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.

Ejemplo

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

```
SELECT *
FROM empleado;

+--
    ------+-----------+--------------+-----------+-----------+---------------------+
| codigo | nif       | nombre       | apellido1 | apellido2 | codigo_departamento
    |
+--
    ------+-----------+--------------+-----------+-----------+---------------------+
|      1 | 32481596F | Aarón        | Rivero    | Gómez     |                   1
    |
|      2 | Y5575632D | Adela        | Salas     | Díaz      |                   2
    |
|      3 | R6970642B | Adolfo       | Rubio     | Flores    |                   3
    |
+--
    ------+-----------+--------------+-----------+-----------+---------------------+
```

```
SELECT *
FROM departamento;

+--------+------------------+-------------+
| codigo | nombre           | presupuesto |
```

```
+--------+------------------+-------------+
|      1 | Desarrollo       |      120000 |
|      2 | Sistemas         |      150000 |
|      3 | Recursos Humanos |      280000 |
+--------+------------------+-------------+
```

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.

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

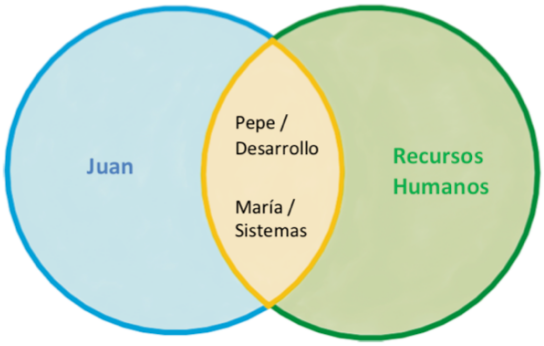
Tabla: empleado

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

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 JOINo JOIN
  - NATURAL JOIN

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 INNERes 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 ONobtendremos el producto cartesiano de lasdos 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 mismonombre 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 quequiero relacionar las dos tablas se llaman igual en las dos tablas. Lo normal es que no suelan tener el mismonombre y que debamos usar una composición de tipo INNER JOIN.

1.3 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
```

1.4 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;
```

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;
```