Aula 10 – BD 1
SQL Parte 2

Profa. Elaine Faria

UFU - 2018

Consultas Aninhadas

- É um dos recursos mais poderosos da SQL
- É aquela que tem outra consulta embutida dentro dela
  - A consulta embutida é chamada de subconsulta
  - A consulta embutida pode ser uma consulta aninhada

Introdução a Consultas Aninhadas

- Reescrever a C1 usando uma subconsulta

  Encontre os nomes de marinheiros que reservaram o barco 103

```
SELECT M.nome_marin
FROM Marinheiros M
WHERE M.id_marin IN ( SELECT R.id_marin
                      FROM Reservas R
                      WHERE R.id_barco = 103)
```

Introdução a Consultas Aninhadas

- Operador IN
  - Testar se um valor está em um determinado conjunto de elementos
- Modificação da consulta C1 para encontrar todos os marinheiros que não reservaram o barco 103
  - Substituir o IN por NOT IN

Introdução a Consultas Aninhadas

- C2: Encontre os nomes dos marinheiros que reservaram um barco vermelho

```
SELECT M.nome_marin
FROM Marinheiros M
WHERE M.id_marin IN (SELECT R.id_marin
                     FROM  Reservas R
                     WHERE R.id_barco IN (
                           SELECT B.id_barco
                           FROM Barcos B
                           WHERE B.cor='vermelho')
```

Introdução a Consultas Aninhadas

- C21: Encontre os nomes dos marinheiros que não reservaram um barco vermelho

```
SELECT M.nome_marin
FROM Marinheiros M
WHERE M.id_marin NOT IN (SELECT R.id_marin
                         FROM  Reservas R
                         WHERE R.id_barco IN (
                             SELECT B.id_barco
                             FROM Barcos B
                             WHERE B.cor='vermelho'))
```

Introdução a Consultas Aninhadas

- Consulta aninhada C2: substituir IN (interno) por NOT IN ??
  - Nomes dos marinheiros que reservaram um barco que não é vermelho
- Consulta aninhada C2: substituir ambas as ocorrências de IN por NOT IN
  - Nomes dos marinheiros que não reservaram um barco que não é vermelho, ou seja, reservaram apenas barcos vermelhos, se é que reservaram algum barco

Consultas aninhadas correlacionadas

- C1: Encontre os nomes de marinheiros que reservaram o barco 103

```
SELECT M.nome_marin
FROM Marinheiros M
WHERE EXISTS (SELECT *
                    FROM Reservas R
                    WHERE R.id_barco = 103 AND
                    R.id_marin = M.id_marin)
```

Consultas aninhadas correlacionadas

- EXISTS é um operador de comparação
  - Testa se um conjunto é não vazio
  - A subconsulta anterior C1 depende da linha atual M e deve ser reavaliada para cada linha de Marinheiros
- Usar NOT EXISTS na consulta C1
  - Nomes dos marinheiros que não reservaram o barco 103

Consultas aninhadas

- Consultas aninhadas correlacionadas ocorrem quando o resultado da subconsulta (consulta interna) muda de acordo com a tupla que está sendo avaliada na consulta externa.
  - Cuidado: isso pode ser muito lento, pois a subconsulta é reavaliada para cada linha da consulta externa. Se houver forma de evitar isso, sua consulta pode ser mais rápida

Operadores de Comparação de Conjuntos

- C22: Encontre os marinheiros cujas avaliações sejam melhores do que as de algum marinheiro chamado Horacio

```
SELECT M.id_marin
FROM Marinheiros M
WHERE M.avaliação > ANY (SELECT M2.avaliação
                         FROM Marinheiros M2
                         WHERE M2.nome_marin='Horacio')
```

Operadores de Comparação de Conjuntos

- Na consulta C22

  Se não houver nenhum marinheiro chamado Horacio → M.avaliação > Any ... é definida para retornar falso e a consulta retorna vazio

Operadores de Comparação de Conjuntos

- C23: Encontre os marinheiros cujas avaliações sejam melhores do que a de todo marinheiro chamado Horacio

  Modificar a consulta C22 substituindo ANY por ALL

  Se não houvesse nenhum marinheiro chamado Horacio a comparação M.avaliação > ALL... seria definida para retornar verdadeiro → a consulta retornaria o nome de todos os marinheiros

Operadores de Comparação de Conjuntos

- C24: Encontre os marinheiros com a maior avaliação

```
SELECT M.id_marin
FROM Marinheiros M
WHERE M.avaliação >= ALL (
            SELECT M2.avaliação
            FROM Marinheiros M2)
```

Operadores de Comparação de Conjuntos

- Observações
  - IN é equivalente a = ANY
  - NOT IN é equivalente a <> ALL

Mais exemplos de consultas aninhadas

- C6: Encontre os nomes dos marinheiros que reservaram um barco vermelho e um barco verde

```
SELECT M.nome_marin
FROM Marinheiros M, Reservas R, Barcos B
WHERE M.id_marin = R.id_marin AND R.id_barco =
    B.id_barco AND B.cor='vermelho' AND M.id_marin IN
                (SELECT M2.id_marin
                FROM Marinheiros M2, Barcos B2, Reservas R2
                WHERE M2.id_marin = R2.id_marin AND
                        R2.id_barco = B2.id_barco AND
                        B2.cor='verde')
```

Mais exemplos de consultas aninhadas

- A consulta C6 ilustra como as consultas envolvendo INTERSECT podem ser reescritas usando IN

- As consultas usando EXCEPT podem ser reescritas usando NOT IN
  - Encontrar os nomes dos marinheiros que reservaram barcos vermelhos, mas não reservaram barcos verdes → substituir IN por NOT IN

Mais exemplos de consultas aninhadas

- C6 usando INTERSECT

```
SELECT M.nome_marin
FROM Marinheiros M
WHERE M.id_marin IN ((SELECT R.id_marin
                      FROM Barcos B, Reservas R
                      WHERE R.id_barco = R.id_barco
                      AND B.cor = ‘vermelho’)
                     INTERSECT
                      (SELECT R2.id_marin
                      FROM Barcos B2, Reservas R2
                     WHERE R2.id_barco = B2.id_barco
                           AND B2.cor=‘verde’))
```

Mais exemplos de consultas aninhadas

- C9: Encontre os nomes dos marinheiros que reservaram todos os barcos

```
SELECT M.nome_marin
FROM Marinheiros M
WHERE NOT EXISTS ((SELECT B.id_barco
                   FROM Barcos B)
                   EXCEPT
                  (SELECT R.id_barco
                   FROM Reservas
                  WHERE R.id_marin = M.id_marin))
```

Operadores Agregados

- A SQL permite o uso de expressões aritméticas
- A SQL suporta 5 operações agregadas que pode ser aplicadas a uma coluna A
  - COUNT ([DISTINCT]): o número de valores (únicos) da coluna A
  - SUM ([DISTINCT]): a soma de todos os valores (únicos) da coluna A
  - AVG (([DISTINCT]): a média de todos os valores (únicos) de A
  - MAX (A): o valor máximo da coluna A
  - MIN (A): o valor mínimo da coluna A

Operadores Agregados

- C25: Encontre a idade média de todos os marinheiros

```
SELECT AVG(M.idade)
FROM Marinheiros M
```

Operadores Agregados

- C26: Encontre a idade média de todos os marinheiros com avaliação 10

```
SELECT AVG (M.idade)
FROM Marinheiros M
WHERE M.avaliação = 10
```

MIN ou MAX podem ser usadas para encontrar a idade do marinheiro mais jovem ou mais velho

Operadores Agregados

- C27: Encontre o nome e a idade do marinheiro mais velho

```
SELECT M.nome_marin, Max(M.idade)
FROM Marinheiros M
```

Essa consulta é ilegal em SQL!

Se a cláusula SELECT usa uma operação, então ela deve usar apenas operações agregadas, a menos que a consulta contenha a cláusula GROUP BY

Operadores Agregados

- C27 deve usar uma consulta aninhada

```
SELECT M.nome_marin, M.idade
FROM Marinheiros M
WHERE M.idade = (SELECT MAX (M2.idade)
                 FROM Marinheiros M2)
```

Operadores Agregados

- C28: Conte o número de marinheiros

```
SELECT Count(*)
FROM Marinheiros M
```

Operadores Agregados

- C29: Conte o número de nomes diferentes de marinheiros

```
SELECT COUNT (DISTINCT M.nome_marin)
FROM Marinheiros M
```

Operadores Agregados

- C30: Encontre os nomes dos marinheiros que são mais velhos do que o marinheiro mais velho que tem avaliação 10

```
SELECT M.nome_marin
FROM Marinheiros M
WHERE M.idade > (SELECT MAX(M2.idade)
                 FROM Marinheiros M2
                 WHERE M2.avaliação = 10)
```

Operadores Agregados

- C30: usando ALL

```
SELECT M.nome_marin
FROM Marinheiros M
WHERE M.idade > ALL (SELECT M2.idade
                     FROM Marinheiros M2
                     WHERE M2.avaliação=10)
```

Referências

- R. Ramakrishnan e J. Gehrke, *Database Management Systems,* 3a Edição, McGraw-Hill, 2003.