5. SQL – Parte III

5.1. Valores nulos

A palavra chave especial *null* pode ser usada como predicado para testar a existência de valores nulos.

- encontrar números de empréstimos que aparecem na relação empréstimo com valores nulos para total:

```
select nro_empréstimo
from empréstimo
where total is null
```

O predicado *is not null* testa a ausência de valores nulos. A existência de valores nulos em expressões aritméticas pode causar diversas complicações:

- resultados nulos
- *unknown* (desconhecido) em comparações

Em operadores agregados (como *sum*), o padrão SQL diz que o operador pode ignorar os valores nulos de entrada. Exemplo:

```
select sum (total)
from empréstimo
```

5.2. Subconsultas aninhadas

Subconsulta: expressão *select-from-where* aninhada dentro de outra consulta. *Aplicações*: testes para membros de conjuntos, comparações de conjuntos e cardinalidade de conjuntos.

5.2.1. Membros de conjuntos

- conectivo *in* testa os membros de um conjunto, no qual o conjunto é a coleção de valores produzidos pela cláusula *select*;
- conectivo *not in* verifica a ausência de membros de um conjunto.

- reescrevendo a consulta encontrar todos os clientes que possuam tanto conta quanto empréstimo no banco:

```
select   distinct nome_cliente
from     devedor
where    nome_cliente in (select nome_cliente
                          from depositante)
```

em vez de:

```
(select  nome_cliente
 from    depositante)
intersect
(select  nome_cliente
 from    devedor)
```

- reescrevendo a consulta encontrar todos os clientes que tenham tanto conta quanto um empréstimo na agência Lages:

```
select   nome_cliente
from     devedor, empréstimo
where    devedor.nro_empréstimo=empréstimo.nro_empréstimo and
         nome_agência = "Lages" and
         (nome_agência, nome_cliente) in
            (select  nome_agência, nome_cliente
             from    depositante, conta
             where   depositante.nro_conta = conta.nro_conta)
```

- para encontrar todos os clientes que tenham um empréstimo no banco, mas que não tenham uma conta, podemos escrever:

```
select   distinct nome_cliente
from     devedor
where    nome_cliente not in     (select nome _cliente
                                  from depositante)
```

- selecionar os nomes dos clientes que possuam um empréstimo no banco e cujos nomes não sejam nem "José Silva" e "João Souza":

```
select   distinct nome_cliente
from     devedor
where    nome_cliente not in ("José Silva", "João Souza")
```

5.2.2. Comparação de conjuntos

Considerando a consulta "*encontrar os nomes das agências que tenham fundos maiores que ao menos uma agência localizada em Blumenau*", temos:

```
select   nome_agência
from     agência
where    fundos > some (select   fundos
                        from     agência
                        where    cidade_agência = "Blumenau")
```

A comparação `> some` (maior que algum) é verdadeira se o valor dos `fundos` da tupla for maior que ao menos um membro do conjunto de todos os valores de `fundos` da agência Blumenau.

Outras comparações possíveis:

`< some`
`<= some`
`>= some`
`= some` (igual a `in`)
`<> some`

A palavra chave `any` é sinônimo de `some` em SQL.

- encontrar os nomes de todas as agências que tenham fundos maiores que cada uma das agências de Blumenau: (`> all` = maior que todos)

```
select nome_agência
```

```
from     agência
where    fundos > all (select    fundos
                       from      agência
                       where     cidade_agência = "Blumenau")
```

Outras comparações possíveis:
< *all*
<= *all*
>= *all*
= *all*
<> *all* (igual a *not in*)

- encontrar a agência que tem o maior saldo médio:

Considerando que funções agregadas não podem ser aninhadas (max (avg (...)) não pode ser usado, por exemplo), escrevemos primeiramente uma consulta para encontrar todos os saldos médios e depois aninhamos com uma subconsulta de uma consulta maior que descubra as agências para cada saldo médio maior ou igual a todos os saldos médios:

```
select   nome_agência
from     conta
group by nome_agência
having   avg (saldo) >= all (select    avg (saldo)
                             from      conta
                             group by  nome_agência)
```

5.2.3. Verificação de relações vazias

A SQL possui meios para testar se o resultado de uma subconsulta possui alguma tupla. O construtor *exists* retorna o valor **true** (verdadeiro) se o argumento de uma subconsulta não for vazio.

- encontrar todos os clientes, que possuam tanto conta quanto empréstimo no banco:

```
select   nome_cliente
from     devedor
where    exists (select    *
                 from      depositante
                 where     depositante.nome_cliente =
                           devedor.nome_cliente)
```

O construtor *not exists* testa a não existência de tuplas na subconsulta.

- encontrar todos os clientes que possuam uma conta em todas as agências localizadas em Blumenau:

```
select   distinct S.nome_cliente
from     depositante as S
where    not exists ((select    nome_agência
                      from      agência
                      where     cidade_agência = "Blumenau")
                     except
                     (select    R.nome_agência
                      from      depositante as T, conta as R
```

```
                    where      T.nro_conta = R.nro_conta and
                           S.nome_cliente = T.nome_cliente))
```

A primeira subconsulta encontra todas as agências de Blumenau. A segunda subconsulta encontra todas as agências nas quais o cliente `S.nome_cliente` tem uma conta.

5.2.4. Teste para a ausência de tuplas repetidas

O construtor *unique* retorna o valor **true** caso o argumento da subconsulta não possua nenhuma tupla repetida.

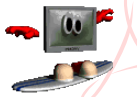
- encontrar todos os clientes que possuam uma conta na agência Lages:

```
select   T.nome_cliente
from     depositante as T
where    unique (select    R.nome_cliente
                 from      conta, depositante as R
                 where     T.nome_cliente = R.nome_cliente and
                           R.nro_conta = conta.nro_conta and
                           conta.nome_agência = “Lages”)
```

O construtor *not unique* permite testar a existência de tuplas repetidas em uma subconsulta.

- encontrar todos os clientes que tenham ao menos duas contas na agência Lages:

```
select   distinct T.nome-cliente
from     depositante as T
where    not unique (select  R.nome_cliente
                    from    conta, depositante as R
                    where   T.nome_cliente=R.nome_cliente and
                            R.nro_conta = conta.nro_conta and
                            conta.nome_agência = “Lages”)
```



Digitally signed
by Cleber Filippin
Location: Lages
Date: 2005.03.09
10:10:22 -03'00'