Tipos de Join

En general, las bases de datos relacionales están normalizadas para eliminar la información duplicada. Con lo cual, para poder combinar información de dos o más tablas, es necesario recurrir a las uniones o JOINs. En este artículo vamos a ver diferentes formas de realizar JOINs en tablas utilizando SQL.

Preparativos

Vamos a crear dos tablas (en cualquier motor de bases de datos): Empleados y Categorías.

Empleados:

Id	Apellido	Categoria
1	González	1
2	Pereyra	5
3	Gutierrez	4
4	Santo	2
5	Liberti	5
6	Pérez	5
7	Rivarola	5
8	Andrade	3
9	Lucca	5
10	Silva	5
11	Zambani	5
12	Di Bari	4
13	De Lucía	5
14	Franco	*NULL*

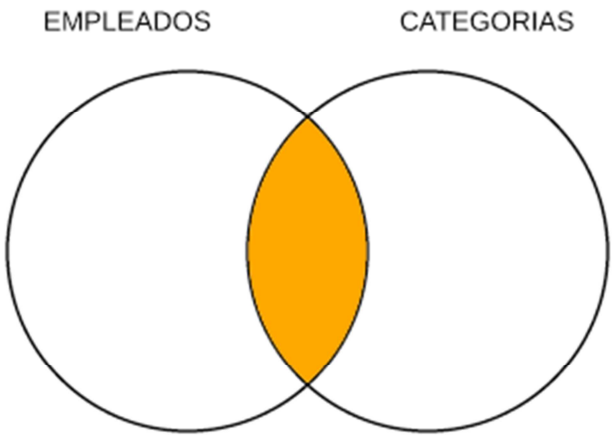
Categorías:

Id	Nombre
1	Gerente General
2	Gerente
3	Sub Gerente
4	Jefe
5	Empleado
6	Cadete

Como podemos ver cada empleado tiene asociada una categoría, con este campo vamos a poder unir a las dos tablas. También podemos observar que hay un empleado (Franco) que no tiene asignada una categoría aún y existe una categoría (cadete) la cual no posee ningún empleado.

INNER JOIN

La forma más utilizada de combinación se llama INNER JOIN, y el resultado es el cálculo cruzado de todos los registros. Se combinan los registros de la tabla Empleados con la tabla Categorías, pero sólo van a permanecer los registros que satisfagan la condición especificada.
Podemos graficar este ejemplo con un diagrama de Venn:



El código a utilizar es el siguiente:

```
SELECT e.Id, e.Apellido, c.Id, c.Nombre FROM Empleados e INNER JOIN Categorías
c ON e.Categoria = c.Id
```

Y el resultado obtenido es este:

	Id	Apellido	Id	Nombre
1	1	González	1	Gerente General
2	2	Pereyra	5	Empleado
3	3	Gutierrez	4	Jefe
4	4	Santo	2	Gerente
5	5	Liberti	5	Empleado
6	6	Pérez	5	Empleado
7	7	Rivarola	5	Empleado
8	8	Andrade	3	Sub Gerente
9	9	Lucca	5	Empleado
10	10	Silva	5	Empleado
11	11	Zambani	5	Empleado
12	12	Di Bari	4	Jefe
13	13	De Lucía	5	Empleado

En este resultado, no aparecen ni el empleado Franco, ni la categoría Cadete. Es importante tener especial cuidado con los valores nulos (null), ya que estos no se combinan ni siquiera con otros valores nulos (a menos que se utilicen los predicados `IS NULL` o `IS NOT NULL`).

De la misma forma que igualamos el `Id` de la categoría con la columna `Categoría` en

empleado, podemos utilizar los operadores de comparación para realizar cualquier otra restricción.

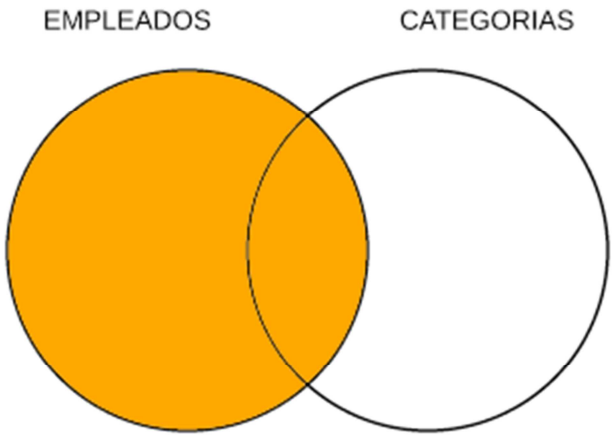
Por ejemplo, si queremos sólo los empleados que tengan un cargo gerencial, podemos hacer lo siguiente:

```
SELECT e.Id, e.Apellido, c.Id, c.Nombre FROM Empleados e INNER JOIN Categorías
c ON e.Categoria = c.Id AND c.Id <= 3
```

	Id	Apellido	Id	Nombre
1	1	González	1	Gerente General
2	4	Santo	2	Gerente
3	8	Andrade	3	Sub Gerente

LEFT JOIN

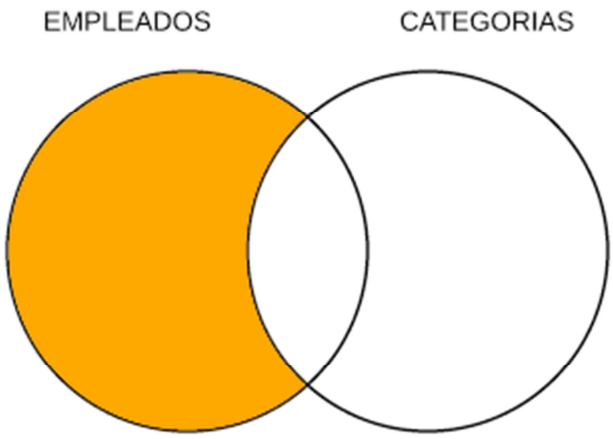
Si lo que necesitamos es una lista de los empleados, sin importar si pertenecen o no a una categoría, entonces lo que tenemos que utilizar es LEFT JOIN. El resultado que produce este JOIN son todas las filas de la tabla que se encuentre a la izquierda, sin importar si coinciden con las filas de la derecha.




```
SELECT e.Id, e.Apellido, c.Id, c.Nombre FROM Empleados e LEFT JOIN Categorías
c ON e.Categoria = c.Id
```

	Id	Apellido	Id	Nombre
1	1	González	1	Gerente General
2	2	Pereyra	5	Empleado
3	3	Gutierrez	4	Jefe
4	4	Santo	2	Gerente
5	5	Liberti	5	Empleado
6	6	Pérez	5	Empleado
7	7	Rivarola	5	Empleado
8	8	Andrade	3	Sub Gerente
9	9	Lucca	5	Empleado
10	10	Silva	5	Empleado
11	11	Zambani	5	Empleado
12	12	Di Bari	4	Jefe
13	13	De Lucía	5	Empleado
14	14	Franco	NULL	NULL

Si lo que buscamos es obtener los empleados que no poseen ninguna categoría asignada, tenemos que agregar la cláusula `WHERE`.

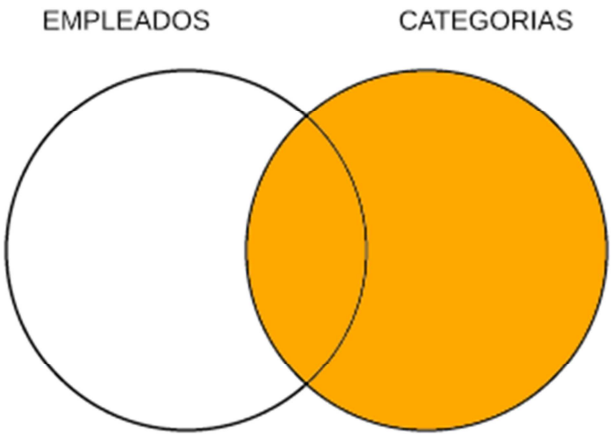



```
SELECT e.Id, e.Apellido, c.Id, c.Nombre FROM Empleados e LEFT JOIN Categorías
c ON e.Categoria = c.Id WHERE c.Id IS NULL
```

	Id	Apellido	Id	Nombre
1	14	Franco	NULL	NULL

RIGHT JOIN

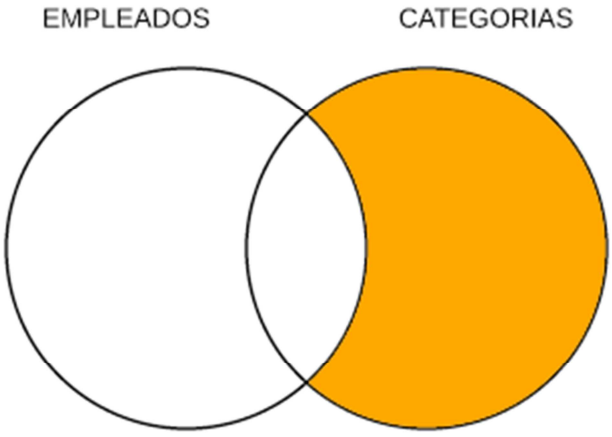
De la misma forma que podemos obtener todas las filas de la tabla de la izquierda con LEFT JOIN, con RIGHT JOIN obtenemos todas las filas de la tabla derecha, sin importar si coinciden con las de la tabla de la izquierda.




```
SELECT e.Id, e.Apellido, c.Id, c.Nombre FROM Empleados e RIGHT JOIN Categorías
c ON e.Categoria = c.Id
```

	Id	Apellido	Id	Nombre
1	1	González	1	Gerente General
2	4	Santo	2	Gerente
3	8	Andrade	3	Sub Gerente
4	3	Gutierrez	4	Jefe
5	12	Di Bari	4	Jefe
6	2	Pereyra	5	Empleado
7	5	Liberti	5	Empleado
8	6	Pérez	5	Empleado
9	7	Rivarola	5	Empleado
10	9	Lucca	5	Empleado
11	10	Silva	5	Empleado
12	11	Zambani	5	Empleado
13	13	De Lucía	5	Empleado
14	NULL	NULL	6	Cadete

Ahora, si queremos las categorías que no posean ningún empleado, deberemos hacer lo siguiente:

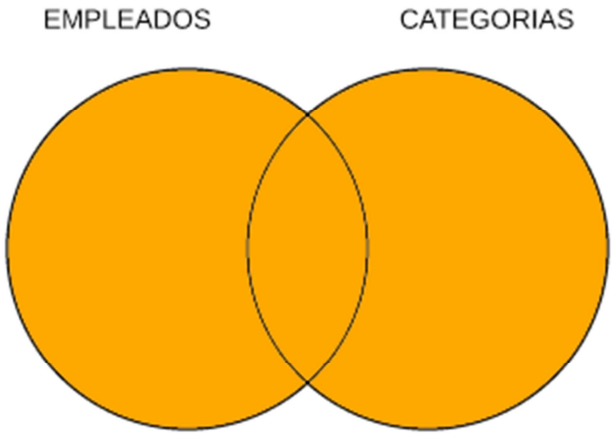


```
SELECT e.Id, e.Apellido, c.Id, c.Nombre FROM Empleados e RIGHT JOIN Categorías
c ON e.Categoria = c.Id WHERE e.Id IS NULL
```

	Id	Apellido	Id	Nombre
1	NULL	NULL	6	Cadete

FULL JOIN

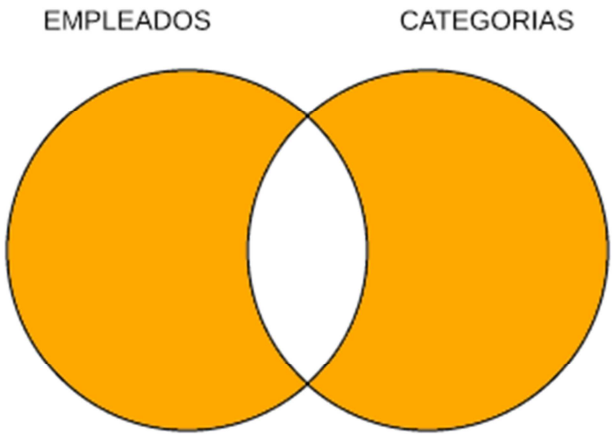
El FULL JOIN se utiliza cuando queremos obtener todas las filas de las dos tablas, sin importarnos que tengan coincidencias. MySQL no soporta este tipo de JOIN, por lo tanto para lograr el mismo resultado, hay que hacer LEFT JOIN + RIGHT JOIN.




```
SELECT e.Id, e.Apellido, c.Id, c.Nombre FROM Empleados e FULL JOIN Categorías
c ON e.Categoria = c.Id
```

	Id	Apellido	Id	Nombre
1	1	González	1	Gerente General
2	2	Pereyra	5	Empleado
3	3	Gutierrez	4	Jefe
4	4	Santo	2	Gerente
5	5	Liberti	5	Empleado
6	6	Pérez	5	Empleado
7	7	Rivarola	5	Empleado
8	8	Andrade	3	Sub Gerente
9	9	Lucca	5	Empleado
10	10	Silva	5	Empleado
11	11	Zambani	5	Empleado
12	12	Di Bari	4	Jefe
13	13	De Lucía	5	Empleado
14	14	Franco	NULL	NULL
15	NULL	NULL	6	Cadete

Para obtener tanto los empleados que no tienen categoría como las categorías que no tienen empleados, podemos hacer uso del FULL JOIN.




```
SELECT e.Id, e.Apellido, c.Id, c.Nombre FROM Empleados e FULL JOIN Categorías
c ON e.Categoria = c.Id WHERE e.Id IS NULL OR c.Id IS NULL
```

	Id	Apellido	Id	Nombre
1	14	Franco	NULL	NULL
2	NULL	NULL	6	Cadete

CROSS JOIN

Finalmente, existe el CROSS JOIN, el cual no puede ser explicado con un diagrama de Venn. Lo que hace, es unir todo con todo, o sea, realiza el producto cartesiano. Como tenemos 14 empleados y 6 categorías (14x6 = 84), obtendremos 84 filas en el resultado. Este tipo de JOIN puede resultar peligroso en el rendimiento de la base de datos, si lo aplicamos a tablas con muchos datos.

Por obvias razones, no adjunto una imagen del resultado.

```
SELECT e.Id, e.Apellido, c.Id, c.Nombre FROM Empleados e CROSS JOIN Categorías
c
```

Resumen

Existen 3 tipos de JOINS:

- INNER
- OUTER (LEFT, RIGHT, FULL)
- CROSS

Material obtenido de http://www.nebaris.com/post/77/tipos-de-join-en-sql