**Consultas sobre varias tablas. Composición interna y cruzada**

Apuntes de BD

Curso 2023/2024

**Índice**

1. [Consultas sobre varias tablas. Composición interna y cruzada](#_bookmark0) 1
   1. [Consultas multitabla SQL 1](#_bookmark1) 1
      1. [Composiciones cruzadas (Producto cartesiano)](#_bookmark2) 1
      2. [Composiciones internas (Intersección)](#_bookmark3) 3
   2. [Consultas multitabla SQL 2](#_bookmark4) 7
      1. [Composiciones cruzadas](#_bookmark5) 7
      2. [Composiciones internas](#_bookmark6) 7
      3. [Composiciones externas](#_bookmark7) 8
   3. [El orden en las tablas no afecta al resultado final](#_bookmark8) 12
   4. [Podemos usar alias en las tablas](#_bookmark9) 13
   5. [Unir tres o más tablas](#_bookmark10) 14
   6. [Unir una tabla consigo misma (self‑equi‑join)](#_bookmark11) 14
   7. [Unir tablas de diferentes bases de datos](#_bookmark12) 15
   8. [Uniones equivalentes (equi‑joins) y Uniones no equivalentes (non‑equi‑joins)](#_bookmark13) 15
2. [Errores comunes](#_bookmark14) 16
3. [Referencias](#_bookmark15) 18

**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.

## Consultas multitabla SQL 1

### Composiciones cruzadas (Producto cartesiano)

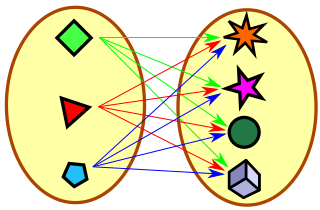
El **producto cartesiano** de dos conjuntos, es una operación que consiste en obtener otro conjunto cuyos elementos son **todas las parejas que pueden formarse entre los dos conjuntos**. Por ejemplo, tendríamos que coger el primer elemento del primer conjunto y formar una pareja con cada uno de los elementos 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](https://es.wikipedia.org/wiki/Producto_cartesiano). Autor: GermanX

#### Ejemplo

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

1

2

3

4

**SELECT** \*

**FROM** empleado;

+--

------+-----------+--------------+-----------+-----------+ +

5

6

| codigo | nif

| nombre

| apellido1 | apellido2 |

codigo\_departamento |

+--

------+-----------+--------------+-----------+-----------+ +

------+-----------+--------------+-----------+-----------+ +

11

12

13

14

15

16

17

**SELECT** \*

**FROM** departamento;

+--------+------------------+ +

| codigo | nombre

| presupuesto |

+--------+------------------+ +

21 +--------+------------------+ +

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | | 1 | | | 32481596F | | Aarón | | | Rivero | | | Gómez | | |
| 8 | | 2 | | | Y5575632D | 1 |  | Adela | | | Salas | | | Díaz | | |
| 9 | | 3 | | | R6970642B | 2 |  | Adolfo | | | Rubio | | | Flores | | |
| 10 +-- |  |  |  | 3 | |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 18 | | 1 | | | Desarrollo | | | 120000 | | |
| 19 | | 2 | | | Sistemas | | | 150000 | | |
| 20 | | 3 | | | Recursos Humanos | | | 280000 | | |

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

1. **SELECT** \*
2. **FROM** empleado, departamento;

El resultado sería el siguiente:

* + - 1. +--------+-----------+--------+-----------+-----------+---------------------+----------------
      2. | codigo | nif | nombre | apellido1 | apellido2 | codigo\_departamento | codigo | nombre | presupuesto | gastos |
      3. +--------+-----------+--------+-----------+-----------+---------------------+----------------

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

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

13 +--------+-----------+--------+-----------+-----------+---------------------+----------------

### 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.

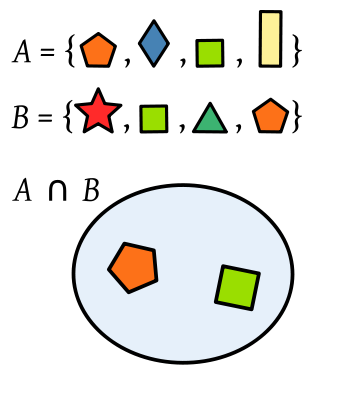


Imagen: Imagen extraída de [Wikipedia](https://es.wikipedia.org/wiki/Intersecci%C3%B3n_de_conjuntos). Autor: Kismalac.

#### 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

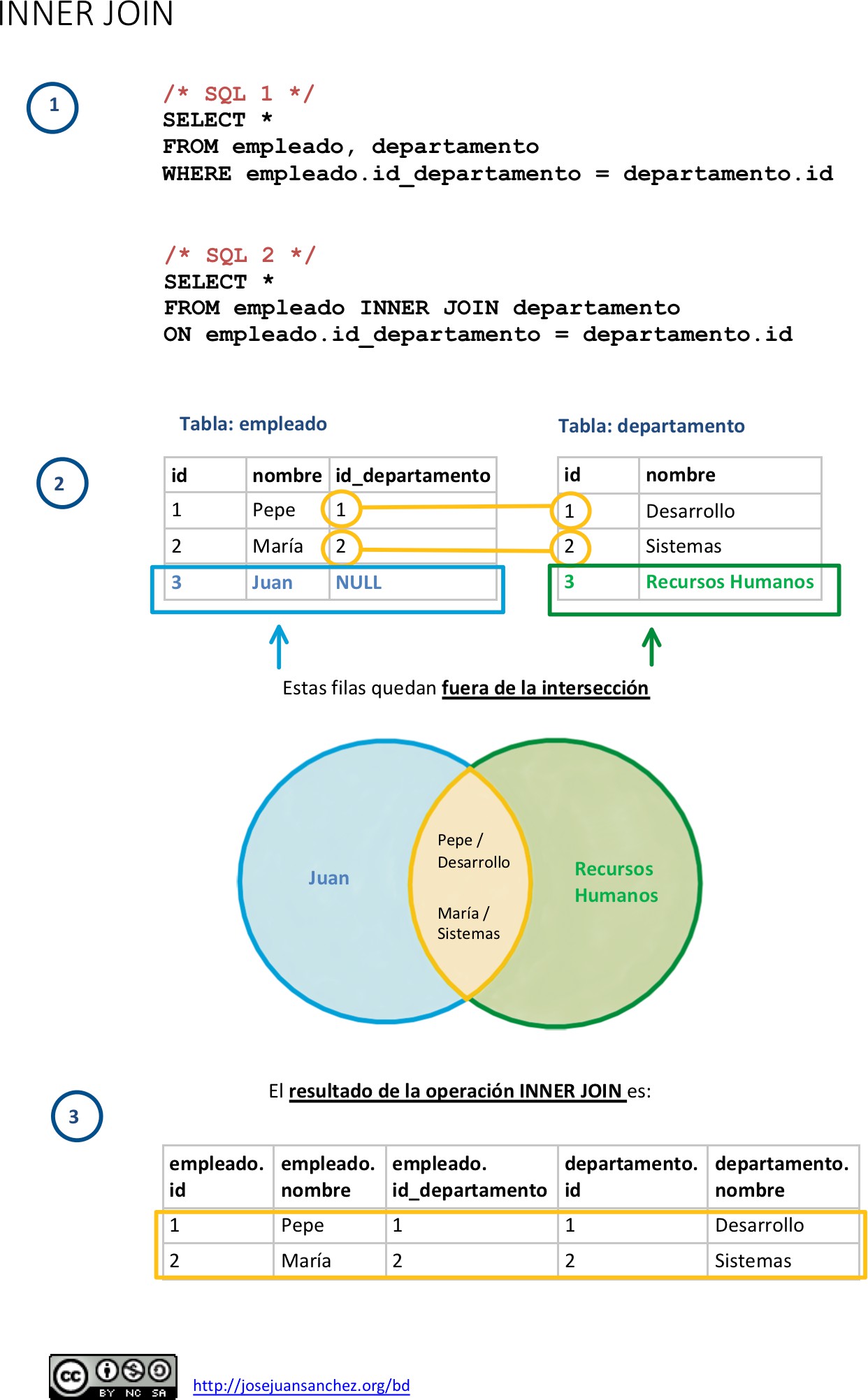
El resultado sería el siguiente:

* + - 1. +--------+-----------+--------+-----------+-----------+---------------------+--------+-------
      2. | codigo | nif | nombre | apellido1 | apellido2 | codigo\_departamento | codigo | nombre | presupuesto | gastos |
      3. +--------+-----------+--------+-----------+-----------+---------------------+--------+-------

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | | 1 | | 32481596F | Aarón | Rivero | | Gómez |  | | | 1 | | | 1 |
| 5 | | 2 | | Desarrollo | 120000  | Y5575632D | Adela | Salas | | 6000  | Díaz | | | | | 2 | | | 2 |
| 6 | | 3 | | Sistemas | 150000  | R6970642B | Adolfo | Rubio | | 21000  | 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.



## Consultas multitabla SQL 2

### 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.

### 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 suelan tener el mismo nombre y que debamos usar una composición de tipo INNER JOIN.

### Composiciones externas

* Join externa

##### LEFT OUTER JOIN

##### RIGHT OUTER JOIN

* + FULL OUTER JOIN (No implementada en MySQL)

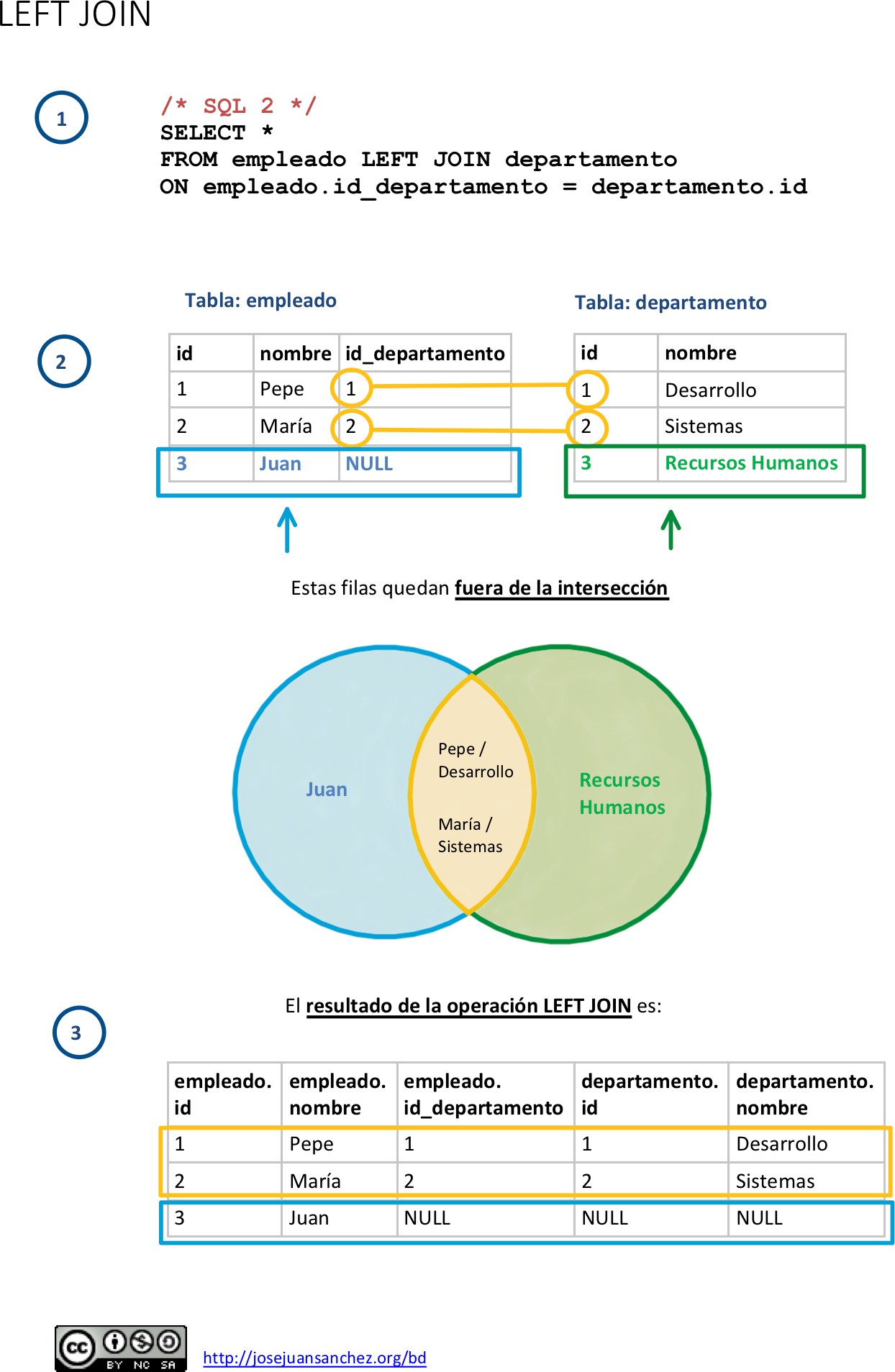
##### NATURAL LEFT OUTER JOIN

##### NATURAL RIGHT OUTER JOIN

#### Ejemplo de LEFT OUTER JOIN:

1. **SELECT** \*
2. **FROM** empleado **LEFT JOIN** departamento
3. **ON** empleado.codigo\_departamento = departamento.codigo

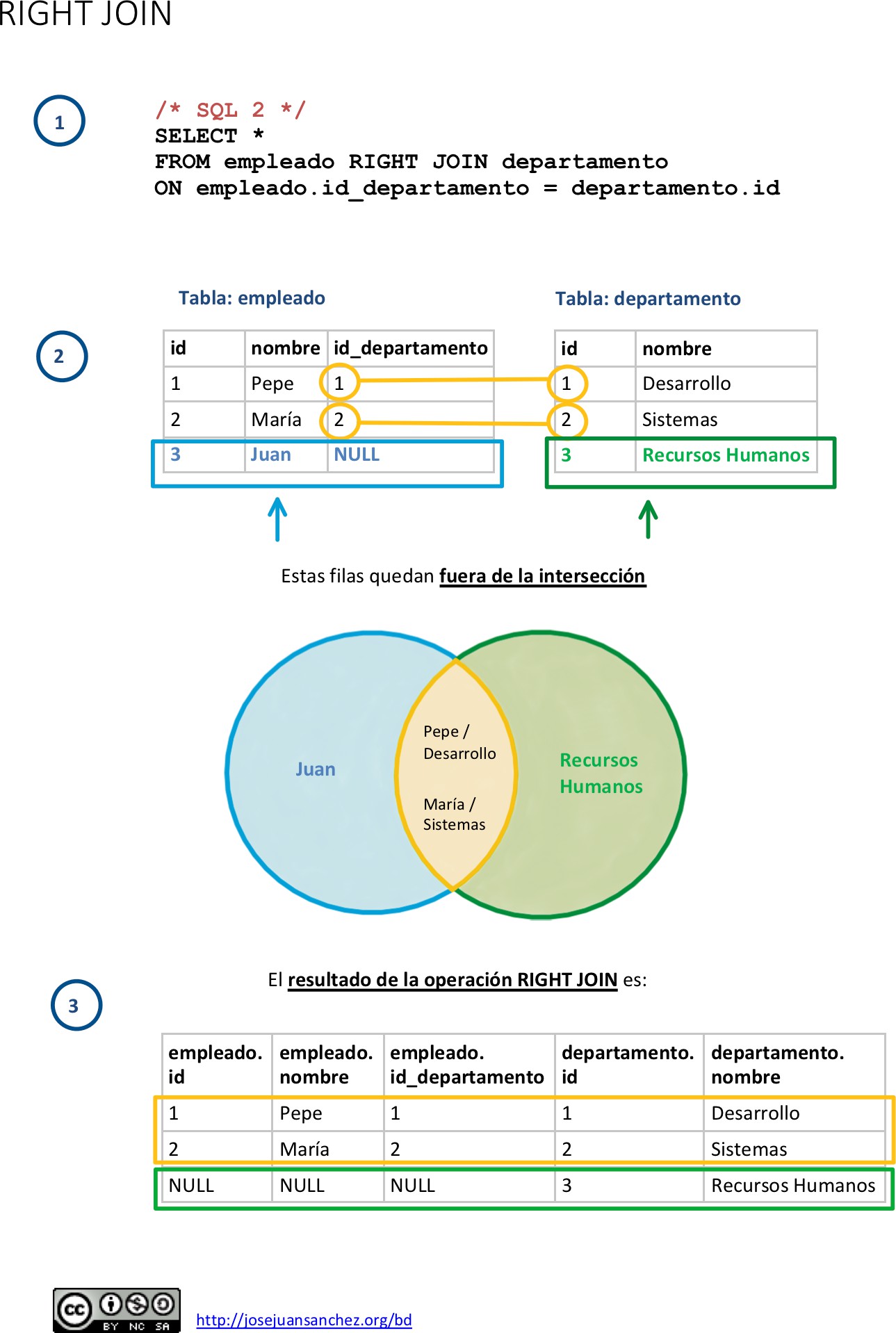
Esta consulta devolverá todas las filas de la tabla que hemos colocado a la izquierda de la composición, en este caso la tabla empleado. Y relacionará las filas de la tabla de la izquierda (empleado) con las filas de la tabla de la derecha (departamento) con las que encuentre una coincidencia. Si no 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.



#### 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 coin‑ cidencia, 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.



#### 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 consul‑ tas.

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 unión de una consulta de tipo LEFT JOIN y una consulta de tipo RIGHT JOIN sobre las mismas tablas.

1

2

3

4

5

6

7

8

9

**SELECT** \*

**FROM** empleado **LEFT JOIN** departamento

**ON** empleado.codigo\_departamento = departamento.codigo

**UNION**

**SELECT**

**FROM** empleado **RIGHT JOIN** departamento

**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.

## 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 con‑ sulta, 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
4. **SELECT** empleado.apellido1, empleado.nombre, departamento.nombre
5. **FROM** departamento **INNER JOIN** empleado
6. **ON** empleado.codigo\_departamento = departamento.codigo

El resultado de ambas consultas sería el mismo:

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

2

3

4

| empleado.apellido1 | empleado.nombre | departamento.nombre |

+--------------------+-----------------+ +

| ...

| ...

| ...

|

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 \*/

1. **SELECT** \*
2. **FROM** empleado **INNER JOIN** departamento
3. **ON** empleado.codigo\_departamento = departamento.codigo

1 /\* 2 \*/

1. **SELECT** \*
2. **FROM** departamento **INNER JOIN** empleado
3. **ON** empleado.codigo\_departamento = departamento.codigo

El resultado de las dos consultas anteriores sería diferente:

1 /\* 1 \*/

1. +--------+-----------+--------+-----------+-----------+---------------------+--------+-------
2. | codigo | nif | nombre | apellido1 | apellido2 | codigo\_departamento | codigo | nombre | presupuesto | gastos |
3. +--------+-----------+--------+-----------+-----------+---------------------+--------+-------

1 /\* 2 \*/

1. +--------+------------------+-------------+--------+--------+-----------+--------+-----------
2. | codigo | nombre | presupuesto | gastos | codigo | nif | nombre | apellido1 | apellido2 | codigo\_departamento |
3. +--------+------------------+-------------+--------+--------+-----------+--------+-----------

## 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 directa‑ mente 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

## 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. **Unir una tabla consigo misma (*self‑equi‑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 tabla haciendo uso de alias.

**Ejemplo:** Cómo unir una tabla consigo misma.

1. **SELECT** empleado.nombre, empleado.apellido1, empleado.apellido2, jefe.nombre, jefe.apellido1, jefe.apellido2
2. **FROM** empleado **INNER JOIN** empleado **AS** jefe
3. **ON** empleado.codigo\_jefe = jefe.codigo\_empleado
4. **SELECT** empleado.nombre, empleado.apellido1, empleado.apellido2, jefe.nombre, jefe.apellido1, jefe.apellido2
5. **FROM** empleado **INNER JOIN** empleado jefe
6. **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
6. **SELECT** empleado.\*, jefe.\*, super\_jefe.\*
7. **FROM** empleado **INNER JOIN** empleado jefe
8. **ON** empleado.codigo\_jefe = jefe.codigo\_empleado
9. **INNER JOIN** empleado super\_jefe
10. **ON** jefe.codigo\_jefe = super\_jefe.codigo\_empleado

## 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. **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
2. **SELECT** \*
3. **FROM** producto, fabricante
4. **WHERE** fabricante.nombre = 'Lenovo';

Consulta correcta

1. **SELECT** \*
2. **FROM** producto, fabricante
3. **WHERE** producto.codigo\_fabricante = fabricante.codigo **AND** fabricante.nombre = ' Lenovo';
4. Nos olvidamos de incluir ON en las consultas de tipo INNER JOIN. Consulta incorrecta
5. **SELECT** \*
6. **FROM** producto **INNER JOIN** fabricante
7. **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';
5. Relacionamos las tablas utilizando nombres de columnas incorrectos. Consulta incorrecta
6. **SELECT** \*
7. **FROM** producto **INNER JOIN** fabricante
8. **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.

Unidad Didáctica 6. Consultas sobre varias tablas. Composición interna y cruzada Curso 2023/2024

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;

* + [Wikibook SQL Exercises](https://en.wikibooks.org/wiki/SQL_Exercises).
  + [Tutorial SQL de w3resource](https://www.w3resource.com/sql/tutorials.php).
  + [MySQL Join Types by Steve Stedman](http://stevestedman.com/wp-content/uploads/VennDiagram2.pdf).
  + 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