EXPLICACIÓN SQL:

- SUBCONSULTAS
- INNER JOIN

Subconsultas

- Se utilizan para realizar filtrados con los datos de otra consulta.
- Consiste en utilizar los resultados de una consulta dentro de otra, que se considera la principal.
- Se pueden aplicar en la cláusula WHERE para filtrar registros.
- Se pueden aplicar en la cláusula HAVING para filtrar grupos.
- Aumenta la eficiencia ya que los valores devueltos de una consulta son los datos de entrada de otra de nivel superior, por ello se dice que están anidadas.
- Tiene que existir un **conector** entre la consulta principal y la anidada.

¿Qué libros se prestaron al estudiante Manuel Díaz?

LIBROS
….
idLibro PK

PRESTAMO
…
idLibro FK
idLector FK

ESTUDIANTE
…
idLector PK

¿Qué libros se prestaron al estudiante Manuel Díaz?

Empezamos por la consulta más interna:

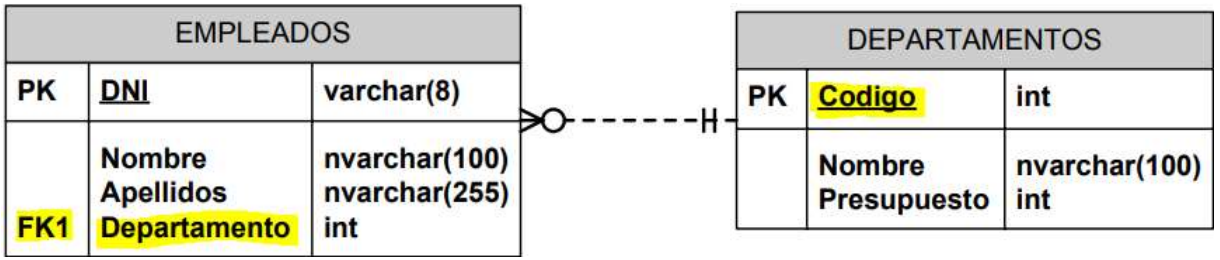
Seleccionamos de la tabla ***estudiante*** los libros cuyo *nombre* coincide con el lector Manuel Díaz.

Seleccionamos de la tabla ***prestamo*** los libros en los que el *idLector* es igual al i*dLector* de la tabla *estudiante*, es decir, los de Manuel Díaz

Seleccionamos de la tabla ***libro*** los libros cuyo *idLibro* es igual al *idLibro* de la tabla *prestamo*, en los que habíamos filtrado previamente los de Manuel Díaz.

```
SELECT *
FROM libro
WHERE idLibro IN(
                SELECT idLibro
                FROM prestamo
                WHERE idLector IN(
                                SELECT idLector
                                FROM estudiante
                                WHERE nombre='Mnauel Díaz'
                                )
                );
```

Obtener los nombres y apellidos de los empleados
que trabajan
en departamentos cuyo presupuesto sea mayor de 60.000 €




```
# Con subconsulta
SELECT Nombre, Apellidos FROM empleados
    WHERE Departamento IN(
                        SELECT Codigo
                        FROM Departamentos
                        WHERE Presupuesto>60000
                        )
```

Obtener los datos de los departamentos

cuyo presupuesto es superior al presupuesto medio de todos los departamentos.

```
SELECT *
  FROM Departamentos
  WHERE presupuesto >
     (
        SELECT AVG(presupuesto)
        FROM Departamentos
     );
```

Obtener los nombres de los departamentos

cuyo departamento tiene más de dos empleados.

```
SELECT nombre
FROM Departamentos
WHERE codigo IN
     (
        SELECT departamento
        FROM Empleados
        GROUP BY departamento
        HAVING COUNT(*) > 2
     );
```

PRODUCTO CARTESIANO (composiciones cruzadas)

Si realizamos una consulta sobre dos tablas sin establecer ninguna condición, lo que obtenemos es el producto cartesiano, es decir, la combinación de cada uno de los registros de una tabla con cada uno de los registros de la otra, aunque no exista relación entre ellos.

Se puede obtener el producto cartesiano de tantas tablas como se quiera.

En este ejemplo, tenemos una tabla de 3 filas y otra de cuatro así que el resultado es una tabla de 3x4=12 filas.

Ejemplo:

TFabricante	
código	nombre
1	Asus
2	Lenovo
3	Hewlett-Packard

TProducto			
código	nombre	precio	fkcodigo
1	Monitor 24 LED Full HD	202	1
2	Monitor 27 LED Full HD	245.99	1
3	Portátil Yoga 520	559	2
4	Portátil Ideapd 320	444	2

La consulta que se ejecutaría sería:

```
SELECT * FROM TFabricante, TProducto
```

cuya salida sería:

TFabricante x TProducto					
código	nombre	código	nombre	precio	fkcodigo
1	Asus	1	Monitor 24 LED Full HD	202	1
2	Lenovo	1	Monitor 24 LED Full HD	202	1
3	Hewlett-Packard	1	Monitor 24 LED Full HD	202	1
1	Asus	2	Monitor 27 LED Full HD	245.99	1
2	Lenovo	2	Monitor 27 LED Full HD	245.99	1
3	Hewlett-Packard	2	Monitor 27 LED Full HD	245.99	1
1	Asus	3	Portátil Yoga 520	559	2
2	Lenovo	3	Portátil Yoga 520	559	2
3	Hewlett-Packard	3	Portátil Yoga 520	559	2
1	Asus	4	Portátil Ideapd 320	444	2
2	Lenovo	4	Portátil Ideapd 320	444	2
3	Hewlett-Packard	4	Portátil Ideapd 320	444	2

JOIN (composiciones internas)

La operación del producto cartesiano no es de las más utilizadas ya que asocia todos los registros de una tabla con cada uno de los de la otra, aunque no exista relación entre ellos.

Lo más normal es que queramos seleccionar los registros de varias tablas que tengan alguna relación, siguiendo algún criterio.

Pondremos alguna condición para que solo aparezcan las filas de una tabla que estén relacionadas con las de la otra.

A esto se le llama **asociar tablas (JOIN).**

La intersección de dos conjuntos es una operación que obtiene otro conjunto que contiene solo los elementos comunes de ambos conjuntos.

Para hacer una composición interna se parte de un producto cartesiano y se eliminan las filas que no cumplen la condición de composición. Solo se emparejarán los campos que tengas valores iguales.

Reglas de composición:

Lo importante en las composiciones internas es emparejar los campos que tengas valores iguales.

Las reglas de composición son:

- Pueden combinarse tantas tablas como se desee
- El criterio de combinación puede estar formado por más de una pareja de columnas.
- En la cláusula ***SELECT*** pueden citarse columnas de ambas tablas, condicionen o no, la combinación.
- Si hay columnas con el mismo nombre en las distintas tablas, deben identificarse especificando la tabla de procedencia o utilizando un alias de tabla.
- Las columnas que aparecen en la cláusula ***WHERE*** se denominan **columnas de emparejamiento** ya que son las que permiten emparejar las filas de las dos tablas. Estas no tienen por qué estar incluidas en la lista de selección.
- Emparejamos tablas que estén relacionadas entre sí y además, **una de las columnas de emparejamiento debe ser clave principal de su tabla**.

JOIN

Vamos a filtrar la información, lo que nos interesa es obtener una relación de los productos pertenecientes a cada fabricante, para eso, tendremos que seleccionar solo aquellos registros de la tabla ***TProducto*** donde la clave ajena que se refiere al fabricante (***fkcodigo***) coincida con el ***código*** del fabricante en la tabla ***Tfabricante***.

Es decir, el campo clave del fabricante debe coincidir con la clave ajena de los productos, que es el campo por el que las dos tablas están relacionadas.

TFabricante x TProducto					
código	nombre	código	nombre	precio	fkcodigo
1	Asus	1	Monitor 24 LED Full HD	202	1
1	Asus	2	Monitor 27 LED Full HD	245.99	1
2	Lenovo	3	Portátil Yoga 520	559	2
2	Lenovo	4	Portátil Ideapd 320	444	2

```
SELECT *
FROM TFabricante, TProducto
WHERE TFabricante.pkcodigo = TProducto.fkcodigo_fabricante;
```

TFabricante x TProducto					
código	nombre	código	nombre	precio	fkcodigo
1	Asus	1	Monitor 24 LED Full HD	202	1
1	Asus	2	Monitor 27 LED Full HD	245.99	1
2	Lenovo	3	Portátil Yoga 520	559	2
2	Lenovo	4	Portátil Ideapd 320	444	2

Podríamos realizar la consulta sin repetir campos y utilizando alias.

```
SELECT TFabricante.nombre as Fabricante, TProducto.nombre as Producto, Precio
FROM TFabricante, TProducto
WHERE TFabricante.codigo = TProducto.fkcodigo;
```

Fabricante	Producto	Precio
Asus	Monitor 24 LED Full HD	202
Asus	Monitor 27 LED Full HD	245.99
Lenovo	Portátil Yoga 520	559
Lenovo	Portátil Ideapd 320	444

JOIN, INNER JOIN

La palabra INNER pude omitirse, INNER JOIN es lo mismo que JOIN.

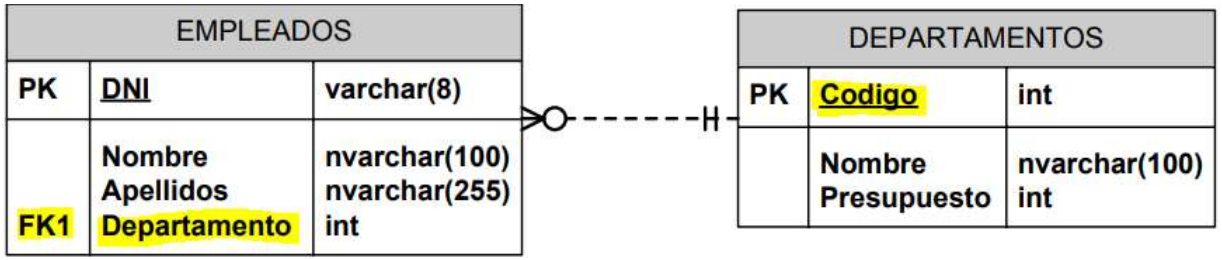
Con la operación de intersección INNER JOIN sólo obtendremos los elementos de existan en ambos conjuntos.

Por lo tanto, en el ejemplo anterior puede ser que existan filas en la tabla fabricante que no aparecen en el resultado porque no tienen ningún producto asociado.

También podrían existir filas en la tabla producto que no aparecen en el resultado porque no tienen ningún

JOIN, INNER JOIN, SELECT

Obtener los nombres y apellidos de los empleados
que trabajan
en departamentos cuyo presupuesto sea mayor de 60.000 €



Estas dos sentencias son equivalentes.

```
# Con subconsulta
SELECT Nombre, Apellidos FROM empleados
    WHERE Departamento IN(
                        SELECT Codigo
                        FROM Departamentos
                        WHERE Presupuesto>60000
                        )
```

```
# Sin subconsulta
SELECT Empleados.Nombre, Apellidos
        FROM Empleados INNER JOIN Departamentos
            ON Empleados.Departamento = Departamentos.Codigo
                AND Departamentos.Presupuesto > 60000
```

NATURAL JOIN

```
SELECT *
FROM TFabricante NATURAL JOIN TProducto;
```

Esta consulta nos devolvería la intersección de las dos tablas, pero solo en el caso de que las columnas que utilizamos para relacionarlas tuviesen el mismo nombre.

Sólo deberíamos utilizar una composición de tipo NATURAL JOIN cuando estemos seguros que los nombres de las columnas sobre las que quiero relacionar las dos tablas se llaman igual en las dos tablas.

Lo normal es que no suelan tener el mismo nombre y que debamos usar una composición de tipo INNER JOIN.

LEFT JOIN, RIGTH JOIN (composiciones externas)

A veces es necesario seleccionar algunas filas de una tabla aunque no tengan correspondencia con las filas de la otra tabla.

A esta composición se le llaman OUTER JOINS: Left Joins / Rigth Joins. Permite seleccionar filas de una tabla resultante de una combinación aunque estas no tengan correspondencia. Dependiendo donde se encuentre se llaman LEFT JOIN o RIGTH JOIN.

JOIN:

1

```
/* SQL 2 */
SELECT *
FROM empleado JOIN departamento
ON empleado.id_departamento = departamento.id
```

2

Tabla: empleado

id	nombre	id_departamento
1	Pepe	1
2	María	2
3	Juan	NULL

Tabla: departamento

id	nombre
1	Desarrollo
2	Sistemas
3	Recursos Humanos

Estas filas quedan **fuera de la intersección**

3

Todas las filas de las dos tablas que quedan fuera de la intersección se descartan

El **resultado de la operación RIGHT JOIN** es:

empleado.id	empleado.nombre	empleado.id_departamento	departamento.id	departamento.nombre
1	Pepe	1	1	Desarrollo
2	María	2	2	Sistemas

LEFT JOIN:

1

```
/* SQL 2 */
SELECT *
FROM empleado LEFT JOIN departamento
ON empleado.id_departamento = departamento.id
```

2

Tabla: empleado

id	nombre	id_departamento
1	Pepe	1
2	María	2
3	Juan	NULL

Tabla: departamento

id	nombre
1	Desarrollo
2	Sistemas
3	Recursos Humanos

Estas filas quedan **fuera de la intersección**

Todas las filas de la izquierda se devuelven, aunque no haya ninguna columna correspondiente en las tablas combinadas

3

El **resultado de la operación LEFT JOIN** es:

empleado.id	empleado.nombre	empleado.id_departamento	departamento.id	departamento.nombre
1	Pepe	1	1	Desarrollo
2	María	2	2	Sistemas
3	Juan	NULL	NULL	NULL

RIGTH JOIN:

1

```
/* SQL 2 */
SELECT *
FROM empleado RIGHT JOIN departamento
ON empleado.id_departamento = departamento.id
```

2

Tabla: empleado

id	nombre	id_departamento
1	Pepe	1
2	María	2
3	Juan	NULL

Tabla: departamento

id	nombre
1	Desarrollo
2	Sistemas
3	Recursos Humanos

Estas filas quedan **fuera de la intersección**

3

El **resultado de la operación RIGHT JOIN** es:

empleado. id	empleado. nombre	empleado. id_departamento	departamento. id	departamento. nombre
1	Pepe	1	1	Desarrollo
2	María	2	2	Sistemas
NULL	NULL	NULL	3	Recursos Humanos

Todas las filas de la derecha se devuelven, aunque no haya ninguna columna correspondiente en las tablas combinadas

Tabla: empleado

id	nombre	id_departamento
1	Pepe	1
2	María	2
3	Juan	NULL

Tabla: departamento

id	nombre
1	Desarrollo
2	Sistemas
3	Recursos Humanos

Estas filas quedan **fuera de la intersección**

2- LEFT JOIN

muestra todos los registros de la tabla izquierda aunque no tengan coincidencia y las de la parte derecha que tengan coincidencia, las de la intersección.

```
SELECT *
FROM empleado LEFT JOIN departamento
ON empleado.id_departamento = departamento.id
```

empleado. id	empleado. nombre	empleado. id_departamento	departamento. id	departamento. nombre
1	Pepe	1	1	Desarrollo
2	María	2	2	Sistemas
3	Juan	NULL	NULL	NULL

1- JOIN

muestra solo coincidencias en las dos tablas, es decir, los registros que quedan en la intersección. Descarta el resto

```
SELECT *
FROM empleado JOIN departamento
ON empleado.id_departamento = departamento.id
```

empleado. id	empleado. nombre	empleado. id_departamento	departamento. id	departamento. nombre
1	Pepe	1	1	Desarrollo
2	María	2	2	Sistemas

3- RIGHT JOIN

muestra todas los registros de la tabla derecha aunque no tengan coincidencia y los de la tabla izquierda que tengan coincidencia, los de la intersección.

```
SELECT *
FROM empleado RIGHT JOIN departamento
ON empleado.id_departamento = departamento.id
```

empleado. id	empleado. nombre	empleado. id_departamento	departamento. id	departamento. nombre
1	Pepe	1	1	Desarrollo
2	María	2	2	Sistemas
NULL	NULL	NULL	3	Recursos Humanos