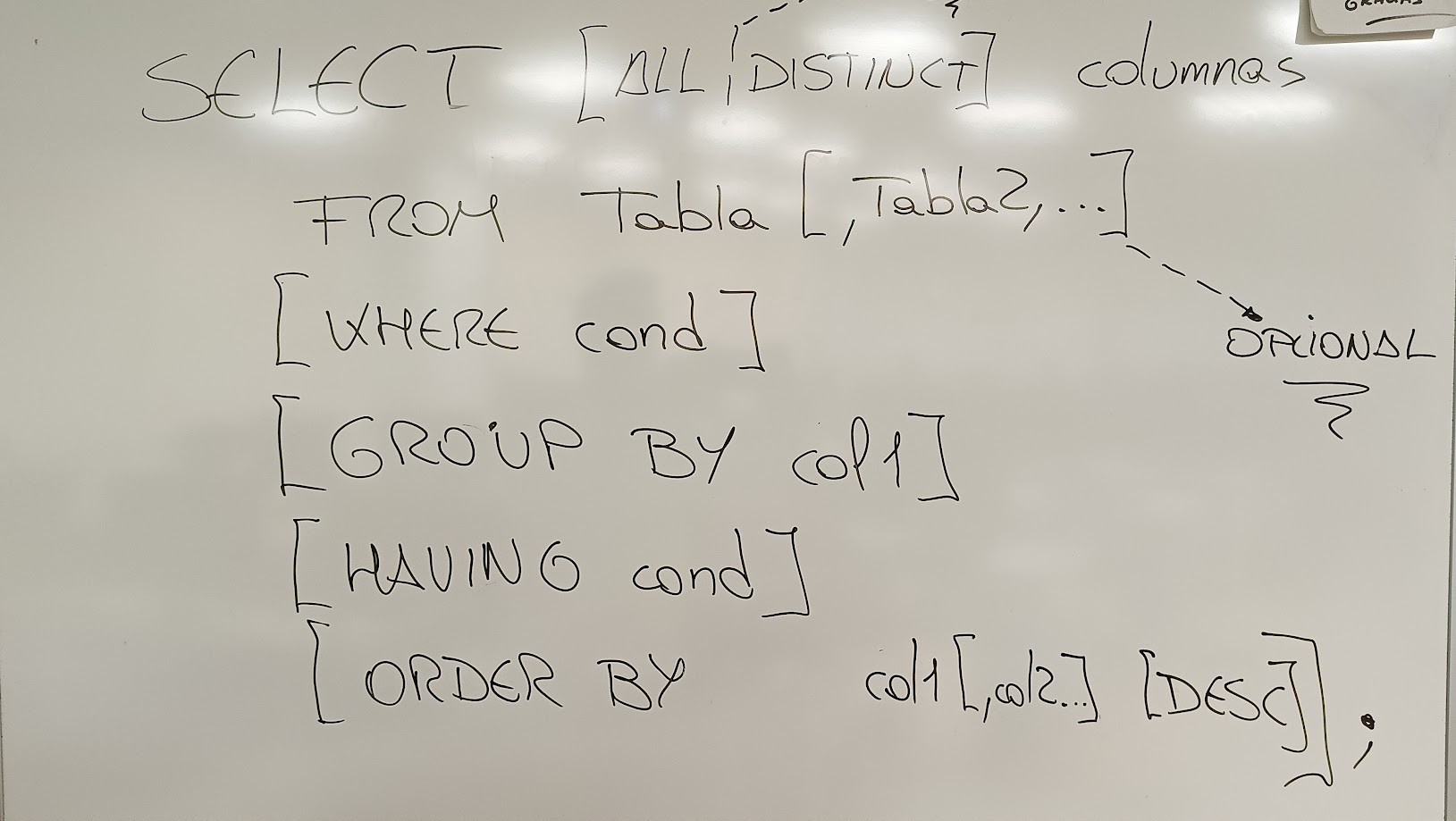
# **Bases de Datos Oracle SQL**



## **Estructura de una Consulta SELECT**

Una consulta básica en SQL sigue esta estructura:

SELECT [DISTINCT] columna1, columna2, ...

FROM nombre\_tabla

[WHERE condición]

[GROUP BY columna(s)]

[HAVING condición]

[ORDER BY columna(s) [ASC|DESC]];

### **Componentes Principales:**

1. **SELECT:** Especifica las columnas que deseas consultar. Puedes usar \* para seleccionar todas las columnas de una tabla.
2. **FROM:** Indica la tabla de donde provienen los datos.
3. **WHERE:** Filtra los resultados según una condición.
4. **GROUP BY:** Agrupa resultados basados en una o más columnas.
5. **HAVING:** Filtra los grupos generados por GROUP BY.
6. **ORDER BY:** Ordena los resultados de la consulta.
   * Orden ascendente por defecto: ORDER BY columna.
   * Orden descendente: ORDER BY columna DESC.

## **Eliminación de Duplicados con DISTINCT**

El uso de DISTINCT elimina filas duplicadas en una columna o conjunto de columnas.

**Ejemplo:**

SELECT DISTINCT oficio

FROM emple

ORDER BY oficio;

**Resultado:**

OFICIO

----------

ANALISTA

DIRECTOR

EMPLEADO

PRESIDENTE

VENDEDOR

**Nota:** Con DISTINCT, los valores repetidos se omiten del resultado.

### **Orden Descendente con DISTINCT**

SELECT DISTINCT oficio

FROM emple

ORDER BY oficio DESC;

**Resultado:**

OFICIO

----------

VENDEDOR

PRESIDENTE

EMPLEADO

DIRECTOR

ANALISTA

## **Uso de la Cláusula WHERE**

La cláusula WHERE filtra filas según condiciones específicas. Para búsquedas de texto, es útil utilizar la función UPPER() para evitar problemas con la distinción entre mayúsculas y minúsculas.

**Ejemplo:** Filtrar empleados con el cargo "DIRECTOR"

SELECT apellido, salario

FROM emple

WHERE UPPER(oficio) = 'DIRECTOR';

**Resultado:**

APELLIDO SALARIO

---------- ----------

JIMENEZ 2900

NEGRO 3005

CEREZO 2885

### **Notas Importantes sobre WHERE:**

* **Búsquedas Insensibles a Mayúsculas:** Usar UPPER(campo) = 'VALOR' asegura que el filtro sea insensible a las mayúsculas.
* **Evitar Errores:** Verifica que los nombres de columnas y tipos de datos coincidan correctamente.

## **Ordenar Resultados con ORDER BY**

El ordenamiento se realiza al final de la consulta:

1. Ordenar por una columna específica:

ORDER BY columna;

1. Ordenar de forma descendente:

ORDER BY columna DESC;

1. Ordenar por múltiples columnas:

ORDER BY columna1, columna2 DESC;

**Ejemplo:** Empleados del departamento 20 ordenados por apellido:

SELECT apellido, oficio, emp\_no, dept\_no

FROM emple

WHERE dept\_no = 20

ORDER BY apellido;

**Ejemplo:** Analistas ordenados por número de empleado:

SELECT apellido, oficio, emp\_no, dept\_no

FROM emple

WHERE UPPER(oficio) = 'ANALISTA'

ORDER BY emp\_no;

## **Cálculo de Promedios**

Puedes realizar cálculos en las consultas utilizando operadores aritméticos:

**Ejemplo:** Calcular la media de tres notas por alumno:

SELECT nombre\_alumno,

(nota1 + nota2 + nota3) / 3 AS promedio

FROM notas\_alumnos;

## **Validaciones y Estructuras de Ejercicios Prácticos**

### **Ejemplo 1: Consultas con Relaciones entre Tablas**

En este ejercicio, se consulta información combinada de dos tablas:

SELECT emple.apellido, emple.oficio, depart.loc

FROM emple, depart

WHERE emple.dept\_no = depart.dept\_no

AND emple.oficio = 'ANALISTA';

**Lecciones Aprendidas:**

* Relacionar tablas usando claves foráneas (emple.dept\_no = depart.dept\_no).
* Filtrar resultados con múltiples condiciones (emple.oficio = 'ANALISTA').

### **Ejemplo 2: Subconsultas en Condiciones**

Seleccionar empleados bajo la dirección de un jefe específico:

SELECT \*

FROM emple, depart

WHERE emple.dir = (

SELECT emp\_no

FROM emple

WHERE apellido = 'CEREZO'

);

**Lecciones Aprendidas:**

* Uso de subconsultas en el WHERE para filtrar datos específicos.
* Seleccionar columnas de múltiples tablas relacionadas.

### **Ejemplo 4: Excluir Valores Específicos**

Consultar departamentos sin empleados asociados:

SELECT \*

FROM depart

WHERE dept\_no NOT IN (

SELECT dept\_no

FROM emple

);

**Lecciones Aprendidas:**

* NOT IN para excluir subconjuntos específicos de datos.
* Identificar relaciones vacías entre tablas.

### **Ejemplo 6: Uso de Operadores Agregados**

Seleccionar empleados cuyo salario sea mayor que cualquier salario en un departamento específico:

SELECT apellido, salario

FROM emple

WHERE salario > ALL (

SELECT salario

FROM emple

WHERE dept\_no = 20

);

**Lecciones Aprendidas:**

* Uso de ALL para comparar con todos los valores de un subconjunto.
* Subconsultas con filtros adicionales.

### **Ejemplo 7: Rangos y Condiciones Múltiples**

Seleccionar registros dentro de un rango específico:

SELECT \*

FROM libreria

WHERE ejemplares >= 8 AND ejemplares <= 15;

**Lecciones Aprendidas:**

* Especificar condiciones de rango con operadores (>=, <=).
* Combinar condiciones con AND para mayor precisión.

### **Ejemplo 9: Comparaciones con Subconsultas**

Consultar temas con menos ejemplares que un tema específico:

SELECT tema

FROM libreria

WHERE ejemplares < (

SELECT ejemplares

FROM libreria

WHERE tema = 'MEDICINA'

);

**Lecciones Aprendidas:**

* Subconsultas para comparar valores específicos en la misma tabla.
* Identificar registros relativos a un criterio de referencia.

### **Ejemplo 11: Búsquedas con Patrones**

Seleccionar asignaturas con un patrón específico en su nombre:

SELECT nombre

FROM asignaturas

WHERE nombre LIKE '%o%o%o%'

AND cod IN (

SELECT cod

FROM notas

WHERE dni IN (

SELECT dni

FROM alumnos

WHERE pobla = 'Madrid'

)

);

**Lecciones Aprendidas:**

* Uso de LIKE para buscar patrones en texto.
* Combinación de múltiples subconsultas para filtrar datos complejos.

## **Buenas Prácticas al Escribir Consultas SQL**

**Comparaciones de Texto:** Siempre usa UPPER() para evitar problemas de mayúsculas/minúsculas.  
  
 WHERE UPPER(oficio) = 'ANALISTA';

1. **Selección de Columnas:** Especifica las columnas necesarias en lugar de usar SELECT \* para mejorar el rendimiento.
2. **Uso de DISTINCT:** Úsalo solo cuando necesites eliminar duplicados, ya que puede afectar el rendimiento en tablas grandes.
3. **Ordenamiento:** Aplica ORDER BY sólo cuando el orden de los resultados sea relevante.

## **Funciones Comunes en SQL**

### **Operaciones Aritméticas:**

-- Cálculo de salario anual

SELECT salario \* 12 AS salario\_anual

FROM emple;

-- Promedios

SELECT (nota1 + nota2 + nota3) / 3 AS promedio

FROM notas\_alumnos;

### **Funciones de Texto:**

1. **Conversión a Mayúsculas/Minúsculas:**

SELECT UPPER(nombre), LOWER(apellido)

FROM emple;

1. **Concatenación de Texto:**

SELECT nombre || ' ' || apellido AS nombre\_completo

FROM emple;

### **Redondeo:**

SELECT ROUND(salario, 2) AS salario\_redondeado

FROM emple;

EN CASO DE QUE TE METAN LOS DATOS MUY MAL Y QUIERA SEGUIR VIENDO TODO:

→ EN EL WHERE PODEMOS UTILIZAR UPPER(CAMPO) = ‘VALOR A BUSCAR’,

LO SUBE A MAYÚSCULAS EN EL SELECT PERO EN NADA MÁS.

SQL> SELECT APELLIDO, SALARIO

2 FROM TEMPLE

3 WHERE UPPER(OFICIO) = 'DIRECTOR';

//Convertimos todo a mayúsculas(nos aseguramos que seleccione todos los directores a pesar de no estar escrito en MAYÚS)

APELLIDO SALARIO

---------- ----------

JIMENEZ 2900

NEGRO 3005

CEREZO 2885

ORDENAR:

AL FINAL DEL TODO PODEMOS PONER UN ORDER BY (COLUMNA QUE QUERAMOS ORDENAR POR, ES DECIR -> ATRIBUTO CAMPO DE UNA TABLA.)

**EMPLEADOS DEL DEPARTAMENTO 20 ORDENADOS POR APELLIDO, QUIERO NUM, APELLIDO, OFICIO Y DEPARTAMENTO**

SELECT APELLIDO, OFICIO, EMP\_NO, DEPT\_NO

FROM EMPLE

WHERE DEPT\_NO = 20

ORDER BY APELLIDO;

**ANALISTAS, ORDENADO POR NÚMERO DE EMPLEADO**

SELECT APELLIDO, OFICIO, EMP\_NO, DEPT\_NO

FROM EMPLE

WHERE UPPER(OFICIO) = 'ANALISTA'

ORDER BY EMP\_NO;

**—-----EN LAS CONDICIONES DE STRING SIEMPRE PONER LOS UPPERS—-----**

**CALCULAR MEDIAS**

SELECT NOMBRE\_ALUMNO, (NOTA1 + NOTA2 + NOTA3) / 3

FROM NOTAS\_ALUMNOS;

**—-COMMIT—--**

Tras hacer la inserción de datos lo mejor es hacer commits ya que si no los haces se guarda en local y ya; si lo haces se guarda a nivel general , al ser multiusuario todo el mundo lo debe ver.

**— COMBINACIÓN DE TABLAS —- CONDICIÓN DE ENLACE —**

Siempre que relacionemos algo usando 2 tablas tendremos que poner la condición where que te iguale el campo que es igual en ambas tablas; si no generamos productos cartesianos, esto se llama condición de enlace; por ejemplo:

SELECT\*

from EMPLE, DEPART

where EMPLE.DEPT\_NO = DEPART.DEPT\_NO (Condición de enlace)

and UPPER (LOC) = 'BARCELONA';

**—- OPERADOR DE CONJUNTOS / SUBCONSULTAS —-**

Otra forma de realizar lo que hemos hecho con la condición de enlace:

SELECT APELLIDO

FROM EMPLE

WHERE DEPT\_NO = (SELECT DEPT\_NO FROM DEPART

WHERE UPPER(LOC) = ‘BARCELONA’);

-> Es como una operación matemática , se ejecuta primero lo de dentro del paréntesis y luego el selecto de fuera.

Es como si hicieramos 3\*(7+3) - Primero paréntesis y luego multiplicación.

La cosa es que la condición de enlace va más rápido y siempre será mejor utilizar las condiciones de enlace; no obstante, cuando no se puedan utilizar las subconsultas habrá que usar esto también.

**EJEMPLO 1**: Habrá casos en los que necesitaré si o si una subconsulta para recuperar datos, por ejemplo cuando quiera saber algo que se basa en una información que puede cambiar a lo largo del tiempo, por ejemplo los apellidos de los empleados que tienen el mismo oficio que ‘Gil’. Si por lo que sea mañana ‘Gil’ cambia de oficio la consulta debe seguir funcionando. Para eso la basaré en una subconsulta que me dará el oficio de gil en el momento de ejecución y de este modo la consulta funcionará siempre aunque cambiemos en el futuro de oficio a Gil.

Le meteremos otro select incrustado para que funcione; da igual que haya muchos select en un mismo query -> así funcionará siempre con cualquier dato que haya; las querys funcionarán siempre.

Composición:

Select apellido from emple

Where upper(oficio) **=** (Select Upper(Oficio) From emple

where upper(apellido)= ‘GIL’);

APELLIDO

----------

GIL

FERNANDEZ

**=** Aquí es dónde vendría nuestra incertidumbre y metemos otro select.

Pero…

Atención: Si añado a un segundo empleado que se apellida ‘Gil’ y además tiene un oficio diferente al primer ‘Gil’...

Probamos.

Insert into EMPLE values (8000, ‘GIL’, ‘VENDEDOR’, NULL, SYSDATE, 1600, 100, 10);



Ahora tenemos dos Gil con dos oficios distintos; uno analista y otro vendedor, **entonces si** **probamos la consulta de antes**:

Select apellido from emple

Where upper(oficio) **=** (Select Upper(Oficio) From emple

where upper(apellido)= ‘GIL’);



Error: La subconsulta de una sola fila devuelve más de una fila.

Dará error porque + cómo solucionarlo…

-Si la subconsulta está bien y la consulta también está bien lo que falla es el ‘=’ ; el tema es que el ‘=’ es un operador binario !!!, es decir sólo funciona poniendo un único valor a cada lado; es decir solo puedes igualar un valor , si tenemos un conjunto de valores para comparar el ‘=’ no sabe hacerlo; entonces tendremos que usar el **IN**; es decir, la subconsulta devuelve dos valores y el ‘=’ no sabe compararlo porque solo funciona con un dato a cada lado, es un comparador binario, usa un operador de conjuntos un comparador de conjuntos sería el **IN**.

**Entonces la solución sería la implementación del IN.**

SOLUCIÓN:

Select apellido from emple

Where upper(oficio) **IN** (Select Upper(Oficio) From emple

where upper(apellido)= ‘GIL’);

APELLIDO

----------

GIL

FERNANDEZ

ARROYO

SALA

MARTIN

TOVAR

GIL

-**COROLARIO** (Enseñanza): Si no estoy seguro de que la subconsulta devuelva un único valor (y sólo lo devolverá cuando la condición sea sobre una PK!!!)... deberé utilizar operadores de conjuntos como ‘IN’ o ‘NOT IN’; si hacemos referencia a la primary key podremos usar un ‘=’ ya que estamos comparando de forma binaria, únicamente con uno ; pero , si comparamos con varios deberemos usar un ‘IN’, por eso lo mejor siempre será curarnos en salud, es decir prevenir y usar un ‘IN’

**EJEMPLO 2**: Empleados que cobren más que Gil… pero atención, porque puede que haya dos empleados que se llamen ‘GIL’ y el comparador ‘>’ no funcione porque también es un operador binario.

**PLANTEAMIENTO A PRIMERA VISTA:**

SELECT APELLIDO FROM EMPLE

WHERE SALARIO > (SELECT SALARIO FROM EMPLE

WHERE UPPER(APELLIDO) = ‘GIL’);

-> ASÍ DARÁ ERROR, EL PROBLEMA ES EL MISMO, ESTAMOS UTILIZANDO UN OPERADOR DE COMPARACIÓN BINARIO Y NO UNO DE CONJUNTOS; EL > ES BINARIO.

**SOLUCIÓN 1**: Utilizar la palabra/s clave o reservada/s ‘ANY’ O ‘ALL’.

ANY: Que cualquiera.

ALL: Que todos.

SELECT APELLIDO FROM EMPLE

WHERE SALARIO > ALL (SELECT SALARIO FROM EMPLE

WHERE UPPER(APELLIDO) = ‘GIL’);

APELLIDO

----------

NEGRO

REY

**SOLUCIÓN 2:** Adelantándonos al tema 4… podría utilizar una función… en concreto ‘MAX’ que da el máximo valor.

SELECT APELLIDO FROM EMPLE

WHERE SALARIO > ALL (**SELECT MAX (SALARIO)** FROM EMPLE

WHERE UPPER(APELLIDO) = ‘GIL’);

-La solución ya iría metida en el propio select.

APELLIDO

----------

NEGRO

REY

**EJERCICIOS PRÁCTICA CONDICIÓN DE ENLACE, SUBCONSULTAS, ETC…**

**Ejercicios de Condición de Enlace**

1. **Muestra todos los empleados junto con el nombre del departamento en el que trabajan.  
   Resultado esperado:** Listado de empleados con su apellido, oficio y nombre del departamento.
2. **Muestra el nombre del departamento y el apellido de los empleados que trabajan en la ciudad de ‘Madrid’.  
   Resultado esperado:** Los apellidos de los empleados que trabajan en el departamento cuya localización es Madrid.
3. **Muestra los empleados que tienen el mismo oficio que ‘JIMÉNEZ’.  
   Resultado esperado:** Listado de empleados con el mismo oficio que JIMÉNEZ.
4. **Lista los empleados cuyo salario sea mayor al salario promedio de todos los empleados.  
   Resultado esperado:** Apellidos de empleados cuyo salario es superior a la media.
5. **Encuentra los empleados cuyo salario sea mayor que el salario del empleado ‘MARTIN’.  
   Resultado esperado:** Apellidos de empleados con salario mayor al de MARTÍN.

**Ejercicios de Subconsultas con Operadores Binarios e IN**

1. **Muestra los empleados que trabajan en el mismo departamento que ‘FERNÁNDEZ’.  
   Resultado esperado:** Listado de apellidos de empleados que trabajan en el mismo departamento que FERNÁNDEZ.
2. **Lista los empleados cuyo salario sea mayor que el salario mínimo de todos los empleados.  
   Resultado esperado:** Apellidos de empleados con un salario mayor al salario mínimo registrado.
3. **Encuentra los empleados cuyo salario es mayor que cualquier salario de los empleados con el oficio de ‘VENDEDOR’.  
   Resultado esperado:** Empleados cuyo salario es mayor que al menos un vendedor.
4. **Muestra los empleados cuyo salario es menor que todos los empleados con el oficio de ‘DIRECTOR’.  
   Resultado esperado:** Empleados que cobran menos que cualquier director.
5. **Encuentra los empleados que trabajan en un departamento que no tiene empleados en ‘Barcelona’.  
   Resultado esperado:** Apellidos de empleados que no están en un departamento ubicado en Barcelona.

### **Ejercicios con IN, NOT IN, ANY, ALL y Funciones Agregadas**

**11. Lista los empleados que ganan más que todos los empleados cuyo oficio sea ‘ANALISTA’.  
Resultado esperado:** Apellidos de empleados con salario superior al máximo salario de los analistas.

**12. Encuentra los empleados cuyo salario está en el mismo rango de los salarios de los empleados con el oficio de ‘EMPLEADO’.  
Resultado esperado:** Empleados cuyo salario es igual a alguno de los empleados con

**13. Muestra los empleados cuyo salario sea superior al de ‘TOVAR’, pero inferior al de ‘NEGRO’.  
Resultado esperado:** Apellidos de empleados cuyo salario esté entre los salarios de TOVAR y NEGRO.

**14. Muestra los empleados que no trabajan en ninguno de los departamentos que tienen empleados con oficio ‘VENDEDOR’.  
Resultado esperado:** Apellidos de empleados que trabajan en departamentos sin vendedores.

**15. Lista los empleados que NO tienen comisión.  
Resultado esperado:** Apellidos de empleados cuyo campo de comisión sea NULL.

**SOLUCIONES**:

1. SELECT APELLIDO, DNOMBRE

FROM EMPLE, DEPART

WHERE EMPLE.DEPT\_NO = DEPART.DEPT\_NO;

1. SELECT APELLIDO, DNOMBRE

FROM EMPLE, DEPART

WHERE DEPART.DEPT\_NO = EMPLE.DEPT\_NO

AND UPPER(LOC) = 'MADRID';

1. SELECT APELLIDO, OFICIO

FROM EMPLE

WHERE OFICIO IN (SELECT OFICIO

FROM EMPLE

WHERE UPPER(APELLIDO) = 'JIMENEZ');

1. SELECT APELLIDO, OFICIO, SALARIO

FROM EMPLE

WHERE SALARIO > (SELECT AVG(SALARIO)

FROM EMPLE);

1. SELECT APELLIDO, OFICIO, SALARIO

FROM EMPLE

WHERE SALARIO > ALL (SELECT SALARIO

FROM EMPLE

WHERE UPPER(APELLIDO) = 'MARTIN');

1. SELECT APELLIDO, OFICIO, DEPT\_NO

FROM EMPLE

WHERE DEPT\_NO IN (SELECT DEPT\_NO

FROM EMPLE

WHERE UPPER(APELLIDO) = 'FERNANDEZ');

1. SELECT APELLIDO, OFICIO, SALARIO

FROM EMPLE

WHERE SALARIO > (SELECT MIN (SALARIO)

FROM EMPLE);

1. SELECT APELLIDO, OFICIO, SALARIO

FROM EMPLE

WHERE SALARIO >ANY (SELECT SALARIO

FROM EMPLE

WHERE UPPER(OFICIO) = 'VENDEDOR');

1. SELECT APELLIDO, OFICIO, SALARIO,

FROM EMPLE

WHERE SALARIO <ALL (SELECT SALARIO

FROM EMPLE

WHERE UPPER(OFICIO) = 'DIRECTOR');

1. SELECT APELLIDO, OFICIO, LOC

FROM EMPLE, DEPART

WHERE EMPLE.DEPT\_NO = DEPART.DEPT\_NO

AND EMPLE.DEPT\_NO NOT IN (

SELECT DISTINCT DEPT\_NO

FROM EMPLE, DEPART

WHERE EMPLE.DEPT\_NO = DEPART.DEPT\_NO

AND UPPER(LOC) = 'BARCELONA');

1. SELECT APELLIDO, OFICIO, SALARIO

FROM EMPLE

WHERE SALARIO > ALL (SELECT SALARIO

FROM EMPLE

WHERE UPPER(OFICIO) = 'ANALISTA');

1. SELECT APELLIDO, OFICIO, SALARIO

FROM EMPLE

WHERE SALARIO IN ( SELECT SALARIO

FROM EMPLE

WHERE UPPER(OFICIO) = 'EMPLEADO');

1. SELECT APELLIDO, OFICIO, SALARIO

FROM EMPLE

WHERE SALARIO > (SELECT SALARIO

FROM EMPLE

WHERE UPPER(APELLIDO) = 'TOVAR')

AND SALARIO < (SELECT SALARIO

FROM EMPLE

WHERE UPPER(APELLIDO) = 'NEGRO');

1. SELECT APELLIDO, OFICIO, DEPT\_NO

FROM EMPLE

WHERE DEPT\_NO NOT IN (SELECT DEPT\_NO

FROM EMPLE

WHERE UPPER(OFICIO) = 'VENDEDOR');

1. SELECT APELLIDO, OFICIO, DEPT\_NO, COMISION

FROM EMPLE

WHERE COMISION IS NULL OR COMISION = 0;

**JOIN ( = REJUNTAR )**

Lo que conocíamos hasta ahora:

Select NOMBRE

FROM CLIENTE, COCHE

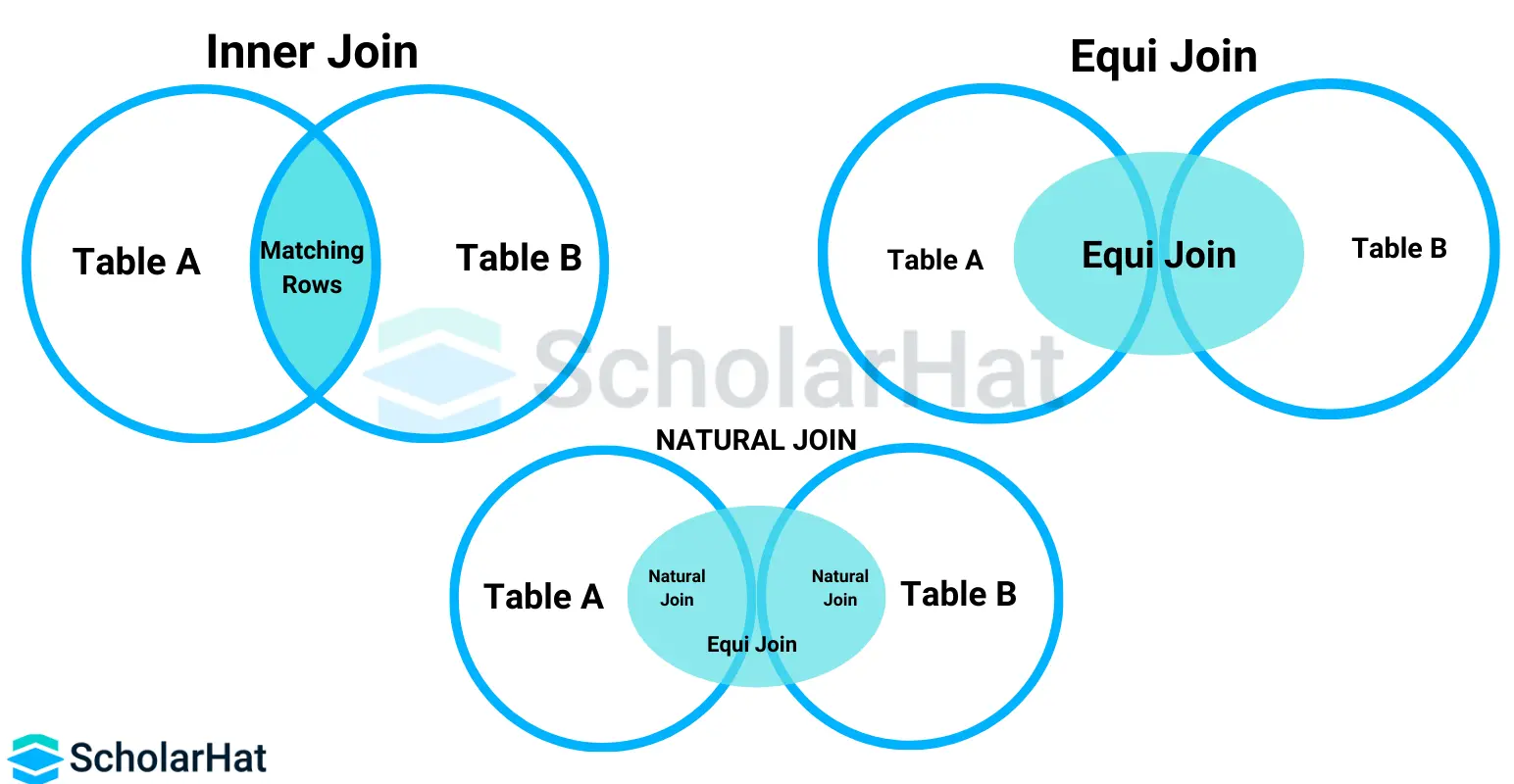
WHERE cliente.ID\_CLIENTE = coche.ID\_CLIENTE

and upper(MARCA) = ‘OPEL’;

-> se puede seguir trabajando así con condiciones de enlace pero hay una serie

de joins que también se emplean y hacen lo mismo que la condición de enlace.

**OPERACIONES DE JOIN**



* **NATURAL Join**: Une dos tablas de forma natural, uniéndose por la condición de enlace, ejemplo: Dime los apellidos de los empleados de los departamentos de ventas.

Hasta ahora…

SELECT APELLIDO

FROM EMPLE, DEPART

WHERE EMPLE.Dept\_no = DEPART.Dept\_no

AND UPPER(DEPART.Dnombre)='VENTAS';

-Si se nos olvida poner el enlace saldrá un producto cartesiano y mezclará todos los empleados con todos los departamentos -> resultados falsos.

— LA CONSULTA EQUIVALENTE CON JOIN SERÍA—

SELECT APELLIDO

FROM EMPLE **JOIN** DEPART

**ON** EMPLE.Dept\_no = DEPART.Dept\_no

WHERE UPPER(DEPART.Dnombre) = ‘VENTAS’;

Te ahorras el where y pones la condición de enlace después del ON, entonces te ahorras la coma también; y el join lo tienes que poner al definir las tablas que vas a usar en el from.

**CON LA NUEVA TABLA CLIENTE:**

**CONSULTA: \*Quiero saber los nombres de los clientes atendidos por empleados de departamentos de ‘VENTAS’.**

**-CON 2 ENLACES DE CONDICION ( VERSIÓN ORIGINAL )**

SELECT CLIENTE.NOMBRE, DEPART.DNOMBRE, EMPLE.APELLIDO

FROM CLIENTE, EMPLE, DEPART

WHERE DEPART.DEPT\_NO = EMPLE.DEPT\_NO

AND CLIENTE.EMP\_NO = EMPLE.EMP\_NO

AND UPPER(DNOMBRE) = ‘VENTAS’;

**-CON JOIN**

SELECT CLIENTE.NOMBRE, DEPART.DNOMBRE, EMPLE.APELLIDO

FROM CLIENTE

JOIN EMPLE

ON CLIENTE.EMP\_NO = EMPLE.EMP\_NO

JOIN DEPART

ON EMPLE.DEPT\_NO = DEPART.DEPT\_NO

WHERE UPPER(DEPART.DNOMBRE) = 'VENTAS';

**EJERCICIOS EDITORIAL HECHOS CON JOIN:**

**TABLAS: LIBRO, EDITOR, IMPRENTA, AUTOR.**

- Nombre de todos los autores que tenemos

- Nombre de los editores

- Nombre de los editores que trabajan con el libro "El Sol Brilla"

- Nombre de los libros que ha editado "Lalo Lález"

- Los libros que aún están en revisión

- Los nombres de imprenta que me han impreso algún libro

- Los nombres de imprenta que me han impreso algún libro del autor "Lolo Lólez"

**- Nombre de todos los autores que tenemos**

SELECT NOMBRE\_AUTOR

FROM AUTOR;

**- Nombre de los editores**

SELECT NOMBRE\_EDITOR

FROM EDITOR;

**- Nombre de los editores que trabajan con el libro "El Sol Brilla"**

CLÁSICO

SELECT TITULO\_LIBRO, NOMBRE\_EDITOR

FROM LIBRO, EDITOR

WHERE LIBRO.ID\_EDITOR = EDITOR.ID\_EDITOR

AND UPPER (LIBRO.TITULO\_LIBRO) = ‘EL SOL BRILLA’;

*CON JOIN:*

SELECT TITULO\_LIBRO, NOMBRE\_EDITOR

FROM LIBRO JOIN EDITOR

ON LIBRO.ID\_EDITOR = EDITOR.ID\_EDITOR

AND UPPER (LIBRO.TITULO\_LIBRO) = ‘EL SOL BRILLA’;

**- Nombre de los libros que ha editado "Lalo Lález"**

CLÁSICO

SELECT TITULO\_LIBRO, NOMBRE\_EDITOR

FROM LIBRO, EDITOR

WHERE LIBRO.ID\_EDITOR = EDITOR.ID\_EDITOR

AND UPPER(NOMBRE\_EDITOR) = ‘LALO LÁLEZ’;

CON JOIN

SELECT TITULO\_LIBRO, NOMBRE\_EDITOR

FROM LIBRO JOIN EDITOR

ON LIBRO.ID\_EDITOR = EDITOR.ID\_EDITOR

AND UPPER(NOMBRE\_EDITOR) = ‘LALO LÁLEZ’;

**- Los libros que aún están en revisión**

SELECT TITULO\_LIBRO, ESTADO\_LIBRO

FROM LIBRO

WHERE ESTADO\_LIBRO = ‘En revisión’;

**- Los nombres de imprenta que me han impreso algún libro**

CLÁSICA

SELECT TITULO\_LIBRO, NOMBRE\_IMPRENTA, ESTADO\_LIBRO

FROM IMPRENTA,LIBRO

WHERE LIBRO.ID\_IMPRENTA = IMPRENTA.ID\_IMPRENTA

AND

UPPER(ESTADO\_LIBRO) = ‘PUBLICADO’;

CON JOIN

SELECT TITULO\_LIBRO, NOMBRE\_IMPRENTA, ESTADO\_LIBRO

FROM IMPRENTA JOIN LIBRO

ON LIBRO.ID\_IMPRENTA = IMPRENTA.ID\_IMPRENTA

AND

UPPER(ESTADO\_LIBRO) = ‘PUBLICADO’;

**- Los nombres de imprenta que me han impreso algún libro del autor "Lolo Lólez"**

CLÁSICO

SELECT NOMBRE\_IMPRENTA

FROM IMPRENTA, LIBRO, AUTOR

WHERE IMPRENTA.ID\_IMPRENTA = LIBRO.ID\_IMPRENTA

AND LIBRO.ID\_AUTOR = AUTOR.ID\_AUTOR

AND AUTOR.NOMBRE\_AUTOR = 'Lolo Lólez'

AND LIBRO.ESTADO\_LIBRO = ‘Publicado’;

CON JOIN

SELECT NOMBRE\_IMPRENTA

FROM LIBRO JOIN IMPRENTA

ON IMPRENTA.ID\_IMPRENTA = LIBRO.ID\_IMPRENTA

JOIN AUTOR

ON LIBRO.ID\_AUTOR = AUTOR.ID\_AUTOR

AND AUTOR.NOMBRE\_AUTOR = ‘Lolo Lólez’

AND LIBRO.ESTADO\_LIBRO = ‘Publicado’;