UT6. CONSULTAS DE RECUPERACIÓN MULTITABLA EN SQL

Módulo: BASES DE DATOS

Curso 2022/2023. 1° DAM

Ruth Lospitao Ruiz

CONTENIDOS JOINS

- Concepto
- Cross Join
- Inner Join
- Valores Null
- Outer Join
- Left Join
- Right Join
- Full Join

CONCEPTO

Un JOIN es una composición entre dos tablas. Es decir, devuelve una nueva tabla compuesta por las dos tablas indicadas

- Es similar al producto cartesiano, con la diferencia de que la condición para componer la nueva tabla no aparece en el WHERE, sino en el propio JOIN.
- Existen varios tipos de JOIN:
  - Composición interna: INNER JOIN (y NATURAL JOIN que no se explicará y se desaconseja su uso).
  - Composición externa:
    - De la tabla izquierda: LEFT OUTER JOIN
    - De la tabla derecha: RIGHT OUTER JOIN
    - Completa: FULL OUTER JOIN
- Producto cartesiano: CROSS JOIN.

CROSS JOIN

Es igual al producto cartesiano que ya conocemos. En vez de separar la tablas por comas, se separan por las palabras CROSS JOIN.

SELECT *

FROM libros

CROSS JOIN editoriales

CROSS JOIN autores;

Titulo	Editorial	Autor	Codigo_e	Nombre	Codigo_a	Nombre	Apellido
Int. Artificial	23	25	23	Rama	25	Pedro	Garcia
Int. Artificial	23	25	56	Paraninfo	25	Pedro	Garcia
Int. Artificial	23	25	48	Mc Graw-Hill	25	Pedro	Garcia
Concep. Y Dis.	23	15	23	Rama	25	Pedro	Garcia
Concep. Y Dis.	23	15	56	Paraninfo	25	Pedro	Garcia
Concep. Y Dis.	23	15	48	Mc Graw-Hill	25	Pedro	Garcia
Turbo C++	23	25	23	Rama	25	Pedro	Garcia
Turbo C++	23	25	56	Paraninfo	25	Pedro	Garcia
Turbo C++	23	25	48	Mc Graw-Hill	25	Pedro	Garcia
Sist. Informac.	56	15	23	Rama	25	Pedro	Garcia
Sist. Informac.	56	15	56	Paraninfo	25	Pedro	Garcia
Sist. Informac.	56	15	48	Mc Graw-Hill	25	Pedro	Garcia
Optimización	48	25	23	Rama	25	Pedro	Garcia
Optimización	48	25	56	Paraninfo	25	Pedro	Garcia
Optimización	48	25	48	Mc Graw-Hill	25	Pedro	Garcia
Int. Artificial	23	25	23	Rama	15	Maria	López
Int. Artificial	23	25	56	Paraninfo	15	Maria	López
Int. Artificial	23	25	48	Mc Graw-Hill	15	Maria	López
Concep. Y Dis.	23	15	23	Rama	15	Maria	López
Concep. Y Dis.	23	15	56	Paraninfo	15	Maria	López
Concep. Y Dis.	23	15	48	Mc Graw-Hill	15	Maria	López
Turbo C++	23	25	23	Rama	15	Maria	López
Turbo C++	23	25	56	Paraninfo	15	Maria	López
Turbo C++	23	25	48	Mc Graw-Hill	15	Maria	López
Sist. Informac.	56	15	23	Rama	15	Maria	López
Sist. Informac.	56	15	56	Paraninfo	15	Maria	López
Sist. Informac.	56	15	48	Mc Graw-Hill	15	Maria	López
Optimización	48	25	23	Rama	15	Maria	López
Optimización	48	25	56	Paraninfo	15	Maria	López
Optimización	48	25	48	Mc Graw-Hill	15	Maria	López

INNER JOIN

El INNER JOIN es una forma de realizar una composición entre 2 tablas para obtener otra.

Al igual que en el producto cartesiano, combina todas las filas de la tabla indicada en primer lugar, con todas las filas de la tabla indicada en segundo lugar. Luego, para cada una de las filas resultados, aplica la condición indicada en el INNER JOIN, después de la palabra ON.

tabla* INNER JOIN *tabla2* ON *condición

La condición puede ser de cualquier tipo (siempre que al aplicarse sobre las filas combinadas devuelva un valor booleano (cierto o falso). Por eso se pueden utilizar las columnas tanto de tabla, como de tabla2. Normalmente se utiliza para comparar una columna de tabla, con una columna de tabla2, que suelen ser la clave primaria, y la clave foránea.

INNER JOIN

SELECT * FROM libros lib INNER JOIN editoriales e ON lib.editorial = e.nombre_e;

Codigo	Titulo	Num_paginas	Editorial	Nombre_e	Direccion	Pais	Ciudad
34590	Int. Artificial	50	Paraninfo	Paraninfo	Virtudes, 7	España	Madrid
1022306	Concep. Y Dis.	48	Rama	Rama	Canillas, 144	España	Madrid
493944	Turbo C++	125	Mc Graw-Hill	Mc Graw-Hill	Basauri, 17	España	Madrid
112314	Sist. Informac.	358	Rama	Rama	Canillas, 144	España	Madrid

En este caso, se asocia cada libro con la editorial a la que pertenece. Es decir, para cada libro se muestran los datos del libro y de la editorial.

Observad como el libro de editorial nula no aparece en los resultados.

Una consulta que daba el mismo resultado ya se había hecho, y era:

SELECT * FROM libros lib, editoriales e WHERE lib.editorial = e.nombre_e;

INNER JOIN

Observad como se puede escribir la consulta anterior de distintas formas, y no cambia el resultado:

SELECT * FROM libros lib INNER JOIN editoriales e ON lib.editorial = e.nombre_e;

SELECT * FROM editoriales e INNER JOIN libros lib ON lib.editorial = e.nombre_e;

SELECT * FROM editoriales e INNER JOIN libros lib ON e.nombre_e = lib.editorial;

Esto se debe a que en las bases de datos, el orden de las columnas no es significativo, pero tened presente que las herramientas que las utilizan si pueden buscar datos por el orden de las columnas, así que la primera instrucción dará la tabla primera de la transparencia siguiente, mientras que las otras dos instrucciones darán como resultado la tabla segunda. Pero ambas tablas son equivalentes.

INNER JOIN

Codigo	Titulo	Num_paginas	Editorial	Nombre_e	Direccion	Pais	Ciudad
34590	Int. Artificial	50	Paraninfo	Paraninfo	Virtudes, 7	España	Madrid
1022306	Concep. Y Dis.	48	Rama	Rama	Canillas, 144	España	Madrid
493944	Turbo C++	125	Mc Graw-Hill	Mc Graw-Hill	Basauri, 17	España	Madrid
112314	Sist. Informac.	358	Rama	Rama	Canillas, 144	España	Madrid

Nombre_e	Direccion	Pais	Ciudad	Codigo	Titulo	Num_paginas	Editorial
Paraninfo	Virtudes, 7	España	Madrid	34590	Int. Artificial	50	Paraninfo
Rama	Canillas, 144	España	Madrid	1022306	Concep. Y Dis.	48	Rama
Mc Graw-Hill	Basauri, 17	España	Madrid	493944	Turbo C++	125	Mc Graw-Hill
Rama	Canillas, 144	España	Madrid	112314	Sist. Informac.	358	Rama

Mirad como la fila de nombre_e = 'Rama' de la tabla editorial aparece repetida 2 veces. Esto es porque hay 2 libros de esa editorial. Entonces, al mostrar para cada editorial sus libros, o para cada libro su editorial, esta editorial aparece 2 veces.

INNER JOIN

```
SELECT *
FROM libros
INNER JOIN editoriales ON libros.editorial = editoriales.codigo_e
INNER JOIN autores ON libros.autor = autor.codigo_a;
```

Titulo	Editorial	Autor	Codigo_e	Nombre	Codigo_a	Nombre	Apellido
Int. Artificial	23	25	23	Rama	25	Pedro	Garcia
Turbo C++	23	25	23	Rama	25	Pedro	Garcia
Optimización	48	25	48	Mc Graw-Hill	25	Pedro	Garcia
Concep. Y Dis.	23	15	23	Rama	15	Maria	López
Sist. Informac.	56	15	56	Paraninfo	15	Maria	López

Se presentan 2 INNER JOIN. En primer lugar, combina cada libro con la editorial del libro. En segundo lugar, combina el resultado previo con el autor del libro.

INNER JOIN

Se observa que la sentencia anterior con INNER JOIN también se puede obtener con un producto cartesiano + filtro en el WHERE. Es decir:

SELECT * FROM tabla INNER JOIN tabla2 ON condición;

es equivalente a:

SELECT * FROM tabla, tabla2 WHERE condición;

Su uso más habitual es para relacionar tablas que tienen una foreign key. Si esta foreign key fuera compuesta, por ejemplo, por campo1 y campo2, la sentencia quedaría:

SELECT * FROM tabla t INNER JOIN tabla2 t2 ON t.campo1=t2.campo1 AND t.campo2=t2.campo2;

INNER JOIN

Aunque su uso más habitual es para unir una tabla con clave foránea con la tabla que tiene la clave primaria, se pueden utilizar para más cosas. Incluso, la condición, no tiene que ser una comparación de igualdad. Por ejemplo:

```
SELECT c.nombre AS nombreComprador, c.num_compras, v.nombre AS nombreVendedor, v.num_ventas FROM compradores c INNER JOIN vendedores v ON c.num_compras > v.num_ventas;
```

Esta sentencia devuelve cada comprador con los vendedores que han vendido menos que él ha comprado.

INNER JOIN

```
SELECT c.nombre AS nombreComprador, c.num_compras, v.nombre AS nombreVendedor, v.num_ventas
FROM compradores c INNER JOIN vendedores v ON c.num_compras > v.num_ventas;
```

Compradores

Codigo	Nombre	Num_compras
34587	Pedro	50
1022305	Maria	48
493942	Juana	125
45307	Lola	50
21566	Carlos	30
12548	Miguel	50
568978	Antonio	70

Vendedores

Nombre	Num_ventas
Luis	65
Juana	15
Maria	23
David	45
Amparo	48
Carlos	56
Teresa	48

nombreComprador	Num_compras	nombreVendedor	Num_ventas
Pedro	50	Juana	15
Pedro	50	Maria	23
Pedro	50	David	45
Pedro	50	Amparo	48
Pedro	50	Teresa	48
Maria	48	Juana	15
Maria	48	Maria	23
Maria	48	David	45
Juana	125	Luis	65
Juana	125	Juana	15
Juana	125	Maria	23
Juana	125	David	45
Juana	125	Amparo	48
Juana	125	Carlos	56
Juana	125	Teresa	48
Lola	50	Juana	15
Lola	50	Maria	23
Lola	50	David	45
Lola	50	Amparo	48
Lola	50	Teresa	48
Carlos	30	Juana	15
Carlos	30	Maria	23
Miguel	50	Juana	15
Miguel	50	Maria	23
Miguel	50	David	45
Miguel	50	Amparo	48
Miguel	50	Teresa	48
Antonio	70	Luis	65
Antonio	70	Juana	15
Antonio	70	Maria	23
Antonio	70	David	45
Antonio	70	Amparo	48
Antonio	70	Carlos	56
Antonio	70	Teresa	48

JOINS: RELACIONES REFLEXIVAS

En cualquier JOIN (así como en los productos cartesianos), se puede utilizar dos veces la misma tabla. Es decir, que tabla y tabla2 sean la misma tabla. En este caso, siempre es necesario el uso de alias para ambas ocurrencias de la tabla.

SELECT emp.id AS idEmpleado, emp.nombre AS nombreEmpleado, jefe.id AS idJefe, jefe.nombre AS nombreJefe FROM empleados emp INNER JOIN empleados jefe ON emp.jefe = jefe.id;

EMPLEADOS		
id	nombre	jefe
1	Maria	NULL
2	Lucas	1
3	David	1
4	Laura	NULL
5	Cristina	4

idEmpleado	nombreEmpleado	idJefe	nombreJefe
2	Lucas	1	Maria
3	David	1	Maria
5	Cristina	4	Laura

Devuelve los empleados junto con su jefe. Se observa que no aparecen (como empleados) ni María ni Laura.

VALORES NULL EN LOS JOINS

En la comparación que aparece en un JOIN (de cualquier tipo) dos valores nulos se consideran distintos.

El siguiente ejemplo obtiene los empleados que tienen el mismo jefe

SELECT emp.id AS idEmpleado1, emp.nombre AS nombreEmpleado,1 emp2.id AS idEmpleado2, emp2.nombre AS nombreEmpleado2 FROM empleados emp INNER JOIN empleados empl2 ON emp.jefe = emp2.jefe;

EMPLEADOS		
id	nombre	jefe
1	Maria	NULL
2	Lucas	1
3	David	1
4	Laura	NULL
5	Cristina	4

idEmpleado1	nombreEmpleado1	idEmpleado2	nombreEmpleado2
2	Lucas	2	Lucas
2	Lucas	3	David
3	David	2	Lucas
3	David	3	David
5	Cristina	5	Cristina

VALORES NULL EN LOS JOINS

SELECT emp.id AS idEmpleado1 , emp.nombre AS nombreEmpleado,1 emp2.id AS idEmpleado2, emp2.nombre AS nombreEmpleado2 FROM empleados emp INNER JOIN empleados empl2 ON emp.jefe = emp2.jefe;

EMPLEADOS		
id	nombre	jefe
1	Maria	NULL
2	Lucas	1
3	David	1
4	Laura	NULL
5	Cristina	4

idEmpleado1	nombreEmpleado1	idEmpleado2	nombreEmpleado2
2	Lucas	2	Lucas
2	Lucas	3	David
3	David	2	Lucas
3	David	3	David
5	Cristina	5	Cristina

En este ejemplo, se observa que se compara cada fila de empleado con todas las filas, y por tanto, también se comparan cada fila consigo misma. Sin embargo, no aparece ninguna fila con jefe nulo, ya que NULL <> NULL.

VALORES NULL EN LOS JOINS

Si en la consulta anterior se quieren quitar los empleados que aparecen relacionados consigo mismo, se puede hacer de dos formas: en la condición del JOIN, y en el WHERE.

1. SELECT emp.id AS idEmpleado1, emp.nombre AS nombreEmpleado,1 emp2.id AS idEmpleado2, emp2.nombre AS nombreEmpleado2 FROM empleados emp INNER JOIN empleados empl2 ON emp.jefe = emp2.jefe AND emp.id <> emp2.id;

EMPLEADOS		
id	nombre	jefe
1	Maria	NULL
2	Lucas	1
3	David	1
4	Laura	NULL
5	Cristina	4

idEmpleado1	nombreEmpleado1	idEmpleado2	nombreEmpleado2
2	Lucas	3	David
3	David	2	Lucas

VALORES NULL EN LOS JOINS

2. SELECT emp.id AS idEmpleado1 , emp.nombre AS nombreEmpleado,1 emp2.id AS idEmpleado2, emp2.nombre AS nombreEmpleado2 FROM empleados emp INNER JOIN empleados empl2 ON emp.jefe = emp2.jefe WHERE emp.id <> emp2.id;

EMPLEADOS		
id	nombre	jefe
1	Maria	NULL
2	Lucas	1
3	David	1
4	Laura	NULL
5	Cristina	4

idEmpleado1	nombreEmpleado1	idEmpleado2	nombreEmpleado2
2	Lucas	3	David
3	David	2	Lucas

Aunque en este caso ambas alternativas dan el mismo resultado, no siempre esto es así. Hay que tener en cuenta el orden de ejecución de las partes del SELECT, ya que, después del JOIN, y antes del WHERE se ejecutan el resto de operaciones indicadas en la parte FROM. Veremos otros tipos de JOINS en los que esto tampoco se cumple.

OUTER JOIN

En este tipo de composición, se garantiza que todas las filas de la tabla indicada aparecerán al menos una vez en el resultado, independientemente de la condición del JOIN. La palabra OUTER es opcional, y no suele ponerse.

Existen 3 casos, que se corresponden con cada una de las posibilidades de tabla indicada:

- LEFT OUTER JOIN: Se garantiza que aparecerán todas las filas de la tabla de la izquierda (la que aparece antes de las palabras LEFT JOIN).
- RIGHT OUTER JOIN: Se garantiza que aparecerán todas las filas de la tabla de la derecha (la que aparece después de las palabras RIGHT JOIN).
- FULL OUTER JOIN: Se garantiza que aparecerán todas las filas de la tabla de la izquierda y de la tabla de la derecha.

OUTER JOIN

El funcionamiento de un OUTER JOIN es muy similar al del INNER JOIN:

- Al igual que en el producto cartesiano, combina todas las filas de la tabla indicada en primer lugar (lado izquierdo o antes del OUTER JOIN), con todas las filas de la tabla indicada en segundo lugar (lado derecho o después del OUTER JOIN).
- Luego, para cada una de las filas resultados, aplica la condición indicada en el OUTER JOIN, después de la palabra ON.
- Por último y de forma diferente al INNER JOIN, comprueba que cada fila del lado izquierdo (en un LEFT JOIN), del lado derecho (RIGHT JOIN) o de ambos lados (FULL JOIN) aparece en el conjunto de resultados. Por cada fila que no aparece, añade una fila más a los resultados, que es la unión de esa fila, y valores NULL para el resto de columnas (correspondiente a la tabla con la que no ha encontrado relación).

LEFT JOIN

SELECT * FROM compradores c LEFT JOIN vendedores v ON c.nombre = v.nombre;

Muestra todos los comprobadores, con su número de compras, y si además han hecho ventas, el número de ventas.

Compradores

Codigo	Nombre	Num_compras
34587	Pedro	50
1022305	Maria	48
493942	Juana	125
45307	Lola	50
21566	Carlos	30
12548	Miguel	50
568978	Antonio	70

Vendedores

Codigo	Nombre	Num_ventas
65154	Luis	65
132155	Juana	15
548965	Maria	23
213564	David	45
568655	Amparo	48
215685	Carlos	56
995485	Teresa	48

c.Codigo	c.Nombre	Num_compras	v.Codigo	v.Nombre	Num_ventas
34587	Pedro	50	NULL	NULL	NULL
1022305	Maria	48	548965	Maria	23
493942	Juana	125	132155	Juana	15
45307	Lola	50	NULL	NULL	NULL
21566	Carlos	30	215685	Carlos	56
12548	Miguel	50	NULL	NULL	NULL
568978	Antonio	70	NULL	NULL	NULL

RIGHT JOIN

SELECT * FROM compradores c RIGHT JOIN vendedores v ON c.nombre= v.nombre;

Muestra todos los vendedores, con su número de ventas, y si además han hecho compras, el número de compras.

Compradores

Codigo	Nombre	Num_compras
34587	Pedro	50
1022305	Maria	48
493942	Juana	125
45307	Lola	50
21566	Carlos	30
12548	Miguel	50
568978	Antonio	70

Vendedores

Codigo	Nombre	Num_ventas
65154	Luis	65
132155	Juana	15
548965	Maria	23
213564	David	45
568655	Amparo	48
215685	Carlos	56
995485	Teresa	48

c.Codigo	c.Nombre	Num_compras	v.Codigo	v.Nombre	Num_ventas
NULL	NULL	NULL	65154	Luis	65
493942	Juana	125	132155	Juana	15
1022305	Maria	48	548965	Maria	23
NULL	NULL	NULL	213564	David	45
NULL	NULL	NULL	568655	Amparo	48
21566	Carlos	30	215685	Carlos	56
NULL	NULL	NULL	995485	Teresa	48

FULL JOIN

SELECT * FROM compradores c FULL JOIN vendedores v ON c.nombre = v.nombre;

Muestra todos los comprobadores y vendedores, con su número de compras y su número de ventas.

Compradores

Codigo	Nombre	Num_compras
34587	Pedro	50
1022305	Maria	48
493942	Juana	125
45307	Lola	50
21566	Carlos	30
12548	Miguel	50
568978	Antonio	70

Vendedores

Codigo	Nombre	Num_ventas
65154	Luis	65
132155	Juana	15
548965	Maria	23
213564	David	45
568655	Amparo	48
215685	Carlos	56
995485	Teresa	48

c.Codigo	c.Nombre	Num_compras	v.Codigo	v.Nombre	Num_ventas
34587	Pedro	50	NULL	NULL	NULL
1022305	Maria	48	548965	Maria	23
493942	Juana	125	132155	Juana	15
45307	Lola	50	NULL	NULL	NULL
21566	Carlos	30	215685	Carlos	56
12548	Miguel	50	NULL	NULL	NULL
568978	Antonio	70	NULL	NULL	NULL
NULL	NULL	NULL	65154	Luis	65
NULL	NULL	NULL	213564	David	45
NULL	NULL	NULL	568655	Amparo	48
NULL	NULL	NULL	995485	Teresa	48

FULL JOIN EN MYSQL

MySQL no implementa FULL JOIN (no se puede realizar). Sin embargo, un FULL JOIN es la unión entre un LEFT JOIN y un RIGHT JOIN iguales. Por eso, en MySQL se puede transformar:

```
SELECT * FROM compradores c FULL JOIN vendedores v ON c.nombre = v.nombre;
```

```
SELECT * FROM compradores c LEFT JOIN vendedores v ON c.nombre = v.nombre
UNION
SELECT * FROM compradores c RIGHT JOIN vendedores v ON c.nombre= v.nombre;
```

OUTER JOIN

Los OUTER JOIN sirven también para recuperar las filas que tienen un campo que está en una tabla, pero no en la otra.

Por ejemplo, si se quieren saber las editoriales que no tienen libros, habría que hacer un LEFT JOIN entre editorial y libros, para saber todas las editoriales y los libros que tienen, y también nos mostrará las editoriales que no tienen libros. Si luego eliminamos las editoriales que tienen libros (es decir, las filas en las que el libro exista), tendremos las editoriales que no tienen libros.

SELECT e.* FROM editoriales e LEFT JOIN libros lib ON e.nombre_e = lib.editorial WHERE lib.editorial IS NULL;

Por supuesto, se puede realizar también con un RIGHT JOIN.

OUTER JOIN

SELECT e.* FROM editoriales e LEFT JOIN libros lib ON e.nombre_e = lib.editorial WHERE lib.codigo IS NULL;

EDITORIALES

Nombre_e	Direccion	Pais	Ciudad
Universal Books	Brown Sq. 23	EEUU	Los Ángeles
Rama	Canillas, 144	España	Madrid
Mc Graw-Hill	Basauri, 17	España	Madrid
Paraninfo	Virtudes, 7	España	Madrid

LIBROS

Codigo	Titulo	Num_paginas	Editorial
34587	Int. Artificial	50	Paraninfo
1022305	Concep. Y Dis.	48	Rama
493942	Turbo C++	125	Mc Graw-Hill
45307	Virus Informát.	50	NULL
21566	Optimización	30	Paraninfo
12548	Seguridad	50	Paraninfo
568978	Hardware	70	Paraninfo
112313	Sist. Informac.	358	Rama
21567	Optimización2	30	NULL

Nombre_e	Direccion	Pais	Ciudad	Codigo	Titulo	Num_paginas	Editorial
Universal Books	Brown Sq. 23	EEUU	Los Ángeles	NULL	NULL	NULL	NULL
Rama	Canillas, 144	España	Madrid	112313	Sist. Informac.	358	Rama
Rama	Canillas, 144	España	Madrid	1022305	Concep. Y Dis.	48	Rama
Mc Graw-Hill	Basauri, 17	España	Madrid	493942	Turbo C++	125	Mc Graw-Hill
Paraninfo	Virtudes, 7	España	Madrid	34587	Int. Artificial	50	Paraninfo
Paraninfo	Virtudes, 7	España	Madrid	21566	Optimización	30	Paraninfo
Paraninfo	Virtudes, 7	España	Madrid	12548	Seguridad	50	Paraninfo
Paraninfo	Virtudes, 7	España	Madrid	568978	Hardware	70	Paraninfo

Nombre_e	Direccion	Pais	Ciudad
Universal Books	Brown Sq. 23	EEUU	Los Ángeles

PRODUCTO CARTESIANO Y JOINS

En la parte del FROM, como se ha visto, pueden aparecer productos cartesianos (indicados por una coma), y JOINS. En el caso de que aparezcan ambos, primero se realizan los JOINS, y luego se realizan los productos cartesianos.

SELECT * FROM a INNER JOIN b ON a.campo = b.campo, c;

Primero se realiza el inner join entre a y b, y sobre el resultado, se hace el producto cartesiano con c.

SELECT * FROM a, b INNER JOIN c ON b.campo = c.campo, d;

Primero se realiza el inner join de b y c. Luego, se hace el producto cartesino de a y el resultado de b y c, y por último, sobre ese resultado se hace el producto cartesiano con d.

SELECT * FROM a, b INNER JOIN c ON a.campo = c.campo;

No se puede ejecutar, porque lo primero que se hace es el producto cartesiano entre b y c, y la columna a.campo no existe en este momento.

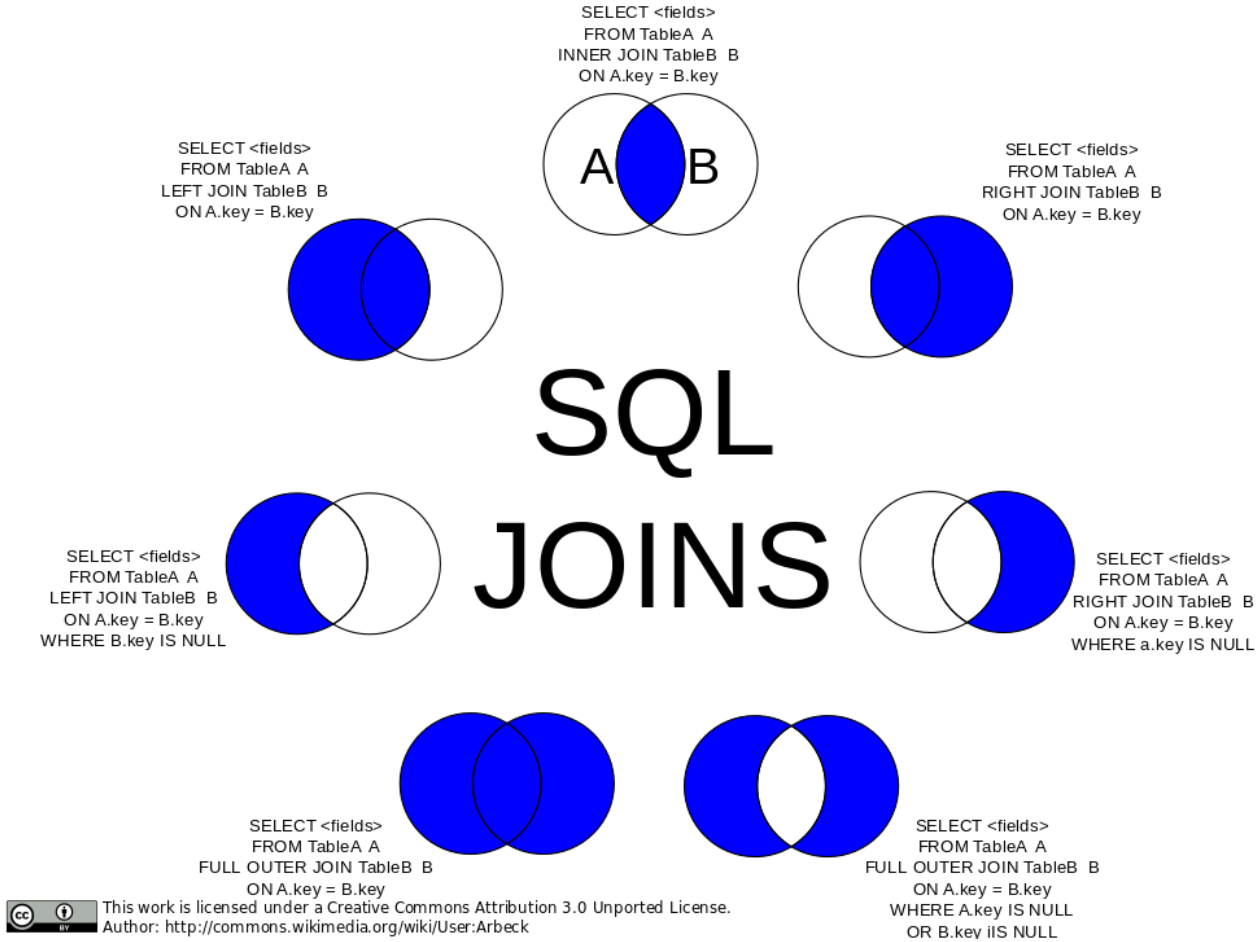
TABLAS DERIVADAS

En el FROM (y en los JOINS) debe aparecer una tabla. Pero el resultado de una SELECT se puede considerar una tabla, al igual que vimos en las subconsultas. Por tanto, en un FROM puede aparecer una subconsulta a la que obligatoriamente le debemos dar un nombre. Esta subconsulta puede contener cualquier cosa de las aquí indicadas, salvo ORDER BY y LIMIT.

```
SELECT MAX(total)
FROM ( SELECT SUM (cantidad * PrecionUnitario) as total, codigoTicket
            FROM lineaTicket
            GROUP BY codigoTicket
) AS totalTicket;
```

Calcula el ticket de mayor importe entre todos los tickets, cuando se tiene los tickets desglosados por líneas.

RESUMEN DE JOINS



OPCIONES EN LA PARTE FROM

table_references*:*

referencia_a_tabla

| ***table_references*** **[, *table_references* …]**

| *table_references* join_condition referencia_a_tabla [[AS] *alias*] **ON *condición***

referencia_a_tabla:

tabla **[[AS] *alias*]**

| *subconsulta* [AS] *alias*

join_condition:

INNER JOIN | CROSS JOIN | LEFT [OUTER] JOIN | RIGHT [OUTER] JOIN

SINTAXIS DE SELECT

```
SELECT [ DISTINCT ] select_expr [, select_expr ...]
FROM table_references
[ WHERE filtro ]
[ GROUP BY expr [, expr … ]
   [ HAVING filtro_grupos ] ]
[ ORDER BY {col_name | expr | position} [ASC | DESC] , ... ]
[ LIMIT [ desplazamiento , ] nfilas ]
```

Se pueden unir varios bloques de este tipo con:
UNION
INTERSECT
EXCEPT

EJECUCIÓN DE UNA SENTENCIA SELECT

1. Si existen subconsultas que no referencien a columnas de la consulta principal, se ejecutan las subconsultas primero siguiendo los pasos aquí descritos.
2. Se toma la tabla indicada en el FROM. Se aplican los JOINS si existen, y los productos cartesianos después (si existen). Si alguna de las tablas es una tabla derivada, se calcula previamente según los pasos aquí descritos.
3. Para cada fila, se aplican las condiciones indicadas en el WHERE. Si en esa fila no se cumple (resultado falso) las condiciones, se elimina la fila del conjunto de resultados. Si hay subconsultas en la parte WHERE que referencian a columnas de la consulta principal, se ejecutan en este momento.
4. Se aplica el GROUP BY, agrupando las filas por las columnas indicadas.
5. Se aplica el HAVING, eliminando las filas agrupadas que no cumplan la condición. Si hay subconsultas en la parte HAVING que referencian a columnas de la consulta principal, se ejecutan en este momento.
6. Para cada fila, se calculan las columnas que se devolverán, indicadas en la parte SELECT. Si hay subconsultas en la parte SELECT que referencian a columnas de la consulta principal, se ejecutan en este momento.
7. Se aplican la UNION, INTERSECCIÓN, DIFERENCIA…
8. Se ordenan las filas según se indica en la parte ORDER BY
9. Solo se muestran las filas que cumplan la condición indicada en LIMIT

UT6. CONSULTAS DE RECUPERACIÓN MULTITABLA EN SQL

Módulo: BASES DE DATOS

Curso 2022/2023. 1° DAM

Ruth Lospitao Ruiz