JOINS

DigitalHouse>

Por que usar joins?

Além de fazer consultas dentro de uma tabela, às vezes é necessário fazer consultas a **tabelas diferentes,** e unir esses resultados com **JOINS.**

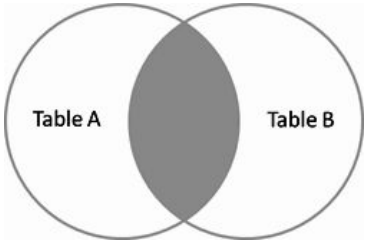
Os **JOINS**, além de outras coisas:

- São mais flexíveis,
- Sua sintaxe é muito mais utilizada,
- Apresentam uma melhor performance.

Inner join

O **INNER JOIN** fará um **cruzamento** entre duas tabelas. Se cruzarmos as tabelas de **clientes** e **vendas** e houver algum cliente **sem vendas**, o INNER JOIN **não traria** esse cliente como resultado.

INNER JOIN



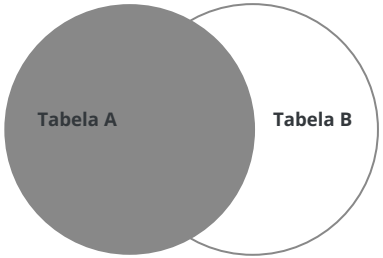
CLIENTES		
id	**nome**	**sobrenome**
1	João	Silva
2	Clara	Sanches
3	Marta	Garcia

VENDAS		
id	**cliente_id**	**data**
1	2	12/03/2019
2	2	22/08/2019
3	1	04/09/2019

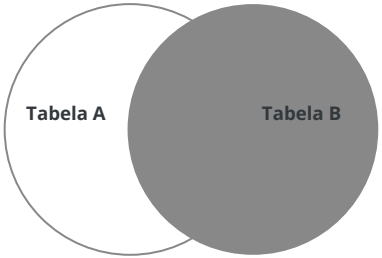
Left join - Right join

Estes tipos de JOINS **não excluem** resultados de alguma das duas tabelas. Se houvesse clientes **sem vendas**, poderíamos incluí-los no resultado usando o **LEFT** ou **RIGHT** JOIN.

LEFT JOIN



RIGHT JOIN



Criando um inner join

Antes *escrevíamos*:

```sql
SELECT clientes.id AS id, clientes.nome, vendas.data
FROM clientes, vendas
```

Agora *escreveremos*:

```sql
SELECT clientes.id AS id, clientes.nome, vendas.data
FROM clientes
INNER JOIN vendas
```

“Embora já tenhamos dado o primeiro passo, que é **cruzar** as duas tabelas, ainda precisamos esclarecer **onde** está esse cruzamento.

Ou seja, qual **chave primária (PK)** irá cruzar com qual **chave estrangeira (FK)**.”

Criando um inner join *(cont.)*

A sintaxe do join **não utiliza** o **WHERE**, mas sim **precisa** da palavra **ON**. É ali onde indicaremos o **filtro** a ser considerado para realizar o cruzamento.

Ou seja, o que antes escrevíamos no **WHERE,** agora escreveremos com **ON**.

SQL

```sql
SELECT clientes.id AS id, clientes.nome, vendas.data
FROM clientes
INNER JOIN vendas
ON clientes.id = vendas.cliente_id
```

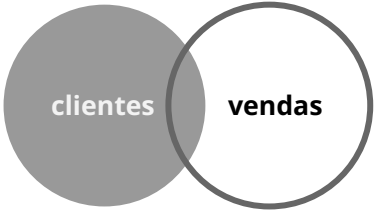
“E se quisermos **incluir** no resultado aqueles **clientes** que **NÃO** tenham **vendas** associadas?”

Criando um left join

Para incluir aqueles clientes sem vendas, basta mudar **INNER** JOIN para **LEFT** JOIN. O **LEFT JOIN** incluirá **todos** os registros da primeira tabela da consulta *(a tabela **esquerda**)*, incluindo quando não exista coincidência com a tabela da direita.

SQL

```sql
SELECT clientes.id AS id, clientes.nome, vendas.data
FROM clientes
LEFT JOIN vendas
ON clientes.id = vendas.cliente_id
```



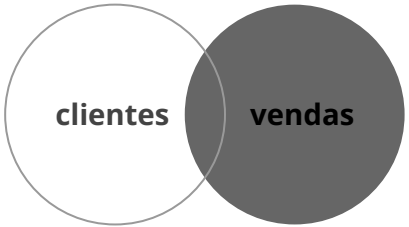
“E para **incluir** no resultado aquelas **vendas** que **NÃO** têm **clientes** associados?”

Criando um right join

Para incluir aquelas vendas sem clientes, basta mudar **LEFT** JOIN por **RIGHT** JOIN. O **RIGHT JOIN** incluirá **todos** os registros da tabela **direita**. *Se observamos a query, a tabela vendas aparece depois da tabela de clientes... à direita!*

SQL

```sql
SELECT clientes.id AS id, clientes.nome, vendas.data
FROM clientes
RIGHT JOIN vendas
ON clientes.id = vendas.cliente_id
```



Cruzando muitas tabelas

No exemplo a seguir, é possível ver como fazer cruzamentos de muitas tabelas em uma mesma consulta usando **joins**:

SQL

```sql
SELECT clientes.id AS id, clientes.nome, vendas.data
FROM clientes
INNER JOIN vendas
ON clientes.id = vendas.cliente_id
INNER JOIN produtos
ON produtos.id = vendas.produto_id
```

DigitalHouse>