UPF

Instituto de Tecnologia - ITec

Ciência da Computação

Banco de Dados

Funções de Agregação e Visões

UPF

- As **FUNÇÕES de AGREGAÇÃO** executam um cálculo em um conjunto de valores e retornam **um único valor**.
- Exemplos de Funções de Agregações:
  - **COUNT**: Retorna o número de linhas afetadas pelo comando
  - **SUM**: retorna a soma os valores de uma coluna
  - **AVG**: retorna a média aritmética dos valores de uma coluna
  - **MAX**: retorna o maior valor de uma coluna
  - **MIN**: identifica o menor valor de uma coluna
- Exemplo: sobre salários, encontre: o total, o maior, o menor e a média salarial da tabela empregados

```
SELECT  SUM(salario), MAX(salario), MIN(salario), AVG(salario)
FROM    Empregado
```

- As funções de agregação normalmente são usadas com a cláusula **GROUP BY** da instrução SELECT.
  - A cláusula GROUP BY agrupa linhas baseado em semelhanças entre elas.
    - Exemplo: Desejamos obter o número de produtos em estoque, agrupados pelo tipo (celular, smartphone, netebook), para que depois seja feita a soma da quantidade existente em cada um dos grupos.
  - Para isso usamos a função SUM() em conjunto com o GROUP BY, como a instrução:

```
SELECT    tipo, SUM(quantidade)
FROM      Produtos
GROUP BY tipo
```

    * no SELECT há o campo tipo e o uso da função de agregação, então isso obriga usar o group by, para agrupar pelo tipo

Funções Agregação com uso de Cláusulas

- A cláusula **HAVING** determina uma condição de busca para um grupo ou um conjunto de registros, definindo critérios para limitar os resultados obtidos a partir do agrupamento de registros/linhas.
  - só pode ser usada em parceria com GROUP BY.
- A cláusula GROUP BY pode ser empregada, entre outras finalidades, para agrupar os produtos de acordo com cada tipo existente.
  - Dentro de cada um dos grupos, a cláusula HAVING pode ser usada para restringir – funcionar como filtro.

Cláusula HAVING

- A cláusula HAVING determina uma condição de busca para um grupo ou um conjunto de registros, definindo critérios para limitar os resultados obtidos a partir do agrupamento de registros.
  - essa cláusula só pode ser usada em parceria com GROUP BY.

- A cláusula GROUP BY pode ser empregada, entre outras finalidades, para agrupar os produtos de acordo com cada tipo existente. Exemplo: dentro de cada um dos grupos, a cláusula HAVING pode ser usada para restringir apenas os registros que possuem uma quantidade superior a 200 unidades no estoque

SINTAXE:

```
SELECT   column_name(s)
FROM     table_name
WHERE    condition
GROUP BY column_name(s)
HAVING   condition
ORDER BY column_name(s);
```

Comparação WHERE/HAVING:

- **WHERE** restringe os resultados obtidos sempre após o uso da cláusula FROM.
- A cláusula **HAVING** filtra o retorno do agrupamento.
  - Exemplo: A cláusula GROUP BY pode ser empregada, entre outras finalidades, para agrupar os produtos de acordo com cada tipo existente. Dentro de cada um dos grupos, a cláusula HAVING pode ser usada para restringir apenas os registros que possuem uma quantidade superior a 20 unidades no estoque.

```
SELECT    tipo, SUM(quantidade)
FROM      Produtos
GROUP BY  tipo
HAVING    SUM (quantidade) > 20
```

Visões (*views*)

- **Visões (views)** são consideradas *tabela virtuais*, ou seja, elas são usadas junto a instrução SELECT para apresentar subconjuntos de dados presentes em tabelas reais.
  - O resultado de uma consulta gera a visão
- Aspecto importante!
  - as Visões possuem permissões separadas, podemos utilizá-las para restringir mais o acesso aos dados pelos usuários, para que veja apenas o que é necessário.
- Por serem tabelas virtuais, não podemos realizar por elas operações DML de inserção, atualização e exclusão(varia conforme SGBD e tipo de visão), mas sim apenas usando o SELECT.

- Podemos criar uma visão de várias formas, até mesmo como combinações de objetos, tais como:
  - Um única tabela; Mais de uma tabela; Outra visão; Visões múltiplas
- Sintaxe de criação de uma visão (entre [opcional]):

```
CREATE [ OR REPLACE ] [ TEMP | TEMPORARY ] [ RECURSIVE ]
VIEW name [ ( column_name [, ...] ) ]
  [ WITH ( view_option_name [= view_option_value] [, ... ] ) ]
  AS SELECT
  [ WITH [ CASCADED | LOCAL ] CHECK OPTION ]
```

fonte: https://www.postgresql.org/docs/current/sql-createview.html

- Podemos consultar visões da mesma forma que consultamos tabelas (substituímos o nome da tabela pelo nome da visão).

- O parâmetro [TEMPORARY | TEMP] é especificado quando queremos que a Visão seja temporária. Estas são eliminadas de forma automática quando a sessão atual é finalizada.
  - Caso tenhamos referenciado tabelas temporárias para a View, esta será criada como temporária mesmo que não tenhamos especificado este parâmetro.
- Exemplo:

  CREATE VIEW visaoTeste AS SELECT……

- O comando CREATE OR REPLACE VIEW cria ou substitui um visão com o mesmo nome
  - Uma visão somente pode ser substituída por uma nova visão que produza um conjunto idêntico de colunas e tipo de dados