"Cuida tus pensamientos, porque se convertirán en tus palabras. Cuida tus palabras, porque se convertirán en tus actos. Cuida tus actos, porque se convertirán en tus hábitos. Cuida tus hábitos, porque se convertirán en tu destino."

Mahatma Gandhi

SELECCIONAR INFORMCIÓN DE VARIAS TABLAS RELACIONADAS

A través de las sentencias *select* sabemos cómo hacer un particionamiento vertical, un particionamiento horizontal y una combinación de ambos para extraer una sección de los datos de la tabla.

Por ejemplo, dada la tabla *CLIENTES*

SELECT Nombre, Email FROM clientes; -- Particionamiento vertical

Cod_cliente	DNI	Nombre	Email	Total_pedidos	Cantidad_total
c1	21492542P	Bruce	brucebanner@marvel.com	140	143000.00
c2	35198860M	Clark	clarkkent@dc.com	3	1285.78
c3	05698721B	Peter	peterparker@marvel.com	37	34987.17
c4	23055754W	Tony	tonystark@marvel.com	986	8845386.23

*SELECT * FROM clientes WHERE cod_cliente = 0000003 OR cod_cliente = 0000004; -- Particionamiento horizontal*

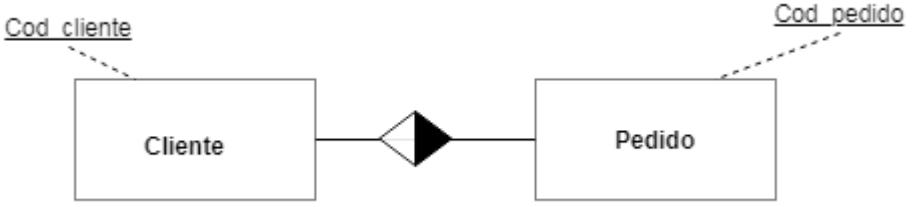
Cod_cliente	DNI	Nombre	Email	Total_pedidos	Cantidad_total
c1	21492542P	Bruce	brucebanner@marvel.com	140	143000.00
c2	35198860M	Clark	clarkkent@dc.com	3	1285.78
c3	05698721B	Peter	peterparker@marvel.com	37	34987.17
c4	23055754W	Tony	tonystark@marvel.com	986	8845386.23

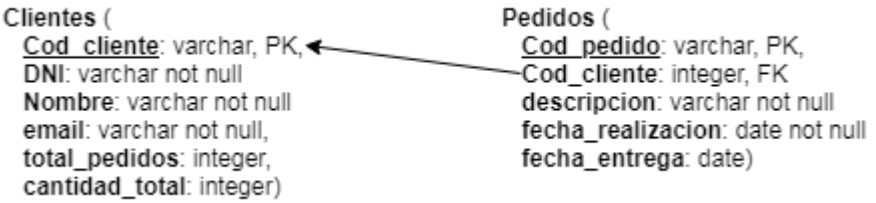
SELECT Total_pedidos, Cantidad_total FROM clientes WHERE (cod_cliente = '0000002' OR cod_cliente = '0000003'); -- particionamiento combinado

Cod_cliente	DNI	Nombre	Email	Total_pedidos	Cantidad_total
c1	21492542P	Bruce	brucebanner@marvel.com	140	143000.00
c2	35198860M	Clark	clarkkent@dc.com	3	1285.78
c3	05698721B	Peter	peterparker@marvel.com	37	34987.17
c4	23055754W	Tony	tonystark@marvel.com	986	8845386.23

Como podemos observar estas sentencias operan sobre una única tabla; pero ¿Qué pasa si queremos relacionar u obtener datos de más de una tabla? Vamos a comenzar estableciendo relaciones de datos entre tablas. Para ello utilizaremos las *Foreign Keys* que establecen relaciones entre los datos de la BD.

Por ejemplo, partiendo del ejemplo anterior, sabemos que los clientes se relacionan con los pedidos que realizan. Por lo tanto, podríamos tener la siguiente situación:

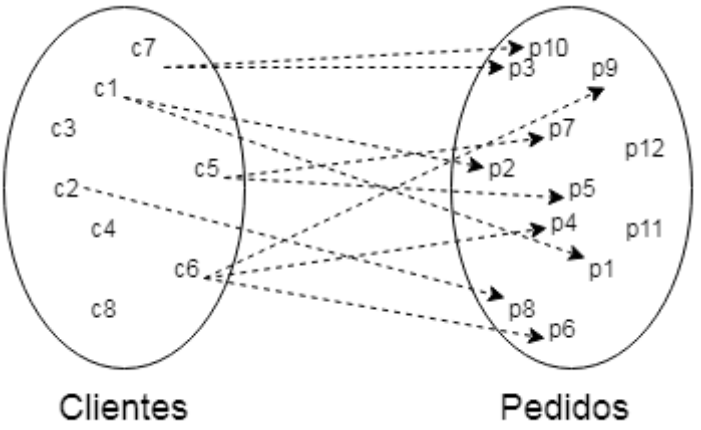




Es decir, vemos que la tabla *Pedido* se relaciona con la tabla *Cliente* a través de una **Foreign Key**, ***Cod_cliente***.

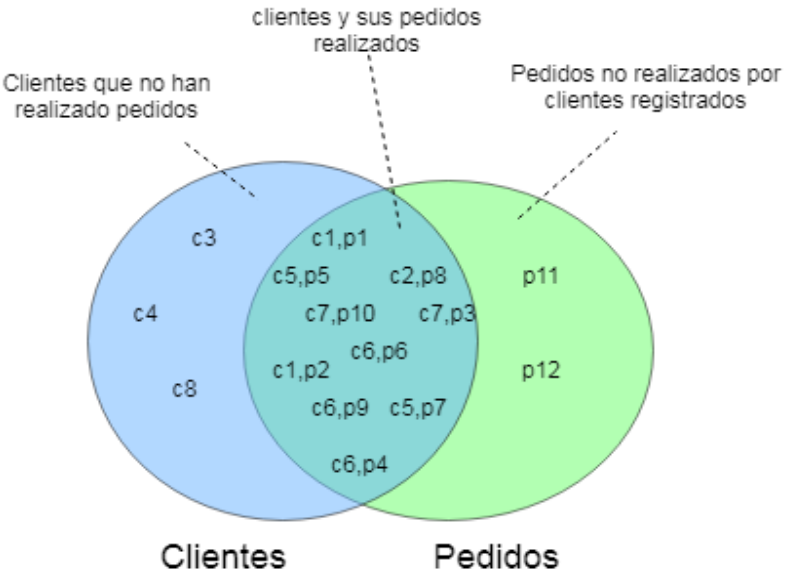
Vamos a usar un poco de Teoría de Conjuntos para mostrar gráficamente todas las tuplas de *Clientes*, así como todas las tuplas de *Pedidos*.

Sean todos los clientes almacenados en la BD: c1, c2, c3, c4, ..., c8.
Sean todos los pedidos almacenados en la BD: p1, p2, p3, p4, …, p10.
Los clientes se relacionan con los pedidos de la siguiente forma:



Observando el diagrama de la figura anterior, vemos que hay clientes que no han realizado pedidos. Otros clientes sí han realizado pedidos y, por último, vemos que hay pedidos que no se han realizado por clientes registrados, los que llamaremos clientes externos.

Empleando una notación matemática (Diagramas de Venn) para expresar las relaciones entre los elementos de los 2 conjuntos:



Las tuplas de la tabla *Clientes* son:

cod_cliente	DNI	Nombre	email	total_pedidos	cantidad_total
c1	21492542P	Bruce	brucebanner@marvel.com	140	143000.0
c2	35198860M	Clark	clarkkent@dc.com	3	1285.78
c3	05698721B	Peter	peterparker@marvel.com	37	34987.2
c4	23055754W	Tony	tonystark@marvel.com	986	8845390.0
c5	09097878K	BruceW	brucewayne@dc.com	5	143.16
c6	17826889A	Flash	barryallen@marvel.com	10	185.65
c7	59470397X	Wonder Woman	dianaprince@marvel.com	370	87.1
c8	54176653S	Thor	Donalblake@marvel.com	98	8845390.0

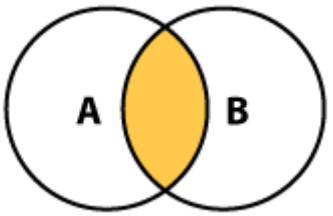
Las tuplas de la tabla *Pedidos* son:

cod_pedido	cod_cliente	descripcion	fecha_realizacion	fecha_entrega
p1	c1	Pedido p1 del cliente c1	2020-01-01	2020-01-10
p10	c7	Pedido p10 del cliente c7	2020-05-01	2020-05-10
p11	(null)	Pedido p11 de un cliente externo	2020-05-15	(null)
p12	(null)	Pedido p12 de un cliente externo	2020-05-15	(null)
p2	c1	Pedido p2 del cliente c1	2020-01-15	2020-01-25
p3	c7	Pedido p3 del cliente c7	2020-01-30	2020-02-10
p4	c6	Pedido p4 del cliente c6	2020-02-15	2020-02-25
p5	c5	Pedido p5 del cliente c5	2020-03-01	2020-03-10
p6	c6	Pedido p6 del cliente c6	2020-03-15	(null)
p7	c5	Pedido p7 del cliente c5	2020-03-30	(null)
p8	c2	Pedido p8 del cliente c2	2020-04-01	2020-04-11
p9	c6	Pedido p9 del cliente c6	2020-04-15	2020-04-25

Cuando se emplea más de una tabla en una query, se está realizando una operación de **join**. Un **join** es un método para relacionar tablas basadas en campos comunes a las tablas que intervienen en la query. Obviamente, los campos comunes deberían ser las **Foreign Keys** de las tablas.

INNER JOIN

Imaginemos, que queremos obtener la información de aquellos clientes que han realizado algún pedido. Es decir, usando la teoría de conjuntos necesitamos obtener la siguiente sección del diagrama:



El join que se debe usar para obtener la intersección de los conjuntos se llama **inner join**. La query patrón para los inner joins es la siguiente:

```
SELECT lista_de_campos
FROM tabla_1
INNER JOIN tabla_2 ON (condicion_del_join);
```

Aplicando el patrón a nuestras tablas:

```
SELECT *
FROM clientes
INNER JOIN pedidos ON (clientes.cod_clientes=pedidos.cod_clientes);
```

Notar que el campo común que comparten las dos tablas *cod_clientes* se llama igual en las dos tablas, por lo que para poder distinguirlos es necesario usar la notación: *nombre_tabla.nombre_campo*.

Podemos acortar la query usando alias para las tablas que intervienen en el join:

```
SELECT *
FROM clientes AS c
INNER JOIN pedidos AS p ON (c.cod_clientes=p.cod_clientes);
```

cod_cliente	DNI	Nombre	email	total_pedidos	cantidad_total	cod_pedido	cod_cliente_1	descripcion	fecha_realizacion	fecha_entrega
c1	21492542P	Bruce	brucebanner@marvel.com	140	143000.0	p1	c1	Pedido p1 del cliente c1	2020-01-01	2020-01-10
c7	59470397X	Wonder Woman	dianaprince@marvel.com	370	87.1	p10	c7	Pedido p10 del cliente c7	2020-05-01	2020-05-10
c1	21492542P	Bruce	brucebanner@marvel.com	140	143000.0	p2	c1	Pedido p2 del cliente c1	2020-01-15	2020-01-25
c7	59470397X	Wonder Woman	dianaprince@marvel.com	370	87.1	p3	c7	Pedido p3 del cliente c7	2020-01-30	2020-02-10
c6	17826889A	Flash	barryallen@marvel.com	10	185.65	p4	c6	Pedido p4 del cliente c6	2020-02-15	2020-02-25
c5	09097878K	BruceW	brucewayne@dc.com	5	143.16	p5	c5	Pedido p5 del cliente c5	2020-03-01	2020-03-10
c6	17826889A	Flash	barryallen@marvel.com	10	185.65	p6	c6	Pedido p6 del cliente c6	2020-03-15	(null)
c5	09097878K	BruceW	brucewayne@dc.com	5	143.16	p7	c5	Pedido p7 del cliente c5	2020-03-30	(null)
c2	35198860M	Clark	clarkkent@dc.com	3	1285.78	p8	c2	Pedido p8 del cliente c2	2020-04-01	2020-04-11
c6	17826889A	Flash	barryallen@marvel.com	10	185.65	p9	c6	Pedido p9 del cliente c6	2020-04-15	2020-04-25

Podemos seleccionar aquella información que pueda ser relevante en un momento dado y descartar la que no sea relevante. Por ejemplo, podríamos querer mostrar el *DNI*, el *nombre* y la *cantidad total* de la tabla *clientes* y, la *descripción* y la *fecha de entrega* de cada pedido de la tabla *pedidos*.

```
SELECT C.DNI, c.nombre, c.cantidad_total, p.descripción, p.fecha_entrega
FROM clientes AS c
INNER JOIN pedidos AS p ON (c.cod_clientes=p.cod_clientes)
ORDER BY c.DNI ASC;
```

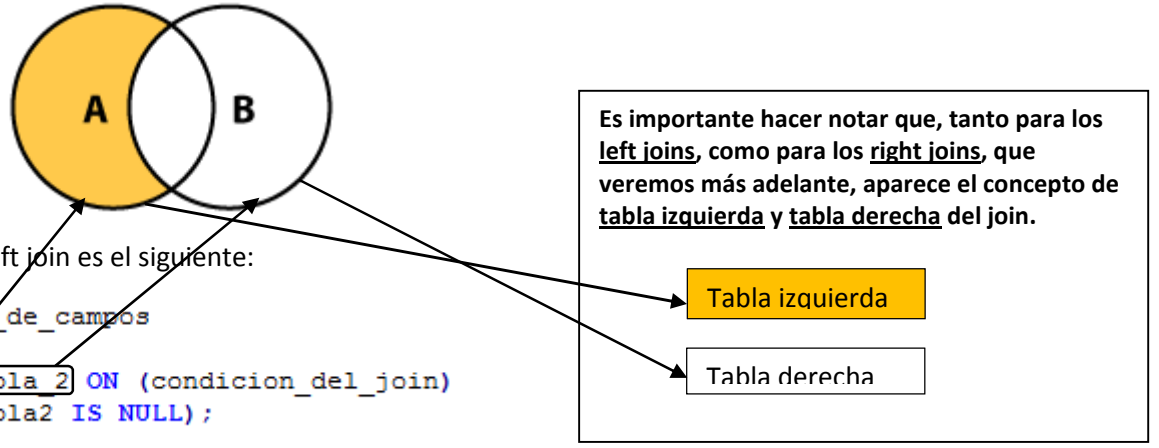
DNI	Nombre	cantidad_total	descripcion	fecha_entrega
09097878K	BruceW	143.16	Pedido p5 del cliente c5	2020-03-10
09097878K	BruceW	143.16	Pedido p7 del cliente c5	(null)
17826889A	Flash	185.65	Pedido p4 del cliente c6	2020-02-25
17826889A	Flash	185.65	Pedido p6 del cliente c6	(null)
17826889A	Flash	185.65	Pedido p9 del cliente c6	2020-04-25
21492542P	Bruce	143000.0	Pedido p1 del cliente c1	2020-01-10
21492542P	Bruce	143000.0	Pedido p2 del cliente c1	2020-01-25
35198860M	Clark	1285.78	Pedido p8 del cliente c2	2020-04-11
59470397X	Wonder Woman	87.1	Pedido p10 del cliente c7	2020-05-10
59470397X	Wonder Woman	87.1	Pedido p3 del cliente c7	2020-02-10

Observa que hemos ordenado los registros obtenidos según el campo DNI de la tabla Clientes.

LEFT JOIN

Ahora pongámonos en la situación de que queremos obtener la información de aquellos clientes que no han realizado ningún pedido. Esta información la obtendremos mediante un **left join**. Describiendo el **left join** en un *diagrama de Venn* tendríamos:

Ilustración 1: Resultado esperado left join



El patrón de un left join es el siguiente:

```
SELECT lista_de_campos
FROM tabla_1
LEFT JOIN tabla_2 ON (condicion_del_join)
WHERE (pk_tabla2 IS NULL);
```

Fíjate que hemos incluido una nueva condición a través de una cláusula **where**. La razón de esta nueva condición es la siguiente. Imaginemos que partimos de la siguiente query (sin la cláusula **where**):

```
SELECT *
FROM clientes AS c
LEFT JOIN pedidos AS p ON (c.cod_clientes=p.cod_clientes);
```

Sentencia 1: Left join incompleto

El resultado que obtenemos es el siguiente:

cod_cliente	DNI	Nombre	email	total_pedidos	cantidad_total	cod_pedido	cod_cliente_1	descripcion	fecha_realizacion	fecha_entrega
c1	21492542P	Bruce	brucebanner@marvel.com	140	143000.0	p1	c1	Pedido p1 del cliente c1	2020-01-01	2020-01-10
c1	21492542P	Bruce	brucebanner@marvel.com	140	143000.0	p2	c1	Pedido p2 del cliente c1	2020-01-15	2020-01-25
c2	35198860M	Clark	clarkkent@dc.com	3	1285.78	p8	c2	Pedido p8 del cliente c2	2020-04-01	2020-04-11
c3	05698721B	Peter	peterparker@marvel.com	37	34987.2	(null)	(null)	(null)	(null)	(null)
c4	23055754W	Tony	tonystark@marvel.com	986	8845390.0	(null)	(null)	(null)	(null)	(null)
c5	09097878K	BruceW	brucewayne@dc.com	5	143.16	p5	c5	Pedido p5 del cliente c5	2020-03-01	2020-03-10
c5	09097878K	BruceW	brucewayne@dc.com	5	143.16	p7	c5	Pedido p7 del cliente c5	2020-03-30	(null)
c6	17826889A	Flash	barryallen@marvel.com	10	185.65	p4	c6	Pedido p4 del cliente c6	2020-02-15	2020-02-25
c6	17826889A	Flash	barryallen@marvel.com	10	185.65	p6	c6	Pedido p6 del cliente c6	2020-03-15	(null)
c6	17826889A	Flash	barryallen@marvel.com	10	185.65	p9	c6	Pedido p9 del cliente c6	2020-04-15	2020-04-25
c7	59470397X	Wonder Woman	dianaprince@marvel.com	370	87.1	p10	c7	Pedido p10 del cliente c7	2020-05-01	2020-05-10
c7	59470397X	Wonder Woman	dianaprince@marvel.com	370	87.1	p3	c7	Pedido p3 del cliente c7	2020-01-30	2020-02-10
c8	54176653S	Thor	Donalblake@marvel.com	98	8845390.0	(null)	(null)	(null)	(null)	(null)

Resultset 1: Left join incompleto

Lo que de una manera gráfica podemos representar como:

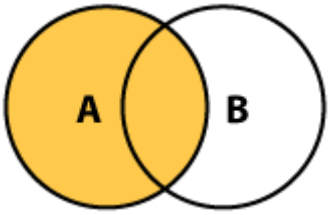


Ilustración 2: Resultado incorrecto left join

Es decir, el *resultset* que obtenemos con esta query es todos los clientes de la tabla **clientes**, hayan realizado o no pedidos.

Ejercicio 1:

*Si la query anterior (Sentencia 1) devuelve todos los clientes de la tabla clientes, ¿Qué diferencia existe con la query, mucho más simple, SELECT * FROM clientes;?*

Si nos fijamos en lo que obtenemos a partir de esta query (*Ilustración 2*) y lo que realmente pretendemos obtener (*Ilustración 1*), nos damos cuenta que necesitamos quitar justo aquellos elementos que caen en la intersección de los 2 conjuntos. Del *Resultset 1*, deducimos que los elementos que queremos obtener tienen como característica el poseer un valor *null* para el campo **cod_pedido** de la tabla **pedidos,** o lo que es lo mismo la **PK de la tabla derecha del join** tiene valor *null*. Es decir, son clientes que no están relacionados con ningún pedido. La query que obtiene los resultados que pretendemos es:

```
SELECT *
FROM clientes AS c
LEFT JOIN pedidos AS p ON (c.cod_clientes=p.cod_clientes)
WHERE (p.cod_pedido IS NULL);
```

El *resulset* después de ejecutar esta query es el siguiente:

cod_cliente	DNI	Nombre	email	total_pedidos	cantidad_total	cod_pedido	cod_cliente_1	descripcion	fecha_realizacion	fecha_entrega
c3	05698721B	Peter	peterparker@marvel.com	37	34987.2	(null)	(null)	(null)	(null)	(null)
c4	23055754W	Tony	tonystark@marvel.com	986	8845390.0	(null)	(null)	(null)	(null)	(null)
c8	54176653S	Thor	Donalblake@marvel.com	98	8845390.0	(null)	(null)	(null)	(null)	(null)

Ejercicio 2:

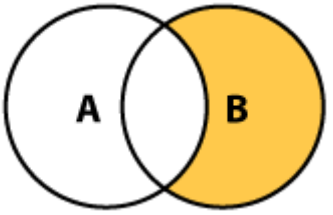
*La explicación teórica se ha realizado tomando como **tabla izquierda del join a clientes** y, como **tabla derecha pedidos**. Compara el resultado obtenido considerando ahora como **tabla izquierda del join a pedidos** y como **tabla derecha a clientes**.*

Ejercicio 3:

*Las tuplas de la query () muetran un valor **null** para todos aquellos campos de la tabla derecha del join; como era de esperar, ya que estamos interesados sólo en aquellos clientes que no han realizado pedido alguno. Modifica la query para obtener sólo los campos de la tabla clientes. ¿Explora en internet si existe alguna notación para referirse a todos los campos de una única tabla de las dos que intervienen en un join?*

RIGHT JOIN

La situación ahora es la de obtener todos aquellos pedidos que no han sido realizados por un cliente del sistema. Es decir, se trata de obtener aquellos pedidos que no están relacionados con ningún cliente de la tabla *clientes*. El tipo de sentencia que nos ayudará a obtener esta información se conoce como *right join*. La representación matemática de un *right join* es la siguiente:



Nota que para este caso la noción de tabla izquierda y derecha del join también es importante ….

Como en los joins anteriores comenzamos mostrando el patrón de la query:

```
SELECT lista_de_campos
FROM tabla_1
RIGHT JOIN tabla_2 ON (condicion_del_join)
WHERE (pk_tabla1 IS NULL);
```

Como en el caso del left join, es necesario añadir una nueva condición a través dela clausula **where**.

Aplicando el patrón a nuestro caso, la query y los resultados obtenidos son los siguientes:

```
SELECT *
FROM clientes AS c
RIGHT JOIN pedidos AS p ON (c.cod_clientes=p.cod_clientes)
WHERE (c.cod_cliente IS NULL);
```

cod_pedido	descripcion	fecha_realizacion	fecha_entrega
p11	Pedido p11 de un cliente externo	2020-05-15	(null)
p12	Pedido p12 de un cliente externo	2020-05-15	(null)

Ejercicio 4:

Utilizando la argumentación teórica empleada para el left join, ¿sabrías explicar por qué es necesario una nueva condición en la cláusula where para el right join?