SQL Avançado
Continuação

Cláusula COMPUTE

- Sintaxe:

SELECT <colunas> FROM <tabelas>
ORDER BY <coluna>
COMPUTE <lista de funções de agregação>
BY <lista de colunas>

- Para que serve?
  - Gerar totalizadores que aparecem como colunas adicionais resumo no final do conjunto resultado
- O que devo fazer ??
  - Trazer 3 exemplos práticos desta cláusula

Cláusula COMPUTE

- Gera totalizadores que aparecem como colunas adicionais resumo no final do conjunto resultado
- Quando utilizada com a cláusula BY, ela gera sub-totais no conjunto resultado
- Não é padrão SQL

Cláusula COMPUTE

```
SELECT <colunas> FROM <tabelas>
ORDER BY <coluna>
COMPUTE <lista de funções de agregação>
BY <lista de colunas>
```

Cláusula COMPUTE

- Exemplo: Mostrar uma soma sumarizada dos preços de custo e venda

```
SELECT p.codP, p.nome, p.tipo,
p.preco_custo, p.preco_venda
FROM produto p, pedido pe
WHERE p.codP = pe.cod_produto
ORDER BY codP
COMPUTE SUM(preco_venda),
    SUM(preco_custo)
```

Cláusula COMPUTE

- **Exemplo com BY**: Mostra uma soma sumarizada dos preços de custo e venda para cada tipo de produto

```
SELECT p.codP, p.nome, p.tipo,
p.preco_custo, p.preco_venda
FROM produto p, pedido pe
WHERE p.codP = pe.cod_produto
ORDER BY codP
COMPUTE SUM(preco_venda),
    SUM(preco_custo) BY codP
```

Agrupando Informações – GROUP BY

- Sintaxe

```
SELECT <colunas> FROM <tabelas>
WHERE <condição>
GROUP BY <coluna>
HAVING <condição>
```

Agrupando Informações – GROUP BY

- GROUP BY: organiza as linhas de resultado em grupos de acordo com os valores das expressões informadas
  - Uso com funções agregadas
- HAVING: (opcional) seleciona os grupos de acordo com os resultados
  - Só pode ser usado junto com o GROUP BY

Agrupando Informações – GROUP BY

- Exemplo: Exibir a quantidade de homens e mulheres na empresa

```
SELECT sexo, count (*)
FROM funcionário
GROUP BY sexo
```

Agrupando Informações – GROUP BY

- Exemplo: Exibir a quantidade de funcionários e o total de salários de cada setor da empresa

```
SELECT setor, COUNT(*)
TotFunc, SUM(salario) TotSal
FROM funcionário
GROUP BY setor
```

Agrupando Informações – GROUP BY

- Exemplo: Exibir a quantidade de pedidos de cada data

```
SELECT data_pedido, COUNT(*)
       quantidade
FROM pedido
GROUP BY data_pedido
```

Agrupando Informações – GROUP BY

- Exemplo: Exibir os códigos dos pedidos que tem mais do que 4 produtos

```
SELECT cod_pedido,
       COUNT(produto)
FROM itens
GROUP BY cod_pedido
HAVING COUNT(produto)>4
```

Agrupando Informações – GROUP BY

- Exemplo: Exibir o valor total de cada pedido (valor total = quantidade*preço-desconto), onde a soma seja maior que R$1.000,00

```
SELECT cod_pedido,
       SUM((quant*preco)-desc)
FROM itens
GROUP BY cod_pedido
HAVING SUM((quant*preco)-desc)
            > 1000
```

Agrupando Informações – GROUP BY

- Exemplo: Exibir a média de idade dos funcionários de cada sexo, em cada setor da empresa. Exibir apenas os setores onde essa média de idade seja superior a 40 anos

```
SELECT setor, sexo, AVG(idade)
FROM funcionario
GROUP BY setor, sexo
HAVING AVG(idade) > 40
```

Sub-Consultas

- Uma sub-consulta é uma consulta SELECT aninhada dentro de outro comando SQL
- Uma sub-consulta deve ser delimitada entre parênteses e é avaliada apenas uma vez
- O resultado de uma sub-consulta retorna um conjunto de linhas para a consulta principal
  - A consulta mais externa depende da sub-consulta

Sub-Consultas

- Retorno de uma sub-consulta
  - Uma sub-consulta de valor único retorna apenas um valor e pode ser usada no lugar de qualquer expressão utilizando operadores (=, <, >, <>)

1 coluna → 1 valor

```
WHERE A=(SELECT b...)  /*Verdade se A=B*/
```

Sub-Consultas

- Exemplo: Exibir o nome e o valor da gratificação das funções que têm a menor gratificação da empresa

```
SELECT nome, gratificacao
FROM funcao
WHERE gratificacao = (SELECT
MIN(gratificacao) FROM funcao)
```

Sub-Consultas

- Exemplo: Exibir o código, o nome e a quantidade em estoque do produto que tem a maior quantidade em, estoque da empresa

```
SELECT codigo, nome, quantEst
FROM produto
WHERE quantEst = (SELECT MAX(quantEst)
FROM produto)
```

Sub-Consultas

- Retorno de uma sub-consulta
  - Uma sub-consulta de valor único retorna múltiplas linhas e pode ser usada apenas em um WHERE utilizando cláusulas especiais

1 coluna → muitos valores

Sub-Consultas

- Uma sub-consulta pode retornar também uma lista de valores, que pode ser usada em comparações com o operador IN

Sub-Consultas

- Exemplo: Exibir código e nome de todos os clientes estrangeiros

```
SELECT codigo, nome FROM cliente
WHERE codPais IN (SELECT codigo
  FROM pais WHERE codigo <>'BRA')
```

```
SELECT c.codigo, c.nome
FROM cliente c, pais p
WHERE c.codPais = p.codigo AND
c.codPais <>'BRA'
```

Sub-Consultas: Lista de Valores Especiais

- > ALL : maior que todos
- < ALL : menor que todos
- <> ALL : diferente de todos (igual a NOT IN)
- = ANY : igual a algum dos elementos da lista (o mesmo que IN)
- > ANY : maior que algum dos elementos da lista
- < ANY: menor que algum dos elementos da lista
- <> ANY : diferente de algum dos elementos da lista

Sub-Consultas

- Exemplo: exibir nome, tipo e preço de venda dos produtos que não sejam dos tipos 3, 4 ou 5, e que tenham preço de venda maior que pelo menos o preço de um destes produtos

```
SELECT nome, tipo, preco_venda
FROM produto
WHERE tipo NOT IN (3,4,5) AND
preco_venda > ANY    (SELECT
preco_venda FROM produto WHERE tipo IN
(3,4,5))
```

União de Conjuntos

- Cláusula UNION
  - Utilizada para combinar resultado de dois comandos do tipo SELECT
  - Os dois comandos podem até trazer dados de tabelas diferentes, desde que com o **mesmo número de colunas** e **tipos de dados compatíveis** para cada coluna correspondente de um com o outro
  - Na união de conjuntos, os elementos repetidos são eliminados (como se utilizasse a cláusula DISTINCT)

União de Conjuntos

- Exemplo: Exiba todas as cidades e estados onde existem autores ou editoras, ordenados pelo estado

```
SELECT cidade, estado FROM autor
   UNION
SELECT cidade, estado
FROM editoras
ORDER BY estado
```

Consulta com Criação de Tabela

- Sintaxe:

```
SELECT <colunas>
INTO <nova_tabela>
FROM <tabela>
```

Consulta com Criação de Tabela

- Exemplo: Criar uma nova tabela que contenha o código do pedido, nome e telefone do cliente que fez cada um deles.

```
SELECT p.codPedido Pedido,
c.nomeCliente Cliente, C.fone Fone
INTO Pedidos
FROM compras p, cliente c
WHERE p.codCli = c.CodCli
```

Consulta com Criação de Tabela

- A nova tabela chamada **Pedidos** será criada no banco de dados (criação física), contendo o código do pedido, o nome e telefone do cliente que fez cada um deles
- Para criar uma tabela temporária, é necessário colocar o caractere # iniciando o nome da tabela
  - #Pedidos – Tabela Local
  - ##Pedidos – Tabela Golbal

Tabelas Temporárias

- Tabelas Temporárias Locais: Criadas com o prefixo #, e possuem visibilidade restrita para a conexão responsável pela sua criação
- Tabelas Temporárias Globais: Criadas com o prefixo ##, e são visíveis para todas as conexões

Tabelas Temporárias

- Os dois tipos de tabelas (locais e globais) só existem enquanto a conexão responsável pela sua criação estiver ativa
  - São eliminadas automaticamente quando a conexão é desfeita

Tabelas Temporárias

- Exemplo: Criar uma tabela com CREATE TABLE e inserir dados na mesma

```
CREATE TABLE #temp (
codCli int,
nome varchar(50)
)
```

```
INSERT INTO #temp
VALUES (1, 'Ana')
```

```
SELECT * FROM #temp
```

Tabelas Temporárias

- Exemplo: selecione a quantidade de pedidos agrupados pela data para uma nova tabela temporária global

```
SELECT data Data, COUNT(*)
    Quant INTO ##temp
FROM pedidos
GROUP BY data
```

Tabelas Temporárias

- Cuidado com a duplicação do nome da tabela temporária durante a criação! Se acontecer, teremos uma situação de erro
- Só utilize tabelas temporárias quando necessário!
  - Utilização exige gravação em disco → baixa performance