Apuntes accesibles selección multitabla

Sitio: FRANCISCO DE GOYA
Curso: Bases de Datos Pendientes
Libro: Apuntes accesibles selección multitabla

Imprimido por: Lucía Hernández Montero
Día: lunes, 24 de octubre de 2022, 09:39

Tabla de contenidos

1. Introducción

Una consulta multitabla es aquella en la que se accede simultáneamente a información de más de una tabla.

Para poder realizar este tipo de consultas debemos utilizar la siguiente sintaxis:

```
SELECT [DISTINCT] select_expr [,select_expr] ...

[FROM table_references]

[WHERE where_definition]

[GROUP BY expr [, expr].... ]

[HAVING having_definition]

[ORDER BY {col_name | expr | position} [ASC | DESC] , ...]
```

La diferencia con las consultas sencillas se halla en la cláusula FROM. Con las consultas multitabla, en la cláusula FROM, en lugar de un único nombre de tabla podemos escribir una lista de nombres de tablas separadas por comas (aquí también podemos usar alias para las tablas), o bien, nombres de tablas unidos por operadores relacionales.

Los operadores relacionales nos permiten manipular tablas completas (Es decir, relaciones o conjuntos enteros de tuplas). Podemos obtener una nueva relación uniendo unas tablas con otras, o quitar las tuplas contenidas en una tabla de otra, fusionar tablas usando ciertos atributos comunes, etc.

Los operadores relacionales son: la unión, diferencia, intersección, el producto cartesiano, la combinación y derivados, la unión externa, división, etc. A continaución, veremos en detalle cada uno de estos operadores.

2. Operador join y desambigüación

El operador join es la herramienta para seleccionar y combinar dos o más tablas. En la mayor parte de los casos el operador ON será obligatorio, y casi siempre indispensable.

También será importante tener claras las referencias de las columnas. Si una columna tiene el mismo nombre en dos tablas, habrá que referirse a cada una con el nombre de la tabla delante y separada por un punto para desambigüar. Por ejemplo:

TabA

nombre	edad	salario
Beatriz	41	1800
David	36	1700
Ana	49	1
Javier	56	2

TabB

nombre	telefono
Beatriz	111111
Beatriz	22222
Ana	33333
Javier	44444
Ana	55555

El campo nombre está en ambas tablas, por lo que al realizar una consulta con join, se deberá indicar en cada campo de qué tabla proviene:

```
select TabA.nombre, TabB.nombre from TabA join TabB on TabA.nombre = TabB.nombre;
```

2.1. Join. Producto cartesiano

La forma más básica y que se corresponde con una operación de conjuntos (del siguiente epígrafe) es realizar todas las combinaciones entre todas las filas de ambas tablas.

Por ejemplo, dos tablas sencillas:

Letra

l
A
B
C
D

Número

n
1
2
3
4

Realizamos un producto cartesiano, esto es, combinamos cada fila de una tabla con todas las filas de la otra. Por ello, el resultado sin filtros resulta en tantas filas como la multiplicación de las filas de una tabla por las de la otra. En este caso 4x4 = 16.

```
select l,n from letra join numero
```

Por compatibilidad con el SQL más antiguo, también se acepta separar las tablas con una coma en vez de con la cláusula join. El resultado es idéntico:

```
select l,n from letra, numero
```

l	n
A	1
A	2
A	3
A	4
B	1
B	2
B	3
B	4
C	1
C	2
C	3
C	4
D	1
D	2
D	3
D	4

Se obtienen, como se puede ver, todas las combinaciones. Esto tiene ciertas utilidades. Una de ellas se puede aplicar a una misma tabla dos veces, esto es, se combinan todas las filas de una tabla consigo misma, aunque hay que otorgar obligatoriamente un alias a cada tabla:

```
select f1.nombre, f1.division, f2.nombre, f2.division from futbol f1 join futbol f2;
```

nombre	division	nombre	division
Alcorcón	2	Alcorcón	2
Barcelona	1	Alcorcón	2
Getafe	1	Alcorcón	2
Real Madrid	1	Alcorcón	2

nombre	division	nombre	division
Alcorcón	2	Barcelona	1
Barcelona	1	Barcelona	1
Getafe	1	Barcelona	1
Real Madrid	1	Barcelona	1
Alcorcón	2	Getafe	1
Barcelona	1	Getafe	1
Getafe	1	Getafe	1
Real Madrid	1	Getafe	1
Alcorcón	2	Real Madrid	1
Barcelona	1	Real Madrid	1
Getafe	1	Real Madrid	1
Real Madrid	1	Real Madrid	1

Una aplicación directa, modificando lo anterior, serviría para enfrentamientos directos, esto es todas las combinaciones pero que una fila no se combine consigo misma.

```
select f1.nombre, f1.division, f2.nombre, f2.division from futbol f1 join futbol f2 where f1.nombre != f2.nombre;
```

nombre	division	nombre	division
Barcelona	1	Alcorcón	2
Getafe	1	Alcorcón	2
Real Madrid	1	Alcorcón	2
Alcorcón	2	Barcelona	1
Getafe	1	Barcelona	1
Real Madrid	1	Barcelona	1
Alcorcón	2	Getafe	1
Barcelona	1	Getafe	1
Real Madrid	1	Getafe	1
Alcorcón	2	Real Madrid	1
Barcelona	1	Real Madrid	1
Getafe	1	Real Madrid	1

En este caso, en vez de 16 filas salen 12, según la fórmula n x n-1 donde n es el número de filas.

2.2. Join - On. Tablas relacionadas

Cuando se quiere utilizar la referencia entre dos tablas, en modo de *Foreign Key*, para mostrar datos combinados de ambas tablas, se usará join entre dos tablas, pero se añadirán unas condiciones tras la cláusula ON.

Esta cláusula no es exáctamente como un where, pero funciona de una forma parecida. Se indica una serie de condiciones pero que se corresponden a las FK entre tablas cuando la relación es directa. A parte de esto y por lo general, si las condiciones afectan a un campo de cada tabla, se indican en el ON, en caso contrario en el WHERE.

Véase un ejemplo en la base de datos **cursos**.

Tenemos la tabla persona, con los datos de las personas. Uno de los campos es ciudad, pero aparece el identificador. Mostramos, por brevedad, solo los que tienen rol = 3:

nombre	apellidos	ciudad
Dario	Gómez	1
Pablo	Caballero	1
Juampe	Lara	6
Eva	De Miguel	5
Isabel	De Lucas	2
Merche	González	7
Concepcion	Jiménez	8
Rosa	Pérez	2

Y la tabla ciudad:

id	nombre
1	Madrid
2	Alcorcón
3	Móstoles
4	Fuenlabrada
5	Getafe
6	Leganés
7	San Sebastián de los Reyes
8	Alcobendas
9	Coslada

Ejemplo 1

Podríamos utilizar directamente un join, de una forma similar a los ejemplos del producto cartesiano, pero esto nos dará un resultado que no es el que queremos:

```
select p.nombre, apellidos, ciudad, c.nombre from persona p join ciudad c where rol = 3;
```

Esta consulta devuelve 72 filas, que es la combinación de cada una de las personas con rol = 3 con todas las ciudades. No nos da la información que buscamos.

Si queremos que aparezcan los datos de las personas y el nombre de la ciudad (o más datos que tuviese la tabla ciudad) en la misma consulta, usaremos datos de ambas mediante la relación de la *foreign key* que las une.

```
select p.nombre, apellidos, ciudad, c.nombre from persona p join ciudad c on p.ciudad = c.id where rol = 3;
```

nombre	apellidos	ciudad	nombre
Dario	Gómez	1	Madrid
Pablo	Caballero	1	Madrid
Juampe	Lara	6	Leganés
Eva	De Miguel	5	Getafe
Isabel	De Lucas	2	Alcorcón
Merche	González	7	San Sebastián de los Reyes
Concepcion	Jiménez	8	Alcobendas
Rosa	Pérez	2	Alcorcón

La consulta se explica de la siguiente forma:

- Cada tabla relacionada con el join deberá ser renombrada con un alias (no es obligatorio si no coinciden nombres de campos, pero sí muy útil). En este caso p y c. Por ello en la lista de columnas se refiere dos veces a un campo nombre: la primera a persona y la segunda a ciudad, utilizando los alias.
- Se ha de indicar en la cláusula on cuáles son los campos relacionados e igualarlos. Ya que p.ciudad es una clave ajena a c.id, se igualan. Posteriormente nos quedamos con el nombre de la ciudad, no el id.
- En el where se indican el resto de condiciones, reservándose para el on lo que se refiere a la clave ajena.

En cuanto al resultado, las filas que aparecen son las filas de la primera tabla que tienen relación con la segunda.

Ejemplo 2

En el caso de que en la primera tabla hubiese una fila que no tuviera relación con la segunda, no aparecerá. Por veamos qué sucede si realizamos la siguiente actualización:

```
update persona set ciudad = null where id = 10;
```

En este momento, las filas anteriormente mostradas cambian:

```
select nombre, apellidos, ciudad from personas where rol = 3;
```

nombre	apellidos	ciudad
Dario	Gómez	1
Pablo	Caballero	1
Juampe	Lara	6
Eva	De Miguel	5
Isabel	De Lucas	2
Merche	González	7
Concepcion	Jiménez	8
Rosa	Pérez	NULL

Si ahora realizamos la misma consulta que antes, Rosa no aparecerá:

```
select p.nombre, apellidos, ciudad, c.nombre from persona p join ciudad c on p.ciudad = c.id where rol = 3;
```

nombre	apellidos	ciudad	nombre
Dario	Gómez	1	Madrid
Pablo	Caballero	1	Madrid
Juampe	Lara	6	Leganés
Eva	De Miguel	5	Getafe
Isabel	De Lucas	2	Alcorcón
Merche	González	7	San Sebastián de los Reyes
Concepcion	Jiménez	8	Alcobendas

Se puede dejar los datos como estaban para que no modifique el resultado de posibles ejercicios con la siguiente consulta, pero aún no se ha de hacer, ya que en el siguiente apartado Left join será útil.

```
update persona set ciudad = 2 where id = 10;
```

2.3. Left join

Teniendo en cuenta el orden de las tablas en un select con join, el resultado que obtendremos con un **left join** es que automáticamente tendrá en cuenta las filas de la primera tabla con las coincidencias que haya en la segunda tabla. Y si no hay referencia, aún así mantiene las filas de la tabla izquierda que se mostrasen.

En el ejemplo anterior, del apartado Join - On. Tablas relacionadas, al actualizar la fila de Rosa poniendo nulo en ciudad, se veía cómo al unir la tabla persona y ciudad, la fila de Rosa desaparecía.

nombre	apellidos	ciudad
Dario	Gómez	1
Pablo	Caballero	1
Juampe	Lara	6
Eva	De Miguel	5
Isabel	De Lucas	2
Merche	González	7
Concepcion	Jiménez	8
Rosa	Pérez	NULL

La diferencia con left join se ve en la siguiente consulta:

```
select p.nombre, apellidos, ciudad, c.nombre from persona p left join ciudad c on p.ciudad = c.id where rol = 3;
```

nombre	apellidos	ciudad	nombre
Dario	Gómez	1	Madrid
Pablo	Caballero	1	Madrid
Juampe	Lara	6	Leganés
Eva	De Miguel	5	Getafe
Isabel	De Lucas	2	Alcorcón
Merche	González	7	San Sebastián de los Reyes
Concepcion	Jiménez	8	Alcobendas
Rosa	Pérez	NULL	NULL

Como se puede observar, Rosa no desaparece y en las columnas que perteneciesen a la tabla ciudad aparecerían campos nulos.

2.4. Right join

De la misma forma que left join incluye todos las filas de la tabla de la izquierda, aunque no tengan relación con una fila de la tabla de la derecha (por la condición en la cláusula ON), en el caso del right join sucede lo mismo con la tabla de la derecha.

Por ejemplo, consultemos la misma combinación, pero sin el filtro de rol = 3:

```
select p.nombre, apellidos, ciudad, c.nombre from persona p left join ciudad c on p.ciudad = c.id;
```

nombre	apellidos	ciudad	nombre
Jesús Angel	García	5	Getafe
Josefa	Valcarcel	1	Madrid
...	...	...	...
Roberto	Gómez	1	Madrid

Devuelve 42 filas. Podemos observar cómo Rosa aparece en la décima fila del resultado. Y no hay ninguna fila con Coslada, ya que nadie vive en Coslada. Pero ahora cambiamos la consulta por un right join, dejando el resto de la consulta intacto:

```
select p.nombre, apellidos, ciudad, c.nombre from persona p right join ciudad c on p.ciudad = c.id;
```

nombre	apellidos	ciudad	nombre
Josefa	Valcarcel	1	Madrid
Darío	Gómez	1	Madrid
...	...	...	...
NULL	NULL	NULL	Coslada

En esta ocasión devuelve también 42 filas pero no son las mismas. Además de cambiar el orden por defecto de las filas, la fila de Rosa ya no aparece y en su lugar aparece la última con Coslada. Dado que en la consulta, la tercera columna se selecciona ciudad de la tabla persona, aparecerá como nulo. Si se hubiese elegido id de ciudad, aparecería el número.

2.5. Otros join

En otros lenguajes SQL se admiten muchos tipos de joins. Algunos son equivalentes a las operaciones de conjuntos que se describen en los siguientes epígrafes.

Algunos de ellos son:

- outer join y full outer join: equivalentes a union y union all.
- inner join: equivalentes a la intersección.

En estos casos, como se verá, se pueden emular igualmente en MySQL, pero mediante joins con ciertas condiciones en el ON.

2.6. Join de la misma tabla

En ocasiones necesitamos obtener uniones de filas de una misma tabla. No hay ningún operador nuevo, sino que con el propio join adecuado se ha de indicar en ambos lados la misma tabla. En este caso, obviamente, sí es obligatorio usar un alias para cada tabla, y así discernir para cada campo a qué "tabla" se refiere, si la izquierda o la derecha, pese a ser la misma.

Ejemplo

Seleccionemos de la base de datos cursos, de la tabla personas, parejas de personas que cumplan los años el mismo mes del mismo año:

```
select p1.nombre, p1.apellidos, p1.fnac, p2.nombre, p2.apellidos,p2.fnac from persona p1 join persona p2 on year(p1.fnac) = year(p2.fnac) and month(p1.fnac) = month(p2.fnac);
```

nombre	apellidos	fnac	nombre	apellidos	fnac
Jesús Ángel	García	1961-10-05	Jesús Ángel	García	1961-10-05
Josefa	Valcárcel	1967-01-15	Josefa	Valcárcel	1967-01-15
...	...	...	...	...	...
Roberto	Gómez	1998-05-11	Roberto	Gómez	1998-05-11

Esto tiene un problema: una persona se podrá emparejar consigo misma. De hecho, hay 42 filas en persona y la consulta devuelve 62. Esto se soluciona añadiendo la condición en el on de que una persona no sea la misma, usando por ejemplo su id:

```
select p1.nombre, p1.apellidos, p1.fnac, p2.nombre, p2.apellidos,p2.fnac from persona p1 join persona p2 on year(p1.fnac) = year(p2.fnac) and month(p1.fnac) = month(p2.fnac) and p1.id != p2.id;
```

nombre	apellidos	fnac	nombre	apellidos	fnac
Alejandro	Yañez	1985-06-15	Darío	Gómez	1985-06-23
Beatriz	Martín	2001-03-30	Juan	Ayuso	2001-03-30
...	...	...	...	...	...
Darío	Gómez	1985-06-23	Alejandro	Yañez	1985-06-15
...	...	...	...	...	...
Sarai	Agbar	1998-05-01	Roberto	Gómez	1998-05-11

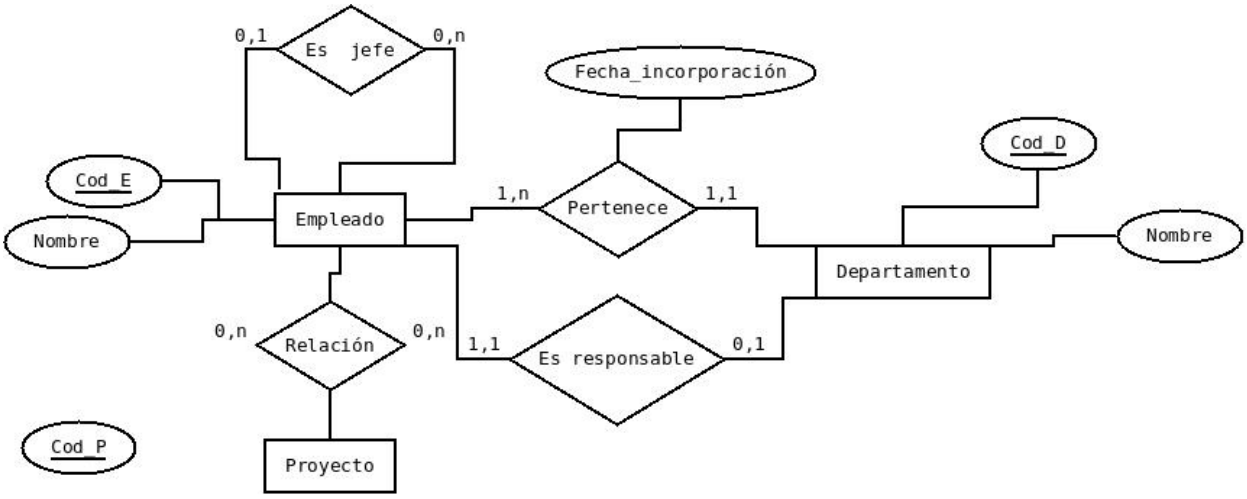
Esto elimina todas las combinaciones de una fila consigo misma. Como se puede observar todas las combinaciones son de personas diferentes que cumplen la condición de nacer el mismo año y mes. Pero observando detenidamente, se puede observar que hay una fila Alejandro Yáñez - Darío Gómez y otra fila Darío Gómez - Alejandro Yáñez. Esto es, la misma combinación pero inversa.

Eliminar estas repeticiones ya es más complejo. Se puede probar, pero no es suficiente con condiciones entre columnas, ya que afecta a diferentes filas del resultado. Tampoco sirve un distinct en una columna ya que eliminaría filas que no se desea eliminar.

3. Join de varias tablas

A la hora de usar una base de datos, en múltiples ocasiones, las tablas están relacionadas entre sí de forma que se puede alcanzar mucha "profundidad", esto es, se pueden unir múltiples tablas relacionadas entre si, con muchas tablas intermediarias.

Por ejemplo, en una base de datos de una empresa se podría saber qué proyectos lleva un jefe de departamento siguiendo un camino a través de las tablas como este: empleado-departamento-empleado-proyecto.

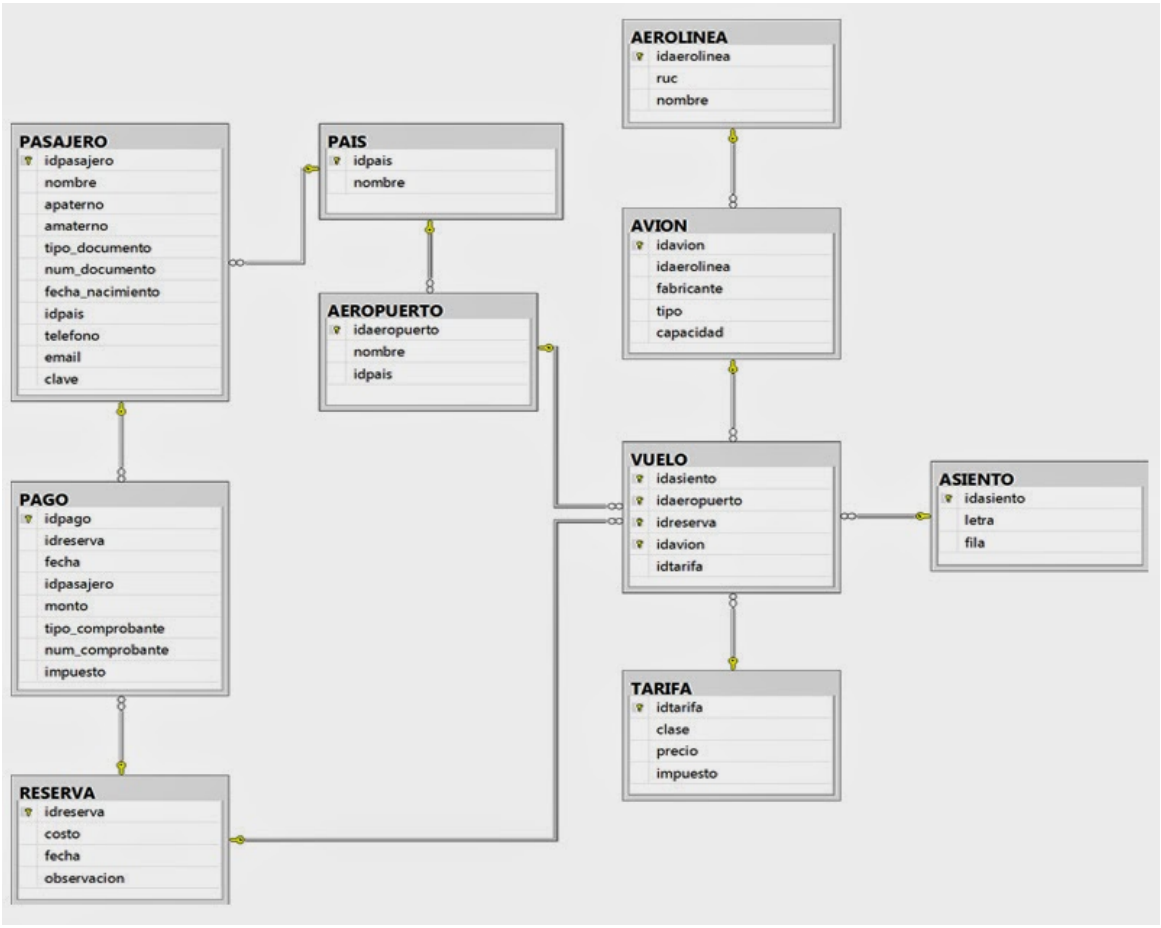


La primera relación es entre el empleado que es jefe de departamento y el departamento, luego del departamento para conseguir todos los empleados del departamento. Posteriormente entre empleado y proyecto para conseguir los proyectos en los que trabajan.

Hay diferentes formas de realizar esto. Se muestran tres de ellas en los siguientes epígrafes. Pero se ha de tener en cuenta que los resultados son iguales, de forma que se podrá usar la forma que mejor se entienda.

3.1. Forma 1. Un Join, un on.

Usaremos como ejemplo la siguiente base de datos, representada por un diagrama de tablas:



Para unir varias tablas es posible hacerlo de la siguiente forma, usando siempre alias para evitar posibles nombres duplicados entre diferentes tablas:

```
select t1.campo1, t2.campo1, t3.campo1

from tabla1 t1 join tabla2 join t2 join tabla3 t3 ...

on t1.campoR2 = t2.id and

t2.campoR3 = t3.id and

t3.campoR4 = t4.id ...

where condiciones_generales
```

De esta forma primero unimos todas las tablas y posteriormente en el on se indican todas las uniones entre tablas, que por lo general serán entre las FKs de una tabla a las PKs de otra.

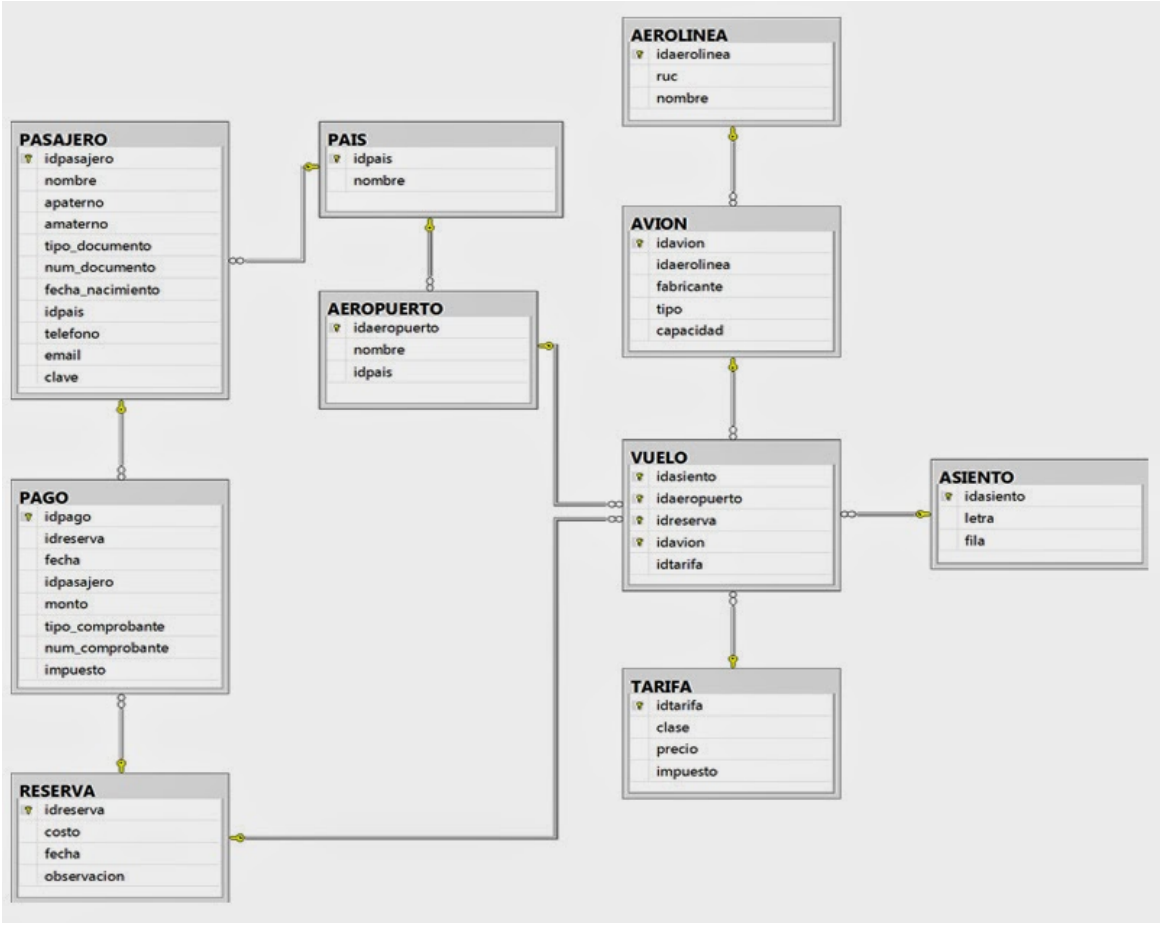
```
select ael.nombre as aerolinea, av.idavion, as.letra, asn.fila, ap.nombre as aeropuerto from

Aerolinea ael join Avion av join Vuelo v join Asiento asn join Aeropuerto ap

on av.idAerolinea = ael.idaerolinea

and v.idavion = av.idavion

and v.idasiento = asn.idasiento

and v.ideaeropuerto = ap.idaeropuerto;
```

Con este código obtendríamos una tabla similar a la siguiente:

aerolinea	idavion	letra	fila	aeropuerto
BAW	RF542	D	33	JFK
QAW	WE839	A	12	HTRW
...	...	...	...	
IBER	PS902	C	34	CHDG

3.2. Forma 2. Un join varios on.

Usaremos la misma base de datos:



De esta otra forma, cada join lleva aparejado un on:

```
select t1.campo1, t2.campo1, t3.campo1 ...

from tabla1 t1 join tabla2 t2

on t1.campoR2 = t2.id

join tabla3 t3

on t2.campoR3 = t3.id

join tabla4 t4 on t3.campoR4 = t4.id ...

where condiciones_generales
```

De esta forma por cada join que hacemos indicamos el on que une ambas tablas.

```
select ael.nombre as aerolinea, av.idavion, as.letra, asn.fila, ap.nombre as aeropuerto from

Aerolinea ael join Avion av

on av.idAerolinea = ael.idaerolinea

join Vuelo v

on v.idavion = av.idavion

join Asiento asn

on v.idasiento = asn.idasiento

join Aeropuerto ap
```

```
on v.ideaeropuerto = ap.idaeropuerto;
```

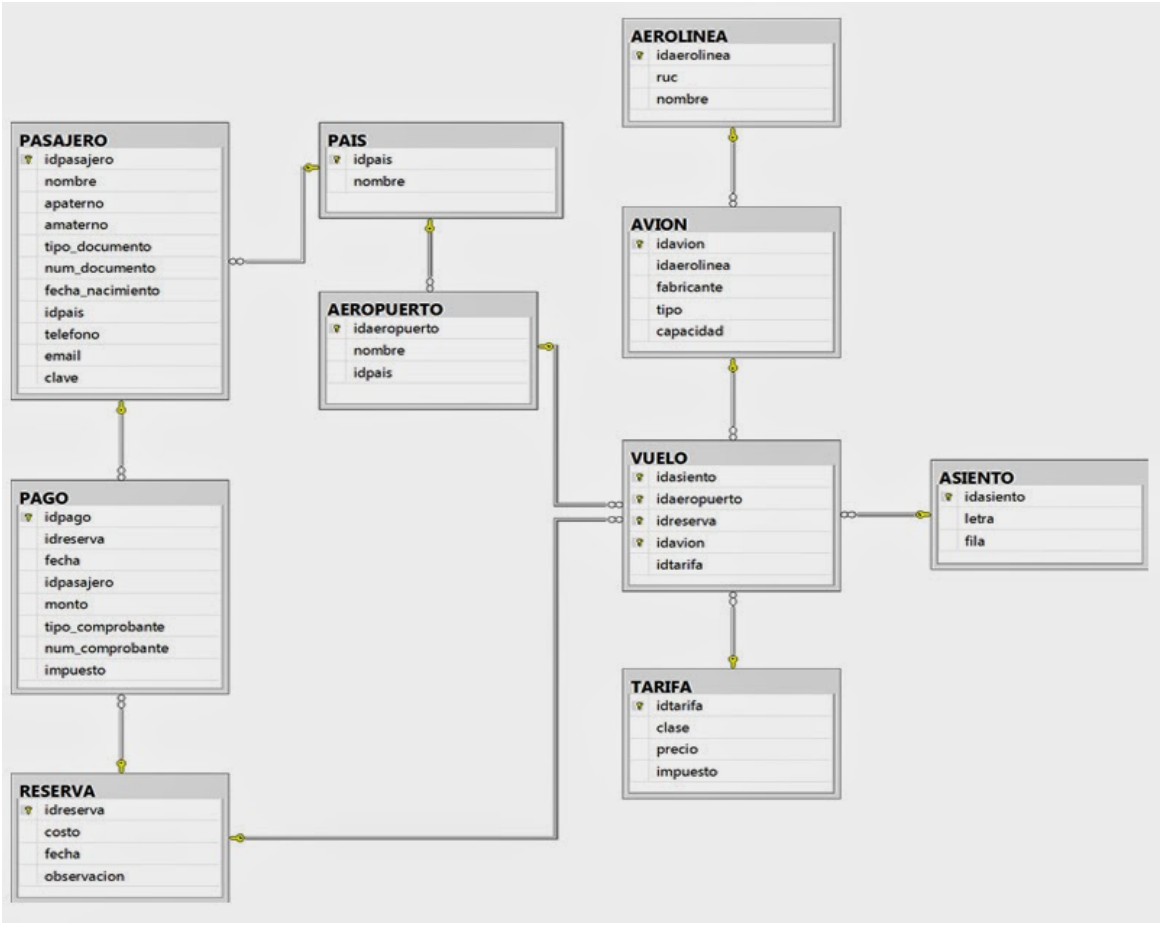
El resultado que obtenemos es idéntico al de la forma anterior. Como se ha dicho, son formas diferentes, pero que funcionan igual.

aerolinea	idavion	letra	fila	aeropuerto
BAW	RF542	D	33	JFK
QAW	WE839	A	12	HTRW
...	...	...	...	
IBER	PS902	C	34	CHDG

3.3. Forma 3. Join de uno en uno con subconsultas.

Esta forma puede ser más larga, pero a la vez es más segura, porque se va construyendo la consulta poco a poco, comprobando que funciona a cada paso.

Seguimos con el mismo ejemplo:



Procedemos uniendo dos tablas con pseudocódigo:

```
select t1.campo1, t2.campo1

from tabla1 t1 join tabla2 t2

on t1.campoR2 = t2.id

where condiciones_t1_y_t2
```

Luego añadimos la tercera tabla, usando la anterior consulta como subconsulta (campoR3 sería cualquiera de los campos campo1 o campo2 que se relacione con tabla3):

```
select s1.campo1, s1.campo1, t3.campo1

from tabla3 t3 join (select t1.campo1, t2.campo1 from tabla1 t1 join tabla2 on t1.campoR2 = t2.idwhere condiciones_t1_t2) as s1

on s1.campoR3 = t3.id

where condiciones_t3
```

Luego añadimos la cuarta tabla:

```
select s2.campo1, s2.campo1, s2.campo1, t4.campo1

from tabla4 join (select s1.campo1, s1.campo1, t3.campo1from tabla3 t3 join (select t1.campo1, t2.campo1from tabla1 t1 join tabla2on t1.campoR2 = t2.idwhere condiciones_t1_t2) as s1on s1.campoR3 = t3.idwhere condiciones_t3) as s2

on s2.campoR4 = t4.id

where condiciones_generales
```

De esta forma vamos uniendo. Primera consulta entre Aerolinea y Avion:

```
select ael.nombre, av.idavion
from Aerolinea ael join Avion av
on ael.idaerolinea = av.idaerolinea;
```

Segunda, añadiendo Vuelo. En esta añadimos campos que se van a usar en condiciones de ON, pero que finalmente no se van a mostrar, como idaeropuerto y idasiento.

```
select v.idasiento, v.idaeropuerto, s1.idavion, s1.nombre
from Vuelo join
(select ael.nombre, av.idavion
from Aerolinea ael join Avion av
on ael.idaerolinea = av.idaerolinea) as s1
on s1.idavion = v.idavion) as s2
```

Tercera, añadiendo Asiento. Ya que el campo nombre de la aerolinea y nombre de aeropuerto se llaman igual, añadimos un alias a aerolinea. Así en la siguiente no tiene conflicto.

```
select asi.letra, asi.fila, s2.idaeropuerto, s2.idavion, s2.nombre as aerolinea
from Asiento asi join
(select v.idasiento, v.idaeropuerto, s1.idavion, s1.nombre
from Vuelo join
(select ael.nombre, av.idavion
from Aerolinea ael join Avion av
on ael.idaerolinea = av.idaerolinea) as s1
on s1.idavion = v.idavion) as s2
on s2.idasiento = asi.idasiento;
```

Última, añadiendo Aeropuerto. No seleccionamos ya los campos id que no vamos a mostrar, como idaerolinea e idaeropuerto.

```
select s3.letra, s3.fila, s3.idavion, s3.aerolinea, aep.nombre as aeropuerto
from Aeropuerto aep join
(select asi.letra, asi.fila, s2.idaeropuerto, s2.idavion, s2.nombre as aerolinea
from Asiento asi join
(select v.idasiento, v.idaeropuerto, s1.idavion, s1.nombre
from Vuelo join
(select ael.nombre, av.idavion
from Aerolinea ael join Avion av
on ael.idaerolinea = av.idaerolinea) as s1
on s1.idavion = v.idavion) as s2
on s2.idasiento = asi.idasiento) as s3
on s3.idaeropuerto = aep.idaeropuerto;
```

Con este código obtendríamos el mismo resultado, pero previamente tenemos resultados parciales que podemos ir comprobando:

aerolinea	idavion	letra	fila	aeropuerto

aerolinea	idavion	letra	fila	aeropuerto
BAW	RF542	D	33	JFK
QAW	WE839	A	12	HTRW
...	...	...	...	
IBER	PS902	C	34	CHDG

4. Operaciones de conjuntos

Las operaciones de conjuntos son unas operaciones multitabla concretas y se describen en los siguientes apartados.

En todos los casos se utiliza left join uniendo por los campos que sean equivalentes. Sin realizar ninguna operación, uniendo todo, resultaría lo siguiente:

Tabla futbol:

nombre	division
Alcorcon	2
Barcelona	1
Getafe	1
Real Madrid	1

Tabla baloncesto

nombre	liga
Barcelona	ACB
Lakers	NBA
Real Madrid	ACB

Para unir ambas tablas por el nombre se realizará la siguiente consulta

```
select * from futbol left join baloncesto on futbol.nombre = baloncesto.nombre;
```

nombre	division	nombre	liga
Barcelona	1	Barcelona	ACB
Real Madrid	1	Real Madrid	ACB
Alcorcón	2	NULL	NULL
Getafe	1	NULL	NULL

El orden es significativo, ya que si lo cambiamos obtenemos:

```
select * from baloncesto left join futbol on futbol.nombre = baloncesto.nombre;
```

nombre	liga	nombre	division
Barcelona	ACB	Barcelona	1
Real Madrid	ACB	Real Madrid	1
Lakers	NBA	NULL	NULL

Por ello es muy importante ver cómo se mezclan ambas tablas y entender lo que significan los nulos: una fila en una tabla que no se encuentra en la otra.

4.1. Unión

Esta operación se representa por el símbolo ∪ . Si R y S son dos relaciones de mismo esquema, la relación resultante R ∪ S es la unión de ambos conjuntos de tuplas.

Por ejemplo, si tenemos por un lado una tabla Pescadores y, por otra, una tabla de Patrones, podemos unir ambas tablas de forma que obtengamos una relación con todos los tripulantes de barco posibles:

Pescadores:

Código	Nombre	Dirección	Ciudad	Volumen_pesca
023	Pepe Rosales	C\Sol 5	Madrid	134
101	Luís Barbudo	La Rúa	Salamanca	230
044	Ana Billar	Gran Vía	Barcelona	171

Patrones:

Código	Nombre	Dirección	Ciudad	Fecha_titulo
102	Lidia Martín	C\de la Luna, 7	Madrid	12/02/2006
21	Manuel Vázquez	C\Pinares 15	Valladolid	04/12/1998
84	Hipólito Verdana	Avda. General	Cádiz	22/10/2000

Tal y como están sus esquemas no son coincidentes, es decir, no tienen exactamente los mismos atributos, y no podríamos aplicar la unión. Pero si proyectamos las tablas, de forma que desaparezcan los atributos Fecha_Título y Volumen_Pesca, sí que podríamos, obteniendo el siguiente resultado:

Código	Nombre	Dirección	Ciudad
023	Pepe Rosales	C\Sol 5	Madrid
101	Luís Barbudo	La Rúa	Salamanca
044	Ana Billar	Gran Vía	Barcelona
102	Lidia Martín	C\de la Luna, 7	Madrid
21	Manuel Vázquez	C\Pinares 15	Valladolid
84	Hipólito Verdana	Avda. General	Cádiz

El orden en que aparecen las tuplas en la relación resultante es lo de menos.

La sentencia SQL (que no MySql) que obtiene la unión de estas dos tablas sería como sigue:

```
Select codigo, nombre, direccion from Patrones UNION Select codigo, nombre, direccion from Pescadores;
```

O bien, equivalentemente:

```
Select codigo, nombre, direccion from Pescadores UNION Select codigo, nombre, direccion from Patrones;
```

Pero hay que tener en cuenta que esto son sentencias SQL, que pueden usarse en algunos SGBD como Oracle, pero no siempre en MySql. En el caso de la unión, sí es posible.

Ejemplos

En MySql podemos usar la base de datos multitabla como ejemplo:

Marino:

nombre	edad	salario
Beatriz	41	1800
David	36	1700

Tabla estribador:

nombre	edad	salario	grúa
Ana	49	2400	1
Javier	56	2800	2

La forma de realizar una unión con MySql sería con el siguiente código

```
select nombre, edad from marino union select nombre, edad from estribador
```

nombre	edad
Ana	49
Javier	56

nombre	edad
Beatriz	41
David	36

Hay que notar que siempre han de tener el mismo número de columnas en ambos lados del union, o dará error. Por ejemplo:

```
select nombre, edad, salario from marino union select nombre, edad from estribador
```

Se ha de tener en cuenta que si coinciden en número, pero son datos diferentes, el resultado puede resultar incongruente. En el siguiente ejemplo, se mezclan grúas con salario en la tercera columna:

```
select nombre, edad,salario from marino union select nombre, edad,grua from estribador;
```

nombre	edad	salario
Beatriz	41	1800
David	36	1700
Ana	49	1
Javier	56	2

4.2. Union all

Una unión une las filas de una y otra tabla, pero elimina las repeticiones. En caso de que queramos que aparezcan todas las filas, sin eliminar repeticiones se realizará un union all:

Tabla futbol:

nombre	division
Alcorcon	2
Barcelona	1
Getafe	1
Real Madrid	1

Tabla baloncesto:

nombre	liga
Barcelona	ACB
Lakers	NBA
Real Madrid	ACB

De esta forma, un union normal resultará en lo siguiente:

```
select nombre from futbol union select nombre from baloncesto;
```

nombre
Alcorcon
Barcelona
Getafe
Real Madrid
Lakers

Obtenemos 5 filas. Sin embargo, si realizamos un union all, obtendremos 7 ya que "Barcelona" y "Real Madrid" aparecerán dos veces:

```
select nombre from futbol union all select nombre from baloncesto;
```

nombre
Alcorcon
Barcelona
Getafe
Real Madrid
Barcelona
Lakers
Real Madrid

4.3. Diferencia

La diferencia entre dos tablas, expresadas como T1 - T2 funciona de forma similar a la unión, con la diferencia de que en el resultado se descartan las filas que estén en la segunda tabla de la primera tabla.

Importante: Oracle soporta el operador minus, pero MySQL no. Por ello se ha de realizar de una forma alternativa utilizando left join.

```
select futbol.nombre from futbol left join baloncesto on futbol.nombre = baloncesto.nombre where baloncesto.nombre is null;
```

nombre
Alcorcon
Getafe

Es importante entender cómo se realiza la diferencia:

1. Se unen dos tablas, seleccionando el nombre de una de las dos. Al haber dos columnas "nombre", una en baloncesto y otra en futbol, en la lista de campos hay que especificar de qué tabla se trata: se desambigü:. **futbol.nombre**
2. Se unen las tablas a través del ON: se indica qué campo de la primera tabla ha de ser igual a qué otro campo de la segunda: **on futbol.nombre = baloncesto.nombre**
3. Se filtran aquellos cuyo nombre de baloncesto es nulo. El campo baloncesto.nombre no se muestra pero explica bastante este filtro. Si se mostrase, el resultado sería:

```
select futbol.nombre,baloncesto.nombre from futbol left join baloncesto on futbol.nombre = baloncesto.nombre;
```

nombre	nombre
Alcorcon	NULL
Barcelona	Barcelona
Getafe	NULL
Real Madrid	Real Madrid

De ahí que al juntar las dos tablas por el campo nombre de ambas, si no coinciden con alguna otra se insertan pero con el campo de la otra tabla en nulo. Por ello es tan sencillo como eliminar esos nulos para hacer la diferencia.

4.4. Intersección

La intersección entre dos tablas T1 ∩ T2 mostrará aquellas filas que estén tanto en la primera como en la segunda tabla. Si por ejemplo tenemos estas dos tablas:

Tabla1

ID
B
C
D
E

Tabla2

ID
A
C
D
E
G

El resultado de T1 ∩ T2 será

ID
C
D
E

Ejemplos

Si utilizamos la base de datos multitabla realizaremos una intersección entre baloncesto y futbol. A diferencia de la diferencia, donde quedaban solo Getafe y Alcorcón, debido a que Real Madrid y Barcelona estaban en ambas, en este caso el resultado cambia:

```
select futbol.nombre from futbol left join baloncesto on futbol.nombre = baloncesto.nombre where baloncesto.nombre = futbol.nombre;
```

nombre
Barcelona
Real Madrid

La consulta se explica de la siguiente forma:

1. Se unen como antes futbol y baloncesto a través del campo nombre: on futbol.nombre = baloncesto.nombre
2. Con ello se consigue una tabla que contenga, además de otros campos, futbol.nombre y baloncesto.nombre. Si solo están en una tabla, en la otra aparece null. Y si están en ambas ambos nombres serán iguales. Por ello, para quedarse con esas filas, en el where se añade la condición: on futbol.nombre = baloncesto.nombre