JOINS

DigitalHouse >
Coding School

¿Por qué usar joins?

Además de hacer consultas dentro de una tabla, a veces es necesario hacer consultas a **distintas tablas**, y unir esos resultados con **JOINS**.

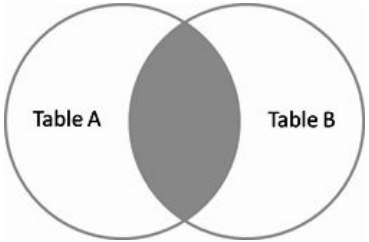
Los **JOINS** dentro de otras cosas**:**

- Son más flexibles.
- Su sintaxis es mucho más utilizada.
- Presentan una mejor performance.

Inner join

El **INNER JOIN** hará una **cruza** entre dos tablas. Si cruzáramos las tablas de **clientes** y **ventas** y hubiese algún cliente **sin ventas**, el INNER JOIN **no traería** a ese cliente como resultado.

INNER JOIN



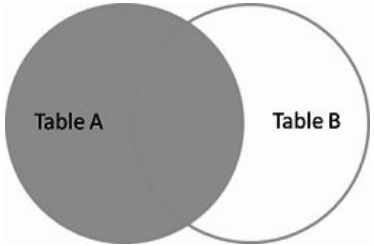
CLIENTES		
id	**nombre**	**apellido**
1	Juan	Perez
2	Clara	Sanchez
3	Marta	García

VENTAS		
id	**cliente_id**	**fecha**
1	2	12/03/2019
2	2	22/08/2019
3	1	04/09/2019

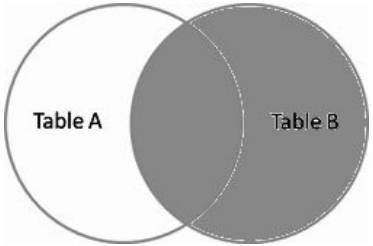
Left join y Right join

Estos tipos de JOINS **no excluyen** resultados de alguna de las dos tablas. Si hubiese clientes **sin ventas,** podríamos incluirlos en el resultado mediante **LEFT** o **RIGHT** JOIN.

LEFT JOIN



RIGHT JOIN



Creando un inner join

Antes escribíamos:

```SQL
SELECT clientes.id AS id, clientes.nombre, ventas.fecha
FROM clientes, ventas
```

Ahora escribiremos:

```SQL
SELECT clientes.id AS id, clientes.nombre, ventas.fecha
FROM clientes
INNER JOIN ventas
```

Si bien ya dimos el primer paso (que es **cruzar** ambas tablas), aún nos falta aclarar **dónde** está ese cruce.

Es decir, qué **clave primaria (PK)** se cruzará con qué **clave foránea (FK)**.

Creando un inner join

La sintaxis del join **no utiliza** el **WHERE**, si no que **requiere** la palabra **ON**. Es ahí en donde indicaremos el **filtro** a tener en cuenta para realizar el cruce.

Es decir, que lo que antes escribíamos en el **WHERE** ahora lo escribiremos en el **ON**.

SQL

```sql
SELECT clientes.id AS id, clientes.nombre, ventas.fecha
FROM clientes
INNER JOIN ventas
ON clientes.id = ventas.cliente_id
```

“

¿Y si quisiéramos **incluir** en el resultado aquellos **clientes** que **NO** tengan **ventas** asociadas**?**

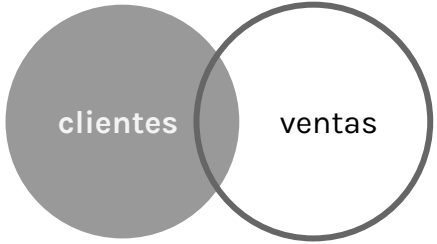
”

Creando un left join

Para incluir aquellos clientes sin ventas basta cambiar **INNER** JOIN por **LEFT** JOIN. El **LEFT JOIN** incluirá **todos** los registros de la primera tabla de la consulta (la tabla **izquierda**) incluso cuando no exista coincidencia con la tabla derecha.

SQL

```sql
SELECT clientes.id AS id, clientes.nombre, ventas.fecha
FROM clientes
LEFT JOIN ventas
ON clientes.id = ventas.cliente_id
```



“

¿Y para **incluir** en el resultado aquellas **ventas** que **NO** tienen **clientes** asociados**?**

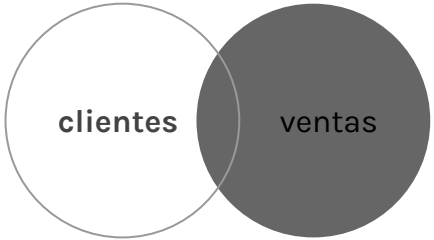
”

Creando un right join

Para incluir aquellas ventas sin clientes basta cambiar **LEFT** JOIN por **RIGHT** JOIN. El **RIGHT JOIN** incluirá **todos** los registros de la tabla **derecha**. Si miramos la query, la tabla ventas aparece posterior a la tabla de clientes... ¡a la derecha!

SQL

```sql
SELECT clientes.id AS id, clientes.nombre, ventas.fecha
FROM clientes
RIGHT JOIN ventas
ON clientes.id = ventas.cliente_id
```



Cruzando muchas tablas

En el siguiente ejemplo se muestra cómo hacer cruces de muchas tablas en una misma consulta usando **joins**:

```sql
SELECT clientes.id AS id, clientes.nombre, ventas.fecha
FROM clientes
INNER JOIN ventas
ON clientes.id = ventas.cliente_id
INNER JOIN productos
ON productos.id = ventas.producto_id
```


DigitalHouse>
Coding School