8) Entendendo e Utilizando sub-consultas

8) Expressões de subconsulta

Esta seção descreve as expressões de subconsulta em conformidade com o padrão SQL disponíveis no PostgreSQL. Todas as formas das expressões documentadas nesta seção retornam resultados booleanos (verdade/falso).

8.1. EXISTS

```
EXISTS ( subconsulta )
```

O argumento do EXISTS é uma declaração SELECT arbitrária, ou uma *subconsulta*. A subconsulta é processada para determinar se retorna alguma linha. Se retornar pelo menos uma linha, o resultado de EXISTS é "verdade"; se a subconsulta não retornar nenhuma linha, o resultado de EXISTS é "falso".

A subconsulta pode referenciar variáveis da consulta que a envolve, que atuam como constantes durante a execução da subconsulta.

A subconsulta geralmente só é processada até ser determinado se retorna pelo menos uma linha, e não até o fim. Não é recomendável escrever uma subconsulta que tenha efeitos colaterais (tal como chamar uma função de seqüência); pode ser difícil prever se o efeito colateral ocorrerá ou não.

Como o resultado depende apenas de alguma linha ser retornada, e não do conteúdo da linha, normalmente não há interesse na saída da subconsulta. Uma convenção de codificação habitual é escrever todos os testes de EXISTS na forma EXISTS(SELECT 1 WHERE ...). Entretanto, existem exceções para esta regra, como as subconsultas que utilizam INTERSECT.

Este exemplo simples é como uma junção interna em col2, mas produz no máximo uma linha de saída para cada linha de tab1, mesmo havendo várias linhas correspondentes em tab2:

```
SELECT col1 FROM tab1
    WHERE EXISTS(SELECT 1 FROM tab2 WHERE col2 = tab1.col2);
```

Exemplo. Utilização das cláusulas CASE e EXISTS juntas

Neste exemplo a tabela frutas é consultada para verificar se o alimento é uma fruta ou não. Caso o alimento conste da tabela frutas é uma fruta, caso não conste não é uma fruta. Abaixo está mostrado o *script* utilizado para criar e carregar as tabelas e executar a consulta. [1]

```
CREATE TEMPORARY TABLE frutas (id SERIAL PRIMARY KEY, nome TEXT);
INSERT INTO frutas VALUES (DEFAULT, 'banana');
INSERT INTO frutas VALUES (DEFAULT, 'maçã');
CREATE TEMPORARY TABLE alimentos (id SERIAL PRIMARY KEY, nome TEXT);
INSERT INTO alimentos VALUES (DEFAULT, 'maçã');
INSERT INTO alimentos VALUES (DEFAULT, 'espinafre');
SELECT nome, CASE WHEN EXISTS (SELECT nome FROM frutas WHERE nome=a.nome)
                  THEN 'sim'
                  ELSE 'não'
             END AS fruta
FROM alimentos a;
```

Abaixo está mostrado o resultado da execução do *script*.

```
   nome    | fruta
-----------+-------
 maçã      | sim
 espinafre | não
```

8.2. IN

```
expressão IN (subconsulta)
```

O lado direito é uma subconsulta entre parênteses, que deve retornar exatamente uma coluna. A expressão à esquerda é processada e comparada com cada linha do resultado da subconsulta. O resultado do IN é "verdade" se for encontrada uma linha igual na subconsulta. O resultado é "falso" se não for encontrada nenhuma linha igual (incluindo o caso especial onde a subconsulta não retorna nenhuma linha).

Deve ser observado que, se o resultado da expressão à esquerda for nulo, ou se não houver nenhum valor igual à direita e uma das linhas à direita tiver o valor nulo, o resultado da construção IN será nulo, e não falso. Isto está de acordo com as regras normais do SQL para combinações booleanas de valores nulos.

Da mesma forma que no EXISTS, não é razoável assumir que a subconsulta será processada até o fim.

```
construtor_de_linha IN (subconsulta)
```

O lado esquerdo desta forma do IN é um construtor de linha, conforme descrito na Seção 4.2.11. O lado direito é uma subconsulta entre parênteses, que deve retornar exatamente tantas colunas quantas forem as expressões na linha do lado esquerdo. As expressões do lado esquerdo são processadas e comparadas, por toda a largura, com cada linha do resultado da subconsulta. O resultado do IN é "verdade" se for encontrada uma linha igual na subconsulta. O resultado é "falso" se não for encontrada nenhuma linha igual (incluindo o caso especial onde a subconsulta não retorna nenhuma linha).

Da maneira usual, os valores nulos nas linhas são combinados de acordo com com as regras normais para expressões booleana do SQL. As linhas são consideradas iguais se todos os seus membros correspondentes forem não-nulos e iguais; as linhas são diferentes se algum membro correspondente for não-nulo e diferente; senão o resultado da comparação é desconhecido (nulo). Se o resultado de todas as linhas for diferente ou nulo, com pelo menos um nulo, o resultado do IN será nulo.

Exemplo. Utilização das cláusulas CASE e IN juntas

Este exemplo é idêntico ao Exemplo 9-16, só que utiliza a cláusula IN para executar a consulta, conforme mostrado abaixo. [2]

```
SELECT nome, CASE WHEN nome IN (SELECT nome FROM frutas)
                  THEN 'sim'
                  ELSE 'não'
             END AS fruta
FROM alimentos;
```

Abaixo está mostrado o resultado da execução do script.

```
   nome     | fruta
------------+-------
 maçã       | sim
 espinafre | não
```

8.3. NOT IN

```
expressão NOT IN (subconsulta)
```

O lado direito é uma subconsulta entre parênteses, que deve retornar exatamente uma coluna. A expressão à esquerda é processada e comparada com cada linha do resultado da subconsulta. O resultado de NOT IN é "verdade" se somente forem encontradas linhas diferentes na subconsulta (incluindo o caso especial onde a subconsulta não retorna nenhuma linha). O resultado é "falso" se for encontrada alguma linha igual.

Deve ser observado que se o resultado da expressão à esquerda for nulo, ou se não houver nenhum valor igual à direita e uma das linhas à direita tiver o valor nulo, o resultado da construção NOT IN será nulo, e não verdade. Isto está de acordo com as regras normais do SQL para combinações booleanas de valores nulos.

Da mesma forma que no EXISTS, não é razoável assumir que a subconsulta será processada até o fim.

```
construtor_de_linha NOT IN (subconsulta)
```

O lado esquerdo desta forma do NOT IN é um construtor de linha, conforme descrito na Seção 4.2.11. O lado direito é uma subconsulta entre parênteses, que deve retornar exatamente tantas colunas quantas forem as expressões na linha do lado esquerdo. As expressões do lado esquerdo são processadas e comparadas, por toda a largura, com cada linha do resultado da subconsulta. O resultado do NOT IN é "verdade" se somente forem encontradas linhas diferentes na subconsulta (incluindo o caso especial onde a subconsulta não retorna nenhuma linha). O resultado é "falso" se for encontrada alguma linha igual.

Da maneira usual, os valores nulos nas linhas são combinados de acordo com com as regras normais para expressões booleana do SQL. As linhas são consideradas iguais se todos os seus membros correspondentes forem não-nulos e iguais; as linhas são diferentes se algum membro correspondente for não-nulo e diferente; senão o resultado da comparação é desconhecido (nulo). Se o resultado de todas as linhas for diferente ou nulo, com pelo menos um nulo, o resultado do NOT IN será nulo.

8.4. ANY/SOME

```
expressão operador ANY (subconsulta)
expressão operador SOME (subconsulta)
```

O lado direito é uma subconsulta entre parênteses, que deve retornar exatamente uma coluna. A expressão à esquerda é processada e comparada com cada linha do resultado da subconsulta usando o operador especificado, devendo produzir um resultado booleano. O resultado do ANY é "verdade" se for obtido algum resultado verdade. O resultado é "falso" se nenhum resultado verdade for encontrado (incluindo o caso especial onde a subconsulta não retorna nenhuma linha). [3]

SOME é sinônimo de ANY. IN equivale a = ANY.

Deve ser observado que se não houver nenhuma comparação bem sucedida, e pelo menos uma linha da direita gerar nulo como resultado do operador, o resultado da construção ANY será nulo, e não falso. Isto está de acordo com as regras normais do SQL para combinações booleanas de valores nulos.

Do mesmo modo que no EXISTS, não é razoável supor que a subconsulta será processada até o fim.

```
construtor_de_linha operador ANY (subconsulta)
construtor_de_linha operador SOME (subconsulta)
```

O lado esquerdo desta forma do ANY é um construtor de linha, conforme descrito na Seção 4.2.11. O lado direito é uma subconsulta entre parênteses, que deve retornar exatamente tantas colunas quantas forem as expressões existentes na linha do lado esquerdo. As expressões do lado esquerdo são processadas e comparadas, por toda a largura, com cada linha do resultado da subconsulta utilizando o operador especificado. Atualmente, somente são permitidos os operadores = e <> em construções ANY para toda a largura da linha. O resultado do ANY é "verdade" se for encontrada alguma linha igual ou diferente, respectivamente. O resultado será "falso" se não for encontrada nenhuma linha deste tipo (incluindo o caso especial onde a subconsulta não retorna nenhuma linha).

Da maneira usual, os valores nulos nas linhas são combinados de acordo com com as regras normais para expressões booleana do SQL. As linhas são consideradas iguais se todos os seus membros correspondentes forem não-nulos e iguais; as linhas são diferentes se algum membro correspondente for não-nulo e diferente; senão o resultado da comparação é desconhecido (nulo). Se houver pelo menos um resultado de linha nulo, então o resultado de ANY não poderá ser falso; será verdade ou nulo.

Exemplo. Utilização das cláusulas CASE e ANY juntas

Este exemplo é idêntico ao Exemplo 9-16, só que utiliza a cláusula ANY para executar a consulta, conforme mostrado abaixo. [4]

```
SELECT nome, CASE WHEN nome = ANY (SELECT nome FROM frutas)
                  THEN 'sim'
                  ELSE 'não'
             END AS fruta
FROM alimentos;
```

Abaixo está mostrado o resultado da execução do script.

```
   nome    | fruta
-----------+-------
 maçã      | sim
 espinafre | não
```

8.5. ALL

```
expressão operador ALL (subconsulta)
```

O lado direito é uma subconsulta entre parênteses, que deve retornar exatamente uma coluna. A expressão à esquerda é processada e comparada com cada linha do resultado da subconsulta usando o operador especificado, devendo produzir um resultado booleano. O resultado do ALL é "verdade" se o resultado de todas as linhas for verdade (incluindo o caso especial onde a subconsulta não

retorna nenhuma linha). O resultado é "falso" se for encontrado algum resultado falso.

NOT IN equivale a <> ALL.

Deve ser observado que se todas as comparações forem bem-sucedidas, mas pelo menos uma linha da direita gerar nulo como resultado do operador, o resultado da construção ALL será nulo, e não verdade. Isto está de acordo com as regras normais do SQL para combinações booleanas de valores nulos.

Do mesmo modo que no EXISTS, não é razoável supor que a subconsulta será processada até o fim.

```
construtor_de_linha operador ALL (subconsulta)
```

O lado esquerdo desta forma do ALL é um construtor de linha, conforme descrito na Seção 4.2.11. O lado direito é uma subconsulta entre parênteses, que deve retornar exatamente tantas colunas quantas forem as expressões existentes na linha do lado esquerdo. As expressões do lado esquerdo são processadas e comparadas, por toda a largura, com cada linha do resultado da subconsulta utilizando o operador especificado. Atualmente, somente são permitidos os operadores = e <> em construções ALL para toda a largura da linha. O resultado do ALL é "verdade" se todas as linhas da subconsulta forem iguais ou diferentes, respectivamente (incluindo o caso especial onde a subconsulta não retorna nenhuma linha). O resultado será "falso" se não for encontrada nenhuma linha que seja igual ou diferente, respectivamente.

Da maneira usual, os valores nulos nas linhas são combinados de acordo com com as regras normais para expressões booleana do SQL. As linhas são consideradas iguais se todos os seus membros correspondentes forem não-nulos e iguais; as linhas são diferentes se algum membro correspondente for não-nulo e diferente; senão o resultado da comparação é desconhecido (nulo). Se houver pelo menos um resultado de linha nulo, então o resultado de ALL não poderá ser verdade; será falso ou nulo.

8.6. Comparação de toda a linha

```
construtor_de_linha operador (subconsulta)
```

O lado esquerdo é um construtor de linha, conforme descrito na Seção 4.2.11. O lado direito é uma subconsulta entre parênteses, que deve retornar exatamente tantas colunas quantas forem as expressões existentes na linha do lado esquerdo. Além disso, a subconsulta não pode retornar mais de uma linha (se não retornar nenhuma linha o resultado será considerado nulo). As expressões do lado esquerdo são processadas e comparadas, por toda a largura, com cada linha do resultado da subconsulta. Atualmente, somente são permitidos os operadores = e <> para comparação por toda a largura da linha. O resultado é "verdade" se as duas linhas forem iguais ou diferentes, respectivamente

Da maneira usual, os valores nulos nas linhas são combinados de acordo com com as regras normais para expressões booleana do SQL. As linhas são consideradas iguais se todos os seus membros correspondentes forem não-nulos e iguais; as linhas são diferentes se algum membro correspondente for não-nulo e diferente; senão o resultado da comparação é desconhecido (nulo).

Notas

[1] Exemplo escrito pelo tradutor, não fazendo parte do manual original.

[2] Exemplo escrito pelo tradutor, não fazendo parte do manual original.

[3] SQL Server 2000 — SOME | ANY comparam um valor escalar com o conjunto de valores de uma única coluna. Sintaxe: expressão_escalar { = | < > | ! = | > | > = | ! > | < | < = | ! < } { SOME | ANY } (subconsulta). A subconsulta possui o conjunto de resultados de uma coluna, e o mesmo tipo de dado da expressão escalar. SOME e ANY retornam verdade quando a comparação especificada é verdade para qualquer par (expressão_escalar, x), onde x é um valor do conjunto de uma única coluna. Senão retorna falso. SOME | ANY (N. do T.)

[4] Exemplo escrito pelo tradutor, não fazendo parte do manual original.

Fonte: http://pgdocptbr.sourceforge.net/pg80/functions-subquery.html

Sub-consultas

Tipos de Sub-consultas:
- Interna (retorna somente um registro)
- Interna (retorna mais de um registro)
- Interna (retorna mais de um registro e mais de um campo)

O resultado do select mais interno serve de base para o select mais externo.

Este tipo tem um bom desempenho.

Uma tabela é consultada e com o retorno será pesquisada a outra.

Pesquisar os produtos com preco maior que o médio.

```
select produto, preco
   from produtos
   where preco > select avg(preco)
            from produtos;
```

Este exemplo tem baixo desempenho, pois executará o select interno para cada registro e o resultado passa para o select externo.

```
select codigo, produto, preco
   from produtos p
   where preco > select avg(preco_venda)
            from produtos
            where codigo = p.codigo;
```

Dica: não usar order by no select interno.

Somente devemos usar um único order by em toda a consulta e este no select externo.

```
select codigo, produto, preco from produtos
   where codigo = select codigo from produtos
            where cod_venda = 2
            and valor > select preco from produtos where cod_venda=2;
```

Sub-consulta na cláusula HAVING

```
select codigo min(preco)
   from produtos
   group by codigo
   having min(preco) >
      select preco from produtos where cod_ven = 6;
```

Sub-consultas com EXISTS

Só mostra o resultado se o select interno retornar um ou mais registros.

```
select cod_fornecedor, fornecedor from fonecedores
   where exists select * from produtos
   where produtos.cod_fornecedor = fornecedores.cod_fornecedor;
```

Caso o select interno retorne:

0 registro – false
>= 1 registro – true

Sub-consulta de Múltiplas linhas

O operador deve ser operador de grupo, como: IN, ANY ou ALL.

IN

Saber que produtos tem preço igual ou menor que o preço de cada fornecedor.

```
1o) select min(preco) from produtos
    group by codigo_fornecedor;

Supor retorno de: 3, 8, 5

2o) Com os valores do retorno anterior
select codigo, nome, preco
    from produtos
    where preco in(3,8,5);

Ou

select codigo, nome, preco
    from produtos
    where preco in
        select min(preco)
            from produtos
            group by codigo_fornecedor;
```

ANY

```
select codigo, nome, preco
    from produtos
    where preco < ANY
        select avg(preco)
            from produtos
            group by codigo_fornecedor;
```

Sub-consultas com múltiplos campos

Geralmente é lenta.

```
select codigo, nome, codigo_fornecedor, codigo_venda
    from produtos
    where codigo_fornecedor != codigo_venda IN
        select codigo_fornecedor, codigo_venda
            from produtos
            group by codigo_fornecedor;
```