dnc

E-BOOK SQL PARA NOOBS!

Conhecendo o SQL

O que é o SQL e para o que serve?

Antes de começarmos a introduzir este assunto, gostaríamos de esclarecer que esse e-book foi pensado e estruturado para quem está **iniciando no mundo do banco de dados relacional e na linguagem SQL.**

Dito isto, vamos ao primeiro conceito... afinal, o que significa e para o que serve SQL?

SQL significa *Structured Query Language* ou Linguagem de Consulta Estruturada e é uma linguagem de programação utilizada para gerenciar dados em banco de dados relacionais, levando maior segurança e consistência de dados para quem a utiliza. Foi criada para que os dados de uma empresa fossem consultados ou modificados de forma descomplicada e unificada, pois opera através de instruções fáceis, simples, intuitivas e declarativas.

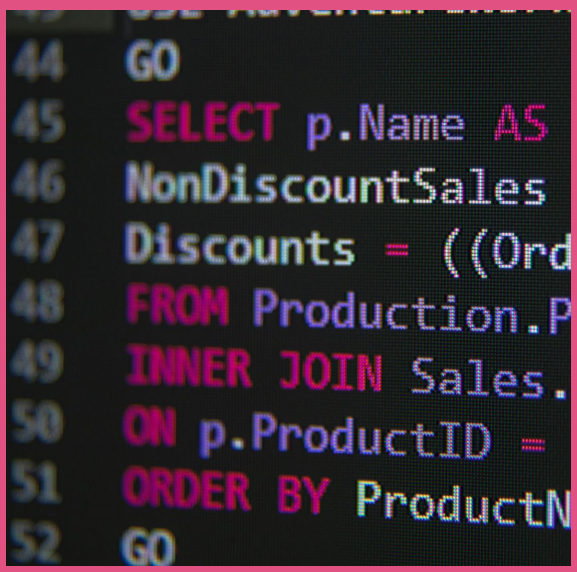
Em resumo, o SQL nos permite "conversar" com o banco de dados.

Conceitos importantes:

Durante o decorrer do e-book algumas palavras serão muito utilizadas, pois estão presentes na execução do SQL, então antes de começarmos aqui estão os significados:

- **Banco de Dados Relacional:** banco de dados que organiza as informações de uma ou mais tabelas. Hoje em dia é essencial para uma empresa registrar toda e qualquer informação, seja no processo de venda, de marketing, durante o serviço ou desenvolvimento de um produto. E esses registros nada mais são do que dados, que, se bem organizados, fornecem informações muito importantes para tomadas de decisões.
- ***Query***: É uma solicitação de informações feita ao banco de dados que retorna uma tabela ou um conjunto delas, figuras, gráficos ou resultados complexos.
- **Tabela:** coleção de dados organizados em linhas e colunas
- **Coluna:** conjunto de valores de dados de um tipo específico
- **Linha:** é um único registro em uma tabela

Primeiros Passos



- **Comandos Select e From**

Agora que já sabemos o que é o SQL, para o que serve e seus principais conceitos, vamos aos comandos que precisamos para extrair dados a partir de um banco de dados, ou seja, vamos aprender como selecionar as colunas que precisam ser consultadas.

A sintaxe básica da instrução de consulta em SQL é:

Comandos que precisamos para extrair dados a partir de um banco de dados.

```
select
*
from Nome da Tabela
```

```
select
Nome da Coluna
from Nome da Tabela
```

Dessa forma, um comando SELECT deve incluir:

- Uma cláusula SELECT que especifica as colunas a serem exibidas, ou seja, o que você vai selecionar.

Obs.: é possível selecionar tudo da tabela consultada usando asterisco (*).

- Uma cláusula FROM que especifica as tabelas que possuem as colunas listadas na cláusula SELECT, ou seja, de onde você vai selecionar.

dnc

Primeiros Passos

Com essa uma linha de código você já pode retornar as informações de um data base relacional.

Já nesse segundo exemplo de comando, os resultados apresentados serão de duas colunas. Basta quando consultados dados específicos de colunas, separar por vírgulas as colunas no comando SELECT.

Basta quando consultados dados específicos de colunas, separar por vírgulas as colunas no comando SELECT.

```
1   select
2   Nome da Coluna,
3   Nome da Coluna II
4   from Nome da Tabela
```

Lembrando: Vírgula apenas entre uma coluna e outra, na última coluna não precisa.

Dica: Sempre abra uma aba só para conseguir ver o banco de dados, veja os tipos de dados que contém na sua tabela, conheça sua base para não correr o risco de puxar informações que não existem.

dnc

Comando Distinct e Where

O famoso “remover duplicatas” do Excel também é possível no SQL.

Em alguns casos, será necessário exibir as informações da tabela consultada de maneira distinta, evitando informações repetidas e redundantes. O famoso “remover duplicatas” do Excel também é possível no SQL, para isso inclua a palavra-chave DISTINCT imediatamente após a palavra SELECT.

A sintaxe do comando DISTINCT é:

```
select distinct
Nome da Coluna
from Nome da Tabela
```

Você pode especificar múltiplas colunas depois da palavra DISTINCT separando elas entre vírgulas. O qualificador DISTINCT afeta todas as colunas selecionadas, e o resultado apresentado é uma combinação distinta das colunas.

Agora que já sabemos selecionar a coluna de uma tabela, é possível filtrar uma tabela e selecionar apenas os dados que interessam. Através da cláusula WHERE conseguimos parametrizar a consulta e filtrar os dados retornados através de uma condição.

A sintaxe do comando WHERE é:

```
select
Nome da Coluna
from Nome da Tabela
where condição
```

Dessa forma, um comando WHERE deve incluir:

- Uma cláusula **SELECT** que especifica as colunas a serem exibidas.
- Uma cláusula **FROM** que especifica as tabelas que possuem as colunas listadas na cláusula SELECT.
- Uma cláusula **WHERE** que especifica uma certa característica definida

dnc

Operadores Lógicos

Para testar a legitimidades das condições WHERE existem **operadores lógicos** que formam as expressões.

Igualdade: este operador realiza operações de igualdade entre dois valores. O sinal utilizado é “=”.

```
select
Nome da Coluna
from Nome da Tabela
where Nome da Coluna = Condição
```

Serão apresentados os dados que são iguais à característica especificada.

Diferença: Para aplicar a operação de desigualdade entre dois valores. O sinal utilizado é “<>”.

```
select
Nome da Coluna
from Nome da Tabela
where Nome da Coluna <> Condição
```

Serão apresentados os dados que diferem da característica especificada.

Maior que: O operador “>” verifica se o valor informado é maior que a condição. Ainda é possível combinar > e = resultando em “>=”, que verifica se o valor é maior ou igual.

```
select
Nome da Coluna
from Nome da Tabela
where Nome da Coluna >= Condição
```

dnc

Operadores Lógicos

Menor que: O operador “<” verifica se o valor informado é menor que a condição. Ainda é possível combinar < e = resultando em “<=”, que verifica se o valor é menor ou igual.

```
select
Nome da Coluna
from Nome da Tabela
where Nome da Coluna <= Condição
```

Between: O operador “Between” recebe um valor mínimo e máximo e retorna os dados que atendem ao critério de intervalo entre esses dois valores.

```
select
Nome da Coluna
from Nome da Tabela
where Condição_1 between Condição_2
```

Dica: Podemos usar o operador NOT antes do BETWEEN - na forma NOT BETWEEN - para retornar os registros que não atendam à condição estabelecida.

Operadores Lógicos

Like: O operador “Like” através do uso do sinal “%” para auxiliar na filtragem permite selecionar valores parecidos.

```
select
Nome da Coluna
from Nome da Tabela
where Nome da Coluna like "1%"
```

Nesse caso o “%” veio depois do valor 1, ou seja, o resultado mostrará todos os campos da coluna que possuem valores que começam com 1 não importando o que vem depois, por exemplo: 11, 123,13.

```
select
Nome da Coluna
from Nome da Tabela
where Nome da Coluna like "%1"
```

Já neste exemplo observamos que se o “%” vem antes do valor, irá retornar tudo que termina com aquele determinado valor. Nesse caso iríamos obter, por exemplo os valores 11, 231,451.

O sinal “%” para auxiliar na filtragem permite selecionar valores parecidos.

Dica: Essa cláusula também é utilizada com textos, colocando entre “%texto%” a parte que contém na palavra, seja início, meio ou final. Além disso, podemos usar o operador NOT antes do LIKE - na forma NOT LIKE - para retornar os registros que não atendam à condição estabelecida.

dnc

É possível criar uma lista de valores que será comparada com os campos da coluna.

Operadores Lógicos

In: Através do operador "In" é possível criar uma lista de valores que será comparada com os campos da coluna. Essa lista pode conter números ou palavras.

```
select
Nome da Coluna
from Nome da Tabela
where Nome da Coluna in nº1,nº2,nº3
```

Com o "in" selecionamos os resultados que conferem com a lista apresentada, que, no caso, seriam todos os campos da coluna selecionada que possuem o valor igual aos números indicados.

Is Null: A condição "is null" quer dizer nulo, ou seja, um campo vazio. Retorna TRUE se um valor NULL for encontrado, caso contrário, retorna FALSE.

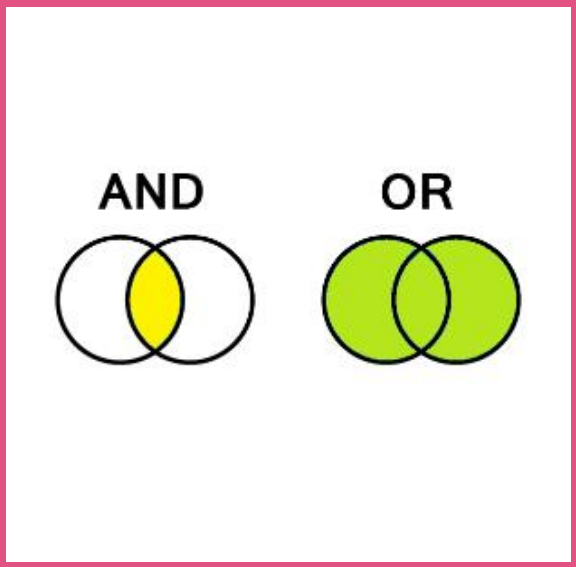
```
select
Nome da Coluna
from Nome da Tabela
where Nome da Coluna is null
```

Nesse caso aparecem somente os valores nulos dessa coluna.

Dica: Podemos usar o operador NOT antes do Null - na forma IS NOT NULL - para retornar os registros que não atendam à condição estabelecida.

dnc

Condições Lógicas



Existem casos que precisamos incluir mais de uma condição na cláusula WHERE, neste caso existem os operadores **AND e OR.**

AND: Apresenta uma condição de inclusão, aplicando a condição lógica E. Observe este exemplo utilizando os conceitos aprendidos anteriormente:

```
select
Nome da Coluna
from Nome da Tabela
where Nome da Coluna_1 is not null
and Nome da Coluna_2 > condição
```

Ou seja, os resultados mostrados serão os campos que possuem valores atribuídos na primeira coluna, mas que também são maiores que 10 na segunda coluna selecionada.

OR: Apresenta uma condição de opção, aplicando a condição lógica OU. Em resumo, os resultados exibidos devem obedecer OU uma condição, OU outra.

```
select
Nome da Coluna
from Nome da Tabela
where Nome da Coluna_1 is not null
or Nome da Coluna_2 > condição
```

Ou seja, os resultados mostrados serão os campos que possuem valores atribuídos na primeira coluna ou que são maiores que 10 na segunda coluna. Caso seja maior que 10 na segunda coluna selecionada e é nulo na primeira coluna também será mostrado.

dnc

Funções de Agregação

Até aqui já percebemos que uma tabela de dados pode conter muitos registros e existem casos que são necessárias apenas informações resumidas de uma determinada tarefa. **Através das funções de agregação é possível processá-las no sistema e exibir apenas o valor desejado.** São elas:

MAX: Valor máximo de conjunto de valores

MIN: Valor mínimo de conjunto de valores

AVG: Média aritmética de um de conjunto de valores

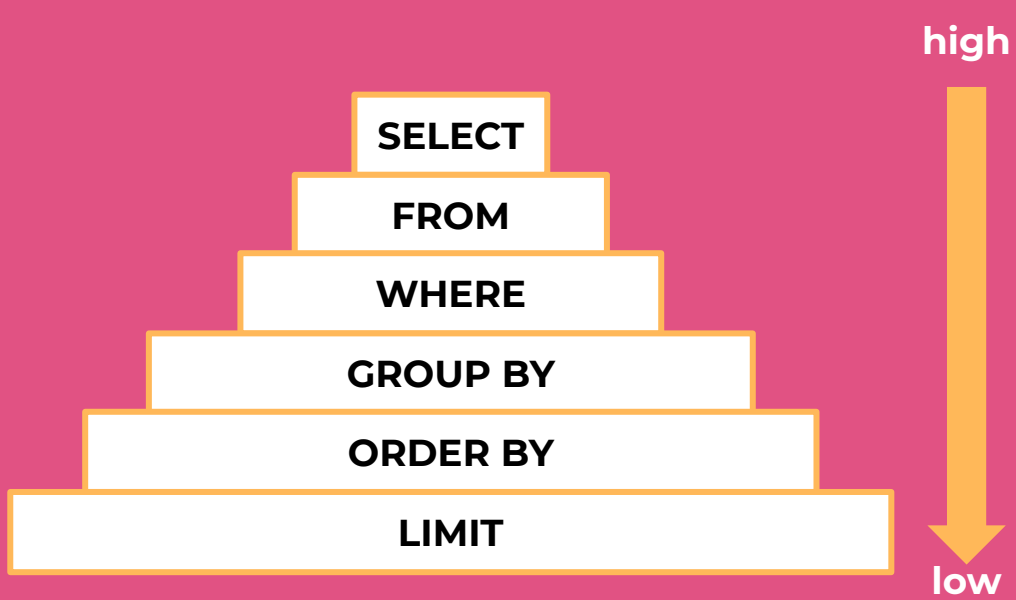
COUNT: Contar quantidade total de itens

SUM: Soma de um conjunto de valores

dnc

dnc

Order By x Group By



ORDER BY

ORDER BY é utilizado para ordenar o conjunto de resultados de registro. Ao utilizarmos o SELECT, podemos utilizar a cláusula ORDER BY para mostrar esse resultado ordenado a partir de uma das colunas em ordem ascendente ou descendente (ordem inversa).

A sintaxe do comando ORDER BY é:

Ascendente (crescente):

```
select
Nome da Coluna
from Nome da Tabela
order by Coluna_que_sera_ordenada
```

Descendente (decrescente):

```
select
Nome da Coluna
from Nome da Tabela
order by Coluna_que_sera_ordenada DESC
```

GROUP BY

GROUP BY divide o resultado da nossa pesquisa em grupos, ou seja, para cada grupo que aplicarmos essa função de agregação você precisar agrupar os dados, é aí que entra a cláusula GROUP BY.

A sintaxe do comando GROUP BY é:

```
select
Nome da Coluna
from Nome da Tabela
group by Nome da Coluna
```

O resultado retornado são todas as linhas do banco de dados agrupadas conforme as condições dadas na query e a deixa compactadas em apenas um resultado.

JOIN'S

O que é possível, por exemplo, no Excel utilizando "procv", no SQL utilizamos a cláusula JOIN.

Muitas vezes só uma tabela não nos dá informações suficientes para o resultado que queremos. Para utilizarmos informações conjuntas de tabelas diferentes, o que é possível, por exemplo, no Excel utilizando "procv", no SQL utilizamos a cláusula JOIN.

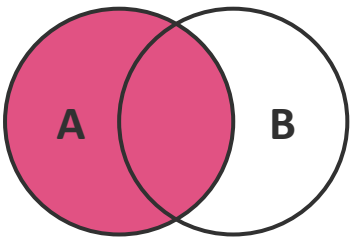
```
select
*
from Nome da Tabela_1
join Nome da Tabela_2 ON Nome da Tabela_1.Nome da coluna = Nome da Tabela_2.Nome da coluna
```

Ou seja, o resultado mostrado será o que queremos tirar da tabela 1 e juntar com a tabela 2 onde a coluna da tabela 1 é igual à coluna da tabela 2.

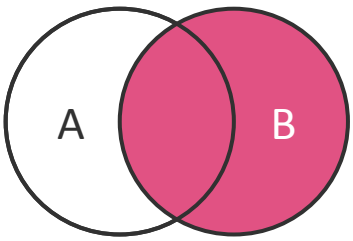
Existem variantes do JOIN e para isso é uma boa prática sempre consultar a representação gráfica, baseada na Teoria dos Conjuntos, que contém todas as sintaxes.

dnc

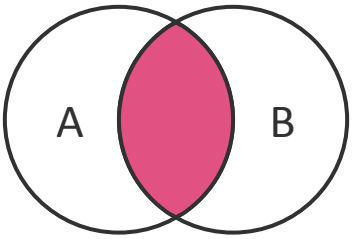
Representação gráfica dos Joins



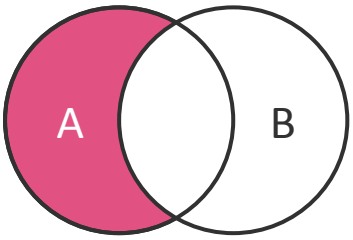

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key=B.Key
```



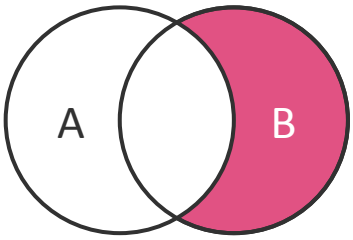

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key=B.Key
```



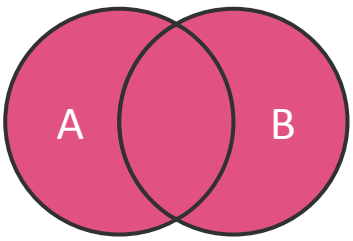

```
SELECT <select_list>
FROM TableA A
INNER JOINT TableB B
ON A.Key=B.Key
```



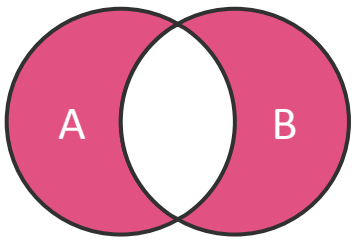

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key=B.Key
WHERE B.Key IS NULL
```




```
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
```

dnc

Conclusão

O mercado de trabalho exige cada vez mais conhecimento em análise de dados e o objetivo deste ebook foi mostrar as principais funções relacionadas ao comando SELECT no SQL para você ser capaz de começar a extrair informações de diferentes bases para relacionar e criar análises estratégicas.

Lembre-se de sempre preparar e organizar os dados, armazená-los em um banco relacional para por fim fazer as perguntas certas ao banco de dados, filtrar elementos, somar valores e realizar cruzamentos, de forma a explorar o conteúdo e extrair informações úteis.

Fontes

- ttps://www.youtube.com/watch?v=UUVO-ancaDE
- https://rockcontent.com/br/blog/query/
- https://support.microsoft.com/pt-br/office/acesso-sql-conceitos-b%C3%A1sicos-vocabul%C3%A1rio-e-sintaxe-444d0303-cde1-424e-9a74-e8dc3e460671
- https://www.devmedia.com.br/sql-clausula-where/37645#sintaxe
- https://www.techonthenet.com/sql/is_null.php

dnc

dnc

E AÍ, GOSTOU?

Agora você já sabe um pouco mais sobre o SQL, esperamos que o material tenha sido útil para você!

#HARDWORK

Confira também nossos outros materiais gratuitos disponíveis!

Acesse: https://conteudo.dnc.group/vitrine-materiais-gratuitos