Imagen que contiene dibujo

Descripción generada automáticamente

TECNÓLOGO EN ANÁLISIS Y DESARROLLO DE SOFTWARE

Ficha: 2900177

Aprendiz: Brayan Santiago Guerrero Mendez

Instructor: Jhon Corredor

Neiva-Huila

Contenido

[1.INNER JOIN 3](#_Toc180392648)

[2. LEFT JOIN 4](#_Toc180392649)

[3. RIGHT JOIN 5](#_Toc180392650)

[4. FULL JOIN 6](#_Toc180392651)

[5. CROSS JOIN 7](#_Toc180392652)

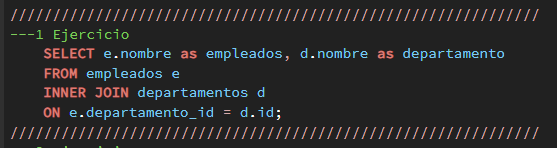
[6. INNER JOIN A LA MISMA TABLA 11](#_Toc180392653)

[7. INNER JOIN MULTIPLES TABLAS 13](#_Toc180392654)

[8. SCRIPT 14](#_Toc180392655)

# 1.INNER JOIN

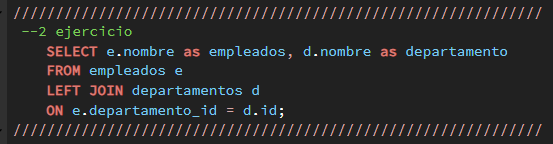
* 1. Realiza un query entre las tablas Empleados y Departamentos para obtener una lista de todos los empleados junto con el nombre de su departamento. Solo muestra a los empleados que están asignados a un departamento.
* El **INNER JOIN** se encarga de mostrar solo a los empleados que estén asociados a un departamento, si no hay registros asociados no se mostraran





# LEFT JOIN

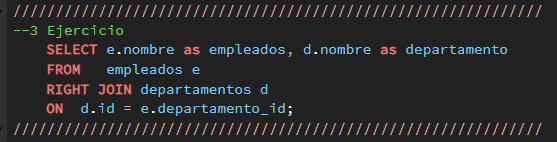
* 1. Realiza un query para obtener todos los empleados, incluso aquellos que no están asignados a un departamento, mostrando el nombre del departamento si está disponible.
* El **LEFT JOIN** muestra todos los registros de la tabla derecha en este nuestro caso la tabla **departamentos**, y los registros que estén asociados a un **empleado**, y de no estarlo completa los campos con NULL

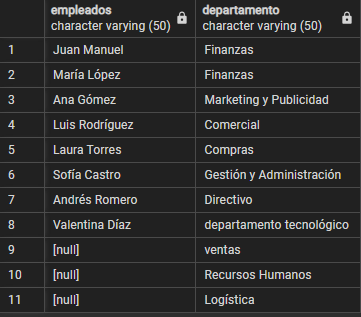




# RIGHT JOIN

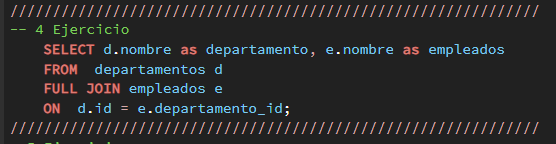
* 1. Realiza un query para listar todos los departamentos, incluso si no tienen empleados asignados.
* El **RIGHT JOIN** muestra todos los registros de la tabla de la izquierda en nuestro caso (**departamentos**) y valida si hay registros asociados a la tabla (**empleados**), si no hay registros asociados completa los campos con NULL

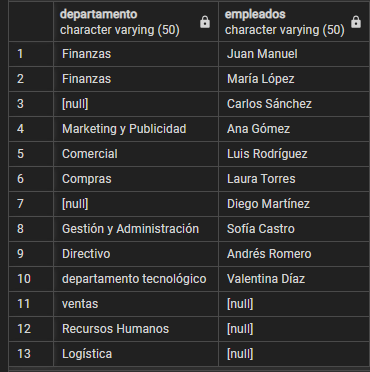




# FULL JOIN

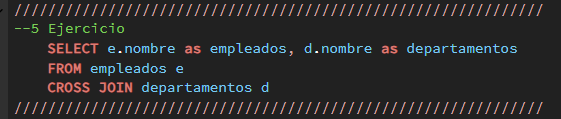
* 1. Usa un query para listar todos los empleados y departamentos, incluyendo aquellos que no tienen una relación con la otra tabla.
* El **FULL JOIN** muestra todos los registros de ambas tablas, y si algún campo de las tablas no tiene un registro asociado les coloca NULL



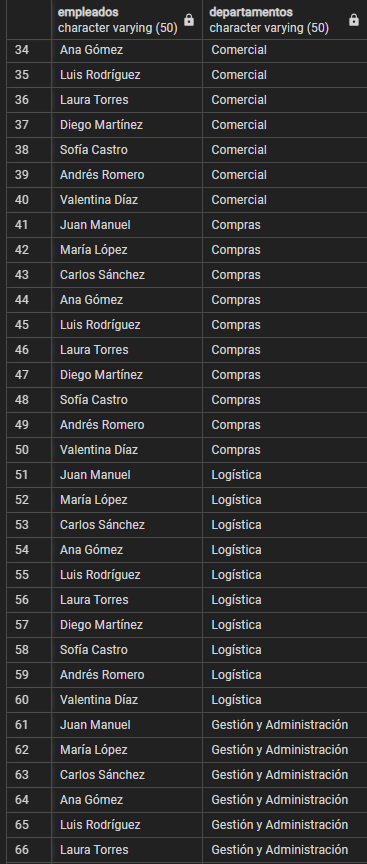
x

# CROSS JOIN

* 1. Realiza un query entre las tablas de empleados y departamentos. Explica cuál es el resultado y cómo difiere de los otros tipos de JOIN.
* El **CROSS JOIN** combina cada registro de una tabla (**empleados**) con cada registro de la tabla (**departamentos**)



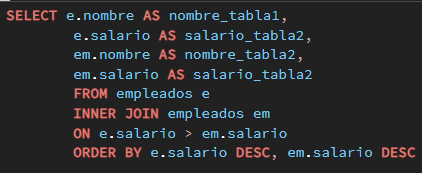






# INNER JOIN A LA MISMA TABLA

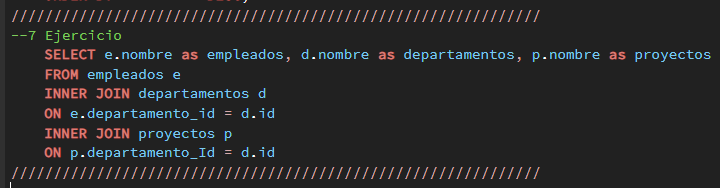
* 1. Usando un query que muestra a los empleados cuyo salario sea mayor que el de otros empleados. Este ejercicio requiere que compares la tabla consigo misma.
* Se realiza un **INNER JOIN** sobre la misma tabla para poder hacer una validación la cual permita conocer cual empleado gana más que otros.

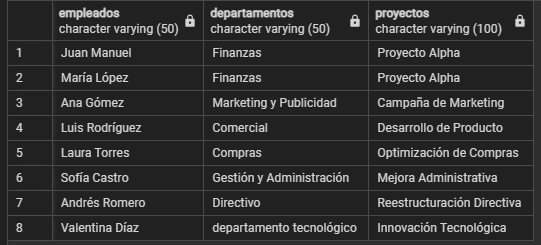




# INNER JOIN MULTIPLES TABLAS

* 1. Realiza un query con más de dos tablas. Agrega una tercera tabla llamada Proyectos (ID, Nombre, DepartamentoID) y realiza un INNER JOIN entre Empleados, Departamentos y Proyectos para mostrar todos los empleados, sus departamentos y los proyectos en los que están involucrados.
* Se usa un INNER JOIN para unir las tablas de **empleados**, **departamentos** con la tabla de **proyectos** y poder mostrar los empleados asociados a un proyecto en cada departamento





# SCRIPT

--CREAR TABLAS

CREATE TABLE departamentos (

id INT NOT NULL,

nombre VARCHAR(50) NOT NULL,

PRIMARY KEY(id)

);

CREATE TABLE empleados(

id INT NOT NULL,

nombre VARCHAR(50) NOT NULL,

departamento\_Id INT,

salario DECIMAL(10,2) NOT NULL,

PRIMARY KEY(id)

);

CREATE TABLE proyectos(

id serial NOT NULL,

nombre varchar(100) NOT NULL,

departamento\_Id integer NOT NULL,

PRIMARY KEY(id)

);

-- Alterar tablas llaves foranes

ALTER TABLE empleados

ADD CONSTRAINT fk\_departamento

FOREIGN KEY (departamento\_Id)

REFERENCES departamentos(id);

ALTER TABLE proyectos

ADD CONSTRAINT fk\_proyectos

FOREIGN KEY (departamento\_Id)

REFERENCES departamentos(id)

-- Insertar datos en las tablas

--Departamentos

INSERT INTO departamentos

(id, nombre)

VALUES (1,'Finanzas'),

(2,'Recursos Humanos'),

(3,'Marketing y Publicidad'),

(4,'Comercial'),

(5,'Compras'),

(6,'Logística'),

(7,'Gestión y Administración'),

(8,'Directivo'),

(9,'departamento tecnológico'),

(10,'ventas');

-- Empleados

INSERT INTO empleados

(id, nombre, departamento\_Id, salario)

VALUES (1, 'Juan Manuel', 1, 7000000),

(2, 'María López', 1, 8802000),

(3, 'Carlos Sánchez', NULL, 6000000),

(4, 'Ana Gómez', 3, 9203000),

(5, 'Luis Rodríguez', 4, 10000000),

(6, 'Laura Torres', 5, 7601000),

(7, 'Diego Martínez', NULL, 8400000),

(8, 'Sofía Castro', 7, 8203200),

(9, 'Andrés Romero', 8, 7802400),

(10, 'Valentina Díaz', 9, 9000000);

--Proyectos

INSERT INTO proyectos (id, nombre, departamento\_Id)

VALUES (1, 'Proyecto Alpha', 1),

(2, 'Iniciativa Beta', 2),

(3, 'Campaña de Marketing', 3),

(4, 'Desarrollo de Producto', 4),

(5, 'Optimización de Compras', 5),

(6, 'Logística Eficiente', 6),

(7, 'Mejora Administrativa', 7),

(8, 'Reestructuración Directiva', 8),

(9, 'Innovación Tecnológica', 9),

(10, 'Estrategia de Ventas', 10);

-- SELECCIONAR DATOS

SELECT \* FROM departamentos;

SELECT \* FROM empleados;

SELECT \* FROM proyectos;

-- INNSERTs

-- INNER JOIN

//////////////////////////////////////////////////////////////

---1 Ejercicio

SELECT e.nombre as empleados, d.nombre as departamento

FROM empleados e

INNER JOIN departamentos d

ON e.departamento\_id = d.id;

//////////////////////////////////////////////////////////////

--2 ejercicio

SELECT e.nombre as empleados, d.nombre as departamento

FROM empleados e

LEFT JOIN departamentos d

ON e.departamento\_id = d.id;

//////////////////////////////////////////////////////////////

--3 Ejercicio

SELECT e.nombre as empleados, d.nombre as departamento

FROM empleados e

RIGHT JOIN departamentos d

ON d.id = e.departamento\_id;

//////////////////////////////////////////////////////////////

-- 4 Ejercicio

SELECT d.nombre as departamento, e.nombre as empleados

FROM departamentos d

FULL JOIN empleados e

ON d.id = e.departamento\_id;

//////////////////////////////////////////////////////////////

--5 Ejercicio

SELECT e.nombre as empleados, d.nombre as departamentos

FROM empleados e

CROSS JOIN departamentos d

//////////////////////////////////////////////////////////////

--6 Ejercicio

SELECT e.nombre as empleados , e.salario as salario

FROM empleados e

INNER JOIN empleados es

ON e.salario>= es.salario

GROUP BY e.id

ORDER BY e.salario DESC;

//////////////////////////////////////////////////////////////

--7 Ejercicio

SELECT e.nombre as empleados, d.nombre as departamentos, p.nombre as proyectos

FROM empleados e

INNER JOIN departamentos d

ON e.departamento\_id = d.id

INNER JOIN proyectos p

ON p.departamento\_Id = d.id