SQL: Consultas

André Luis Schwerz
andreluis@utfpr.edu.br

Universidade Tecnológica Federal do Paraná

Banco de Dados 1
2017/1

Agenda

1. Contextualização
2. SQL:1999
3. Visão Geral da SQL
   - Formato de uma consulta básica
   - Exemplos de consultas básicas
4. UNION, INTERSEC, EXCEPT
5. Consultas Aninhadas
   - Consultas Aninhadas Correlacionadas
6. Operadores Agregados
   - GROUP BY e HAVING
7. Valores Nulos
8. Conclusão

Objetivos

Entender:

- O que é SQL:1999
- O que está incluído na SQL
- Como são as consultas em SQL

Agenda

1. Contextualização
2. SQL:1999
3. Visão Geral da SQL
4. UNION, INTERSEC, EXCEPT
5. Consultas Aninhadas
6. Operadores Agregados
7. Valores Nulos
8. Conclusão

Contextualização

- SQL inclui:
  - DML — consulta, inserção, exclusão e alteração de tuplas (registros)
  - DDL — criação, exclusão e alteração das definições de tabelas e visões
  - Gatilhos e restrições de integridade avançadas
  - SQL embutida e dinâmica
  - Execução Cliente-Servidor e acesso a BD Remoto
  - Gerenciamento de **transação**
  - Segurança
  - Recursos avançados — OO, consultas recursivas, consultas de apoio a decisão, mineração de dados, dados espaciais, gerenciamento de texto e dados XML

Agenda

1. Contextualização
2. SQL:1999
3. Visão Geral da SQL
4. UNION, INTERSEC, EXCEPT
5. Consultas Aninhadas
6. Operadores Agregados
7. Valores Nulos
8. Conclusão

SQL:1999

Conformidade com os Padrões SQL: O SQL:1999 tem uma coleção de recursos chamada Core SQL, que um fabricante deve implementar para alegar conformidade com o padrão SQL:1999. É estimado que todos os principais fabricanes podem se ajustar ao Core SQL com pouco esforço. Muitos dos recursos restantes estão organizados em **pacotes**.

Por exemplo, os pacotes tratam cada um dos seguintes recursos (com os capítulos relevantes entre parênteses): *data e horário melhorados, gerenciamento de integridade melhorado e banco de dados ativos* (neste capítulo), *interfaces de linuagem externa* (Capítulo 6), *OLAP* (Capítulo 25), e *recursos de objeto* (Capítulo 23). O padrão SQL/MM complementa o SQL:1999 definindo pacotes adicionais que suportam a *mineração de dados* (Capítulo 26), *dados espaciais* (Capítulo 28) e *documentos de texto* (Capítulo 27). O suporte aos dados e consultas XML está por vir.
`http://web.cecs.pdx.edu/~len/sql1999.pdf`

Agenda

1. Contextualização
2. SQL:1999
3. Visão Geral da SQL
   - Formato de uma consulta básica
   - Exemplos de consultas básicas
4. UNION, INTERSEC, EXCEPT
5. Consultas Aninhadas
6. Operadores Agregados
7. Valores Nulos
8. Conclusão

Visão Geral

Sobre os exemplos

```
Marinheiros(id-marin: integer, nome-marin: string, avaliacao: integer,
    idade: real)
```

```
Barcos(id-barco: integer, nome-barco: string, cor: string)
```

```
Reservas(id-marin: integer, id-barco: integer, dia: date)
```

Formato de uma consulta básica

```
SELECT [ DISTINCT ] lista-selecao
FROM                lista-from
WHERE               qualificacao
```

Tabela: **Instância M3 de Marinheiros**

id-marin	nome-marin	avaliação	idade
22	Dustin	7	45,0
29	Brutus	1	33,0
31	Lubber	8	55,5
32	Andy	8	25,5
58	Rusty	10	35,0
64	Horatio	7	35,0
71	Zorba	10	16,0
74	Horatio	9	35,0
85	Art	3	25,5
95	Bob	3	63,5

Tabela: **Instância R2 de Reservas**

id-marin	id-barco	data
22	101	10/10/98
22	102	10/10/98
22	103	10/08/98
22	104	10/07/98
31	102	11/10/98
31	103	11/06/98
31	104	11/12/98
64	101	09/05/98
64	102	09/05/98
74	103	09/05/98

Formato de uma consulta básica

Tabela: **Instância B1 de Barcos**

id-barco	nome-barco	cor
101	Interlake	azul
102	Interlake	vermelho
103	Clipper	verde
104	Marine	vermelho

(C15) Encontre os nomes e as idades de todos os marinheiros.

```
SELECT  DISTINCT    M.nome-marin, M.idade
FROM                Marinheiros M
```

A resposta de uma consulta é sempre um multiconjunto de linhas (ou seja, uma outra relação)

Formato de uma consulta básica

(C15) Encontre os nomes e as idades de todos os marinheiros.

```
SELECT  DISTINCT    M.nome-marin, M.idade
FROM                Marinheiros M
```

Tabela: **C15**

nome-marin	idade
Dustin	45,0
Brutus	33,0
Lubber	55,5
Andy	25,5
Rusty	35,0
Horatio	35,0
Zorba	16,0
Art	25,5
Bob	63,5

Tabela: **C15 sem DISTINCT**

nome-marin	idade
Dustin	45,0
Brutus	33,0
Lubber	55,5
Andy	25,5
Rusty	35,0
Horatio	35,0
Zorba	16,0
Horatio	35,0
Art	25,5
Bob	63,5

Formato de uma consulta básica

(C11) Encontre todos os marinheiros com uma avaliação acima de 7.

```
SELECT  M.id-marin, M.nome-marin, M.avaliacao, M.idade
FROM    Marinheiros AS M
WHERE   M.avaliacao > 7
```

Formato de uma consulta básica

```
SELECT [ DISTINCT ] lista-selecao
FROM                lista-from
WHERE               qualificacao
```

A **lista-from** da cláusula FROM é uma lista de nomes de tabela. Um nome de tabela pode ser seguido por uma **variável de intervalo** *(range variable)*, que é particulamente útil quando o mesmo nome de tabela aparece mais do que uma vez na lista-from.
A **lista-seleção** é uma lista de (expressões envolvendo) nomes de coluna das tabelas nomeadas na lista-from. Os nomes de coluna podem ser prefixados por uma variável de intervalo.
A **qualificação** da cláusula FROM é uma combinação booleana (isto é, uma expressão usando conectivos lógicos AND, OR e NOT) de condições no formato *expressão op expressão*, onde *op* é um dos operadores de comparação $<$, $<=$, $=$, $<>$, $>=$, $>$. Uma *expressão* é um nome de coluna, uma constante ou uma expressão (aritmética ou de string).
A palavra reservada DISTINCT é opcional. Ela indica que a tabela computada como uma resposta a essa consulta não deve conter duplicatas, ou seja, duas cópias da mesma linha. O padrão é que as duplicatas não sejam eliminadas.

Formato de uma consulta básica

- A resposta é dada pela seguinte avaliação conceitual:
  1. Compute o produto cartesiano das tabelas de **lista-from**
  2. Exclua as linhas no produto cartesiano que não satisfazem as condições de **qualificação**
  3. Exclua as colunas que não aparecem na **lista-seleção**
  4. Se **DISTINCT** for especificado, elimine as linhas duplicadas

Formato de uma consulta básica

```
SELECT  M.nome-marin
FROM    Marinheiros M, Reservas R
WHERE   M.id-marin = R.id-marin
        AND R.id-barco = 103
```

Formato de uma consulta básica

- A resposta é dada pela seguinte avaliação conceitual:
  1. Compute o produto cartesiano das tabelas de **lista-from**
  2. Exclua as linhas no produto cartesiano que não satisfazem as condições de **qualificação**
  3. Exclua as colunas que não aparecem na **lista-seleção**
  4. Se **DISTINCT** for especificado, elimine as linhas duplicadas

Formato de uma consulta básica

Tabela: **M4 x R3**

id-marin	nome-marin	avaliação	idade	id-marin	id-barco	dia
22	dustin	7	45,0	22	101	10/10/96
22	dustin	7	45,0	58	103	11/12/96
31	lubber	8	55,5	22	101	10/10/96
31	lubber	8	55,5	58	103	11/12/96
58	rusty	10	35,0	22	101	10/10/96
58	rusty	10	35,0	58	103	11/12/96

Formato de uma consulta básica

- A resposta é dada pela seguinte avaliação conceitual:
  1. Compute o produto cartesiano das tabelas de **lista-from**
  2. Exclua as linhas no produto cartesiano que não satisfazem as condições de **qualificação**
  3. Exclua as colunas que não aparecem na **lista-seleção**
  4. Se **DISTINCT** for especificado, elimine as linhas duplicadas

Formato de uma consulta básica

Tabela: **M4 x R3**

id-marin	nome-marin	avaliação	idade	id-marin	id-barco	dia
22	dustin	7	45,0	22	101	10/10/96
22	dustin	7	45,0	58	103	11/12/96
31	lubber	8	55,5	22	101	10/10/96
31	lubber	8	55,5	58	103	11/12/96
58	rusty	10	35,0	22	101	10/10/96
58	rusty	10	35,0	58	103	11/12/96

Formato de uma consulta básica

- A resposta é dada pela seguinte avaliação conceitual:
  1. Compute o produto cartesiano das tabelas de **lista-from**
  2. Exclua as linhas no produto cartesiano que não satisfazem as condições de **qualificação**
  3. Exclua as colunas que não aparecem na **lista-seleção**
  4. Se **DISTINCT** for especificado, elimine as linhas duplicadas

Formato de uma consulta básica

nome-marin
Rusty

Formato de uma consulta básica

Exemplo de consultas SQL básicas

```
SELECT  nome-marin
FROM    Marinheiros M, Reservas R
WHERE   M.id-marin = R.id-marin
        AND R.id-barco = 103
```

```
SELECT  nome-marin
FROM    Marinheiros, Reservas
WHERE   Marinheiros.id-marin = Reservas.id-marin
        AND id-barco = 103
```

Formato de uma consulta básica

Exemplo de consultas SQL básicas

(C16) Encontre os id-marins dos marinheiros que reservaram um barco vermelho.

```
SELECT  R.id-marin
FROM    Barcos B, Reservas R
WHERE   B.id-barco = R.id-barco
        AND B.cor = 'vermelho'
```

(C2) Encontre os nomes dos marinheiros que reservaram um barco vermelho.

```
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'
```

Formato de uma consulta básica

Exemplo de consultas SQL básicas

(C3) Encontre as cores dos barcos reservados por Lubber.

```
SELECT  B.cor
FROM    Marinheiros M, Reservas R, Barcos B
WHERE   M.id-marin = R.id-marin
        AND R.id-barco = B.id-barco
        AND M.nome-marin = 'Lubber'
```

(C4) Encontre os nomes dos marinheiros que reservaram pelo menos um barco.

```
SELECT  M.nome-marin
FROM    Marinheiros M, Reservas R
WHERE   M.id-marin = R.id-marin
```

- **Lista-seleção** não é apenas uma lista de colunas.
  - Pode ser `expressão AS nome-coluna`
    - `expressão` pode ser uma expressão aritmética ou de string e constantes
    - `nome-coluna` é um novo nome para essa coluna na saída da consulta
  - Pode conter **funções agregadas**
    - Exemplo: `count, sum`
  - Pode haver expressões envolvendo valores de data e hora
  - Embora não seja padrão, implementações de SGBDs podem oferecer funções embutidas
    - Exemplo: `sqrt, sin, mod`

Formato de uma consulta básica

Exemplo de consultas SQL básicas

- Comparações de strings são feitas usando operadores tradicionais: por exemplo, $<$, $>$, $<=$, $>=$
- **Collation** é um conceito genérico usado para ordenar strings em uma ordem não alfabética: por exemplo, ordenar por mês, Janeiro, Fevereiro, Março ...
  - `http://dev.mysql.com/doc/refman/5.7/en/charset-charsets.html`
  - `http://dev.mysql.com/doc/refman/5.7/en/charset-collate.html`
- **Like** é um operador especial para correspondência de padrão
  - Uso dos símbolos coringa: % e _
  - Exemplo: '_AB%'
  - Caracteres em branco são considerados
- No padrão SQL:1999 foi definido o uso de expressões regulares por meio de um comando mais poderoso que o **Like**, o **Similar**
  - `http://dev.mysql.com/doc/refman/5.7/en/regexp.html`

Formato de uma consulta básica

Exemplo de consultas SQL básicas

(C18) Encontre as idades dos marinheiros cujos nomes começam e terminam com B e têm no mínimo três caracteres.

```
SELECT  M.idade
FROM    Marinheiros M
WHERE   M.nome-marin LIKE 'B_%B'
```

Agenda

1. Contextualização
2. SQL:1999
3. Visão Geral da SQL
4. UNION, INTERSEC, EXCEPT
5. Consultas Aninhadas
6. Operadores Agregados
7. Valores Nulos
8. Conclusão

UNION, INTERSECT, EXCEPT

- União, intersecção e diferença de conjuntos
  - Muitos sistemas suportam apenas UNION
  - A palavra MINUS é usada, em alguns sistemas, como sinônimo de EXCEPT
  - Outras operações de Conjunto
    - IN: Verificar se um elemento existe em um conjunto (aceita NOT)
    - ANY, ALL: Comparar um valor com elementos de um conjunto por meio do operador op
    - EXISTS: Verificar se um conjunto é vazio (aceita NOT)

UNION, INTERSECT, EXECPT

(C5) Encontre os nomes dos marinheiros que reservaram um barco vermelho ou 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' OR B.cor = 'verde')
```

Facilmente feito com OR...

UNION, INTERSECT, EXECPT

(C6) Encontre os nomes dos marinheiros que reservaram um barco vermelho ou um barco verde.

Facilmente feito com AND?

```
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 B.cor = 'verde')
```

Não é facilmente feito com AND...

UNION, INTERSECT, EXECPT

Considere agora a seguinte consulta:

(C6) Encontre os nomes dos marinheiros que reservaram um barco vermelho ou um barco verde.

```
SELECT  M.nome-marin
FROM    Marinheiros M, Reservas R1, Barcos B1,  Reservas R2, Barcos
     B2
WHERE   M.id-marin = R1.id-marin
        AND R1.id-barco = B1.id-barco
        AND M.id-marin = R2.id-marin
        AND R2.id-barco = B2.id-barco
        AND B1.cor = 'vermelho'
        AND B2.cor = 'verde'
```

Solução difícil de entender e ineficiente de executar.

UNION, INTERSECT, EXCEPT

- A consulta C5, substituindo uso do OR por UNION

```
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'
UNION
SELECT  M2.nome-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'
```

UNION, INTERSECT, EXCEPT

- De forma análoga, a consulta C6, substituindo uso do AND (que não funciona), por INTERSECT:

```
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'
INTERSECT
SELECT  M2.nome-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'
```

Em verdade há um erro sutil nessa consulta! Qual?

UNION, INTERSECT, EXCEPT

- Essa consulta exemplifica a diferença de conjuntos:

(C19) Encontre os id-marins de todos os marinheiros que reservaram barcos vermelhos, mas não barcos verdes.

```
SELECT  M.id-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'
EXCEPT
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'
```

UNION, INTERSECT, EXCEPT

(C19) Encontre os id-marins de todos os marinheiros que reservaram barcos vermelhos, mas não barcos verdes.
Como id-marin está em Reservas, uma solução mais simples seria:

```
SELECT  R.id-marin
FROM    Reservas R, Barcos B
WHERE   R.id-barco = B.id-barco
        AND B.cor = 'vermelho'
EXCEPT
SELECT  R.id-marin
FROM    Barcos B2, Reservas R2
WHERE   R2.id-barco = B2.id-barco
        AND B2.cor = 'verde'
```

UNION, INTERSECT, EXCEPT

- UNION, INTERSECT e EXCEPT podem ser usados em duas tabelas quaisquer que sejam compatíveis com a união
  - Mesmo número de colunas com os mesmos tipos (considerando a ordem das colunas)

UNION, INTERSECT, EXCEPT

- UNION, INTERSECT e EXCEPT podem ser usados em duas tabelas quaisquer que sejam compatíveis

(C20) Encontre todos os id-marins de marinheiros que têm uma avaliação 10 ou reservaram o barco 104.

```
SELECT  M.id-marin
FROM    Marinheiros M
WHERE   M.avaliacao = 10
UNION
SELECT  R.id-marin
FROM    Reservas R
WHERE   R.id-barcos = 104
```

UNION, INTERSECT, EXCEPT

- Detalhes de UNION, INTERSECT e EXCEPT
  - DISTINCT é **sempre** aplicado na operações UNION (ou seja, não há linhas duplicadas)
  - Deve-se usar UNION ALL, INTERSECT ALL e EXCEPT ALL para manter as duplicatas.
    - UNION ALL: O número de cópias é m + n, onde m e n são o número de vezes que a linha aparece em ambas as partes da união
    - INTERSECT ALL: O número de cópias é min(m,n)
    - EXCEPT ALL: o número de cópias é m-n, onde m é a primeira relação (tabela)

Agenda

1. Contextualização
2. SQL:1999
3. Visão Geral da SQL
4. UNION, INTERSEC, EXCEPT
5. Consultas Aninhadas
   - Consultas Aninhadas Correlacionadas
6. Operadores Agregados
7. Valores Nulos
8. Conclusão

Consultas Aninhadas

- **Consultas aninhadas** são consultas que possuem outras consultas embutidas
- A consulta embutida é chamada de **subconsulta**
- A consulta embutida pode ser ela própria uma consulta aninhada
- Consultas aninhadas são usadas quando precisamos expressar uma **condição** que se refere a uma tabela que deve ser ela própria computada
  - Essa tabela computada é criada a partir da subconsulta
    - Tipicamente na clausula WHERE, mas também pode existir em FROM e em HAVING (explicada nos slides seguintes)

Consultas Aninhadas

Exemplo

(C1) 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)
```

Operador IN

Para os marinheiros que não reservaram o barco 103, usa-se NOT IN

Consultas aninhadas

Avaliação conceitual modificada

- Para entender uma consulta aninhada a avaliação conceitual deve ser:

1. Compute o produto cartesiano das tabelas de **lista-from**
2. Exclua as linhas no produto cartesiano que não satisfazem as condições de **qualificação**. No entanto, para cada linha do produto cartesiano, ao testar a cláusula WHERE, (re)compute a subconsulta
3. Exclua as colunas que não aparecem na **lista-seleção**
4. **DISTINCT** for especificado, elimine as linhas duplicadas

Porque recomputar, se a subconsulta interior do nosso exemplo não depende da linha "atual" da consulta mais externa?

Consultas aninhadas

Exemplo com múltiplos níveis

(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'))
```

Consultas aninhadas

Exemplo com múltiplos níveis

E se trocarmos a consulta (C2), que encontrava os nomes dos marinheiros que reservaram um barco vermelho, para a seguinte consulta?

```
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'))
```

Consultas aninhadas

Exemplo com múltiplos níveis

O que essa consulta retornaria?

```
SELECT  M.nome-marin
FROM    Marinheiros M
WHERE   M.id-marin IN ( SELECT R.id-marin
            FROM Reservas R
            WHERE R.id-barco NOT IN ( SELECT B.id-barco
                        FROM Barcos B
                        WHERE B.cor = 'vermelho'))
```

Consultas aninhadas

Exemplo com múltiplos níveis

E, ainda, o que essa consulta retornaria?

```
SELECT  M.nome-marin
FROM    Marinheiros M
WHERE   M.id-marin NOT IN ( SELECT R.id-marin
            FROM Reservas R
            WHERE R.id-barco NOT IN ( SELECT B.id-barco
                        FROM Barcos B
                        WHERE B.cor = 'vermelho'))
```

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)
```

EXISTS testa se um conjunto não é vazio.
A ocorrência de M na subconsulta é chamada de correlação.
Uso do * apenas para verificar se uma linha qualificada existe!

Usar NOT EXISTS buscaria os que não reservaram o barco 103.

Usar UNIQUE ao invés de EXISTS retorna verdadeiro caso não haja linhas duplicadas na subconsulta e falso caso contrário.

Consultas aninhadas correlacionadas

Operadores de comparação de conjuntos

- UNIQUE, IN e EXISTS são operadores de comparação de conjuntos
  - Podem ser negados usando NOT
- Existem outros que funcionam juntamente com os operadores de comparação $>$, $<$, $>=$, ...
  - op ANY e op ALL
    - SOME pode ser usado, ele é sinônimo de ANY

Consultas aninhadas correlacionadas

Operadores de comparação de conjuntos

(C22) Encontre os marinheiros cujas avaliações sejam melhores que as de algum marinheiro chamado Horatio

```
SELECT  M.id-marin
FROM    Marinheiros M
WHERE   M.avaliacao>ANY (SELECT  M2.avaliacao
                         FROM    Marinheiros M2
                         WHERE   M2.nome-marin = 'Horatio')
```

- E se não houver marinheiros com o nome Horatio?
  - A comparação com ANY retorna falso

Consultas aninhadas correlacionadas

Operadores de comparação de conjuntos

(C23) Encontre os marinheiros cujas avaliações sejam melhores que a de todo marinheiro chamado Horatio

```
SELECT  M.id-marin
FROM    Marinheiros M
WHERE   M.avaliacao>ALL (SELECT  M2.avaliacao
                         FROM    Marinheiros M2
                         WHERE   M2.nome-marin = 'Horatio')
```

- E se não houver marinheiros com o nome Horatio?
  - A comparação com ALL retorna verdadeiro

Consultas aninhadas correlacionadas

Operadores de comparação de conjuntos

(C24) Encontre os marinheiros com a maior avaliação

```
SELECT  M.id-marin
FROM    Marinheiros M
WHERE   M.avaliacao >= ALL ( SELECT M2.avaliacao
                             FROM Marinheiros M2)
```

Observe que IN e NOT IN é equivalente ao uso de =ANY e <>ALL, respectivamente.

Consultas aninhadas correlacionadas

Mais exemplos

(C6) Encontre os nomes dos marinheiros que reservaram um barco vermelho e um barco verde

```
SELECT  M.id-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')
```

Pode-se usar IN para simular INTERSECT. E, de maneira similar, pode usar NOT IN para simular EXCEPT.

Consultas aninhadas correlacionadas

Mais exemplos

(C6) Encontre os nomes dos marinheiros que reservaram um barco vermelho e um barco verde

```
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 = B.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' ))
```

Usar INTERSECT, nesse caso, faz a consulta ficar mais complexa.

Consultas aninhadas correlacionadas

Mais exemplos

(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 R
                      WHERE     R.id-marin = M.id-marin ))
```

Consultas aninhadas correlacionadas

Mais exemplos

(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
            WHERE  NOT EXISTS (SELECT  R.id-barco
                    FROM    Reservas R
                    WHERE   R.id-barco = B.id-barco
                            AND R.id-marin = M.id-marin ))
```

Solução alternativa sem o uso de EXCEPT. Mais difícil de entender.

Agenda

1. Contextualização
2. SQL:1999
3. Visão Geral da SQL
4. UNION, INTERSEC, EXCEPT
5. Consultas Aninhadas
6. Operadores Agregados
   - GROUP BY e HAVING
7. Valores Nulos
8. Conclusão

Operadores Agregados

1. COUNT ([DISTINCT] A): O número de valores (únicos) da coluna A.
2. SUM ([DISTINCT] A): A soma de todos os valores (únicos) da coluna A.
3. AVG ([DISTINCT A): A média de todos os valores (únicos) da coluna A.
4. MAX (A): O valor máximo da coluna A.
5. MIN (A): O valor mínimo da coluna A.

- O padrão SQL:1999 expande essa lista
  - No entanto esses outros operadores não estão no pacote "core"

Operadores Agregados

Exemplos

(C25) Encontre a idade média de todos os marinheiros.

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

(C26) Encontre a idade média dos marinheiros com avaliação 10.

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

Operadores Agregados

Exemplos

(C27) Encontre o nome e a idade do marinheiro mais velho.

Considere a seguinte tentativa de responder a esta consulta:

```
SELECT  M.nome-marin, MAX (M.idade)
FROM    Marinheiros M
```

O uso do MAX faz a consulta ilegal! A menos que seja usada a clausula GROUP BY (que veremos adiante).

Operadores Agregados

Exemplos

(C27) Encontre o nome e a idade do marinheiro mais velho.

```
SELECT  Mnome-marin, M.idade
FROM    Marinheiros M
WHERE   M.idade = (SELECT MAX (M2.idade)
                   FROM Marinheiros M2)
```

É necessário usar uma subconsulta!

Mas, respostas de consultas não são tabelas!?! Como pode-se comparar uma tabela com um campo?

Operadores Agregados

Exemplos

(C27) Encontre o nome e a idade do marinheiro mais velho.

```
SELECT  Mnome-marin, M.idade
FROM    Marinheiros M
WHERE   (SELECT MAX (M2.idade)
         FROM Marinheiros M2) = M.idade
```

Solução similar permitida pela SQL:1999. No entanto, não suportada por muitos SGBDs!

Operadores Agregados

Exemplos

(C28) Conte o número de marinheiros.

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

(C29) Conte o número de nomes diferentes de marinheiros.

```
SELECT  COUNT (DISTINCT M.nome-marin)
FROM    Marinheiros M
```

Operadores Agregados

Exemplos

(C30) Encontre os nomes dos marinheiros que são mais velhos do que o marinheiros 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.avaliacao = 10 )
```

Uso de uma consulta aninhada para substituir ALL.

Operadores Agregados

Exemplos

(C30) Encontre os nomes dos marinheiros que são mais velhos do que o marinheiros mais velho que tem avaliação 10.

```
SELECT  M.nome-marin
FROM    Marinheiros M
WHERE   M.idade > ALL ( SELECT M2.idade
                        FROM Marinheiros M2
                        WHERE M2.avaliacao = 10 )
```

Como ficaria usando ALL.

Operadores Agregados

GROUP BY e HAVING

- Em alguns casos é necessário aplicar operações agregadas em **grupos** de linhas ao invés de em cada linha
- Por exemplo, considere:

(C31) Encontre a idade do marinheiro mais jovem para cada nível de avaliação.

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

Sabendo que avaliação é um inteiro que vai de 1..10! Escrever 10 consultas como essa!!! Tedioso!!

Operadores Agregados

GROUP BY e HAVING

```
SELECT      [ DISTINCT ] lista-selecao
FROM        lista-from
WHERE       qualificacao
GROUP BY    lista-agrupamento
HAVING      qualificacao-grupo
```

Sintaxe completa do SELECT!

Operadores Agregados

GROUP BY e HAVING

(C31) Encontre a idade do marinheiro mais jovem para cada nível de avaliação.

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

Solução com GROUP BY!
E se houver NULL no campo usado na clausula GROUP BY?

Operadores Agregados

GROUP BY e HAVING

```
SELECT      [ DISTINCT ] lista-selecao
FROM        lista-from
WHERE       qualificacao
GROUP BY    lista-agrupamento
HAVING      qualificacao-grupo
```

- Em **lista-seleção** existem:
  1. Uma lista de nome de colunas
  2. Uma lista de termos `opag (nome_coluna) AS novo_nome`
- Toda coluna que aparece em 1 deve também aparecer em lista-agrupamento

Operadores Agregados

GROUP BY e HAVING

```
SELECT      [ DISTINCT ] lista-selecao
FROM        lista-from
WHERE       qualificacao
GROUP BY    lista-agrupamento
HAVING      qualificacao-grupo
```

- Algumas vezes pode-se usar uma chave primária para verificar se uma coluna tem valor único em todas as linhas de cada grupo
  - Se a **lista-agrupamento** tem uma chave primária de uma tabela em lista-from, toda coluna dessa tabela tem valor único dentro de cada grupo
  - Em SQL:1999 é permitido que tais colunas apareçam em 1 **lista-seleção**

Operadores Agregados

GROUP BY e HAVING

```
SELECT      [ DISTINCT ] lista-selecao
FROM        lista-from
WHERE       qualificacao
GROUP BY    lista-agrupamento
HAVING      qualificacao-grupo
```

- As expressões que aparecem na **qualificação-grupo** da cláusula HAVING devem ter um único valor por grupo
  - Em SQL-92 uma coluna que aparece em **qualificação-grupo** deve aparecer como argumento de um operador de agregação ou deve aparecer em **lista-agrupamento**
  - Em SQL-1999 duas novas funções de conjunto foram introduzidas que permite verificar se EVERY ou ANY linha em um grupo satisfaz uma condição

Operadores Agregados

GROUP BY e HAVING

```
SELECT      [ DISTINCT ] lista-selecao
FROM        lista-from
WHERE       qualificacao
GROUP BY    lista-agrupamento
HAVING      qualificacao-grupo
```

- Se GROUP BY for omitido, a tabela inteira é considerada um único grupo

- Explicando a semântica considerando a instância M3

(C32) Encontre a idade do marinheiro mais jovem que 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
```

Tabela: **Instância M3 de Marinheiros**

id-marin	nome-marin	avaliação	idade
22	Dustin	7	45,0
29	Brutus	1	33,0
31	Lubber	8	55,5
32	Andy	8	25,5
58	Rusty	10	35,0
64	Horatio	7	35,0
71	Zorba	10	16,0
74	Horatio	9	35,0
85	Art	3	25,5
95	Bob	3	63,5
96	Frodo	3	25,5

Operadores Agregados

GROUP BY e HAVING

- Para entender uma consulta aninhada a avaliação conceitual deve ser:
  1. Compute o produto cartesiano das tabelas de **lista-from**
  2. Exclua as linhas no produto cartesiano que não satisfazem as condições de **qualificação**. No entanto, para cada linha do produto cartesiano, ao testar a cláusula WHERE, (re)compute a subconsulta
  3. Exclua as colunas que não aparecem na **lista-seleção** considerando que apenas as colunas que aparecem em SELECT, GROUP BY e HAVING são necessárias
  4. Ordene a tabela resultante de acordo com GROUP BY para identificar os grupos
  5. Aplique a condição que está em **qualificação-grupo** da cláusula HAVING
  6. Gere uma linha de resposta para cada grupo remanescente considerando os operadores de agregação
  7. Se **DISTINCT** for especificado no SELECT, elimine as linhas duplicadas

- Para entender uma consulta aninhada a avaliação conceitual deve ser:
  1. Compute o produto cartesiano das tabelas de **lista-from**
  2. Exclua as linhas no produto cartesiano que não satisfazem as condições de **qualificação**. No entanto, para cada linha do produto cartesiano, ao testar a cláusula WHERE, (re)compute a subconsulta
  3. Exclua as colunas que não aparecem na **lista-seleção** considerando que apenas as colunas que aparecem em SELECT, GROUP BY e HAVING são necessárias
  4. Ordene a tabela resultante de acordo com GROUP BY para identificar os grupos
  5. Aplique a condição que está em **qualificação-grupo** da cláusula HAVING
  6. Gere uma linha de resposta para cada grupo remanescente considerando os operadores de agregação
  7. Se **DISTINCT** for especificado no SELECT, elimine as linhas duplicadas

Formato de uma consulta básica

Tabela: **Instância M3 de Marinheiros**

id-marin	nome-marin	avaliação	idade
22	Dustin	7	45,0
29	Brutus	1	33,0
31	Lubber	8	55,5
32	Andy	8	25,5
58	Rusty	10	35,0
64	Horatio	7	35,0
71	Zorba	10	16,0
74	Horatio	9	35,0
85	Art	3	25,5
95	Bob	3	63,5
96	Frodo	3	25,5

- Para entender uma consulta aninhada a avaliação conceitual deve ser:
  1. Compute o produto cartesiano das tabelas de **lista-from**
  2. Exclua as linhas no produto cartesiano que não satisfazem as condições de **qualificação**. No entanto, para cada linha do produto cartesiano, ao testar a cláusula WHERE, (re)compute a subconsulta
  3. Exclua as colunas que não aparecem na **lista-seleção** considerando que apenas as colunas que aparecem em SELECT, GROUP BY e HAVING são necessárias
  4. Ordene a tabela resultante de acordo com GROUP BY para identificar os grupos
  5. Aplique a condição que está em **qualificação-grupo** da cláusula HAVING
  6. Gere uma linha de resposta para cada grupo remanescente considerando os operadores de agregação
  7. Se **DISTINCT** for especificado no SELECT, elimine as linhas duplicadas

Formato de uma consulta básica

Tabela: **Instância M3 de Marinheiros**

id-marin	nome-marin	avaliação	idade
22	Dustin	7	45,0
29	Brutus	1	33,0
31	Lubber	8	55,5
32	Andy	8	25,5
58	Rusty	10	35,0
64	Horatio	7	35,0
71	Zorba	10	16,0
74	Horatio	9	35,0
85	Art	3	25,5
95	Bob	3	63,5
96	Frodo	3	25,5

¡ que 18 anos.

- Para entender uma consulta aninhada a avaliação conceitual deve ser:
  1. Compute o produto cartesiano das tabelas de **lista-from**
  2. Exclua as linhas no produto cartesiano que não satisfazem as condições de **qualificação**. No entanto, para cada linha do produto cartesiano, ao testar a cláusula WHERE, (re)compute a subconsulta
  3. Exclua as colunas que não aparecem na **lista-seleção** considerando que apenas as colunas que aparecem em SELECT, GROUP BY e HAVING são necessárias
  4. Ordene a tabela resultante de acordo com GROUP BY para identificar os grupos
  5. Aplique a condição que está em **qualificação-grupo** da cláusula HAVING
  6. Gere uma linha de resposta para cada grupo remanescente considerando os operadores de agregação
  7. Se **DISTINCT** for especificado no SELECT, elimine as linhas duplicadas

Operadores Agregados

GROUP BY e HAVING

avaliação	idade
7	45,0
1	33,0
8	55,5
8	25,5
10	35,0
7	35,0
10	16,0
9	35,0
3	25,5
3	63,5
3	25,5

Linhas idênticas!

- Para entender uma consulta aninhada a avaliação conceitual deve ser:
  1. Compute o produto cartesiano das tabelas de **lista-from**
  2. Exclua as linhas no produto cartesiano que não satisfazem as condições de **qualificação**. No entanto, para cada linha do produto cartesiano, ao testar a cláusula WHERE, (re)compute a subconsulta
  3. Exclua as colunas que não aparecem na **lista-seleção** considerando que apenas as colunas que aparecem em SELECT, GROUP BY e HAVING são necessárias
  4. Ordene a tabela resultante de acordo com GROUP BY para identificar os grupos
  5. Aplique a condição que está em **qualificação-grupo** da cláusula HAVING
  6. Gere uma linha de resposta para cada grupo remanescente considerando os operadores de agregação
  7. Se **DISTINCT** for especificado no SELECT, elimine as linhas duplicadas

Operadores Agregados

GROUP BY e HAVING

avaliação	idade
1	33,0

3	25,5
3	25,5
3	63,5

7	45,0
7	35,0

8	55,5
8	25,5

9	35,0

10	35,0

- Para entender uma consulta aninhada a avaliação conceitual deve ser:
  1. Compute o produto cartesiano das tabelas de **lista-from**
  2. Exclua as linhas no produto cartesiano que não satisfazem as condições de **qualificação**. No entanto, para cada linha do produto cartesiano, ao testar a cláusula WHERE, (re)compute a subconsulta
  3. Exclua as colunas que não aparecem na **lista-seleção** considerando que apenas as colunas que aparecem em SELECT, GROUP BY e HAVING são necessárias
  4. Ordene a tabela resultante de acordo com GROUP BY para identificar os grupos
  5. Aplique a condição que está em **qualificação-grupo** da cláusula HAVING
  6. Gere uma linha de resposta para cada grupo remanescente considerando os operadores de agregação
  7. Se **DISTINCT** for especificado no SELECT, elimine as linhas duplicadas

Operadores Agregados

GROUP BY e HAVING

avaliação	idade
1	33,0

3	25,5
3	25,5
3	63,5

7	45,0
7	35,0

8	55,5
8	25,5

9	35,0

10	35,0

- Para entender uma consulta aninhada a avaliação conceitual deve ser:
  1. Compute o produto cartesiano das tabelas de **lista-from**
  2. Exclua as linhas no produto cartesiano que não satisfazem as condições de **qualificação**. No entanto, para cada linha do produto cartesiano, ao testar a cláusula WHERE, (re)compute a subconsulta
  3. Exclua as colunas que não aparecem na **lista-seleção** considerando que apenas as colunas que aparecem em SELECT, GROUP BY e HAVING são necessárias
  4. Ordene a tabela resultante de acordo com GROUP BY para identificar os grupos
  5. Aplique a condição que está em **qualificação-grupo** da cláusula HAVING
  6. Gere uma linha de resposta para cada grupo remanescente considerando os operadores de agregação
  7. Se **DISTINCT** for especificado no SELECT, elimine as linhas duplicadas

Operadores Agregados

GROUP BY e HAVING

avaliação	minIdade
3	25,5
7	35,0
8	25,5

- Para entender uma consulta aninhada a avaliação conceitual deve ser:
  1. Compute o produto cartesiano das tabelas de **lista-from**
  2. Exclua as linhas no produto cartesiano que não satisfazem as condições de **qualificação**. No entanto, para cada linha do produto cartesiano, ao testar a cláusula WHERE, (re)compute a subconsulta
  3. Exclua as colunas que não aparecem na **lista-seleção** considerando que apenas as colunas que aparecem em SELECT, GROUP BY e HAVING são necessárias
  4. Ordene a tabela resultante de acordo com GROUP BY para identificar os grupos
  5. Aplique a condição que está em **qualificação-grupo** da cláusula HAVING
  6. Gere uma linha de resposta para cada grupo remanescente considerando os operadores de agregação
  7. Se **DISTINCT** for especificado no SELECT, elimine as linhas duplicadas

- A SQL:1999 introduziu dois novos operadores de conjunto EVERY e ANY
- Substituindo, na consulta anterior, a cláusula HAVING por:
  - HAVING COUNT(*) $>$ 1 AND EVERY (M.idade $<=$ 60)

Operadores Agregados

GROUP BY e HAVING

avaliação	idade
1	33,0

3	25,5
3	25,5
3	63,5

7	45,0
7	35,0

8	55,5
8	25,5

9	35,0

10	35,0

HAVING COUNT(*) $>$ 1 AND EVERY (M.idade $<=$ 60)

avaliação	minIdade
7	35,0
8	25,5

- Contrastando o uso da condição, colocando-a na cláusula WHERE ao invés de na HAVING:

```
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
```

avaliação	minIdade
3	25,5
7	35,0
8	25,5

Operadores Agregados

GROUP BY e HAVING

(C33) Para cada barco vermelho, encontre o número de reservas desse barco.

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

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

Essa última consulta é ilegal, porque?

Operadores Agregados

GROUP BY e HAVING

(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 idadeMedia
FROM        Marinheiros M
GROUP BY    M.avaliacao
HAVING      COUNT (*) > 1
```

avaliação	idadeMédia
3	38,2
7	40,0
8	40,5
10	25,5

Operadores Agregados

GROUP BY e HAVING

Solução alternativa à anterior.

(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 idadeMedia
FROM        Marinheiros M
GROUP BY    M.avaliacao
HAVING      1 < (SELECT COUNT(*)
                 FROM Marinheiros M2
                 WHERE M.avaliacao =  M2.avaliacao)
```

avaliação	idadeMédia
3	38,2
7	40,0
8	40,5
10	25,5

Operadores Agregados

GROUP BY e HAVING

(C35) Encontre a idade média dos marinheiros que possuem idade mínima de 18 anos para cada nível de avaliação que tenha no mínimo dois marinherios.

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

avaliação	idadeMédia
3	38,2
7	40,0
8	40,5
10	35,0

Operadores Agregados

GROUP BY e HAVING

(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 idadeMedia
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)
```

avaliação	idadeMédia
3	38,2
7	40,0
8	40,5

Simplificação da C36

(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 idadeMedia
FROM        Marinheiros M
WHERE       M.idade >= 18
GROUP BY    M.avaliacao
HAVING      COUNT (*) > 1
```

Ainda, outra forma da C36

(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  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
```

Operadores Agregados

GROUP BY e HAVING

(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)
```

Operações agregadas não podem ser aninhadas!
Portanto, essa é uma consulta ilegal!

Exemplo válido da C37

(C37) Encontre as avaliações para as quais a idade média dos marinheiros seja a mínima considerando todas as avaliações.

```
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)
```

Operadores Agregados

GROUP BY e HAVING

Essas consultas computam o mesmo resultado que a solução anterior para 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)
```

```
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
```

Agenda

1. Contextualização
2. SQL:1999
3. Visão Geral da SQL
4. UNION, INTERSEC, EXCEPT
5. Consultas Aninhadas
6. Operadores Agregados
7. Valores Nulos
8. Conclusão

Valores Nulos

- NULL
  - Valor desconhecido ou não aplicável
- Comparações com NULL retornam valor desconhecido
- Operadores especiais
  - IS NULL
  - IS NOT NULL
- Conectivos lógicos AND, OR e NOT

Valores Nulos

- Conectivos lógicos AND, OR e NOT
  - NOT desconhecido = desconhecido
  - OR = verdadeiro, se um dos dois argumentos for verdadeiro e outro desconhecido
  - OR = desconhecido, se um dos dois argumentos for falso e o outro desconhecido
  - OR = desconhecido, se os dois argumentos forem desconhecidos
  - AND = falso, se um dos dois argumentos for falso e o outro desconhecido
  - AND = desconhecido, se um dos dois argumentos for verdadeiro e o outro desconhecido
  - AND = desconhecido, se os dois argumentos forem desconhecidos

Valores Nulos

Impactos nos construtores da SQL

- Quando ocorrer um valor desconhecido em uma clausula WHERE, a linha deve ser considerada válida ou não?
  - Desconsidera-se, mas isso tem um impacto sutil e importante, principalmente em EXISTS e UNIQUE
- Quando duas linhas de uma tabela são consideradas duplicadas?
  - Por definição, em SQL, se as colunas tiverem valores iguais
  - Mas, se compararmos NULL = NULL, o resultado não é desconhecido?
  - No entanto, isso é tratado como verdadeiro, o que é uma anomalia!

Valores Nulos

Impactos nos construtores da SQL

- Os operadores aritméticos $+$, -, * e $/$ retornam NULL se um de seus operandos for NULL
- Mas, count(*) trata NULL exatamente como outros valores, ou seja, eles são contados!
- Todas as outras operações agregadas (SUM, AVG, MIN, MAX, e variações usando DISTINCT) desconsideram o valor NULL

Valores Nulos

Junções Externas

- Junções descartam valores não correspondentes
  - Junção Interna
- Junções externas funcionam de maneira diferente
  - Uma junção externa não requer que os registros de uma tabela possuam registros equivalentes em outra
    - NULL

Valores Nulos

Junções Externas

- Junção Externa
  - Esquerda
  - Direita
  - Completa
- É possível desabilitar valores nulos quando criamos campos em tabelas, usando NOT NULL

Junções

Recordando...

- Junção Interna usando a sintaxe SQL:99
  - INNER JOIN e ON
    - Nomes dos campos usados na junção devem ser especificados

```
SELECT  Marinheiros.nome-marin
FROM    Marinheiros, Reservas
WHERE   Marinheiros.id-marin = Reservas.id-marin
```

```
SELECT      Marinheiros.nome-marin
FROM        Marinheiros
INNER JOIN  Reservas
    ON (Marinheiros.id-marin = Reservas.id-marin)
```

Junções

Recordando...

- Junção Interna usando a sintaxe SQL:99
  - INNER JOIN e USING
    - Nomes dos campos usados na junção não precisam ser especificados se forem iguais nas duas tabelas
    - Pode ser usado quando outros campos das tabelas tem o mesmo nome e não se deseja usá-los na junção
    - Requer que o nome do campo seja colocado sem o nome da tabela

```
SELECT      Marinheiros.nome-marin
FROM        Marinheiros
INNER JOIN  Reservas
    USING (id-marin)
```

Junções

Recordando...

- Junção Interna usando a sintaxe SQL:99
  - NATURAL JOIN
    - Nomes dos campos usados na junção não precisam ser especificados se forem iguais nas duas tabelas
    - Pode ser usado apenas quando todos campos das tabelas tem o mesmo nome participam da junção

```
SELECT          Marinheiros.nome-marin
FROM            Marinheiros
NATURAL JOIN    Reservas
```

Junções

Recordando...

- Junção Externa
  - LEFT OUTER JOIN
    - O resultado desta seleção sempre contém todos os registros da tabela esquerda (isto é, a primeira tabela mencionada na consulta), mesmo quando não exista registros correspondentes na tabela direita
    - Pode-se usar ON, USING ou NATURAL

```
SELECT          Marinheiros.nome-marin
FROM            Marinheiros
LEFT OUTER JOIN Reservas
        ON (Marinheiros.id-marin = Reservas.id-marin)
```

Junções

Recordando...

- Junção Externa
  - RIGHT OUTER JOIN
    - O resultado desta seleção sempre contém todos os registros da tabela direita (isto é, a segunda tabela mencionada na consulta), mesmo quando não exista registros correspondentes na tabela esquerda
    - Pode-se usar ON, USING ou NATURAL

```
SELECT              Marinheiros.nome-marin
FROM                Marinheiros
RIGHT OUTER JOIN    Reservas
    USING (id-marin)
```

- Junção Externa
  - FULL OUTER JOIN
    - O resultado desta seleção apresenta todos os dados das tabelas à esquerda e à direita, mesmo que não possuam correspondência entre as tabelas

```
SELECT  Marinheiros.nome-marin
FROM    Marinheiros
NATURAL FULL OUTER JOIN Reservas
```

Agenda

1. Contextualização
2. SQL:1999
3. Visão Geral da SQL
4. UNION, INTERSEC, EXCEPT
5. Consultas Aninhadas
6. Operadores Agregados
7. Valores Nulos
8. Conclusão

O que aprendemos?

Considerações Finais

- SQL:1999 é um padrão que normatiza como os bancos de dados devem implementar comandos fundamentais da linguagem SQL
- Está incluído na SQL vários comandos de diversos tipos (DDL, DML, ...) e diversos conceitos fundamentais de um SGBD
- Consultas usando SELECT...