Aula 11 – SBD
SQL Parte 3

Profa. Elaine Faria

UFU - 2018

Group by e *Having*

- Às vezes deseja-se aplicar operações agregadas a cada um dos vários grupos de linhas em um relação

Group by e *Having*

- C31: Encontre a idade do marinheiro mais jovem para cada nível de avaliação

```
SELECT Min(M.idade)
FROM Marinheiros M
WHERE M.avaliacao = i
```

Fazer 10 consultas variando o valor de i de 1 a 10

Group by e *Having*

- C31

  E se não soubermos quantos níveis de avaliação existem?

```
SELECT M.avaliacao, MIN (M.idade)
FROM Marinheiros M
GROUP BY M.avaliacao
```

Group by e *Having*

- Formato geral da consulta com *Group By* e *Having*

SELECT [DISTINCT] lista-seleção
FROM lista-from
WHERE qualificação
GROUP BY lista-agrupamento
HAVING qualificação-grupo

Group by e *Having*

- Lista-seleção
  - Consiste de
    - (1) Uma lista de nomes de colunas
    - (2) Lista de termos tendo o formato **opAg** (nome-coluna) as novo-nome
  - Toda coluna que aparece em (1) também deve aparecer na lista-agrupamento
    - Cada linha no resultado corresponde a um grupo

Group by e *Having*

- As expressões que aparecem na qualificação-grupo da cláusula HAVING devem ter um único valor por grupo
  - *Having*: determina se uma linha de resposta deve ser gerada para determinado grupo
  - Uma coluna que aparece na qualificação-grupo deve aparecer como argumento de um operador de agregação ou na lista-agrupamento
- Se o GROUP BY é omitido, a tabela inteira é considerada um único grupo

GROUP BY: atenção!

- A partir do SQL 1999, é possível colocar outros atributos que não estão agrupados no SELECT desde que o atributo usado no agrupamento seja uma chave

```
SELECT M.id_marin, M.nome_marin, count(*)
FROM Reservas R, Marinheiros M
WHERE R.id_marin = M.id_marin
GROUP BY M.id_marin
```

Group by e *Having*

- C32: Encontre a idade do marinheiro mais jovem que pode votar (ou seja, tenha no mínimo 18 anos) para cada nível de avaliação com no mínimo dois marinheiros desse tipo

```
SELECT M.avaliacao, Min(M.idade) as minIdade
FROM Marinheiros M
WHERE M.idade >=18
GROUP BY M.avaliacao
HAVING COUNT(*) > 1
```

Group by e *Having*

- Etapas para avaliação da consulta C32
  - Construir o produto cartesiano das tabelas da lista-from
  - Aplicar a qualificação da cláusula WHERE
  - Eliminar as colunas indesejadas
    - Apenas as colunas na cláusula SELECT, GROUP BY e HAVING são necessárias
  - Ordenar a tabela de acordo com a cláusula GROUP BY
  - Aplicar a qualificação-grupo da cláusula HAVING
  - Gerar uma linha de resposta para cada grupo remanescente
  - Se a consulta contém DISTINCT as duplicatas são eliminadas em uma etapa adicional e final

Group by e *Having*

- SQL:1999 introduziu duas novas funções de conjunto → EVERY e ANY
  - Ex: HAVING COUNT (*) > 1 AND EVERY (M.idade <=60)

```
SELECT M.avaliacao, Min(M.idade) AS minIdade
FROM Marinheiros M
WHERE M.idade >=18
GROUP BY M.avaliacao
HAVING COUNT(*) > 1 AND EVERY(M.idade <=60)
```

Group by e *Having*

- Condição sobre a idade no WHERE ou no HAVING → Qual a diferença?

```
SELECT M.avaliacao, Min(M.idade) AS minIdade
FROM Marinheiros M
WHERE M.idade >=18 AND M.idade <=60
GROUP BY M.avaliacao
HAVING COUNT(*) > 1
```

Group by e *Having*

- Condição sobre a idade no WHERE ou no HAVING → Qual a diferença?

Avaliação	Idade
7	45
1	33
8	55
8	25
10	35
7	35
9	35
3	25
3	63
3	25

Mais exemplos de consultas agregadas

- C33: Para cada barco vermelho, encontre o número de reservas desse barco

```
SELECT B.id-barco, COUNT(*) AS contagemR
FROM Barcos B, Reservas R
WHERE R.id-barco = B.id-barco AND
            B.cor='vermelho'
GROUP BY B.id-barco
```

Mais exemplos de consultas agregadas

```
SELECT B.id-barco, COUNT(*) AS contagemR
FROM Barcos B, Reservas R
WHERE R.id-barco = B.id-barco
GROUP BY B.id-barco
HAVING B.cor='vermelho'
```

Essa consulta é legal?

Não! Apenas as colunas que aparecem no GROUP BY podem aparecer no HAVING, a menos que elas apareçam como argumentos de um operador agregado

Mais exemplos de consultas agregadas

- C34: Encontre a idade média dos marinheiros de cada nível de avaliação que tenha no mínimo dois marinheiros

```
SELECT M.avaliacao, AVG(M.idade) AS idadeM
FROM Marinheiros M
GROUP BY M.avaliacao
HAVING COUNT(*) > 1
```

Mais exemplos de consultas agregadas

- C34

```
SELECT M.avaliacao, AVG(M.idade) as idadeM
FROM Marinheiros M
GROUP BY M.avaliacao
HAVING 1 < (SELECT COUNT(*)
FROM Marinheiros M2
WHERE M.avaliacao = M2. avaliacao)
```

Mais exemplos de consultas agregadas

- C35: Encontre a idade média dos marinheiros que possuem idade mínima de 18 anos para cada nível de avaliação que tenha no mínimo dois marinheiros

```
SELECT M.avaliacao, AVG(M.idade) as idadeM
FROM Marinheiros M
WHERE M.idade >=18
GROUP BY M.avaliacao
HAVING 1 < (SELECT COUNT (*)
                    FROM Marinheiros M2
                    WHERE M.avaliacao = M2.avaliacao)
```

Mais exemplos de consultas agregadas

- C36: Encontre a idade média dos marinheiros que possuem idade mínima de 18 anos para cada nível de avaliação que tenha no mínimo dois marinheiros que satisfazem essa condição

```
SELECT M.avaliacao, AVG(M.idade) as idadeM
FROM Marinheiros M
WHERE M.idade >=18
GROUP BY M.avaliacao
HAVING 1 < (SELECT COUNT (*)
                    FROM Marinheiros M2
                    WHERE M.avaliacao = M2.avaliacao AND
                              M2.idade >=18)
```

Mais exemplos de consultas agregadas

- C36

```
SELECT M.avaliacao, AVG(M.idade) as IdadeM
FROM Marinheiros M
WHERE M.idade >=18
GROUP BY M.avaliação
HAVING COUNT(*) > 1
```

Mais exemplos de consultas agregadas

- É possível realizar consultas sobre os resultados obtidos em outras consultas. Isso pode ser feito adicionando a consulta na cláusula FROM

Mais exemplos de consultas agregadas

- C36

```
SELECT Temp.avaliacao, Temp.idadeMedia
FROM (SELECT M.avaliacao, AVG(M.idade) AS idadeMedia,
        COUNT(*) AS contagemAvaliacao
        FROM Marinheiros M
        WHERE M.idade >=18
        GROUP BY M.avaliacao ) AS TEMP
WHERE Temp.contagemAvaliacao > 1
```

Mais exemplos de consultas agregadas

- C37: Encontre as avaliações para as quais a idade média dos marinheiros seja a mínima considerando todas as avaliações

```
SELECT M.avaliacao
FROM Marinheiros M
WHERE AVG(M.idade) = (SELECT MIN(AVG(M2.idade))
                      FROM Marinheiros M2
                      GROUP BY M2.avaliacao)
```

Essa consulta funciona???

Não!

Mais exemplos de consultas agregadas

- C37

```
SELECT Temp.avaliacao, Temp.idadeMedia
FROM (SELECT M.avaliacao, AVG(M.idade) as idadeMedia
        FROM Marinheiros M
        GROUP BY M.avaliacao) As Temp
WHERE Temp.idadeMedia =
      (SELECT MIN(Temp.idadeMedia) FROM Temp)
```

Mais exemplos de consultas agregadas

- C37 – versão Postgresql

```
SELECT Temp.avaliacao, Temp.idadeMedia
FROM (SELECT M.avaliacao, AVG(M.idade) as idadeMedia
        FROM Marinheiros M
        GROUP BY M.avaliacao) As Temp
WHERE Temp.idadeMedia =
      (SELECT MIN(Temp2.idadeMedia) FROM
            (SELECT M.avaliacao, AVG(M.idade) as idadeMedia
            FROM Marinheiros M
            GROUP BY M.avaliacao) as Temp2
      )
```

Mais exemplos de consultas agregadas

- A consulta abaixo computa o mesmo resultado que a C37?

```
SELECT Temp.avaliacao, MIN(Temp.idadeMedia)
FROM (SELECT M.avaliacao, AVG(M.idade) as idadeMedia
        FROM Marinheiros M
        GROUP BY M.avaliacao) AS TEMP
GROUP BY Temp.avaliacao
```

ORDER BY

- Permite ordenar as linhas retornadas por um comando SELECT em order crescente ou decrescente baseado em um critério especificado
- É preciso especificar a coluna que você deseja ordenar na cláusula ORDER BY
  - Se você deseja ordenar usando múltiplas colunas, use vírgula para separar as colunas
- Use ASC para ordenar de forma crescente ou DESC para decrescente
  - Default: ASC

ORDER BY

```
SELECT id_marin, nome_marin
FROM Marinheiros
ORDER BY Avaliacao DESC
```

```
SELECT id_marin, nome_marin
FROM Marinheiros
ORDER BY Avaliacao,Idade DESC
```

```
SELECT id_marin, nome_marin
FROM Marinheiros
ORDER BY Avaliacao DESC
                 Idade ASC
```

LIMIT

- Permite recuperar apenas uma porção das linhas que são geradas pela consulta
- Limita o nro de resultados
  - É importante usar a cláusula ORDER BY

```
SELECT DISTINCT idade
FROM Marinheiros
ORDER BY idade DESC
LIMIT 3
```

OFFSET

- Indica quantas linhas devem ser removidas do ínicio da solução da consulta
- Indica o início da leitura

```
SELECT DISTINCT idade
FROM Marinheiros
ORDER BY idade DESC
LIMIT 3
OFFSET 2
```

Valores Nulos

- Os valores das colunas podem ser desconhecidos
  - Ex: modificação na tabela Marinheiros para incluir uma coluna nomeSolteiro
    - Apenas as mulheres casadas que assumem o sobrenome do marido têm um sobrenome de solteira
    - Para as mulheres que não assumem o sobrenome do marido e para as homens a coluna nomeSolteiro é não aplicável
- A SQL provê um valor especial de coluna chamado nulo (*null*)
  - Usa-se *null* quando o valor da coluna é desconhecido ou não aplicável

Comparações usando NULL

- Ex: Suponha a linha (98, Dan, null, 39)
  - Comparação avaliacao = 8?
  - Resultado: desconhecido
- IS NULL
  - Testar se o valor de uma coluna é nulo
- IS NOT NULL

Conectivos Lógico AND, OR e NOT

- Ex: Suponha a linha (98, Dan, nulo, 39)
  - Qual o resultado da expressão avaliacao=9 OR idade < 40? → verdadeiro
  - Qual o resultado da expressão avaliacao=8 AND idade < 40? → desconhecido
- Comparação de dois valores desconhecidos → desconhecido
- Extensão da interpretação de AND, OR e NOT
  - NOT desconhecido → desconhecido
  - OR de dois argumentos é desconhecido se um dos argumentos for falso e o outro desconhecido
  - AND de dois argumentos é desconhecido se um argumento for desconhecido e o outro verdadeiro ou desconhecido

Impactos nos Construtores da SQL

- WHERE
  - Na presença de valores nulos, qualquer linha avaliada como falso ou desconhecido é eliminada
- Duplicatas
  - Duas linhas são duplicatas se as colunas correspondentes são iguais ou se ambas contêm nulo
- Operadores +, - , * e /
  - Retornam nulo se um dos argumentos for nulo
- Count(*)
  - Trata os valores nulo como os demais valores → eles são contados
- Todas as outras operações agregadas
  - Têm uma etapa preliminar para descartar os valores nulos

Junções Externas

- Considere a junção $M \bowtie_c R$
  - As tuplas de Marinheiros que não correspondem a alguma linha em Reservas, de acordo com a condição c não aparecem no resultado

- Em uma junção externa
  - As linhas de Marinheiros sem uma linha em Reservas correspondente aparecem exatamente uma vez no resultado com as colunas de Resultado herdadas de Reserva com os valores nulos

Junções Externas

- Junção externa esquerda
  - As linhas de Marinheiros sem uma linha de Reservas correspondente aparecem no resultado, mas não vice-versa
- Junção externa direita
  - As linhas de Reservas sem uma linha de Marinheiros correspondente aparecem no resultado, mas não vice-versa
- Junção externa completa
  - As linhas de Marinheiros e de Reservas sem correspondência aparecem no resultado

As linhas com uma correspondência sempre aparecem no resultado em todas as variações

Junções Externas

- O tipo de junção pode ser especificado na cláusula FROM

SELECT M.id_marin, R.id_barco

FROM Marinheiros M NATURAL LEFT OUTER JOIN Reservas R

Natural: significa que a condição de junção é igualdade em todos os atributos comuns e a cláusula WHERE não é exigida

Junções Externas

- No PostgreSQL temos:
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN

  Ex:

```
SELECT *
FROM Marinheiros M FULL OUTER JOIN Reservas R ON
   R.id_marin = M.id_marin

SELECT *
FROM Marinheiros M NATURAL LEFT OUTER JOIN Reservas R
```

Referências

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