Principales tipos de JOINS en SQL

Los JOINS en SQL permiten combinar las filas de dos o más tablas para poder mostrar result-sets de forma relacional

Contenido modificable

Si ves errores o quieres modificar/añadir contenidos, puedes crear un pull request. Gracias

Los JOINs en SQL sirven para combinar filas de dos o más tablas basándose en un campo común entre ellas, devolviendo por tanto datos de diferentes tablas. Un JOIN se produce cuando dos o más tablas se juntan en una sentencia SQL.

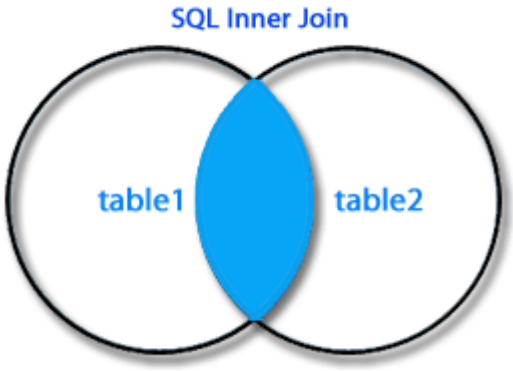
Existen más tipos de joins en SQL que los que aquí se explican, como CROSS JOIN, O SELF JOIN, pero no todos ellos están soportados por todos los sistemas de bases de datos. Los más importantes son los siguientes:

1. INNER JOIN: Devuelve todas las filas cuando hay al menos una coincidencia en ambas tablas.
2. LEFT JOIN: Devuelve todas las filas de la tabla de la izquierda, y las filas coincidentes de la tabla de la derecha.
3. RIGHT JOIN: Devuelve todas las filas de la tabla de la derecha, y las filas coincidentes de la tabla de la izquierda.
4. OUTER JOIN: Devuelve todas las filas de las dos tablas, la izquierda y la derecha. También se llama FULL OUTER JOIN.

INNER JOIN selecciona todas las filas de las dos columnas siempre y cuando haya una coincidencia entre las columnas en ambas tablas. Es el tipo de JOIN más común.

```
SELECT nombreColumna(s)
FROM tabla1
INNER JOIN tabla2
ON tabla1.nombreColumna=table2.nombreColumna;
```

Se ve más claro utilizando una imagen:



Vamos a verlo también con un ejemplo, mediante las tablas Clientes y Pedidos:

Clientes:

ClienteID	NombreCliente	Contacto
1	Marco Lambert	456443552
2	Lydia Roderic	445332221
3	Ebbe Therese	488982635
4	Sofie Mariona	412436773

Pedidos:

PedidoID	ClienteID	Factura
234	4	160
235	2	48
236	3	64
237	4	92

La siguiente sentencia SQL devolverá todos los clientes con pedidos:

```
SELECT Clientes.NombreCliente, Pedidos.PedidoID FROM Clientes
INNER JOIN Pedidos ON Clientes.ClienteID=Pedidos.ClienteID
```

Si hay filas en Clientes que no tienen coincidencias en Pedidos, los Clientes no se mostrarán. La sentencia anterior mostrará el siguiente resultado:

NombreCliente	PedidoID
Ebbe Therese	236
Lydia Roderic	235
Sofie Mariona	234
Sofie Mariona	237

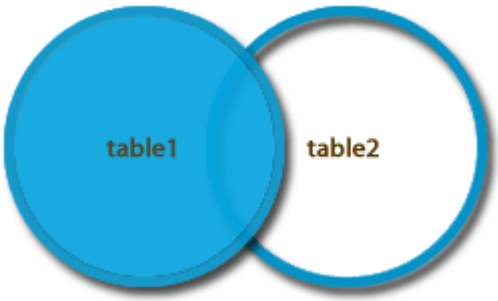
Sofie Mariona aparece dos veces ya que ha realizado dos pedidos. No aparece *Marco Lambert*, pues no ha realizado ningún pedido.

2. LEFT JOIN

LEFT JOIN mantiene todas las filas de la tabla izquierda (la tabla1). Las filas de la tabla derecha se mostrarán si hay una coincidencia con las de la izquierda. Si existen valores en la tabla izquierda pero no en la tabla derecha, ésta mostrará null.

```
SELECT nombreColumna(s)
FROM tabla1
LEFT JOIN tabla2
ON tabla1.nombreColumna=tabla2.nombreColumna;
```

La representación de LEFT JOIN en una imagen es:



Tomando de nuevo las tablas de Productos y Pedidos, ahora queremos mostrar todos los clientes, y cualquier pedido que pudieran haber encargado:

```
SELECT Clientes.NombreCliente, Pedidos.PedidoID
FROM Clientes LEFT JOIN Pedidos
ON Clientes.ClienteID=Pedidos.ClienteID
ORDER BY Clientes.NombreCliente;
```

La sentencia anterior devolverá lo siguiente:

NombreCliente	PedidoID

Marco Lambert	(null)
Sofie Mariona	234
Sofie Mariona	237

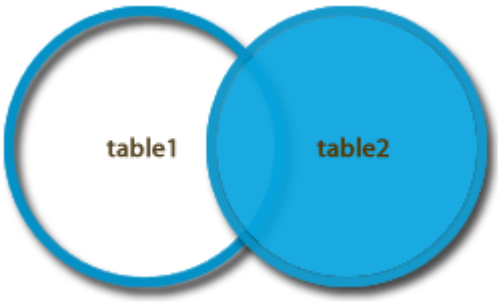
Ahora vemos que se muestran todas las filas de la tabla Clientes, que es la tabla de la izquierda, tantas veces como haya coincidencias con el lado derecho. *Marco Lambert* no ha realizado ningún pedido, por lo que se muestra null.

3. RIGHT JOIN

Es igual que LEFT JOIN pero al revés. Ahora se mantienen todas las filas de la tabla derecha (tabla2). Las filas de la tabla izquierda se mostrarán si hay una coincidencia con las de la derecha. Si existen valores en la tabla derecha pero no en la tabla izquierda, ésta se mostrará null.

```
SELECT nombreColumna(s)
FROM tabla1
RIGHT JOIN tabla2
ON tabla1.nombreColumna=tabla2.nombreColumna;
```

La imagen que representa a RIGHT JOIN es:



De nuevo tomamos el ejemplo de Clientes y Pedidos, y vamos a hacer el mismo ejemplo anterior, pero cambiado LEFT por RIGHT:

```
SELECT Pedidos.PedidoID, Clientes.NombreCliente
FROM Clientes RIGHT JOIN Pedidos
ON Clientes.ClienteID=Pedidos.ClienteID
ORDER BY Pedidos.PedidoID;
```

Ahora van a aparecer todos los pedidos, y los nombres de los clientes que han realizado un pedido. Nótese que también se ha cambiado el orden, y se han ordenado los datos por PedidoID.

PedidoID	NombreCliente

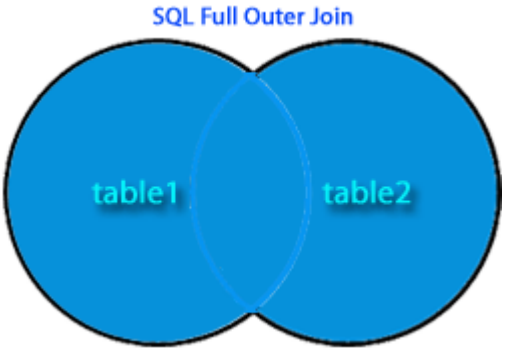
236	Ebbe Therese
237	Sofie Mariona

4. OUTER JOIN

OUTER JOIN o FULL OUTER JOIN devuelve todas las filas de la tabla izquierda (tabla1) y de la tabla derecha (tabla2). Combina el resultado de los joins LEFT y RIGHT. Aparecerá null en cada una de las tablas alternativamente cuando no haya una coincidencia.

```
SELECT nombreColumna(s)
FROM tabla1
OUTER JOIN tabla2
ON tabla1.nombreColumna=tabla2.nombreColumna;
```

La imagen que representa el OUTER JOIN es la siguiente:



Vamos a obtener todas las filas de las tablas Clientes y Pedidos:

```
SELECT Clientes.NombreCliente, Pedidos.PedidoID
FROM Clientes OUTER JOIN Pedidos
ON Clientes.ClienteID=Pedidos.ClienteID
ORDER BY Clientes.NombreCliente;
```

La sentencia devolverá todos los Clientes y todos los Pedidos, si un cliente no tiene pedidos mostrará null en PedidoID, y si un pedido no tuviera un cliente mostraría null en NombreCliente (en este ejemplo no sería lógico que un Pedido no tuviera un cliente).

La sintaxis de OUTER JOIN o FULL OUTER JOIN no existen en MySQL, pero se puede conseguir el mismo resultado de diferentes formas, esta es una:

```
SELECT Clientes.NombreCliente, Pedidos.PedidoID
FROM Clientes
LEFT JOIN Pedidos ON Clientes.ClienteID=Pedidos.ClienteID
```

```
SELECT Clientes.NombreCliente, Pedidos.PedidoID
FROM Clientes
RIGHT JOIN Pedidos ON Clientes.ClienteID=Pedidos.ClienteID;
```