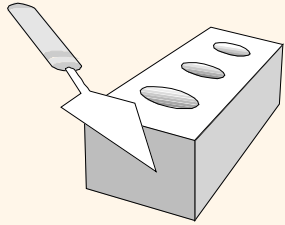
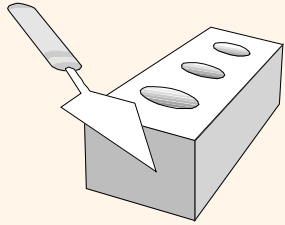


SQL: Consultas, Programação, Gatilhos

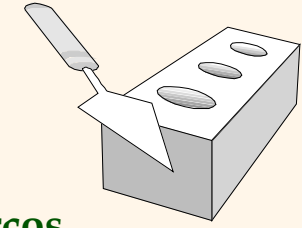
Capítulo 5

Introdução



- O que é SQL ? *Structured Query Language*
  - Linguagem comercial de banco de dados mais utilizada no mercado.
- Origem
  - Originalmente chamada de SEQUEL e desenvolvida pela IBM como parte do projeto Sistema R no início dos anos 70.
- Padrão ANSI/ISSO
  - Nem todos os fabricantes implementam 100% SQL padrão
  - SQL-86, SQL-89, SQL-92, SQL:1999, SQL:2003 : evolução do padrão
- Objetivo
  - A linguagem SQL tem diversas partes: definição de dados, manipulação de dados, autorização, integridade, controle de transações...
  - O foco deste capítulo é a manipulação dos dados = DML.
- Tópicos a serem discutidos
  - Consultas básicas e aninhadas, conjuntos de operadores, valores nulos, restrições de integridade e *triggers*.

Instâncias Exemplos



- Utilizaremos estas instâncias das relações "Marinheiros" (*Sailors*), "Barcos" (*Boats*) e "Reservas" (*Reserves*) em nossos exemplos.

Marinheiros

sid	sname	rating	age
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

S3

Reservas

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

R2

Barcos

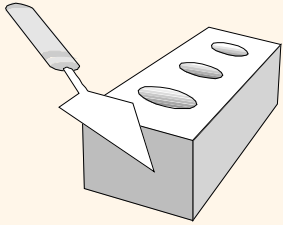
bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

B1

- Se a chave da relação de Reservas tivesse apenas os atributos *sid* e *bid*, o que seria diferente na semântica?

Retornar último slide

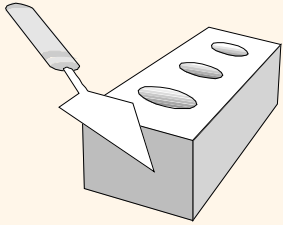
Consulta Básica SQL



```
SELECT    [DISTINCT]  select-list
FROM      from-list
WHERE     qualification
```

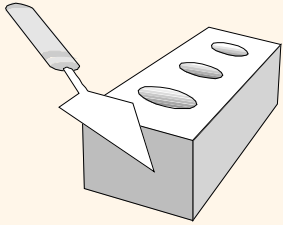
- *from-list*: lista de nomes de relações, possivelmente com um variável tupla ("apelido" ou "*range variable*") para cada nome.
- *select-list*:lista de atributos das relações que estão na lista de relações.
- *qualification*: comparações (Atrib *op* Const ou Atrib1 *op* Atrib2, onde *op* pode ser <, >, =, ≥, ≤ ,≠) combinadas com AND, OR, e NOT.
- DISTINCT é uma palavra chave opcional indicando que a resposta não deveria conter linhas duplicadas. Por padrão, as linhas duplicadas não são eliminadas! (em outras palavras, SQL trabalha com *multi-sets*, enquanto que álgebra relacional trabalha com *sets*)

Estratégia de Avaliação Conceitual



- A semântica de uma consulta SQL é definida pela seguinte estratégia de avaliação conceitual:
  - Compute o produto cartesiano da lista de relações.
  - Elimine as linhas resultantes que não atendem às condições de qualificação.
  - Eliminar todos os atributos que não estão na lista de atributos.
  - Se DISTINCT é especificado, eliminar as linhas duplicadas.
- Esta estratégia é provavelmente o caminho menos eficiente de executar uma consulta! Um otimizador encontrará estratégias mais eficientes para obter as mesmas respostas.

Exemplo da Avaliação Conceitual



```
SELECT  S.sname
FROM    Sailors S, Reserves R
WHERE   S.sid=R.sid AND R.bid=103
```

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

S4

sid	bid	day
22	101	10/10/98
58	103	11/12/96

R3

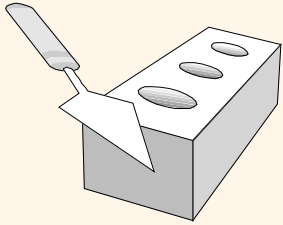
(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

S4 x R3

sname
rusty

Resultado

Uma nota sobre variáveis tupla



- Realmente necessário apenas se a mesma relação aparecer mais de uma vez na cláusula FROM. A consulta anterior pode ser escrita assim:

```
SELECT  S.sname
FROM    Sailors S, Reserves R
WHERE   S.sid=R.sid AND
        bid=103
```

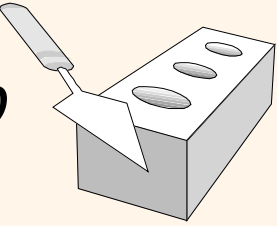
OU

```
SELECT  sname
FROM    Sailors, Reserves
WHERE   Sailors.sid=Reserves.sid
        AND bid=103
```

É considerado um "bom estilo", porém recomenda-se utilizar sempre variáveis tupla para melhorar a legibilidade de suas consultas.

O bid=103 não precisou de variável tupla este atributo só existe na tabela de Reserves.
Mas é recomendado utilizar .

Encontre os marinheiros que reservaram pelo menos um barco.



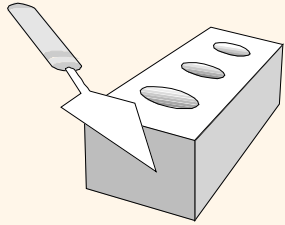
```
SELECT  S.sid
FROM  Sailors S, Reserves R
WHERE  S.sid=R.sid
```

- Adicionar DISTINCT a essa consulta faria alguma diferença?

- Qual é o efeito de substituir *S.sid* por *S.sname* na cláusula SELECT ? Adicionar DISTINCT a essa variação faria alguma diferença?

Ver Instâncias Exemplos

Expressões e Strings



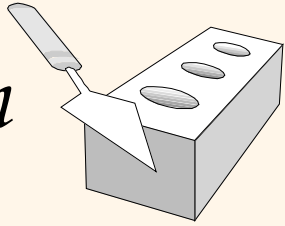
```
SELECT  S.age
FROM  Sailors S
WHERE  S.sname
        LIKE 'B_%B'
```

```
SELECT  S.sname, S.rating+1 AS rating
FROM  Sailors S, Reserves R1, Reserves R2
WHERE  S.sid = R1.sid AND S.sid = R2.sid
AND R1.day = R2.day
AND R1.bid <> R2.bid
```

- Mostra o uso de expressões aritméticas e combinação de string. Encontre as idades dos marinheiros cujos nomes começam e terminam com B e contenham pelo menos três caracteres.
- LIKE é usado para comparações de strings.
  - '_' pode ser qualquer caracter
  - % pode ser 0 ou mais caracteres arbitrários.
- SIMILAR é uma evolução do LIKE, mais parecido com expressões regulares ("*regexp*") => SQL:1999 em diante

Ver Instâncias
Exemplos

Encontre sid's dos marinheiros que reservaram um barco vermelho ou um barco verde



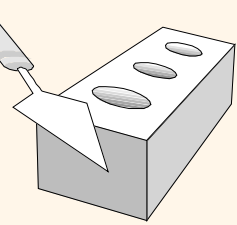
- UNION: Pode ser usado para calcular a união de qualquer dois conjuntos de tuplas compatíveis (que são elas mesmas o resultado das consultas SQL).
- Se substituir OR por AND na primeira versão, o que obtemos?
- O que obtemos se nós substituirmos UNION por EXCEPT?

```
SELECT  S.sid
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid
  AND (B.color='red' OR B.color='green')
```

```
SELECT  S.sid
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid
            AND B.color='red'
UNION
SELECT  S.sid
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid
            AND B.color='green'
```

Ver Instâncias Exemplos

Encontre sid's dos marinheiros que reservaram um barco vermelho e um barco verde

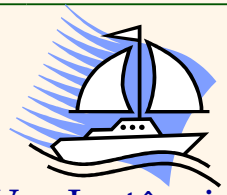


- INTERSECT: Pode ser usado para calcular a interseção de qualquer dois conjuntos de tuplas compatíveis.
- Incluído no padrão SQL/92, mas alguns sistemas não suportam.
- Compare a simetria do UNION com INTERSECT com as outras versões das mesmas consultas.

```
SELECT  S.sid
FROM  Sailors S, Boats B1, Reserves R1,
          Boats B2, Reserves R2
WHERE  S.sid=R1.sid AND R1.bid=B1.bid
  AND  S.sid=R2.sid AND R2.bid=B2.bid
  AND (B1.color='red' AND B2.color='green')
```

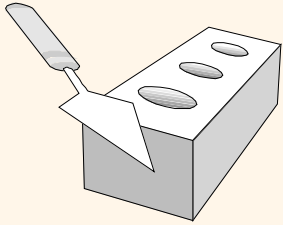
Campo chave!

```
SELECT  S.sid
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid
            AND B.color='red'
INTERSECT
SELECT  S.sid
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid
            AND B.color='green'
```



Ver Instâncias Exemplos

Consultas Aninhadas

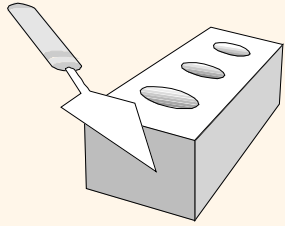


Encontre os nomes dos marinheiros que reservaram o barco #103

```
SELECT  S.sname
FROM  Sailors S
WHERE  S.sid IN  (SELECT  R.sid
                  FROM  Reserves R
                  WHERE  R.bid=103)
```

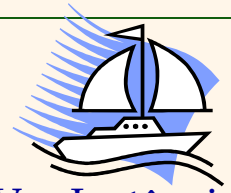
- Uma característica muito poderosa do SQL: a cláusula WHERE pode conter nela mesma uma consulta SQL! (De fato, cláusulas FROM e HAVING também podem).
- Para encontrar marinheiros que não reservaram o barco #103, use NOT IN.
- Para entender a semântica da consulta aninhada, pense numa avaliação de laços aninhados. Para cada tupla de marinheiro, verifique a condição computando a sub-consulta.

Consultas Aninhadas Correlacionadas



Consulta nomes de navegadores que reservaram o barco #103

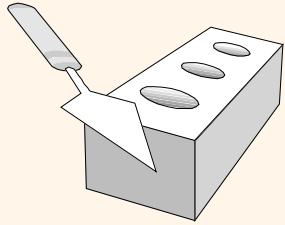
```
SELECT  S.sname
FROM  Sailors S
WHERE   EXISTS  (SELECT  *
                 FROM  Reserves R
                 WHERE  R.bid=103 AND S.sid=R.sid)
```



Ver Instâncias Exemplos

- EXISTS é outro operador de comparação, como IN.
- Se UNIQUE for usado e * for substituído por *R.bid*, encontre os marinheiros com no máximo uma reserva do barco #103. (UNIQUE verifica as tuplas duplicadas; * significa todos os atributos. Porque temos que alterar * por *R.bid*?)
- Este exemplo mostra por que a sub-consulta deve ser re-computada para cada tupla de marinheiros.

Mais operadores comparação-conjunto



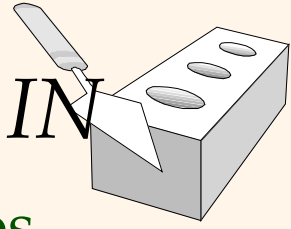
- Nós já vimos IN, EXISTS e UNIQUE. Podemos também utilizar NOT IN, NOT EXIST e NOT UNIQUE.
- Também estão disponíveis: *op* ANY, *op* ALL, onde *op* pode ser >, <, =, ≥, ≤, ≠
- Encontre os marinheiros cujo nível é melhor do que algum marinheiro chamado Horatio:

```
SELECT  *
FROM  Sailors S
WHERE  S.rating > ANY  (SELECT  S2.rating
                        FROM  Sailors S2
                        WHERE S2.sname='Horatio')
```

DICA de avaliação conceitual: substituir ANY por SOME

Ver Instâncias Exemplos

Reescrevendo consultas INSERSECT usando IN

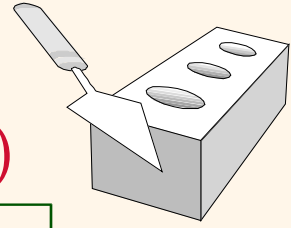


Encontre *sid´s* dos marinheiros que reservaram ambos um barco vermelho e um verde:

```
SELECT  S.sid
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid AND B.color='red'
          AND S.sid IN  (SELECT  S2.sid
                              FROM  Sailors S2, Boats B2, Reserves R2
                              WHERE  S2.sid=R2.sid AND R2.bid=B2.bid
                                        AND  B2.color='green')
```

- De modo similar, consultas usando EXCEPT podem ser reescritas usando NOT IN.
- Para encontrar nomes (e não *sid's*) dos marinheiros que reservaram barcos vermelhos e verdes, apenas substitua *S.sid* por *S.sname* na cláusula SELECT. (Como fazer esta alteração numa consulta com INTERSECT?)

Divisão em SQL



(2)

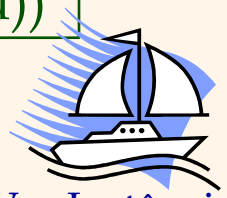
Encontre os marinheiros que reservaram todos os barcos.

- Faremos de modo difícil, sem EXCEPT (também chamado de MINUS em alguns sistemas).

```
SELECT  S.sname
FROM  Sailors S
WHERE  NOT EXISTS
          ((SELECT  B.bid
            FROM  Boats B)
           EXCEPT
            (SELECT  R.bid
             FROM  Reserves R
             WHERE  R.sid=S.sid))
```

(1)

```
SELECT  S.sname
FROM  Sailors S
WHERE  NOT EXISTS  (SELECT  B.bid
                    FROM  Boats B
                    WHERE  NOT EXISTS  (SELECT  R.bid
                                        FROM  Reserves R
                                        WHERE  R.bid=B.bid
                                               AND R.sid=S.sid))
```



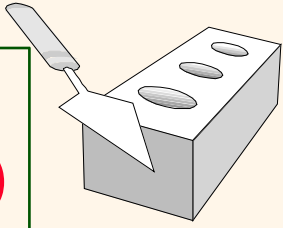
Ver Instâncias Exemplos

Marinheiros tais que ...

não há barco sem...

uma reserva de marinheiro para barco

Operadores Agregados



- Extensão significativa da álgebra de relacional.

```
COUNT (*)
COUNT ( [DISTINCT] A)
SUM ( [DISTINCT] A)
AVG ( [DISTINCT] A)
MAX (A)
MIN (A)
```

Coluna simples

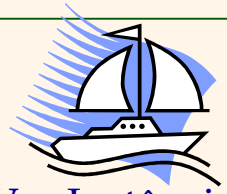
```
SELECT  COUNT (*)
FROM  Sailors S
```

```
SELECT  COUNT (DISTINCT S.sname)
FROM  Sailors S
```

```
SELECT  AVG (S.age)
FROM  Sailors S
WHERE  S.rating=10
```

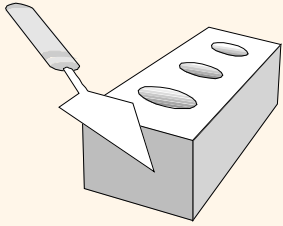
```
SELECT  S.sname, S.age
FROM  Sailors S
WHERE  S.age=
     (SELECT  MAX(S2.age)
              FROM  Sailors S2)
```

```
SELECT  AVG (DISTINCT S.age)
FROM  Sailors S
WHERE  S.rating=10
```



Ver Instâncias Exemplos

Encontre o nome e a idade do marinheiro mais velho



- A primeira consulta é ilegal! (Nós veremos a razão mais tarde quando discutirmos sobre GROUP BY).

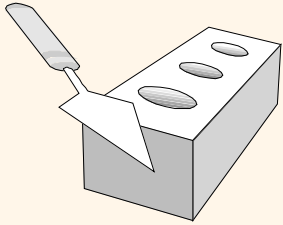
- A terceira consulta é equivalente à segunda, e é permitida pelo padrão SQL/92, mas não é suportada por alguns sistemas.

```
SELECT  S.sname, MAX (S.age)
FROM  Sailors S
```

```
SELECT  S.sname, S.age
FROM  Sailors S
WHERE  S.age =
          (SELECT  MAX (S2.age)
           FROM  Sailors S2)
```

```
SELECT  S.sname, S.age
FROM  Sailors S
WHERE  (SELECT  MAX (S2.age)
           FROM  Sailors S2)
           = S.age
```

Motivação para Agrupar

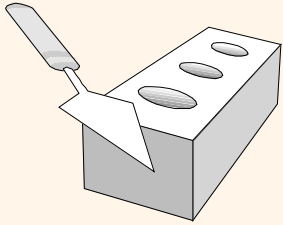


- Até agora, temos aplicados operadores de agregação para todas as tuplas (qualificadas). Algumas vezes, queremos aplicá-los a vários grupos de tuplas.
- Para cada nível, encontre a idade do marinheiro mais jovem.
  - Em geral nós não sabemos quantos níveis existem, e quais os seus valores.
  - Suponha que sabemos que valores dos níveis são de 1 a 10, podemos escrever 10 consultas como esta:

For i = 1, 2, ... , 10:

```
SELECT  MIN (S.age)
FROM  Sailors S
WHERE  S.rating = i
```

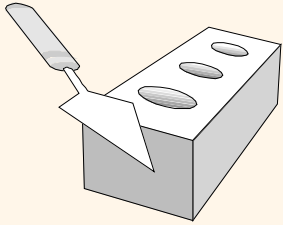
Consultas com GROUP BY e HAVING



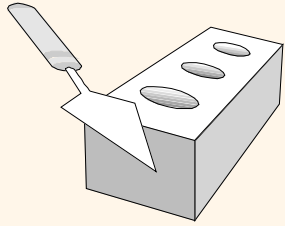
```
SELECT      [DISTINCT]  select-list
FROM        from-list
WHERE       qualification
GROUP BY    grouping-list
HAVING      group-qualification
```

- A lista de atributos ("*select-list*") contém: (i) nomes dos atributos e (ii) termos com operações agregadas (ex: MIN(S.age)).
  - A lista de atributos (i) deve ser um sub-conjunto da lista de grupos. Intuitivamente, cada tupla de resposta corresponde a um grupo, e estes atributos devem ter um único valor por grupo. (Um grupo é um conjunto de tuplas que têm o mesmo valor para todos os atributos na lista de grupos).

Avaliação Conceitual



- O produto cartesiano da lista de relação é calculado, as tuplas que falharam na qualificação são descartadas, os campos "desnecessários" são excluídos e as tuplas restantes são divididas em grupos por valor dos atributos definidos na lista de grupos.
- A qualificação do grupo (cláusula HAVING) é então aplicada para eliminar alguns grupos. Expressões na qualificação do grupo devem ter um único valor por grupo!
  - Para todos os efeitos, um atributo na qualificação do grupo que não é argumento de um operador de agregação também aparece na lista de grupos.
  - Uma tupla resposta é gerada por grupo de qualificação.



Encontre a idade do marinheiro mais jovem com idade ≥ 18, para cada nível com pelo menos 2 desses marinheiros

```
SELECT  S.rating,  MIN (S.age)
                          AS minage
FROM  Sailors S
WHERE  S.age >= 18
GROUP BY  S.rating
HAVING  COUNT (*) > 1
```

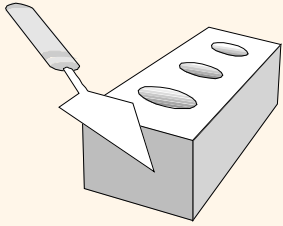
Relação resposta:

rating	minage
3	25.5
7	35.0
8	25.5

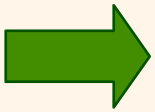
Instância de "Marinheiro"

sid	sname	rating	age
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

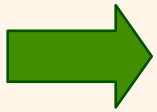
Encontre a idade do marinheiro mais jovem com idade ≥ 18, para cada nível com pelo menos 2 desses marinheiros



rating	age
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

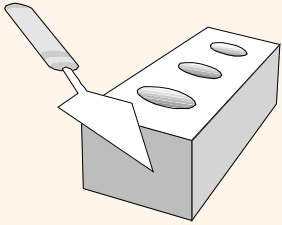


rating	age
1	33.0
3	25.5
3	63.5
3	25.5
7	45.0
7	35.0
8	55.5
8	25.5
9	35.0
10	35.0



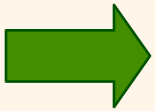
rating	minage
3	25.5
7	35.0
8	25.5

Encontre a idade do marinheiro mais jovem com idade ≥ 18, para cada nível com pelo menos 2 marinheiros entre 18 e 60

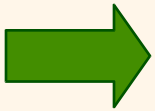


HAVING COUNT (*) > 1 AND EVERY (S.age <=60)

rating	age
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



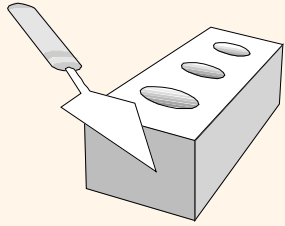
rating	age
1	33.0
3	25.5
3	63.5
3	25.5
7	45.0
7	35.0
8	55.5
8	25.5
9	35.0
10	35.0



rating	minage
7	35.0
8	25.5

Se trocar EVERY por ANY, qual será o resultado?

Encontre a idade do marinheiro mais jovem com idade ≥ 18, para cada nível com pelo menos 2 marinheiros entre 18 e 60



```
SELECT  S.rating,  MIN (S.age)
                          AS minage
FROM  Sailors S
WHERE  S.age >= 18 AND S.age <= 60
GROUP BY  S.rating
HAVING  COUNT (*) > 1
```

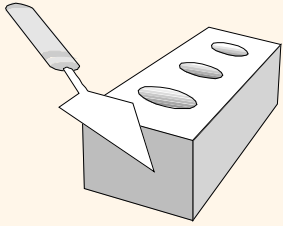
Instância de "Marinheiro"

sid	sname	rating	age
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

Relação resposta:

rating	minage
3	25.5
7	35.0
8	25.5

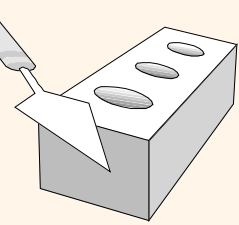
Encontre a média de idade dos marinheiros com idade > 18, para cada nível com pelo menos 2 marinheiros (de qualquer idade)



```
SELECT  S.rating,  AVG (S.age) as avgage
FROM  Sailors S
WHERE  S.age > 18
GROUP BY  S.rating
HAVING  1  <  (SELECT  COUNT (*)
                FROM  Sailors S2
                WHERE  S.rating=S2.rating)
```

- Mostra que a cláusula HAVING também pode conter uma sub-consulta.
- Compare essa consulta com a que consideramos somente níveis com pelo menos 2 marinheiros maiores que 18 anos.
- O que acontece se a cláusula HAVING for substituída por:
  - HAVING COUNT(*) > 1

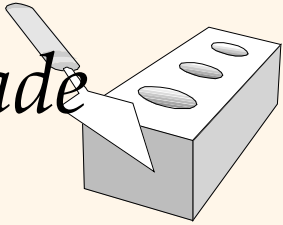
Para cada barco vermelho, encontre o número de reservas para este barco



```
SELECT  B.bid,  COUNT(*) AS reservationcount
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid AND B.color='red'
GROUP BY  B.bid
```

- Agrupando sobre uma junção de 3 relações.
- O que obtemos removendo B.color=‘red’ da cláusula WHERE e adicionando uma cláusula HAVING com essa condição?
- E se retiramos a relação “Marinheiros” e a condição envolvendo *S.sid*?

Encontre os níveis para qual a média de idade é a mais baixa sobre todos os níveis



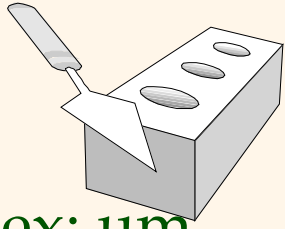
- Operadores agregados não podem ser aninhados. ERRADO:

```
SELECT  S.rating
FROM  Sailors S
WHERE  AVG(S.age) =  (SELECT  MIN (AVG (S2.age))  FROM Sailors S2)
```

- Solução correta (em SQL/92)

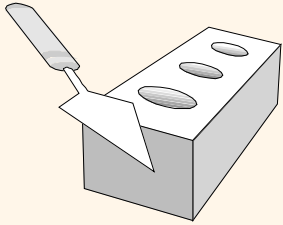
```
SELECT  Temp.rating, Temp.avgage
FROM  (SELECT  S.rating, AVG (S.age) AS avgage
          FROM  Sailors S
          GROUP BY  S.rating) AS Temp
           WHERE  Temp.avgage =
                     (SELECT  MIN (Temp.avgage)
                      FROM  Temp)
```

Valores Nulos



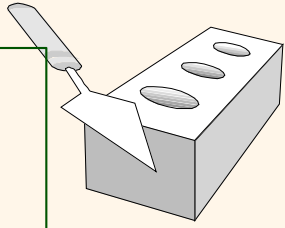
- Às vezes os valores em uma tupla são desconhecidos (ex: um nível não foi determinado) ou não aplicáveis (ex: nome de solteira para um homem).
  - Para essas situações o SQL tem o valor especial *null*.
- A presença do *null* complica em várias questões:
  - Operadores especiais precisam verificar se o valor é ou não nulo.
  - Nível > 8 é verdadeiro ou falso quando o nível é nulo? E as condições contendo os conectivos AND, OR, e NOT?
  - Precisamos ter lógica de 3 valores (verdadeiro, falso,desconhecido)
  - O significado das construções deve ser definido cuidadosamente (ex: cláusula WHERE elimina linhas que não são verdadeiras.)
  - Novos operadores (em particular, *outer joins*) são possíveis/necessários.

Restrições de Integridade (Revisão)



- Uma RI descreve as condições que toda instância válida de uma relação deve satisfazer.
  - Inserts/deletes/updates que violam as RI's são rejeitadas.
  - Pode ser usado na semântica da aplicação (ex: *sid* é a chave) ou prevenir inconsistências (ex: *sname* tem que ser string, *age* deve ser < 200).
- *Tipos de RI's*: restrições de domínio, restrições de chave primária, restrições de chave estrangeira, restrições gerais.
  - *Restrições de domínio:* Valores de campo devem ser do tipo correto. Sempre obrigatórios.

Restrições gerais

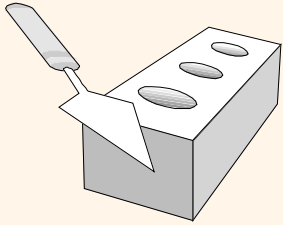


- Úteis quando são envolvidas mais RI's do que chaves
- Pode utilizar consulta para expressar uma restrição.
- Restrições podem ser nomeadas.

```
CREATE TABLE   Sailors
        ( sid  INTEGER,
        sname  CHAR(10),
        rating  INTEGER,
        age  REAL,
        PRIMARY KEY  (sid),
        CHECK  ( rating >= 1
                AND rating <= 10 )
```

```
CREATE TABLE  Reserves
        (sid INTEGER,
        bid  INTEGER,
        day  DATE,
        FOREIGN KEY  (sid) REFERENCES Sailors
        FOREIGN KEY  (bid) REFERENCES Boats
        CONSTRAINT  noInterlakeRes
        CHECK  (`Interlake' <>
                ( SELECT  B.bname
                FROM  Boats B
                WHERE  B.bid=Reserves.bid)))
```

Restrições sobre Múltiplas Relações



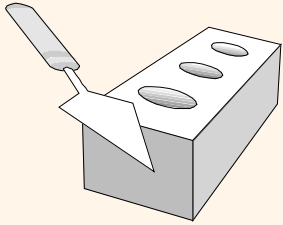
- Inadequado e errado!
- Se "Marinheiros" está vazia, o número de tuplas de "Barcos" pode ser qualquer um!
- ASSERTION é a solução correta; não associado com umas das duas tabelas.

Número de barcos mais número de "Marinheiros é < 100.

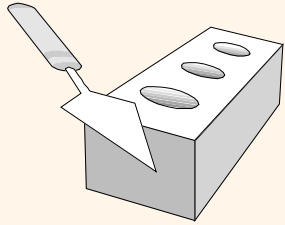
```
CREATE TABLE   Sailors
            ( sid  INTEGER,
            sname  CHAR(10),
            rating  INTEGER,
            age  REAL,
            PRIMARY KEY  (sid),
            CHECK
            ( (SELECT COUNT (S.sid) FROM Sailors S)
            + (SELECT COUNT (B.bid) FROM Boats B) < 100 )
```

```
CREATE ASSERTION  smallClub
CHECK
( (SELECT COUNT (S.sid) FROM Sailors S)
+ (SELECT COUNT (B.bid) FROM Boats B) < 100 )
```

Gatilhos (Triggers)



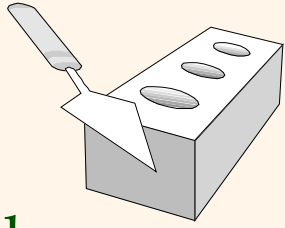
- Gatilhos: procedimentos que são iniciados automaticamente se mudanças especificadas ocorrerem no banco de dados.
- 3 partes:
  - Evento (ativa o gatilho).
  - Condição (testa se os gatilhos deveriam ser executados).
  - Ação (o que acontece se o gatilho for executado).
- Gatilhos devem ser usados com parcimônia, pois podem tornar o comportamento do banco de dados difícil de se prever.



Gatilhos: Exemplo (SQL:1999)

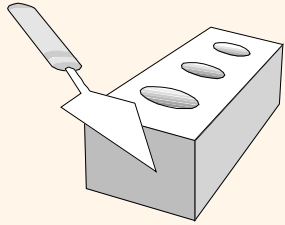
```
CREATE TRIGGER youngSailorUpdate
    AFTER INSERT ON SAILORS
REFERENCING NEW TABLE NewSailors
FOR EACH STATEMENT
    INSERT
        INTO YoungSailors(sid, name, age, rating)
        SELECT sid, name, age, rating
        FROM NewSailors N
        WHERE N.age <= 18
```

Sumário



- SQL foi um fator importante para a rápida aceitação do modelo relacional; mais natural do que rápida, linguagens de consulta procedural.
- Relacionalmente completa. De fato, tem poder expressivo significativamente maior que a álgebra relacional (Ex: agrupamento e consultas aninhadas)
- Consultas expressas em álgebra relacional podem ser expressas, de modo mais natural, em SQL.
- Existem várias maneiras de se escrever uma consulta, o otimizador deve escolher o plano de avaliação mais eficiente.
  - Na prática, usuários devem estar atentos de como as consultas são otimizadas para um melhor resultado.

Sumário (Continuação)



- NULL para valores de campos desconhecidos trazem muitas complicações.
- NULL também pode trazer problemas de performance (ex. uso de NULL para períodos sem começo ou fim).
- SQL permite especificação de ricas restrições de integridade.
- Gatilhos respondem às alterações realizadas no Banco de Dados.