
Unidad Didáctica 6. Consultas sobre varias tablas. Composición interna y cruzada

Apuntes de BD para DAW, DAM y ASIR

José Juan Sánchez Hernández

Curso 2023/2024

Índice

1 Consultas sobre varias tablas. Composición interna y cruzada	1
1.1 Consultas multitable SQL 1	1
1.1.1 Composiciones cruzadas (Producto cartesiano)	1
1.1.2 Composiciones internas (Intersección)	3
1.2 Consultas multitable SQL 2	7
1.2.1 Composiciones cruzadas	7
1.2.2 Composiciones internas	7
1.2.3 Composiciones externas	8
1.3 El orden en las tablas no afecta al resultado final	12
1.4 Podemos usar alias en las tablas	13
1.5 Unir tres o más tablas	14
1.6 Unir una tabla consigo misma (<i>self-equi-join</i>)	14
1.7 Unir tablas de diferentes bases de datos	15
1.8 Uniones equivalentes (<i>equi-joins</i>) y Uniones no equivalentes (<i>non-equi-joins</i>)	15
2 Errores comunes	16
3 Referencias	18
4 Licencia	19

Índice de figuras

Índice de cuadros

1 Consultas sobre varias tablas. Composición interna y cruzada

Las consultas multitabla nos permiten consultar información en más de una tabla. La única diferencia respecto a las consultas sencillas es que vamos a tener que especificar en la cláusula **FROM** cuáles son las tablas que vamos a usar y cómo las vamos a relacionar entre sí.

Para realizar este tipo de consultas podemos usar dos alternativas, la sintaxis de **SQL 1** (SQL-86), que consiste en realizar el producto cartesiano de las tablas y añadir un filtro para relacionar los datos que tienen en común, y la sintaxis de **SQL 2** (SQL-92 y SQL-2003) que incluye todas las cláusulas de tipo **JOIN**.

1.1 Consultas multitabla SQL 1

1.1.1 Composiciones cruzadas (Producto cartesiano)

El **producto cartesiano** de dos conjuntos, es una operación que consiste en obtener otro conjunto cuyos elementos son **todas las parejas que pueden formarse entre los dos conjuntos**. Por ejemplo, tendríamos que coger el primer elemento del primer conjunto y formar una pareja con cada uno de los elementos del segundo conjunto. Una vez hecho esto, repetimos el mismo proceso para cada uno de los elementos del primer conjunto.

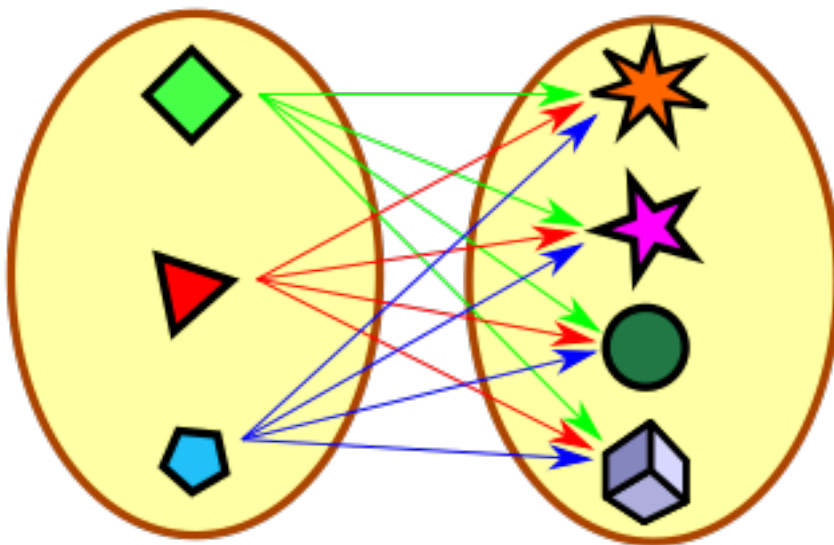


Imagen: Imagen extraída de [Wikipedia](#). Autor: [GermanX](#)

Ejemplo

Suponemos que tenemos una base de datos con dos tablas: `empleado` y `departamento`.

```

1 SELECT *
2 FROM empleado;
3
4 +--
5 | codigo | nif          | nombre      | apellido1 | apellido2 |
6 |-----+-----+-----+-----+-----+
7 |      1 | 32481596F   | Aarón      | Rivero    | Gómez     |
8 |      2 | Y5575632D  | Adela      | Salas     | Díaz      |
9 |      3 | R6970642B  | Adolfo     | Rubio     | Flores    |
10 +--
11
12 SELECT *
13 FROM departamento;
14
15 +-----+-----+-----+
16 | codigo | nombre          | presupuesto |
17 +-----+-----+-----+
18 |      1 | Desarrollo      | 120000     |
19 |      2 | Sistemas        | 150000     |
20 |      3 | Recursos Humanos | 280000     |
21 +-----+-----+-----+
    
```

El **producto cartesiano** de las dos tablas se realiza con la siguiente consulta:

```

1 SELECT *
2 FROM empleado, departamento;
    
```

El resultado sería el siguiente:

	codigo	nif	nombre	apellido1	apellido2	codigo_departamento	
	codigo	nombre	presupuesto	gastos			
4	1	32481596F	Aarón	Rivero	Gómez	1	1
5	2	Y5575632D	Adela	Salas	Díaz	2	1
6	3	R6970642B	Adolfo	Rubio	Flores	3	1
7	1	32481596F	Aarón	Rivero	Gómez	1	2

8	2	Y5575632D	Adela	Salas	Díaz	2	2
		Sistemas		150000	21000		
9	3	R6970642B	Adolfo	Rubio	Flores	3	2
		Sistemas		150000	21000		
10	1	32481596F	Aarón	Rivero	Gómez	1	3
		Recursos Humanos		280000	25000		
11	2	Y5575632D	Adela	Salas	Díaz	2	3
		Recursos Humanos		280000	25000		
12	3	R6970642B	Adolfo	Rubio	Flores	3	3
		Recursos Humanos		280000	25000		
13	+-----+-----+-----+-----+-----+-----+-----+						

1.1.2 Composiciones internas (Intersección)

La **intersección de dos conjuntos** es una operación que resulta en otro conjunto que contiene **sólo los elementos comunes** que existen en ambos conjuntos.

$$A = \{ \text{pentágono naranja}, \text{rombo azul}, \text{cuadrado verde}, \text{rectángulo amarillo} \}$$

$$B = \{ \text{estrella roja}, \text{cuadrado verde}, \text{triángulo verde}, \text{pentágono naranja} \}$$

$$A \cap B$$

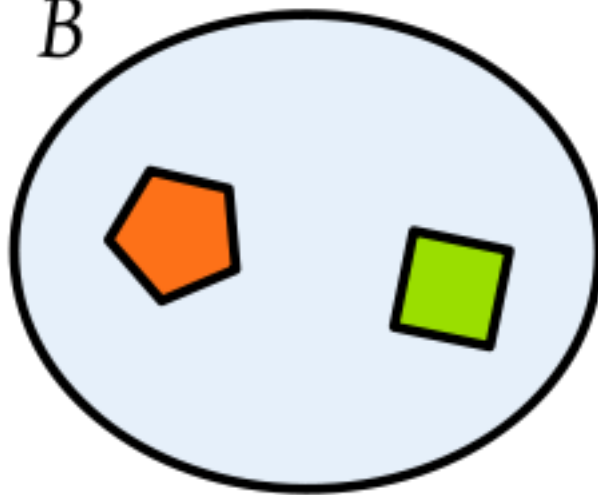


Imagen: Imagen extraída de [Wikipedia](#). Autor: Kismalac.

Ejemplo

Para poder realizar una **operación de intersección** entre las dos tablas debemos utilizar la cláusula **WHERE** para indicar la columna con la que queremos relacionar las dos tablas. Por ejemplo, para obtener un listado de los empleados y el departamento donde trabaja cada uno podemos realizar la siguiente consulta:

```

1 SELECT *
2 FROM empleado, departamento
3 WHERE empleado.codigo_departamento = departamento.codigo
    
```

El resultado sería el siguiente:

1	codigo	nif	nombre	apellido1	apellido2	codigo_departamento
2	codigo	nombre	presupuesto	gastos		

3								
4	1	32481596F	Aarón	Rivero	Gómez	1	1	
		Desarrollo	120000	6000				
5	2	Y5575632D	Adela	Salas	Díaz	2	2	
		Sistemas	150000	21000				
6	3	R6970642B	Adolfo	Rubio	Flores	3	3	
		Recursos Humanos	280000	25000				
7								

Nota: Tenga en cuenta que con la **operación de intersección** sólo obtendremos los elementos de existan en ambos conjuntos. Por lo tanto, en el ejemplo anterior puede ser que existan filas en la tabla `empleado` que no aparecen en el resultado porque no tienen ningún departamento asociado, al igual que pueden existir filas en la tabla `departamento` que no aparecen en el resultado porque no tienen ningún empleado asociado.

INNER JOIN

1

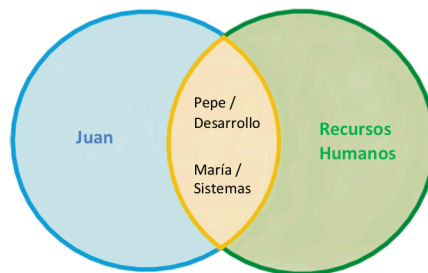
```
/* SQL 1 */
SELECT *
FROM empleado, departamento
WHERE empleado.id_departamento = departamento.id
```

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

2

Tabla: empleado			Tabla: departamento	
id	nombre	id_departamento	id	nombre
1	Pepe	1	1	Desarrollo
2	María	2	2	Sistemas
3	Juan	NULL	3	Recursos Humanos

Estas filas quedan **fuera de la intersección**



3

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

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



<http://josejuansanchez.org/bd>

1.2 Consultas multitabla SQL 2

1.2.1 Composiciones cruzadas

- Producto cartesiano
 - `CROSS JOIN`

Ejemplo de `CROSS JOIN`:

```
1 SELECT *
2 FROM empleado CROSS JOIN departamento
```

Esta consulta nos devolvería el producto cartesiano de las dos tablas.

1.2.2 Composiciones internas

- Join interna
 - `INNER JOIN` o `JOIN`
 - `NATURAL JOIN`

Ejemplo de `INNER JOIN` utilizando la cláusula `ON`:

```
1 SELECT *
2 FROM empleado INNER JOIN departamento
3 ON empleado.codigo_departamento = departamento.codigo
```

Esta consulta nos devolvería la intersección entre las dos tablas.

La palabra reservada `INNER` es opcional, de modo que la consulta anterior también se puede escribir así:

```
1 SELECT *
2 FROM empleado JOIN departamento
3 ON empleado.codigo_departamento = departamento.codigo
```

NOTA: Tenga en cuenta que **si olvidamos incluir la cláusula `ON` obtendremos el producto cartesiano de las dos tablas.**

Por ejemplo, la siguiente consulta nos devolverá el producto cartesiano de las tablas `empleado` y `departamento`.

```
1 SELECT *
2 FROM empleado INNER JOIN departamento
```

Cuando queremos realizar una composición interna entre dos tablas y las columnas que queremos relacionar tienen el mismo nombre en ambas tablas podemos utilizar la cláusula `USING`.

Ejemplo de `INNER JOIN` utilizando la cláusula `USING`:

Supongamos que la clave primaria de la tabla `departamento` se llama `codigo_departamento` y la clave ajena de a tabla `empleado` se llama también `codigo_departamento`. En este caso podríamos realizar la siguiente consulta:

```
1 SELECT *
2 FROM empleado INNER JOIN departamento
3 USING (codigo_departamento)
```

Ejemplo de NATURAL JOIN:

```
1 SELECT *
2 FROM empleado NATURAL JOIN departamento
```

Esta consulta nos devolvería la intersección de las dos tablas, pero utilizaría las columnas que tengan el mismo nombre para relacionarlas. En este caso usaría las columnas `código` y `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 suelen tener el mismo nombre y que debemos usar una composición de tipo `INNER JOIN`.

1.2.3 Composiciones externas

- Join externa
 - LEFT OUTER JOIN
 - RIGHT OUTER JOIN
 - FULL OUTER JOIN (No implementada en MySQL)
 - NATURAL LEFT OUTER JOIN
 - NATURAL RIGHT OUTER JOIN

Ejemplo de LEFT OUTER JOIN:

```
1 SELECT *
2 FROM empleado LEFT JOIN departamento
3 ON empleado.codigo_departamento = departamento.codigo
```

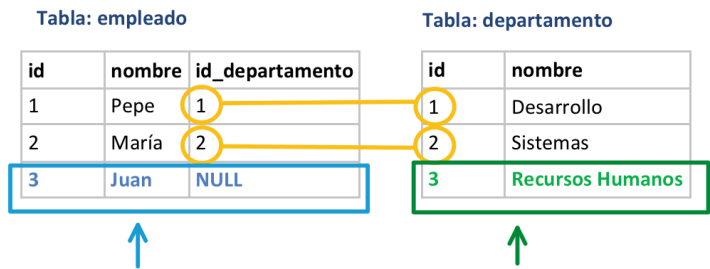
Esta consulta devolverá todas las filas de la tabla que hemos colocado a la izquierda de la composición, en este caso la tabla `empleado`. Y relacionará las filas de la tabla de la izquierda (`empleado`) con las filas de la tabla de la derecha (`departamento`) con las que encuentre una coincidencia. Si no encuentra ninguna coincidencia, se mostrarán los valores de la fila de la tabla izquierda (`empleado`) y en los valores de la tabla derecha (`departamento`) donde no ha encontrado una coincidencia mostrará el valor `NULL`.

LEFT JOIN

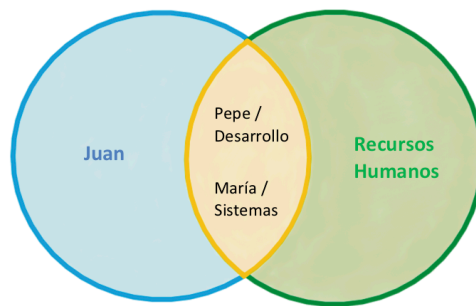
1

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

2



Estas filas quedan **fuera de la intersección**



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



<http://josejuansanchez.org/bd>

Ejemplo de RIGHT OUTER JOIN:

```
1 SELECT *
2 FROM empleado RIGHT JOIN departamento
3 ON empleado.codigo_departamento = departamento.codigo
```

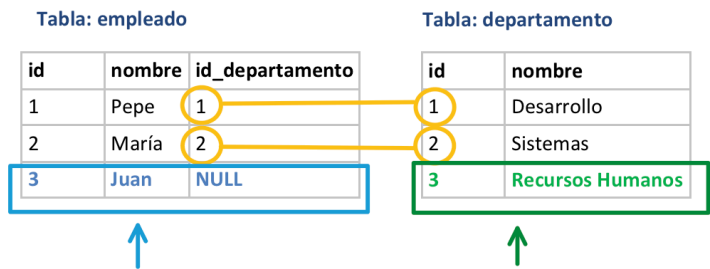
Esta consulta devolverá todas las filas de la tabla que hemos colocado a la derecha de la composición, en este caso la tabla `departamento`. Y relacionará las filas de la tabla de la derecha (`departamento`) con las filas de la tabla de la izquierda (`empleado`) con las que encuentre una coincidencia. Si no encuentra ninguna coincidencia, se mostrarán los valores de la fila de la tabla derecha (`departamento`) y en los valores de la tabla izquierda (`empleado`) donde no ha encontrado una coincidencia mostrará el valor `NULL`.

RIGHT JOIN

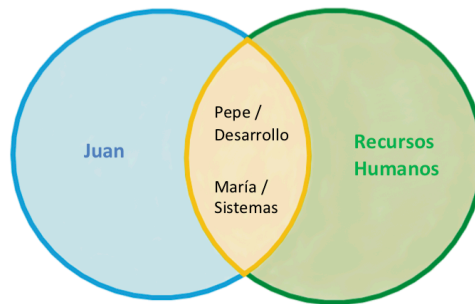
1

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

2



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



<http://josejuansanchez.org/bd>

Ejemplo de FULL OUTER JOIN:

La composición `FULL OUTER JOIN` **no está implementada en MySQL**, por lo tanto para poder simular esta operación será necesario hacer uso del operador `UNION`, que nos realiza la union del resultado de dos consultas.

El resultado esperado de una composición de tipo `FULL OUTER JOIN` es obtener la intersección de las dos tablas, junto las filas de ambas tablas que no se puedan combinar. Dicho con otras palabras, el resultado sería el equivalente a realizar la union de una consulta de tipo `LEFT JOIN` y una consultas de tipo `RIGHT JOIN` sobre las mismas tablas.

```

1 SELECT *
2 FROM empleado LEFT JOIN departamento
3 ON empleado.codigo_departamento = departamento.codigo
4
5 UNION
6
7 SELECT
8 FROM empleado RIGHT JOIN departamento
9 ON empleado.codigo_departamento = departamento.codigo

```

Ejemplo de `NATURAL LEFT JOIN`:

```

1 SELECT *
2 FROM empleado NATURAL LEFT JOIN departamento

```

Esta consulta realiza un `LEFT JOIN` entre las dos tablas, la única diferencia es que en este caso no es necesario utilizar la cláusula `ON` para indicar sobre qué columna vamos a relacionar las dos tablas. **En este caso las tablas se van a relacionar sobre aquellas columnas que tengan el mismo nombre.** Por lo tanto, sólo deberíamos utilizar una composición de tipo `NATURAL LEFT JOIN` cuando estemos seguros de que los nombres de las columnas sobre las que quiero relacionar las dos tablas se llaman igual en las dos tablas.

1.3 El orden en las tablas no afecta al resultado final

El orden en las tablas a la hora de realizar la operación de `INNER JOIN` no afecta al resultado final de la consulta, siempre que se indique el mismo orden de las columnas en la cláusula `SELECT`.

En el siguiente ejemplo se muestran dos consultas donde hemos modificado el orden en el que aparecen las tablas al realizar la operación de `INNER JOIN`. Sin embargo, las dos consultas devuelven el mismo resultado porque en la cláusula `SELECT` hemos indicado el mismo orden de las columnas.

```

1 SELECT empleado.apellido1, empleado.nombre, departamento.nombre
2 FROM empleado INNER JOIN departamento
3 ON empleado.codigo_departamento = departamento.codigo

```

```

1 SELECT empleado.apellido1, empleado.nombre, departamento.nombre
2 FROM departamento INNER JOIN empleado
3 ON empleado.codigo_departamento = departamento.codigo

```

El resultado de ambas consultas sería el mismo:

```

1 +-----+-----+-----+

```


2	empleado.apellido1	empleado.nombre	departamento.nombre	
3	+-----+-----+-----+			
4	

Si en las consultas anteriores hubiésemos utilizado el operador `*` en la cláusula `SELECT`, el resultado de ambas consultas habría sido diferente, ya que el orden de las columnas dependería del orden en el que aparecen las tablas en la operación de `INNER JOIN`.

Por ejemplo, estas dos consultas devolverían resultados diferentes:

```
1 /* 1 */
2 SELECT *
3 FROM empleado INNER JOIN departamento
4 ON empleado.codigo_departamento = departamento.codigo
```

```
1 /* 2 */
2 SELECT *
3 FROM departamento INNER JOIN empleado
4 ON empleado.codigo_departamento = departamento.codigo
```

El resultado de las dos consultas anteriores sería diferente:

```
1 /* 1 */
2 +-----+-----+-----+-----+-----+-----+
3 | codigo | nif      | nombre | apellido1 | apellido2 | codigo_departamento |
4 |      | nombre  |      | presupuesto | gastos |                       |
5 +-----+-----+-----+-----+-----+-----+-----+
```

```
1 /* 2 */
2 +-----+-----+-----+-----+-----+-----+
3 | codigo | nombre  | presupuesto | gastos | codigo | nif      |
4 |      | nombre | apellido1 | apellido2 | codigo_departamento |
5 +-----+-----+-----+-----+-----+-----+-----+
```

1.4 Podemos usar alias en las tablas

Para crear un alias en una tabla podemos utilizar la palabra reservada `AS` o escribir el nombre del alias directamente después del nombre de la tabla.

A continuación se muestra un ejemplo de cada caso.

Ejemplo: Cómo crear un alias de una tabla utilizando la palabra reservada `AS`.

```
1 SELECT *
2 FROM empleado AS e INNER JOIN departamento AS d
3 ON e.codigo_departamento = d.codigo
```

Ejemplo: Cómo crear un alias de una tabla sin utilizar la palabra reservada `AS`.

```
1 SELECT *
2 FROM empleado e INNER JOIN departamento d
3 ON e.codigo_departamento = d.codigo
```

1.5 Unir tres o más tablas

Podemos unir tres o más tablas en una misma operación de `INNER JOIN`.

Ejemplo:

```
1 SELECT *
2 FROM cliente INNER JOIN empleado
3 ON cliente.codigo_empleado_rep_ventas = empleado.codigo_empleado
4 INNER JOIN pago
5 ON cliente.codigo_cliente = pago.codigo_cliente;
```

1.6 Unir una tabla consigo misma (*self-join*)

Para poder hacer una operación de `INNER JOIN` sobre la misma tabla es necesario utilizar un alias para la tabla. A continuación se muestra un ejemplo de las dos formas posibles de hacer una operación de `INNER JOIN` sobre la misma tablas haciendo uso de alias.

Ejemplo: Cómo unir una tabla consigo misma.

```
1 SELECT empleado.nombre, empleado.apellido1, empleado.apellido2, jefe.nombre,
2 jefe.apellido1, jefe.apellido2
3 FROM empleado INNER JOIN empleado AS jefe
4 ON empleado.codigo_jefe = jefe.codigo_empleado
```

```
1 SELECT empleado.nombre, empleado.apellido1, empleado.apellido2, jefe.nombre,
2 jefe.apellido1, jefe.apellido2
3 FROM empleado INNER JOIN empleado jefe
4 ON empleado.codigo_jefe = jefe.codigo_empleado
```

Ejemplo: Cómo unir una tabla consigo misma varias veces.

```
1 SELECT empleado.*, jefe.*, super_jefe.*
2 FROM empleado INNER JOIN empleado AS jefe
3 ON empleado.codigo_jefe = jefe.codigo_empleado
4 INNER JOIN empleado AS super_jefe
5 ON jefe.codigo_jefe = super_jefe.codigo_empleado
```

```
1 SELECT empleado.*, jefe.*, super_jefe.*
2 FROM empleado INNER JOIN empleado jefe
3 ON empleado.codigo_jefe = jefe.codigo_empleado
4 INNER JOIN empleado super_jefe
5 ON jefe.codigo_jefe = super_jefe.codigo_empleado
```

1.7 Unir tablas de diferentes bases de datos

En algunos casos puede ser necesario unir tablas de diferentes bases de datos en una misma consulta. En este caso, tenemos que especificar el nombre de la base de datos antes del nombre de la tabla.

Ejemplo:

```
1 SELECT *
2 FROM db1.tabla1 AS t1 INNER JOIN db2.tabla2 AS t2
3 ON t1.codigo = t2.codigo
```

1.8 Uniones equivalentes (*equi-joins*) y Uniones no equivalentes (*non-equi-joins*)

TODO

2 Errores comunes

1. Nos olvidamos de incluir en el **WHERE** la condición que nos relaciona las dos tablas.

Consulta incorrecta

```
1 SELECT *
2 FROM producto, fabricante
3 WHERE fabricante.nombre = 'Lenovo';
```

Consulta correcta

```
1 SELECT *
2 FROM producto, fabricante
3 WHERE producto.codigo_fabricante = fabricante.codigo AND fabricante.nombre = 'Lenovo';
```

2. Nos olvidamos de incluir **ON** en las consultas de tipo **INNER JOIN**.

Consulta incorrecta

```
1 SELECT *
2 FROM producto INNER JOIN fabricante
3 WHERE fabricante.nombre = 'Lenovo';
```

Consulta correcta

```
1 SELECT *
2 FROM producto INNER JOIN fabricante
3 ON producto.codigo_fabricante = fabricante.codigo
4 WHERE fabricante.nombre = 'Lenovo';
```

3. Relacionamos las tablas utilizando nombres de columnas incorrectos.

Consulta incorrecta

```
1 SELECT *
2 FROM producto INNER JOIN fabricante
3 ON producto.codigo = fabricante.codigo;
```

Consulta correcta

```
1 SELECT *
2 FROM producto INNER JOIN fabricante
3 ON producto.codigo_fabricante = fabricante.codigo;
```

4. Cuando hacemos la intersección de tres tablas con **INNER JOIN** nos olvidamos de incluir **ON** en alguna de las intersecciones.

Consulta incorrecta

```
1 SELECT DISTINCT nombre_cliente, nombre, apellido1
2 FROM cliente INNER JOIN empleado
3 INNER JOIN pago
4 ON cliente.codigo_cliente = pago.codigo_cliente;
```

Consulta correcta

```
1 SELECT DISTINCT nombre_cliente, nombre, apellido1
2 FROM cliente INNER JOIN empleado
3 ON cliente.codigo_empleado_rep_ventas = empleado.codigo_empleado
4 INNER JOIN pago
5 ON cliente.codigo_cliente = pago.codigo_cliente;
```

3 Referencias

- [Wikibook SQL Exercises](#).
- [Tutorial SQL de w3resource](#).
- [MySQL Join Types by Steve Stedman](#).
- [Guía visual de SQL Joins](#).
- **Bases de Datos**. 2ª Edición. Grupo editorial Garceta. Iván López Montalbán, Manuel de Castro Vázquez y John Ospino Rivas.
- **INNER JOIN**.
- **LEFT JOIN**.
- **RIGHT JOIN**.

4 Licencia

Esta página forma parte del curso Bases de Datos de José Juan Sánchez Hernández y su contenido se distribuye bajo una licencia Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.