

---

# SQL - Structured Query Language

## Unidade 3 – Manipulação Avançada de Dados – Parte 2

Prof. Daniel Callegari  
Escola Politécnica – PUCRS

---

### 1. Introdução

Nessa parte, aprenderemos sobre funções sobre linhas, funções sobre conjuntos de linhas, agregações e agrupamentos.

### 2. Funções

Em SQL podemos aplicar dois tipos de funções sobre linhas de uma tabela:

- Funções sobre linhas, as quais operam sobre cada linha do resultado individualmente; e
- Funções sobre conjuntos de linhas, que operam sobre diversas linhas, calculando valores sobre todo o conjunto (para determinar totais, médias, o maior valor, entre outras possibilidades).

As funções numéricas mais comuns são:

ABS(n), ACOS(n), ASIN(n), ATAN(n), ATAN2(n), CEIL(n), COS(n), COSH(n), EXP(n), FLOOR(n), LN(n), LOG(n), MOD(n,m), POWER(n,m), ROUND(n,m), SIGN(n), SIN(n), SINH(n), SQRT(n), TAN(n), TANH(n), TRUNC(n,m)

As funções sobre caracteres mais comuns são:

LOWER(s), UPPER(s), INITCAP(s), LTRIM(s1,s2), RTRIM(s1,s2), CONCAT(s1,s2), LPAD(s1,n,s2), RPAD(s1,n,s2), LENGTH(s), SUBSTR(s,n,m), REPLACE(s1,s2,s3), CHR(n), SOUNDEX(s), TRANSLATE(s1,s2,s3).

Nota 1: Mais uma vez, estes exemplos são do SGBD Oracle. Procure na documentação do seu SGBD as funções embutidas disponíveis.

Nota 2: Não veremos em detalhe cada uma das funções, mas você pode pesquisar a documentação da linguagem SQL para obter mais informações.

### 3. Funções de Agregação

Uma função de agregação (ou função agregada) é uma função que opera sobre um conjunto de linhas. As funções de agregação permitem calcular:

- Valores totais para toda uma tabela; e
- Subtotais para toda uma tabela, agrupando o resultado por determinado atributo e apresentando-o como uma nova coluna.

A forma geral é:

```
SELECT função_agregada  
FROM nome_da_tabela  
[...]
```

Já aprendemos sobre uma das funções de agregação anteriormente (lembre-se de COUNT). As funções agregadas mais comuns são:

```
COUNT (*)  
COUNT ([ALL|DISTINCT] nome_da_coluna)  
SUM ([ALL|DISTINCT] nome_da_coluna)  
AVG ([ALL|DISTINCT] nome_da_coluna)  
MAX([ALL|DISTINCT] nome_da_coluna)  
MIN([ALL|DISTINCT] nome_da_coluna)  
STDDEV ([ALL|DISTINCT] nome_da_coluna)  
VARIANCE ([ALL|DISTINCT] nome_da_coluna)
```

Alguns exemplos:

```
SELECT AVG(preco) MEDIA FROM PRODUTOS;  
SELECT AVG(NVL(preco,0)) MEDIA FROM PRODUTOS;  
SELECT MAX(preco) FROM PRODUTOS;  
SELECT COUNT(*) NUM_CLIENTES FROM CLIENTES;  
SELECT COUNT(ddd) FROM TELEFONES;
```

Dica: A função **NVL()** converte valores nulos em um valor computável. Compare os resultados dos dois primeiros exemplos e tente identificar a diferença.

## 4. Agrupamento via cláusula GROUP BY

A cláusula GROUP BY pode ser utilizada em um comando SELECT para agrupar os resultados de uma consulta, gerando subtotais por grupos em novas colunas.

A forma geral do comando é então:

```
SELECT nome_da_coluna [, ...], função_agregada [, ...]
FROM nome_da_tabela [, ...]
GROUP BY [ALL] nome_da_coluna [,...]
ORDER BY colunas
```

Observações sobre a cláusula GROUP BY:

- O operador ALL inclui no resultado todos os grupos, incluindo aqueles que não atendem às condições de busca;
- A(s) coluna(s) contidas na cláusula SELECT deve(m) estar todas obrigatoriamente na cláusula GROUP BY;
- As colunas da cláusula GROUP BY não precisam estar na cláusula SELECT;
- Pode-se agrupar também por mais de uma coluna;
- A cláusula ORDER BY não é obrigatória, mas é bastante comum nesses casos porque organiza o resultado da consulta.

### 4.1 Exemplos

```
CREATE TABLE PRODS
(
    codigo NUMERIC(3) NOT NULL,
    nome VARCHAR(50) NOT NULL,
    preco NUMERIC (5,2) NOT NULL,
    tipo CHAR(1) NULL,
    -- [S]uprimento, [C]omponente, [P]eriférico
    CONSTRAINT PK1 PRIMARY KEY (codigo)
);
```

```

INSERT INTO PRODS
VALUES( 10, 'HD', 200, 'C');

INSERT INTO PRODS
VALUES( 11, 'Memoria', 250, 'C');

INSERT INTO PRODS
VALUES( 12, 'Impressora', 680, 'P');

INSERT INTO PRODS
VALUES( 13, 'Processador', 600, 'C');

INSERT INTO PRODS
VALUES( 14, 'DVD-RW', 2, 'S');

INSERT INTO PRODS
VALUES( 15, 'Papel A4', 19, 'S');

INSERT INTO PRODS
VALUES( 16, 'Scanner', 199, 'P');

```

 Escreva comandos SELECT para as seguintes consultas:

- a) Quantos produtos existem na tabela PRODS?
- b) Quantos tipos de produtos existem na tabela PRODS?
- c) Quantos produtos existem de cada tipo?
- d) Qual a média de preço de todos os produtos?
- e) Qual a média de preço dos suprimentos (tipo 'S')?
- f) Qual a média de preço dos produtos de cada tipo?

Seguir as instruções do professor para executar os seguintes comandos adicionais:

```

ALTER TABLE PRODS ADD (usuario NUMBER(1) NULL);

UPDATE PRODS
SET usuario = 1
WHERE codigo IN (10,12,13,14);

UPDATE PRODS
SET usuario = 2
WHERE usuario IS NULL;

SELECT tipo, usuario, AVG(preco)
FROM PRODS
GROUP BY tipo, usuario
ORDER BY tipo, usuario;

```

```
UPDATE PRODS
  SET usuario = 2
  WHERE codigo = 14;
```

```
-- Deixando um produto sem usuario associado
UPDATE PRODS
  SET usuario = NULL
  WHERE codigo = 13;

-- Executando novamente
SELECT tipo, usuario, AVG(preco)
FROM PRODS
GROUP BY tipo, usuario
ORDER BY tipo, usuario;
```


## 5. Agrupamento via cláusulas GROUP BY e HAVING

A cláusula HAVING é usada em conjunto com a cláusula GROUP BY. Ela determina as condições sobre as quais será realizada a composição dos grupos. Em outras palavras a cláusula HAVING serve para decidir quais dos grupos gerados farão parte do resultado final. Os grupos que não satisfizerem as condições da cláusula HAVING são descartados.


A forma geral do comando é:

```
SELECT nome_da_coluna [, ...], função_agregada [, ...]
FROM nome_da_tabela [, ...]
GROUP BY [ALL] nome_da_coluna [,...]
HAVING condições
ORDER BY colunas
```

Dica: As novas colunas geradas pelo cálculo das funções agregadas podem ser referidas na cláusula HAVING.

 Sob o BD do estudo de caso (Oracle), experimente o comando a seguir e tente identificar qual o seu propósito:

```
SELECT CID.nome, COUNT(*) QTD
FROM CIDADES CID, ENDERECOS END
WHERE CID.cod_cidade = END.cod_cidade
GROUP BY CID.nome
HAVING COUNT(*) > 10;
```

 Crie outras consultas sobre o banco de dados do Estudo de Caso, com apoio do professor.

## 6. Prática

Considerando nosso Estudo de Caso, escreva comandos SELECT para as questões a seguir:

- a) Qual a quantidade de endereços por estado?
- b) Qual a quantidade de clientes de cada estado?
- c) Qual o ranking de estados por quantidade de clientes, ou seja, em ordem decrescente de quantidade de clientes?
- d) Qual o ranking de regiões por quantidade de clientes, ou seja, em ordem decrescente de quantidade de clientes?
- e) Qual a quantidade de pedidos por região?
- f) Qual a quantidade de pedidos por ano e por região, considerando apenas os pedidos feitos nos anos de 2000 até 2004?
- g) Qual o valor total gasto por cliente, ordenado em ordem decrescente de valor total?
- h) Qual o valor total gasto por cliente, ordenado em ordem decrescente de valor total, considerando apenas os clientes do Rio Grande do Sul?
- i) Qual o valor total vendido por autor?
- j) Qual o valor médio faturado com as vendas por produto?
- k) Qual o valor total de cada pedido?
- l) Qual o valor médio dos pedidos por estado?

### Dicas finais desta aula

- Há vários tipos de funções que podem ser aplicados aos dados em SQL.
- Há funções que operam sobre valores individuais e funções que operam sobre conjuntos de valores.
- As cláusulas GROUP BY e HAVING complementam o comando SELECT para computar valores para grupos de registros.
- Os agrupamentos podem ser também aplicados a dados provenientes de várias tabelas.
- Uma forma de lembrar o propósito da cláusula HAVING é pensa-la como sendo “o WHERE do GROUP BY”, ou seja, um filtro sobre grupos de registros.

-X-