Banco de Dados

Junção de Tabelas (Join)

Até agora, utilizamos a cláusula WHERE para fazer filtros de dados e também para fazer junção de tabelas, ou seja, utilizar dados de várias tabelas para obter o resultado desejado baseado nas chaves primárias e estrangeiras que relacionam as tabelas em questão.

Hoje vamos aprender a forma de fazer junção de tabelas através do uso da cláusula JOIN dentro do nosso comando SELECT. Vejamos os diferentes tipos de JOINs e a utilização de cada um deles:

Para melhor explicar os comandos, teremos como base o Banco de Dados de **Fornecimento de produtos**, salve-o e execute-o.

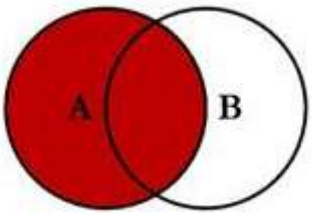
Execute os seguintes comandos SQL para conhecer o conteudo das tabelas criadas:

```
SELECT * FROM join_Produto
SELECT * FROM join_Fornecedor

SELECT * FROM join_Fornecimento Fo, join_Fornecedor F, join_Produto P
   WHERE Fo.idProduto=P.id AND Fo.idFornecedor=F.id
```

Como sabemos, o último SELECT acima nos mostra todos os dados da Tabela Fornecimento relacionando os nomes os dados de Produtos e Fornecedores.

LEFT JOIN

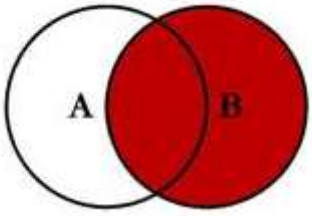


Para o LEFT JOIN, precisamos entender que a tabela mandatória do comando será o conjunto de dados da tabela da ESQUERDA à cláusula JOIN, portanto todos os dados deste conjunto serão exibidos. Se existirem correspondentes na tabela da direita ele será exibito pela junção, mas se não existirem dados correspondentes pelas Pks e Fks, os dados da tabela à esquerda serão exibidos da mesma forma!

Vejamos então a sintaxe de um LEFT JOIN para exibir todos os fornecedores cadastrados juntamente com o código e preço dos produtos que fornece. Mesmo que ele não forneça nenhum produto ele deve ser exibido.

```
SELECT F.nome, Fo.preco, Fo.idproduto FROM join_Fornecedor F
   LEFT JOIN join_Fornecimento Fo ON id = Fo.IdFornecedor
ORDER BY F.Nome
```

RIGHT JOIN



Para o RIGHT JOIN, precisamos entender que a tabela mandatória do comando será a tabela da DIREITA, portanto todos os dados desta tabela serão exibidos. Se existirem correspondentes na tabela da esquerda ele será exibito pela junção, mas se não existires dados correspondentes pelas Pks e Fks, os dados da tabela é direita serão exibidos da mesma forma!

Execute o comando acima apenas trocando o LEFT pelo RIGHT e veja a diferença!

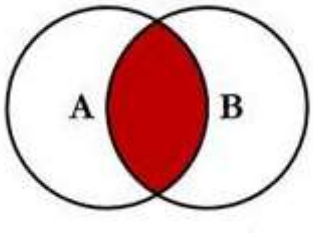
```
SELECT F.nome, Fo.preco, Fo.idproduto FROM join_Fornecedor F
   RIGHT JOIN join_Fornecimento Fo ON id = Fo.IdFornecedor
ORDER BY F.Nome
```

Vejamos então a sintaxe de um RIGHT JOIN para exibir todos os fornecedores cadastrados juntamente com o código e preço dos produtos que fornece. Mesmo que ele não forneça nenhum produto ele deve ser exibido.

```
SELECT nome, preco, idproduto FROM join_Fornecimento Fo
  RIGHT JOIN join_Fornecedor F ON id = Fo.IdFornecedor
ORDER BY F.nome
```

INNER JOIN

O INNER JOIN é o JOIN mais usado de todos, pois ele faz a junção das tabelas com referencia aos dados relacionados exibindo apenas aquelas linhas que tem correspondente em ambas as tabela, funcionando exatamente como o select com a cláusula WHERE ligando as chaves primárias e estrangeiras das tabelas.




```
SELECT nome, preco, idproduto FROM join_Fornecimento Fo
  INNER JOIN join_Fornecedor F ON id = Fo.IdFornecedor
ORDER BY F.Nome
```

Podemos omitir a palavra INNER que o comando funcionará da mesma forma.

```
SELECT nome, preco, idproduto FROM join_Fornecimento Fo
  JOIN join_Fornecedor F ON id = Fo.IdFornecedor
ORDER BY F.Nome
```

FULL JOIN

Já o FULL JOIN vai fazer a junção de todos os dados das duas tabelas mesmo que eles não tenham correspondente em alguma das tabelas. Dificilmente se encontrará utilização para um FULL JOIN se o objetivo é unir tabelas por chaves estrangeiras, porém ele é muito útil quando temos dados de tabelas identicas que contém apenas dados distribuidos.

```
SELECT nome, preco, idproduto FROM join_Fornecimento Fo
  FULL JOIN join_Fornecedor F ON id = Fo.IdFornecedor
ORDER BY F.Nome
```

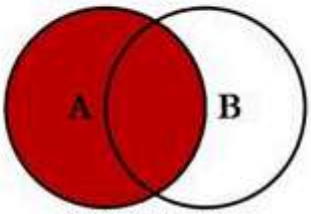
Podemos fazer JOIN de várias tabelas, por exemplo se quisermos saber no nosso banco de Fornecimento, quais os fornecedores e o nome de seus produtos devemos fazer junção das 3 tabelas do banco. Teriamos então ...

```
SELECT F.nome, P.Nome, preco FROM join_Fornecimento Fo
JOIN join_Fornecedor F  ON F.id = Fo.IdFornecedor
JOIN join_Produto P ON P.id = Fo.IdProduto
```

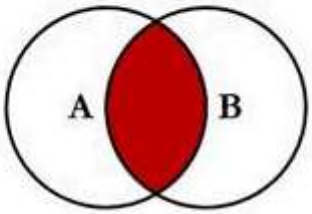
Temos então a sintaxe básica de um comando SELECT com JOIN:

```
SELECT [*|campo]
   FROM Tabela_A
       [INNER | LEFT | RIGHT] JOIN Tabela_B ON relacionamento_de_campos
   WHERE condição_logica
```

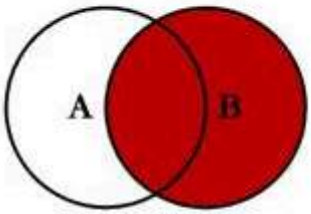
SQL JOINS



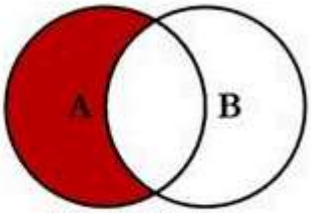

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



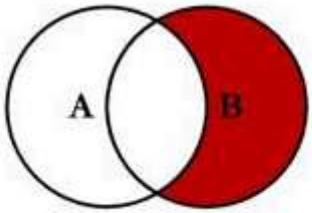

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



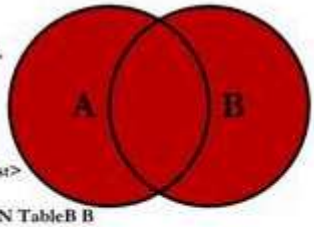

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



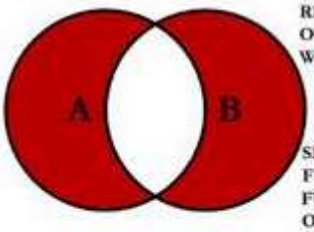

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```