Relación entre tablas

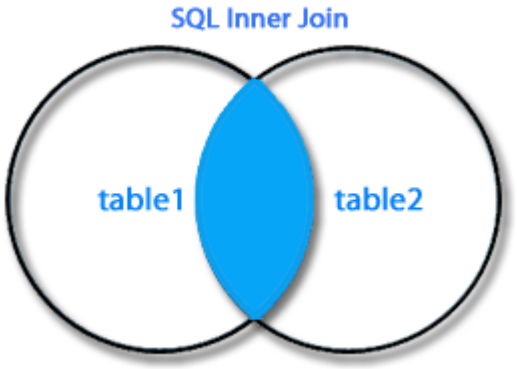
JOIN

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.
Los más importantes son los siguientes:
INNER JOIN: Devuelve todas las filas cuando hay al menos una coincidencia en ambas tablas.
LEFT JOIN: Devuelve todas las filas de la tabla de la izquierda, y las filas coincidentes de la tabla de la derecha.
RIGHT JOIN: Devuelve todas las filas de la tabla de la derecha, y las filas coincidentes de la tabla de la izquierda.
OUTER JOIN: Devuelve todas las filas de las dos tablas, la izquierda y la derecha. También se llama **FULL OUTER JOIN**.
NATURAL JOIN se usa cuando los campos por los cuales se enlazan las tablas tienen el mismo nombre. También se puede usar **NATURAL** con **LEFT JOIN** y **RIGHT JOIN**

INNER 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;
```



Como se muestra en el gráfico el INNER JOIN devuelve la intersección entre las tablas que intervienen.
Una consulta puede tener más de un INNER JOIN, dependiendo de la cantidad de tablas que se quieren relacionar.

Ejemplo INNER JOIN

Si tenemos las tablas Clientes y Pedidos:

Clientes

ClienteID	NombreCliente	Contacto
1	Marco Lambert	456443552
2	Lydia Roderic	45332221
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
ORDER BY Clientes.NombreCliente;
```

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

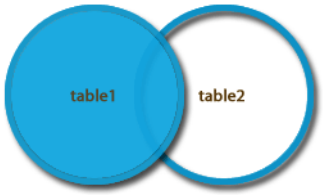
NombreCliente	PedidoID
Ebbe Therese	236
Lydia Roderic	235
Sofie Mariona	234
Sofie Mariona	237

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

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;
```



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;
```

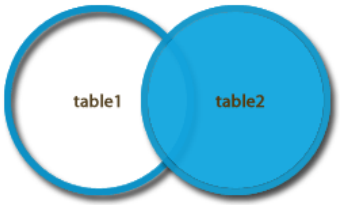
NombreCliente	PedidoID
Ebbe Therese	236
Lydia Roderic	235
Marco Lambert	(null)
Sofie Mariona	234
Sofie Mariona	237

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.

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;
```



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;
```

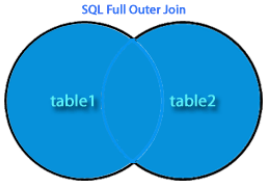
PedidoID	NombreCliente
234	Sofie Mariona
235	Lydia Roderic
236	Ebbe Therese
237	Sofie Mariona

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.

OUTER JOIN / FULL 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;
```



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 d*evolverá 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 haciendo UNION de dos consultas. para poder utilizar UNION, es que en todos los SELECT debe haber los mismos campos seleccionados.

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

UNION

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

NATURAL JOIN

NATURAL JOIN se usa cuando los campos por los cuales se enlazan las tablas tienen el mismo nombre.
Tenemos las tablas "libros" y "editoriales" de una librería.

LIBROS
codigo (clave primaria)
titulo
autor
codigoeditorial
precio

EDITORIALES
codigoeditorial(clave primaria)
nombre

Como en ambas tablas, el código de la editorial se denomina codigoeditorial, podemos omitir la parte ON que indica los nombres de los campos por el cual se enlazan las tablas, empleando **NATURAL JOIN**, se unirán por el campo que tienen en común:

```
select titulo,nombre
from libros as l
natural join editoriales as e;
```

Es equivalente a

```
select titulo,nombre
from libros as l
inner join editoriales as e
on l.codigoeditorial=e.codigoeditorial;
```

Es decir, con NATURAL JOIN no se coloca la parte "on" que especifica los campos por los cuales se enlazan las tablas, porque MySQL busca los campos con igual nombre y enlaza las tablas por ese campo.

NATURAL JOIN

También se puede usar **NATURAL** con **LEFT JOIN** y **RIGHT JOIN**:

```
select nombre,titulo
from editoriales as e
natural left join libros as l;
```

Es equivalente a

```
select nombre,titulo
from editoriales as e
left join libros as l
on e.codigoeditorial=l.codigoeditorial;
```

Hay que tener cuidado con este tipo de **JOIN** porque ***si ambas tablas tiene más de un campo con igual nombre, MySQL no sabrá por cual debe realizar*** la unión. Por ejemplo, si el campo "titulo" de la tabla "libros" se llamara "nombre", las tablas tendrían 2 campos con igual nombre ("codigoeditorial" y "nombre").

Otro problema que puede surgir es el siguiente. Tenemos la tabla "libros" con los siguientes campos: codigo (del libro), titulo, autor y codigoeditorial, y la tabla "editoriales" con estos campos: codigo (de la editorial) y nombre. Si usamos "natural join", unirá las tablas por el campo "codigo", que es el campo que tienen igual nombre, pero el campo "codigo" de "libros" no hace referencia al código de la editorial sino al del libro, así que la salida será errónea.