Banco de Dados

AULA 9 - SUBCONSULTAS E OPERADORES PARA TESTE EM CONJUNTOS DE VALORES

Usando uma subconsulta para solucionar um problema

- Suponha que você queira criar uma consulta para descobrir quem ganha um salário maior do que o salário do Paulo.
- Para solucionar este problema serão necessárias duas consultas: uma para localizar quanto Paulo ganha e outra para localizar quem ganha mais que esse valor.
- É possível solucionar este problema combinando as duas consultas, colocando uma consulta dentro de outra consulta.

Usando uma subconsulta para solucionar um problema

- A consulta interna (ou subconsulta) retorna um valor que é usado pela consulta externa (ou consulta principal).
- Usar uma subconsulta é equivalente a realizar duas consultas sequenciais e usar o resultado da primeira consulta como o valor de pesquisa da segunda consulta.
- A subconsulta é executada antes da consulta principal.
- As subconsultas podem ser muito úteis quando é preciso selecionar linhas de uma tabela com uma condição que que depende dos dados na própria tabela.

Sintaxe de subconsulta

```
SELECT  select_list

FROM table

WHERE expr operator (

     SELECT select_list

     FROM  table);
```

Exemplo de subconsultas no **WHERE**

```
SELECT e.firstname, e.salary

FROM employee e

WHERE e.salary > (

    SELECT es.salary

    FROM employee es

    WHERE es.firstname = 'Paulo');
```

Subconsultas no **`FROM`**

- Em SQL é possível criar consultas que sejam utilizadas em cláusulas **`WHERE`**, **`HAVING`**, **`FROM`**, **`SELECT`** ou de uma outra consulta.
- Em uma cláusula **`FROM`** o resultado da subconsulta é tratado com uma relação, semelhante a uma tabela. Neste caso é possível fazer uma junção entre uma tabela e um resultado de uma consulta.

```
FROM evento e INNER JOIN (SELECT....) a ON e... =
                                        a...
```

Subconsultas no **FROM**

Por exemplo: Mostrar o código do projeto, a sequência das atividades, o nome das atividades e a descrição das atividades que ocorreram a partir de 01/01/2012.

```
SELECT a.idproj, a.sequencia, a.nome, oc.descricao
FROM atividade a, (
    SELECT o.idproj, o.sequencia, o.descricao
    FROM ocorrencia o
    WHERE a.dataocor >= '2012-01-01') oc
WHERE  a.idproj  =  oc.idproj  AND  a.sequencia  =
oc.sequencia;
```

Subconsultas no **`SELECT`**

- Em uma cláusula **`SELECT`** uma subconsulta pode ser usada para retornar um ou mais atributos. Neste caso, para cada linha retornada pela consulta principal, a subconsulta somente pode retornar um valor.
- Na maioria das situações as subconsultas são utilizadas em operações de seleção, na cláusula **`WHERE`**. A busca de um valor no resultado de uma subconsulta ou comparação de valores com este resultado são os casos mais comuns.

Subconsultas no `SELECT`

- Por exemplo: Mostrar o código do projeto, o nome do projeto e a maior duração entre as atividades dos projeto.

```
SELECT p.id, p.nome, (
    SELECT MAX (a.duracaoest)
    FROM atividade a
    WHERE p.id = a.idproj) AS maiorduracaoativ
FROM projeto p;
```

Regras para o uso de subconsultas

- Insira as subconsultas entre parênteses.
- Insira subconsultas do lado direito da condição de comparação por questão de legibilidade. Entretanto, a subconsulta pode ser exibida nos dois lados do operador de comparação.
- Use operadores de uma única linha (`>`, `=`, `>=`, `<`, `<>`, `<=`) com subconsultas de uma única linha e operadores de várias linhas (**`IN`**, **`ALL`**, **`SOME`**, **`ANY`**, **`EXISTS`**).

Exemplo de subconsulta de uma única linha

```
SELECT lastname, job_id, salary
FROM employee
WHERE job_id = (
    SELECT job_id
    FROM employee
    WHERE lastname = 'Lima')
AND salary > (
    SELECT salary
    FROM employee
    WHERE lastname = 'Lima');
```

Exemplo de subconsulta com uma função de grupo

```
SELECT firstname, salary
FROM employee
WHERE salary = (
    SELECT MIN (salary)
    FROM employee);
```

Subconsultas no **HAVING**

- É possível utilizar subconsultas também na cláusula **HAVING**.
- Por exemplo: Exibir todos os departamentos que possuem um salário mínimo maior que o departamento 4.

```
SELECT department_id, MIN (salary)
FROM employee
GROUP BY department_id
HAVING MIN (salary) > (
    SELECT MIN (salary)
    FROM employee
    WHERE department_id = 4);
```

Exercício 1: o que há de errado com esta instrução?

```
SELECT lastname, job_id
FROM employee
WHERE job_id = (
    SELECT job_id
    FROM employee
    WHERE lastname = 'Fereira');
```

Exercício 2: o que há de errado com esta instrução?

```
SELECT employee_id, lastname
FROM employee
WHERE salary = (
    SELECT MIN (salary)
    FROM employee
    GROUP BY department_id);
```

Operador IN

- O operador **`IN`** permite testar se um determinado valor existe em um conjunto de valores.
- Retorna mais de uma linha e usa operadores de comparação de várias linhas.
- O conjunto de valores normalmente é resultado de um **`SELECT`** (subconsulta de várias linhas).
- A instrução que retorna os valores a serem comparados será executada antes das comparações.

Operador IN

```
SELECT employee_id, lastname
FROM employee
WHERE salary IN (
    SELECT MIN (salary)
    FROM employee
    GROUP BY department_id);
```

Operador IN

- Por exemplo: Mostrar o código e o nome das pessoas que participam de pelo menos uma atividade na função 'Auxiliar'.

```
SELECT codpess, nome
FROM pessoa
WHERE codpess IN (
    SELECT codpess
    FROM participanteativ p, funcao f
    WHERE f.codfunc = p.codfunc AND f.nome =
    'Auxiliar');
```

Operador IN

- Na consulta anterior, cada valor do campo "codpess" na tabela pessoa é comparado ao conjunto de valores retornados pela consulta entre parênteses.
- Somente serão mostrados no resultado final as pessoas cujo código apareça na lista de valores resultantes.
- Para testar a não existência em um conjunto basta incluir o **`NOT`** antes do **`IN`**, no caso anterior teríamos **`codpess NOT IN`** ...

Operador IN

- Outro exemplo: Mostrar a lista de pessoas que participam de alguma atividade como 'Auxiliar' e também participam de pelo menos uma, como 'Analista'.

```
SELECT codpess, nome
FROM pessoa
WHERE codpess IN (
    SELECT codpess
    FROM participanteativ p, funcao f
    WHERE f.codfunc = p.codfunc AND f.nome =
    'Auxiliar')
AND codpess IN (
    SELECT codpess
    FROM participanteativ p, funcao f
    WHERE f.codfunc = p.codfunc AND f.nome =
    'Analista');
```

Operador IN

- Neste caso, somente as pessoas que exercem ambas as funções serão mostradas.
- Caso fosse necessário mostrar aquelas que exercem uma função, mas não outra, bastaria colocar o operador de **`NOT`** no segundo teste.
- Como visto anteriormente, este operador pode ser usado também com valores constantes como:

```
codcid IN (1, 5)
```

Operador IN

- Se o valor de um campo é comparado ao conjunto de valores retornados por um **`SELECT`**, somente um campo pode constar na cláusula **`SELECT`** da subconsulta.
- O tipo do campo a ser comparado deve ser o mesmo ou compatível com os valores retornados pela subconsulta.
- Também é possível comparar dois campos com dois valores retornados por uma subconsulta.

Comparação entre conjuntos

- A comparação com **IN** permite comparar um valor a valores de um conjunto.
- Entretanto, existem outras situações que é importante comparar uma relação de valores com outra.
- Estas operações podem ser realizadas usando os operadores **SOME** (algum), **ANY** (algum) e **ALL** (todos). **ANY** e **SOME** são sinônimos.

Operador ALL

- Compara um valor com todos os valores retornados por uma subconsulta.
- Por exemplo: Mostrar os funcionários que não são administradores e cujos salários são menores que os salários de todos os funcionários com o cargo de administrador.

```
SELECT id, lastname, job_id, salary
FROM employee
WHERE salary < ALL (
    SELECT salary
    FROM employee
    WHERE job_id = 7)
AND job_id <> 7;
```

Operador SOME ou ANY

- Compara um valor com cada valor retornado por uma subconsulta.
- Por exemplo: Exibir os funcionários que não são administradores e cujos salários são menores do que qualquer administrador.

```
SELECT id, lastname, job_id, salary
FROM employee
WHERE salary < SOME (
    SELECT salary
    FROM employee
    WHERE job_id = 7)
AND job_id <> 7;
```

Comparação entre conjuntos

- Junto com as cláusulas **`ALL`** e **`SOME`** é possível utilizar `>`, `>=`, `<`, `<=`.
- Usando `>=` por exemplo, comparando uma lista de valores com outra lista igual, é possível obter o maior valor desta lista.
- Similarmente, `<=` permite obter o menor valor da lista.
- Quando os valores das listas não são os mesmos, para achar o maior ou menor, usa-se `>` e `<`.

Operador EXISTS

- É possível testar se uma subconsulta retornou algum registro em seu resultado usando o operador **`EXISTS`**.
- Este operador retorna o valor **`True`** se o argumento, que é uma subconsulta, não está vazio.
- Em muitas situações este operador é uma alternativa ao uso do operador **`IN`**, descrito anteriormente
- A inclusão de **`NOT`**, permite verificar se o conjunto está vazio.

Operador EXISTS

- Por exemplo: mostrar a lista dos gerentes que ganham um salário maior que R$4.000,00.

```
SELECT e.id, e.salary, e.lastname
FROM employee e
WHERE EXISTS (
    SELECT x.id
    FROM employee x
    WHERE x.manager_id = e.id
    AND x.salary > 4000);
```

Operador EXISTS

- Outro exemplo: mostrar a lista das pessoas que participam de atividades como 'Auxiliar'.

```
SELECT pe.codpess, pe.nome
FROM pessoa pe
WHERE EXISTS (
    SELECT *
    FROM participanteativ pa, funcao f
    WHERE f.codfunc = pa.codfunc
    AND pa.codpess = pe.codpess
    AND f.nome = 'Auxiliar');
```

Operador EXISTS

- Esta consulta poderia ser desenvolvida com o uso do **`IN`** ou mesmo sem testes de pertinência ou existência de valores em conjuntos.
- Existem diversas situações nas quais é possível optar por uma estratégia diferente.
- Neste caso, cada valor do código de pessoa será utilizado como parâmetro da subconsulta.
- A subconsulta será executada tantas vezes quanto for o número de pessoas.

Operador EXISTS

- O operador EXISTS pode ser importante para dividir consultas complexas e aproveitar o resultado de uma na construção da outra.
- Sua utilização é comum em **`procedures`** para verificar se uma consulta retorna valores.
- Na maioria dos casos, quando existe uma quantidade grande de valores a serem usados como parâmetro para execução da subconsulta esta estratégia será ineficiente se comparada à forma tradicional.
- A diferença para o operador **`IN`** é que no caso anterior a subconsulta não utiliza dados das tabelas envolvidas na consulta principal.

Exercícios

Ver Lista 6