*

POWER(x, y) → *Eleva **X** a la potencia **Y**.*

SQRT( ) → *Devuelve la raíz cuadrada de un número.*

SIGN( ) → *Indica el signo de un número (1 positivo, -1 negativo, 0 si es cero).*

```
DECLARE
  contador NUMBER;
BEGIN
  contador := 1;
  WHILE contador <= 10 LOOP
    DBMS_OUTPUT.PUT_LINE(contador);
    contador := contador + 1;
  END LOOP;
END;
```

```
DECLARE
  contador NUMBER;
BEGIN
  contador := 1;
  LOOP
    EXIT WHEN contador > 10;
    DBMS_OUTPUT.PUT_LINE(contador);
    contador := contador + 1;
  END LOOP;
END;
```

```
-- Media aritmética
media := (n1 + n2 + n3) / 3;

-- Varianza
media := (n1 + n2 + n3) / 3;
varianza := ((n1 - media)**2 + (n2 - media)**2 + (n3 - media)**2) / 3;

-- Área de círculo
area := 3.1416 * r * r;

-- Perimetro Rectangulo
perimetro := 2 * (base + altura);

-- Volumen de esfera
volumen := (4/3) * 3.1416 * POWER(r,3);

-- Volumen de prisma
volumen := largo * ancho * alto;

-- Área de rombo
area := (D * d) / 2;

-- Ecuación 1er grado (ax + b = 0)
x := -b / a;
```