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

JOSÉ JUAN SÁNCHEZ HERNÁNDEZ

Índice general

1 Consultas sobre varias tablas. Composición interna y cruzada 4

1.1 Consultas multitabla `SQL 1` 4

1.1.1 Composiciones cruzadas (Producto cartesiano) 4

1.1.2 Composiciones internas (Intersección) 6

1.2 Consultas multitabla `SQL 2` 9

1.2.1 Composiciones cruzadas 9

1.2.2 Composiciones internas 10

1.2.3 Composiciones externas 11

1.3 El orden en las tablas no afecta al resultado final 15

1.4 Podemos usar alias en las tablas 16

1.5 Unir tres o más tablas 16

1.6 Unir una tabla consigo misma (*self-equi-join*) 16

1.7

2 Errores comunes 18

3 Referencias 20

4 Licencia 21

Capítulo 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.

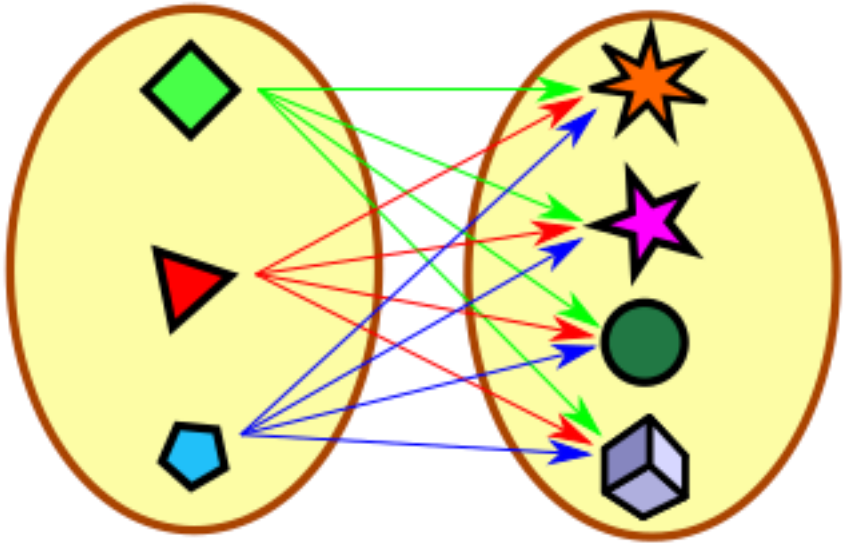


Imagen: Imagen extraída de Wikipedia. Autor: GermanX

Ejemplo

Tomemos de ejemplo la base de datos EMPLEADOS con dos tablas: `empleado` y `departamento`.

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

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

El resultado sería el siguiente:

```
| codigo | nif       | nombre | apellido1 | apellido2 | codigo_departamento |
    codigo | nombre           | presupuesto | gastos |

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

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.

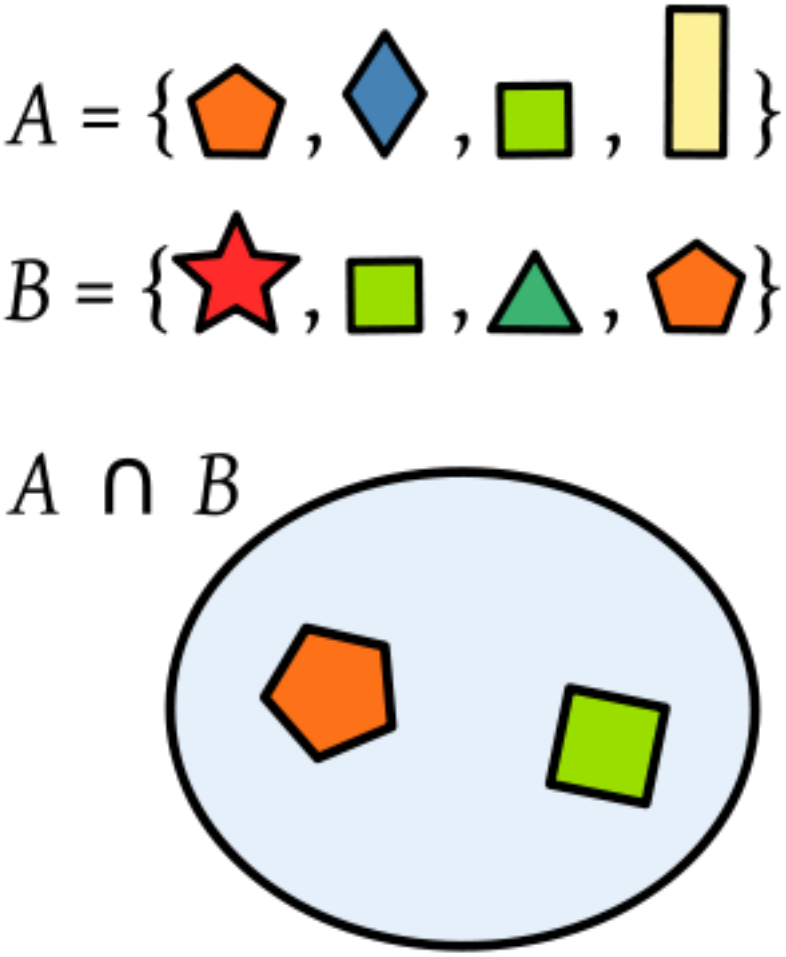


Imagen: Imagen extraída de Wikipedia. Autor: Kismalac.

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:

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

El resultado sería el siguiente:

```
| codigo | nif       | nombre | apellido1 | apellido2 | codigo_departamento |
   codigo | nombre           | presupuesto | gastos |

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

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

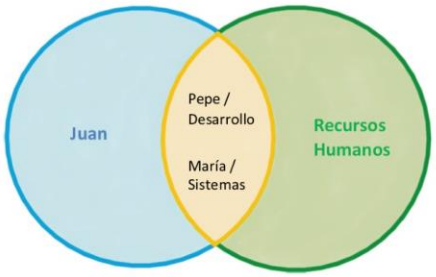
Tabla: empleado

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 INNER JOIN** es:

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

En nuestra BD

SELECT *

FROM empleado INNER JOIN departamento

ON empleado.codigo_departamento = departamento.codigo;

1.2 Consultas multitabla SQL 2

1.2.1 Composiciones cruzadas

- Producto cartesiano – `CROSS JOIN`

Ejemplo de `CROSS JOIN`:

```
SELECT *
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`

La operación JOIN o combinación permite mostrar columnas de varias tablas como si se tratase de una sola tabla, combinando entre sí los registros relacionados usando para ello claves externas. Las tablas relacionadas se especifican en la cláusula FROM, y además hay que hacer coincidir los valores que relacionan las columnas de las tablas.

Las combinaciones internas se realizan mediante la instrucción INNER JOIN. Devuelven únicamente aquellos registros/filas que tienen valores idénticos en los dos campos que se comparan para unir ambas tablas. Es decir, aquellas que tienen elementos en las dos tablas, identificados éstos por el campo de relación.

Ejemplo de `INNER JOIN`:

```
SELECT *
FROM empleado INNER JOIN departamento
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:

```
SELECT *
FROM empleado JOIN departamento
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`.

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

Ejemplo de `NATURAL JOIN`:

```
SELECT *
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 `código`. 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`.

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`:

```
SELECT *
FROM empleado LEFT JOIN departamento
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

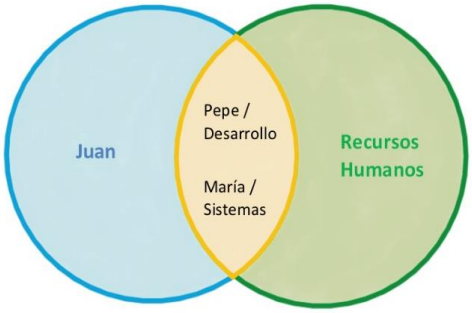
Tabla: empleado

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

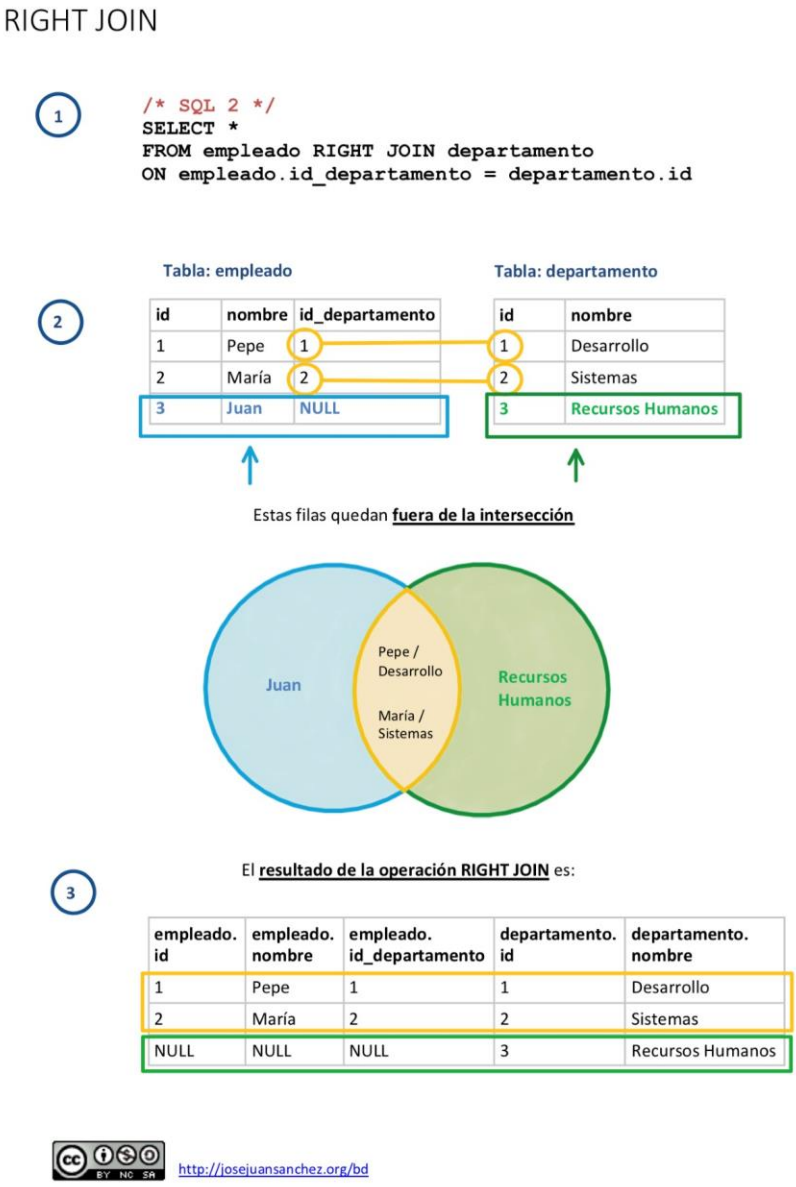
El **resultado de la operación LEFT JOIN** es:

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

Ejemplo de `RIGHT OUTER JOIN`:

```
SELECT *
FROM empleado RIGHT JOIN departamento
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. Sino 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`.



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 unión 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 a 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.

```
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
```

Para poder utilizar el operador UNION entre dos o más consultas deberá tener en cuenta que:
• Deben tener el mismo número de columnas.
• Las columnas que se van a unir tienen que tener tipos de datos similares.
Para ordenar los resultados tras aplicar una operación de UNION existen dos soluciones:
• Usar la posición de la columna sobre la que queremos ordenar los resultados en el ORDER BY.
• Crear un alias en las columnas del primer SELECT sobre la que queremos ordenar los resultados y usarlo en el ORDER BY.
Ejemplo:

```
/* Solución 1 */

SELECT departamento.nombre, empleado.apellido1, empleado.apellido2, empleado. nombre
FROM empleado LEFT JOIN departamento
ON empleado.codigo_departamento=departamento.codigo

UNION

SELECT departamento.nombre, empleado.apellido1, empleado.apellido2, empleado.nombre
FROM empleado RIGHT JOIN departamento
ON empleado.codigo_departamento=departamento.codigo
ORDER BY 1, 2, 3, 4;
```

```
/* Solución 2 */
```

```
SELECT departamento.nombre AS nombre_departamento, empleado.apellido1, empleado.apellido2,
empleado.nombre
FROM empleado LEFT JOIN departamento
ON empleado.codigo_departamento=departamento.codigo

UNION

SELECT departamento.nombre, empleado.apellido1, empleado.apellido2, empleado.nombre
FROM empleado RIGHT JOIN departamento
ON empleado.codigo_departamento=departamento.codigo
ORDER BY nombre_departamento;
```

Ejemplo de `NATURAL LEFT JOIN`:

```
SELECT *
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. Ídem para `NATURAL RIGHT JOIN`

1.3 El orden en las tablas no afecta al resultado final

Estas dos consultas devuelven el mismo resultado:

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

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

1.4 Podemos usar alias en las tablas

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

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

1.5 Unir tres o más tablas

Ejemplo:

```
SELECT *
FROM cliente INNER JOIN empleado
ON cliente.codigo_empleado_rep_ventas = empleado.codigo_empleado
INNER JOIN pago
ON cliente.codigo_cliente = pago.codigo_cliente;
```

1.6 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:

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

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

Capítulo 2

Errores comunes

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

Consulta incorrecta

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

Consulta correcta

```
SELECT *
FROM producto, fabricante
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

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

Consulta correcta

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

3. Relacionamos las tablas utilizando nombres de columnas incorrectos.

Consulta incorrecta

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

Consulta correcta

```
SELECT *
FROM producto INNER JOIN fabricante
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

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

Consulta correcta

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

5. Cuando estamos usando `LEFT JOIN` o `RIGHT JOIN` no deberíamos tener varias condiciones en la cláusula `ON`.

Consulta incorrecta

```
SELECT *
FROM fabricante LEFT JOIN producto
ON fabricante.codigo = producto.codigo_fabricante AND
   producto.codigo_fabricante IS NULL;
```

Consulta correcta.

```
SELECT *
FROM fabricante LEFT JOIN producto
ON fabricante.codigo = producto.codigo_fabricante
WHERE producto.codigo_fabricante IS NULL;
```

Capítulo 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.

Capítulo 4

Licencia