SELECT, WHERE, operadores, DISTINCT, ORDER BY, Agregação

Prof. Dr. Nazareno de Oliveira Pacheco
nazareno.pacheco@prof.sc.senac.br

Exibindo registros

- Até agora, nós só inserimos registros
- Uma das funcionalidades mais importantes de um banco de dados é a capacidade de retornar registros
- Para isto podemos utilizar o comando SELECT
  - O SELECT é provavelmente o comando mais importante de um banco de dados

SELECT-FROM

- A sintaxe básica de um SELECT é a seguinte:

```
SELECT
    nome_col1,
    nome_col2,
    ...
FROM nome_tabela
```

Experimente

```
CREATE TABLE funcionario (
   codigo   SERIAL NOT NULL,
   nome     VARCHAR(100) NOT NULL,
   funcao   VARCHAR(50) NOT NULL,
   salario  NUMERIC(8,2) NOT NULL,
   PRIMARY KEY (codigo)
);

INSERT INTO funcionario (nome, funcao, salario)
VALUES ('Charles Wust', 'Presidente', 25000);

INSERT INTO funcionario (nome, funcao, salario)
VALUES ('Alessandra Anacleto Wust', 'Diretora', 22000);

INSERT INTO funcionario (nome, funcao, salario)
VALUES ('Leonardo Anacleto Wust', 'Estagiário(a)', 5000);

INSERT INTO funcionario (nome, funcao, salario)
VALUES ('Aline Anacleto', 'Estagiário(a)', 400);
```

Experimente

```
SELECT
    nome,
    salario
FROM funcionario;

SELECT
    nome
FROM funcionario;

SELECT
    *
FROM funcionario;
```

SELECT-FROM

- Este comando exibirá os valores das colunas informadas de todos os registros da tabela informada
- Se quisermos exibir todas as colunas da tabela informada, podemos utilizar o símbolo *
  - A utilização do * é apropriada apenas para testes pessoais
  - Qualquer aplicação (em Java, p. ex.) que consulte uma tabela deve informar sempre o nome de cada coluna desejada

Filtrando registros

- O comando anterior mostra todas as linhas da tabela
- Porém, em muitos casos só queremos exibir registros específicos
  - Queremos exibir apenas o regi
- Nestes casos, podemos utilizar a cláusula WHERE

SELECT-FROM-WHERE

```
SELECT
   nome_col1,
   ...,
   ...
FROM nome_tabela
WHERE
   CONDICAO
```

- Este comando vai retornar apenas os registros da tabela informada que atendam a condição

Operadores relacionais

- Normalmente, as condições são definidas em expressões que utilizam operadores relacionais
- Os operadores relacionais do SQL são os seguintes:

=	Igual
<> ou !=	Diferente*
>	Maior que
>=	Maior ou igual a
<	Menor que
<=	Menor ou igual a

* Nem todos os SGBD's aceitam !=

Experimente

```
SELECT
   codigo,
   nome,
   salario
FROM funcionario
WHERE
   salario > 20000.00;
--retorna todos os funcionarios que ganham mais de 20000

SELECT
   nome
FROM funcionario
WHERE
   codigo = 3
--retorna o nome do funcionario de codigo = 3
```

Combinando condições

- É possível combinar múltiplas condições utilizando operadores lógicos
- Os operadores lógicos são:
  - AND (e)
  - OR (ou)
  - NOT

Experimente

```
SELECT
    codigo,
    nome,
    salario
FROM funcionario
WHERE
    funcao = 'Estagiário(a)' AND
    salario > 2000
--retorna todos os estagiarios privilegiados demais;
```

Exercícios

Para os exercícios desta aula, crie uma base de dados e rode este script que foi enviado separadamente

script-19-07.sql

Exercícios (10)

1 – Crie um SELECT que retorne o nome de todos os produtos do fabricante Motorola
2 – Crie um SELECT que retorne o nome e preço de todas as câmeras da Nikon
3 – Crie um SELECT que retorne todos os tablets na faixa de preço entre 100 e 150
4 – Crie um SELECT que retorne o nome de todos os produtos da Sony com preço acima de 500
5 – Crie um SELECT que retorne todos os dados dos celulares com preço abaixo de 100
6 – Crie um SELECT que retorne o nome e preço de todos os notebooks de HP ou Sony com preço entre 500 e 800

Outros operadores

- O SQL possui ainda um conjunto de operadores adicionais úteis em nossas comparações:
- BETWEEN
  - Verifica se um valor está dentro de um intervalo
  - Exemplo:
    - preco BETWEEN 100 AND 200
  - O exemplo acima é equivalente a:
    - (preco >= 100 AND preco <=200)

Outros operadores

- LIKE
  - Operador poderoso para comparação de textos.
  - Permite realizar comparação utilizando curingas (*wildcards*), representados pelo caractere %
  - Exemplo de uso (1):
    - nome LIKE ‘tablet%’
    - Este exemplo vai retornar todos os nomes que começam com a palavra tablet

Outros operadores

- LIKE
  - Exemplo de uso (2):
    - nome LIKE ‘%silver’
    - Este exemplo vai retornar todos os nomes que terminam com a palavra silver
  - Exemplo de uso (3):
    - nome LIKE ‘%portable%’
    - Este exemplo vai retornar todos os nomes que contenham a palavra portable no meio

Outros operadores

- NOT LIKE
  - Similar ao like, mas verifica se o texto não bate com o padrão informado
  - Exemplo de uso (3):
    - nome NOT LIKE ‘%portable%’
    - Este exemplo vai retornar todos os nomes que NÃO contenham a palavra portable no meio

Outros operadores

- O SQL possui ainda um conjunto de operadores adicionais úteis em nossas comparações:
- BETWEEN
  - Verifica se um valor esta dentro de um intervalo
  - Exemplo:
    - preco BETWEEN 100 AND 200
  - O exemplo acima é equivalente a:
    - (preco >= 100 AND preco <=200)

Outros operadores

- IN
  - Este operador verifica se um determinado valor está dentro de uma lista de outros valores
  - Exemplo de uso
    - fabricante IN (‘Sony’, ‘Motorola’, ‘Asus’)
    - A comparação acima é equivalente a:
    - fabricante = ‘Sony’ OR fabricante = ‘Motorola’ OR fabricante = ‘Asus’
  - Também é possível utilizar o IN para ver se um valor não está no conjunto de valores retornados por um outro comando SQL (subselect) (será visto mais tarde)

Verificando nulos

- Valores nulos tem uma característica bastante peculiar:
  - Pergunta: null = null ?
  - Resposta: FALSO!
- Porém, muitas vezes precisamos localizar registros em nossas tabelas que tenham determinada coluna contendo o valor null
- Neste caso, é necessário utilizar o comparador IS NULL

Verificando nulos

- Valores nulos tem uma característica bastante peculiar:
  - Pergunta: null = null ?
  - Resposta: FALSO!
- Porém, muitas vezes precisamos localizar registros em nossas tabelas que tenham determinada coluna contendo o valor null
- Neste caso, é necessário utilizar o comparador IS NULL
- Exemplo de uso:
  - fabricante IS NULL
  - Retorna todos os registros cujo o campo fabricante não tenha sido informado

Exercícios (11)

1 – Crie um SELECT que retorne o nome, categoria, fabricante e preço de todos os produtos dos fabricantes Coby, Genesis, Samsung e Orange

2 – Crie um SELECT que retorne o nome e preço de todas as câmeras da Nikon, Canon e Sony na faixa de preço entre 100 e 250

3 – Crie um SELECT que retorne o código e nome de todos os tablets que contenham a palavra vermelho no meio

4 – Crie um SELECT que retorne o nome, categoria e preço de todos os produtos da Sony, Panasonic ou LG que terminem com a palavra preto que estejam na faixa entre 100 e 300

5 – Crie um SELECT que retorne os dados de todos os produtos que não tenham o fabricante informado

Valores distintos

- Às vezes, nossos SELECT podem retornar registros idênticos
- Experimente:
  - ```
    SELECT fabricante
    FROM produto
    WHERE categoria = 'Câmera';
    ```
    ```

SELECT DISTINCT

- Nestes casos, é possível utilizar o comando SELECT DISTINCT
  - O comando SELECT DISTINCT exibe apenas resultados não repetidos
- Experimente:
  - ```
    SELECT DISTINCT fabricante
    FROM produto
    WHERE categoria = 'Câmera';
    ```
    ```

SELECT DISTINCT

- **Atenção**: o SELECT DISTINCT só filtra registros se os valores de todas as colunas forem idênticos.
- Se apenas uma coluna tenha valor diferente, o SELECT DISTINCT exibirá ambos os valores
- Experimente:
  - ```
    SELECT DISTINCT fabricante, categoria
    FROM produto
    WHERE categoria = 'Câmera';
    ```
  - ```
    SELECT DISTINCT fabricante, categoria, preco
    FROM produto
    WHERE categoria = 'Câmera';
    ```

Exercícios (12)

1 – Crie um SELECT que retorne o nome de todos as categorias de produtos produzidas pela Motorola

2 – Crie um SELECT que retorne o nome de todos os fabricantes de tablets

Ordenando registros

- Até agora, a ordem dos resultados de nossos SELECTs foi imprevisível
  - Quando não tomamos cuidados específicos, não há nenhuma garantia de que os registros venham ordenados na mesma ordem que foram cadastrados
- É possível definir a ordem de retorno de nossos registros
- Para isso, é necessário adicionar a cláusula ORDER BY em nossos SELECTs

ORDER BY

- A sintaxe de um comando SELECT com ORDER BY é a seguinte:

```
SELECT
   nome_col1,
   nome_col2,
   ...
FROM nome_tabela
WHERE
  condicoes
ORDER BY
   primeira coluna a ordenar,
   segunda coluna a ordenar,
   ...
```

ORDER BY

- O ORDER BY vai ordenar os resultados em ordem crescente das colunas informadas
- Exemplo:
  - ```
    SELECT
        codigo,
        nome,
        preco
    FROM produto
    ORDER BY
        preco
    ```
- Este SQL retorna os produtos em ordem crescente de preço
    ```

Múltiplas colunas no ORDER BY

- É possível informar diversas colunas no ORDER BY
- Quando isto acontece, os resultados são ordenados primeiro pela primeira coluna informada no ORDER BY, e depois pelas demais colunas, em sequência

Múltiplas colunas no ORDER BY

- Exemplos:
  - ```
    SELECT DISTINCT
       fabricante,
       categoria
    FROM produto
    ORDER BY
       fabricante,
       categoria
    ```
- Ordena primeiro por fabricante (em ordem alfabética), depois por categoria
    ```

Múltiplas colunas no ORDER BY

- Exemplos:
  - ```
    SELECT DISTINCT
       categoria,
       fabricante
    FROM produto
    ORDER BY
       categoria,
       fabricante
    ```
- Ordena primeiro por categoria, e depois por fabricante
    ```

Invertendo a ordem

- Às vezes, queremos ordenar de forma inversa
  - Por exemplo: primeiro os produtos mais caros
- Nestes casos, podemos utilizar a palavra DESC após nossa coluna de ordenação
  - O DESC faz com que a ordenação seja feita de forma descendente (primeiro o maior, depois o menor)
- Também é possível utilizar a palavra ASC depois de uma coluna de ordenação
  - Nestes casos, a ordenação será feita em ordem ascendente (primeiro o menor, depois o maior
  - É desnecessário informar ASC, pois ele já é o comportamento padrão

Invertendo a ordem

- Exemplos:
  - ```
    SELECT
       nome,
       fabricante,
       preco
    FROM produto
    WHERE
       categoria = 'Notebook'
    ORDER BY
       preco DESC,
       fabricante ASC
    ```
- Retorna todos os notebooks, do mais caro ao mais barato. Se o preço é igual, ordena por fabricante alfabeticamente
    ```

Exercícios (13)

1 – Faça um SELECT que retorne o nome e preço de todas as câmeras da Nikon em ordem alfabética

2 – Faça um SELECT que retorne o nome, fabricante e preço de todos os todos os celulares, ordenados por preço e depois por fabricante

3 – Faça um SELECT que retorne todos os tablets da Samsung, do mais caro ao mais barato

Agregando valores

- Suponha que você precise responder à seguinte pergunta:
  - Qual é a média salarial dos empregados?
- Uma resposta como esse depende de combinar diversos valores da nossa base de dados para gerar um único resultado
  - Em outras palavras, precisamos *agregar* estes valores

Funções de agregação

- Nestes casos, podemos utilizar funções de agregação
- Uma função de agregação combina todos os valores de uma determinada coluna para gerar um novo valor

Um exemplo de função de agregação

- AVG(coluna)
  - Retorna a média dos valores não nulos

```
SELECT AVG(preco)
FROM produto
WHERE categoria='Tablet';
--retorna o preço médio de um tablet
```

Funções de agregação

- Note que, quando uma função agregadora é utilizada, é retornado apenas uma linha no resultado
  - Esta linha agrega todos os valores da coluna informada em um único valor
- Por esta razão, não é possível combinar funções de agregação com valores de outras colunas

Funções de agregação

- Note que, quando uma função agregadora é utilizada, é retornado apenas uma linha no resultado
  - Esta linha agrega todos os valores da coluna informada em um único valor
- Por esta razão, não é possível combinar funções de agregação com valores de outras colunas
  - Entretanto, é possível combinar várias funções de agregação em um mesmo SELECT

Experimente

```
SELECT AVG(preco), nome
FROM produto
WHERE categoria='Tablet';
--ERRO!
```

- Se o resultado só mostra uma linha, qual nome de produto o SQL deveria exibir???

Experimente

```
SELECT
    AVG(preco),
    MAX(preco),
    MIN(preco)
FROM produto
WHERE categoria='Tablet';
```

- Retorna o preço médio, o preço máximo e o preço mínimo de um tablet

Outras funções de agregação

- Existem diversas funções de agregação disponíveis
- As mais comuns são:
  - COUNT(): retorna a quantidade de valores não nulos
  - MAX(): retorna o maior valor
  - MIN(): retorna o menor valor
  - AVG(): retorna a média dos valores
  - SUM(): retorna o somatório dos valores

Agregando por grupos

- Porém, sabemos que não é possível combinar uma coluna (categoria) com uma função de agregação diretamente
- Nestes casos, precisamos informar ao banco de dados que queremos agrupar nossas médias pelo campo categoria
- Isto se consegue utilizando a cláusula GROUP BY

Agregando por grupos

- Tente responder à seguinte pergunta:
  - Qual é o valor médio de cada categoria de produto
- Neste caso, uma primeira tentativa poderia ser algo do estilo:

```
SELECT categoria, AVG(preco)
FROM produto
```

GROUP BY

- Para retornar o preço médio de cada categoria, podemos utilizar o seguinte SELECT:

```
SELECT categoria, AVG(preco)
FROM produto
GROUP BY
    categoria
```

- Este SELECT retornará a média de cada uma das categorias disponíveis

Agregação

- É possível informar mais de uma coluna em um GROUP BY
  - Neste caso a agregação é feita pelas diversas colunas informadas

```
SELECT
   categoria,
   fabricante,
   AVG(preco)
FROM produto
GROUP BY
   categoria,
   fabricante;
--retorna a média de preço de cada categoria por cada fabricante
```

Agregação

- Na listagem de colunas a ser exibidas em um SELECT que utilize agregação, só é permitido utilizar:
  - Funções de agregação,
  - Colunas utilizadas na cláusula GROUP BY

Combinando tudo

- A sintaxe final de um SELECT é a seguinte:

```
SELECT
    nome_col1,
    nome_col2...
FROM nome_tabela
WHERE condicoes
GROUP BY colunas
ORDER BY colunas
```

Exercícios (14)

1 – Faça um SELECT a quantidade de produtos da fabricante Sony por categoria
2 – Faça um SELECT que retorne o preço médio dos celulares da Motorola
3 – Faça um SELECT que retorne o preço do produto mais barato de cada categoria
4 – Faça um SELECT que retorne o preço do produto mais caro de cada fabricante
5 – Faça um SELECT que retorne a quantidade de produtos por categoria e fabricante. Ordene por categoria e depois por fabricante
6 – Faça um SELECT que o preço médio de todos os tablets vermelhos
7 – Faça um SELECT que retorne o preço do menor celular com valor acima de $100