TÉCNICO LISBOA

Bases de Dados 2015/2016
Linguagem SQL

Bibliografia

- Raghu Ramakrishnan, *Database Management Systems*, Cap. 3 e 5

Sumário

- Linguagem SQL

História

- Princípio dos anos 70
  - IBM desenvolve a linguagem Sequel para o System R
- Renomeada para SQL (*Structured Query Language*)
- Standards ISO e ANSI
  - SQL-86, SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2008, SQL:2011
- Sistemas comerciais suportam:
  - a maior parte do SQL-92
  - algumas das funcionalidades dos standards mais recentes
  - algumas funcionalidades específicas e proprietárias

Sumário

- Definição de dados
  - **SQL/DDL** – Data Definition Language
- Manipulação de dados interactiva
  - **SQL/DML** – Data Manipulation Language
- Restrições de integridade
- Definição de vistas
- Autorização e segurança
- Inclusão em linguagens de programação
- Controlo de transações

Definição de dados

- Permite especificar relações e as características de cada relação
  - esquema da relação
  - domínio de cada coluna
  - restrições de integridade
  - índices para a relação
  - privilégios de acesso
  - estrutura física de armazenamento no disco

Definição do esquema

Uma **tabela** define-se com o comando:

```
create table tabela (
            coluna1 tipo1,
            coluna2 tipo2,
                  …
            restrição-integridade1,
            restrição-integridade2,
                  ... )
```

- *tabela* é o nome da tabela (dar um nome adequado)
- $coluna_i$ é o nome de uma coluna

Esquema Relacional para exemplos

Sailors(sid, sname, rating, age)

Boats(bid, bname, color)

Reserves(sid, bid, day)

- *sid: FK(Sailors)*
- *bid: FK(Boats)*

Definição de Esquema – exemplo

```
create table Sailors (
        sid    integer,
        sname  char(15),
        rating numeric(12,2),
        age int,
        primary key (sid))

create table Reserves (
        sid    integer,
        bid    integer,
        day date,
        primary key (sid, bid, day)
        foreign key (sid) references Sailors(sid),
        foreign key (bid) references Boats(bid))
```

Tipos básicos de domínio

Tipos de dados suportados em SQL

- **char(*n*)** – cadeia de caracteres de tamanho fixo (pouco usado)
- **varchar(*n*)** – tamanho variável, máximo *n*
- **integer**
- **smallint**
- **numeric(*p*,*d*)** – *p* dígitos, *d* casas decimais
- **real**
- **double**
- **float(*n*)** – pelo menos *n* dígitos

Alteração de relações

Remover colunas

alter table *tabela*
drop *coluna*

Inserir colunas

alter table *tabela*
add *coluna tipo*

- todos os tuplos ficam com ***null*** neste novo atributo

Remover Tabela

Remover a tabela Sailors da base de dados

```
drop table sailors
```

Inserção e remoção de tuplos

```
insert
into Sailors(sid, sname, rating, age)
values(1, 'Manuel Iscas', 4, 24)

delete
from Sailors
where sid = 1

delete from Sailors
```

Actualização de dados

```
update Sailors
set age = 25
where sid = 1
```

Forma básica de uma interrogação

select[**distinct**] *select-list*
from *from-list*
[where *qualification***]**

- ***select-list***: lista atributos a extrair
  - `select *` lista todas as colunas
  - É possível usar expressões aritméticas aplicadas a constantes ou atributos
- ***from-list*:** lista tabelas de onde extrair dados
- ***qualification*:** filtra quais as linhas a seleccionar
  - condição booleana: *expressão op expressão*, em que op: <, <=, >, >=, <> e outros
  - *expressão*: nome atributo, constante, ou expressão aritmética
- **`distinct`:** elimina linhas duplicadas

SQL vs Álgebra Relacional

Consulta típica em SQL:

select distinct c_1, c_2, ..., c_n
from T_1, T_2, ..., T_m
where P

$$\Pi_{c_1, c_2, c_n}(\sigma_P(T_1 \times T_2 \times \ldots \times T_m))$$

Expressão equivalente à consulta típica em Álgebra Relacional

Observações

select c_1, c_2, ..., c_n
from T_1, T_2, ..., T_m
where P

- se **where** for omitida é como se P = **true**
- o resultado pode conter tuplos duplicados!
- SGBD pode reformular a expressão algébrica para uma forma mais eficiente

Estratégia de avaliação conceptual de uma interrogação

1. Calcula o produto cartesiano de todas as tabelas no ***from-list***
2. Elimina as linhas que falham a condição ***qualification***
3. Elimina as colunas que não aparecem na ***select-list***
4. Elimina as linhas duplicadas caso o **`distinct`** seja usado

Tabelas Exemplo

Sailors

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Boats

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

Reserves

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

Exemplo básico

Q15: Quais os nomes e idades de todos os marinheiros?

```
select distinct sname, age
from Sailors
```

snarne	*age*
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

Figure 5.4 Answer to Q15

sname	*age*
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

Figure 5.5 Answer to Q15 without DISTINCT

Filtro de linhas

Q11: Quais os marinheiros com um *rating* maior que 7?

```
select S.sid, S.sname, S.rating, S.age
from Sailors as S
where S.rating > 7
```

- **`as`** permite fazer etiquetagem de tabelas
  - variável de tuplo
  - é opcional

- **`select *`** também poderia ser usado para escolher todos os atributos

Uso de 2 tabelas

Q1: Quais os nomes dos marinheiros que reservaram o barco 103?

$$\rho_{sname}((\sigma_{bid=103} Reserves) \bowtie Sailors)$$

```
select distinct S.sname
from Sailors S, Reserves R
where S.sid = R.sid and R.bid=103
```

Cálculo para o Exemplo

- Consideramos:

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

Figure 5.6 Instance *R3* of Reserves

sid	*sname*	*Tating*	*age*
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Figure 5.7 Instance 54 of Sailors

- Produto cartesiano:

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

Figure 5.8 *S4* x *R3*

- Resultado final:

sname
rusty

Exemplo - Q16

Q16: Quais os identificadores dos marinheiros que reservaram um barco vermelho?

$$\rho_{sid}((\sigma_{color='red'} Boats) \bowtie Reserves \bowtie Sailors)$$

```
select distinct R.sid
from Boats B, Reserves R, Sailors S
where B.bid = R.bid
and S.sid = R.sid
And B.color = 'red'
```

Alternativa para Junção Natural

```
select distinct R.sid
from Boats natural join Reserves
  R natural join  Sailors S
Where color = 'red’
```

Uso de 3 tabelas

Q2: Quais os nomes dos marinheiros que reservaram um barco vermelho?

$$\rho_{sname}((\sigma_{color='red'} Boats) \bowtie Reserves \bowtie Sailors)$$

```
select distinct S.sname
from Sailors S, Boats B, Reserves R
where S.sid = R.sid and B.bid = R.bid
  and B.color = 'red'
```

Mais um exemplo (1)

Q3: Quais as cores dos barcos reservados pelo Lubber?

```
select B.color
from Sailors S, Reserves R, Boats B
where S.sid = R.sid and R.bid = B.bid
  and S.sname = 'Lubber'
```

Mais um exemplo (2)

Q4: Quais os nomes dos marinheiros que reservaram pelo menos um barco?

$$\rho_{sname}(\text{Re}serves \bowtie Sailors)$$

```
select distinct S.sname
from Sailors S, Reserves R
where S.sid = R.sid
```

Ou

```
select distinct sname
from Sailors natural join Reserves
```

Expressões no *select-list* e renomeação em SQL

- Podemos ter
  - *Expressão* **AS** *nome_coluna*
- *expression* pode ser uma qualquer expressão aritmética, um nome de atributo ou uma constante
  - **AS** serve para dar um nome novo
- Ou funções de agregação
  - A ver mais à frente

Exemplo

Q17: Calcular o incremento dos *ratings* dos marinheiros que reservaram dois barcos para o mesmo dia.

```
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
```

Com expressões na *qualification*

```
select S1.sname AS name1,
  S2.sname as name2
from Sailors S1, Sailors S2
where 2*S1.rating = S2.rating-1
```

Operador LIKE para *pattern matching*

- O caracter **%** representa zero ou mais caracteres arbitrários
- O caracter _ representa um caracter arbitrário
- `s` **`like`** `'_AB%'`
  - Filtra cadeia de caracteres começadas por um qualquer caracter, seguido de um A e depois de um B e depois de qualquer coisa
- `'jeff' = 'jeff '` **`true`**
- `'jeff' LIKE 'jeff '` **`false`**

Uso do operador LIKE

Q18: Quais as idades dos marinheiros cujo nome começa e termina com um B e tem, no mínimo, 3 caracteres?

```
select S.age
from Sailors S
where S.sname like 'B_%B'
```

Observações

- se a sequência de caracteres dada contiver ‘%’ ou ‘_’?
  - ‘ab\%cd%’ aplicável a todas as cadeias de caracteres com “ab%cd”
  - ‘ab\\cd%’ aplicável a todas as cadeias de caracteres com “ab\cd”
- **`not like`** também é um operador
- Conversão com **upper**() e **lower**()
- Muitas outras funções
  - concatenação, inserção, procura, substituição, inversão, etc.
  - sintaxe diferente em cada sistema
  - **`Similar to`** (em SQL:1999) – expressões regulares

Ponto da situação

- Linguagem SQL
  - SELECT / FROM / WHERE
  - **Conjuntos**
  - Sub-interrogações
  - Divisão
  - Funções de Agregação
  - GROUP BY / HAVING
  - NULL
  - OUTER JOIN
  - Vistas
  - Autorização e Segurança

Constructores de conjuntos

- União

`select` … **`union select`** …

- Intersecção

`select` … **`intersect select`** …

- Excepto

`select` … **`except select`** …

- Por pré-definição, eliminam duplicados
- Para forçar a não eliminação de duplicados:
  - **`union all`**
  - **`intersect all`**
  - **`except all`**

Operações sobre Conjuntos – Duplicados

Se um tuplo ocorre

- *m* vezes em *R*
- *n* vezes em *S*

então

- ocorre *m* + *n* vezes em (*R* **union all** *S*)
- ocorre *min(m,n)* vezes em (*R* **intersect all** *S*)
- ocorre *max(0, m – n)* vezes em (*R* **except all** *S*)

União simples

Q5: Quais os nomes dos marinheiros que reservaram um barco verde **ou** vermelho?

```
select S.sname
from Sailors S, Reserves R, Boats B
where S.sid = R.sid
and R.bid = B.bid
and (B.color = 'red'
or B.color = 'green')
```

Agora com UNION

```
select S.sname
from Sailors S, Reserves R, Boats B
where S.sid = R.sid and R.bid = B.bid and
  B.color = 'red'
union
select S.sname
from Sailors S, Reserves R, Boats B
where S.sid = R.sid and R.bid = B.bid and
  B.color = 'green'
```

Assume S.name como chave candidata; se não for o que acontece?

Outro exemplo

Q6: Quais os nomes dos marinheiros que reservaram um barco verde **e** vermelho?

```
select S.sname
from Sailors S, Reserves R1, Boats B1,
  Reserves R2, Boats B2
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'
```

Agora com INTERSECT

```
select S.sname
from Sailors S, Reserves R, Boats B
where S.sid = R.sid and R.bid = B.bid and
  B.color = 'red'
intersect
select S.sname
from Sailors S, Reserves R, Boats B
where S.sid = R.sid and R.bid = B.bid and
  B.color = 'green'
```

Assume S.name como chave candidata, senão?

Exemplo de EXCEPT (ou MINUS)

Q19: Quais os identificadores do marinheiros que reservaram barcos vermelhos mas não verdes?

```
select R.sid
from Boats B, Reserves R
where R.bid = B.bid and B.color = 'red’
except
select R2.sid
from Boats B2, Reserves R2
where R2.bid = B2.bid and B2.color = 'green'
```

União de tabelas diferentes

- Quais os identificadores dos marinheiros com um *rating* de 10 ou com uma reserva para o barco 104?

```
select S.sid
from Sailors S
where S.rating = 10
union
select R.sid
from Reserves R
where R.bid = 104
```

Ponto da situação

- Linguagem SQL
  - SELECT / FROM / WHERE
  - Conjuntos
  - **Sub-interrogações**
  - Divisão
  - Funções de Agregação
  - GROUP BY / HAVING
  - NULL
  - OUTER JOIN
  - Vistas
  - Autorização e Segurança

Sub-Interrogações

- Um comando `SELECT` pode conter outros comandos `SELECT`
- Podem aparecer nas cláusulas:
  - `WHERE`
  - `HAVING` (a ver mais adiante)
  - `FROM`
- Aplicações típicas
  - ocorrência num conjunto
  - comparação de conjuntos
  - número de elementos num conjunto

Exemplo com IN

Q1: Quais 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 )
```

- O operador `IN` permite testar se um valor pertence a um conjunto de elementos
- Qual a instrução em SQL equivalente a esta?

Múltiplas sub-interrogações

Q2: Quais os nomes dos marinheiros que reservaram um barco vermelho?

```
select S.sname
from Sailors S
where S.sid in(
  select R.sid
  from Reserves R
  where R. bid in(
    select B.bid
    from Boats B
    where B.color = 'red'))
```

Exemplo com NOT IN

Q21: Quais os nomes dos marinheiros que **não** reservaram um barco vermelho?

```
select S.sname
from Sailors S
where S.sid not in(
  select R.sid
  from Reserves R
  where R.bid in (
    select B.bid
    from Boats B
    where B.color = 'red'))
```

Interrogações correlacionadas

Q1: Quais os nomes dos marinheiros que reservaram o barco 103?

```
select S.sname
from Sailors S
where exists(
        select *
        from Reserves R
        where R.bid = 103 and
              R.sid = S.sid )
```

- **EXISTS** permite testar se um conjunto não é vazio
- A sub-interrogação usa a tabela da interrogação principal

Predicado UNIQUE (NOT UNIQUE)

- Quando aplicado a uma sub-interrogação, devolve true se não houver duplicados na resposta da sub-interrogação
  - Também retorna true se a resposta fôr vazia
- Quais os nomes dos marinheiros que reservaram o barco 103 uma vez, no máximo?

```
select S.sname
from Sailors S
where UNIQUE(
      select sid
      from Reserves R
      where R.bid = 103 and
            R.sid = S.sid )
```

Comparação de conjuntos com ANY

Q22: Quais os marinheiros cujo *rating* é maior que o de algum dos marinheiros chamados *Horatio*?

```
select S.sid
from Sailors S
where S.rating > ANY (
      select S2.rating
      from Sailors S2
      where S2.sname = 'Horatio' )
```

- E se não existir nenhum marinheiro chamado Horatio?
  - A comparação com `ANY` retorna falso

Comparação de conjuntos com ALL

Q23: Quais os marinheiros cujo *rating* é maior que o de todos os marinheiros chamados *Horatio*?

```
select S.sid
from Sailors S
where S.rating > ALL (
      select S2.rating
      from Sailors S2
      where S2.sname = 'Horatio' )
```

- E se não existir nenhum marinheiro chamado Horatio?
  - A comparação com `ALL` retorna verdadeiro

Exemplo de escolha do máximo

Q24: Quais os marinheiros com maior *rating*?

```
select S.sid
from Sailors S
where S.rating >= ALL (
      select S2.rating
      from Sailors S2)
```

ALL, ANY

- Fórmula geral:
  - *op* `ANY` (ou *op* `SOME`)
  - *op* `ALL`

- `IN` equivalente a `= ANY`
- `NOT IN` equivalente a `<> ALL`

Exemplo do IN para intersecções

Q6: Quais os nomes dos marinheiros que reservaram um barco verde **e** vermelho?

```
select S.sname
from Sailors S, Reserves R, Boats B
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' )
```

Alternativa

Q6: Quais os nomes dos marinheiros que reservaram um barco verde **e** vermelho?

```
select S.sname
from Sailors S
where S.sid in(
   select R.sid
   from Boats B, Reserves R
   where R.bid = B.bid and B.color = 'red'
   intersect
   select R2.sid
   from Boats B2, Reserves R2
   where R2.bid = B2.bid and B.color = 'green'
)
```

Ponto da situação

- Linguagem SQL
  - SELECT / FROM / WHERE
  - Conjuntos
  - Sub-interrogações
  - **Divisão**
  - Funções de Agregação
  - GROUP BY / HAVING
  - NULL
  - OUTER JOIN
  - Vistas
  - Autorização e Segurança

Divisão

Q9: Quais os nomes dos marinheiros que reservaram todos os barcos?

```
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 ))
```

- Para cada marinheiro S, verificamos se não existe nenhum barco que não tenha sido reservado por esse marinheiro.

Exemplo de divisão com restrição

- Quais os nomes dos marinheiros que reservaram todos os barcos vermelhos?

```
select S.sname
from Sailors S
where not exists(
      select B.bid
      from Boats B
      where B.color = 'red'
      and not exists (
            select R. bid
            from Reserves R
            where R.bid = B.bid
            and R.sid = S.sid ))
```

Divisão com EXCEPT

Q9: Quais os nomes dos marinheiros que reservaram todos os barcos?

```
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 ))
```

- Para cada marinheiro S, verificamos se o conjunto dos barcos reservados por S contém todos os barcos
  - A contains B ⇔ not exists (B except A)

Divisão com EXCEPT

Q9: Quais os nomes dos marinheiros que reservaram todos os barcos?

```
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 ))
```

Para cada marinheiro S

Todos os barcos - B

Barcos reservados por S - A

- Para cada marinheiro S, verificamos se o conjunto dos barcos reservados por S contém todos os barcos
  - A contains B ⇔ not exists (B except A)

Ponto da situação

- Linguagem SQL
  - SELECT / FROM / WHERE
  - Conjuntos
  - Sub-interrogações
  - Divisão
  - **Funções de Agregação**
  - GROUP BY / HAVING
  - NULL
  - OUTER JOIN
  - Vistas e Segurança

Operadores de Agregação

- `COUNT ([DISTINCT] A)`
  - O número de valores (únicos) na coluna A
- `SUM ([DISTINCT] A)`
  - A soma dos valores (únicos) na coluna A
- `AVG ([DISTINCT] A)`
  - A média dos valores (únicos) na coluna A
- `MAX (A)`
  - O máximo valor da coluna A
- `MIN (A)`
  - O mínimo valor da coluna A

Exemplos de AVG

Q25: Qual a média de idades de todos os marinheiros?

```
SELECT AVG(S.age)
FROM Sailors S
```

Q26: Qual a média de idades dos marinheiros com classificação 10?

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

Exemplos de MAX

- Qual a idade do marinheiro mais velho?

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

Q27: Qual a o nome e idade do marinheiro mais velho?

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

Exemplo de COUNT

Q28: Qual o número de marinheiros?

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

Q29: Qual o número de nomes de marinheiros distintos?

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

- `COUNT(*)` equivalente a `COUNT(sname)`

Exemplo com sub-interrogações

Q30: Quais os nomes dos marinheiros que são mais velhos que o marinheiro mais velho com *rating* de 10?

```
SELECT S.sname
FROM Sailors S
WHERE S.age > (
      SELECT MAX(S2.age)
      FROM Sailors S2
      WHERE S2.rating = 10 )
```

Agrupamento e Filtragem

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

- `GROUP BY`
  - Operador que junta linhas com valores iguais nas colunas do *grouping-list* num único grupo
- `HAVING`
  - Filtra os grupos que não satisfazem a *group-qualification*

Exemplo de GROUP BY

Q31: Qual a idade do mais novo marinheiro para cada classificação?

```
SELECT S.rating, MIN(S.age)
FROM Sailors S
GROUP BY S.rating
```

Exemplo de GROUP BY e HAVING

Q32: Qual a idade do mais novo marinheiro com mais de 18 anos para cada valor de classificação com no mínimo 2 marinheiros com mais de 18 anos?

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

Regras

- As colunas da select-list consistem em: lista de nomes de colunas ou lista de termos `op-agreg` (*coluna*)
  - As colunas da *select-list* que não são agregadas por uma função têm que constar da *grouping-list*
- As expressões da *grouping-qualification* têm que ter um único valor por grupo
  - Uma coluna da *grouping-qualification* tem que constar como argumento de um operador de agregação ou tem que pertencer à *grouping-list*

- Se não existe `GROUP BY`
  - a tabela toda é considerada um grupo
- predicados de `HAVING` são aplicados **depois** do agrupamento
- predicados de `WHERE` são aplicados **antes** do agrupamento

Exemplo de GROUP BY (1)

Q33: Qual o número de reservas para cada barco vermelho?

```
SELECT B.bid, COUNT (*) AS reservationcount
FROM Boats B, Reserves R
WHERE R.bid = B.bid
GROUP BY B.bid
HAVING B.color = 'red'  <-- Ilegal em SQL-92

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

Exemplo de GROUP BY (2)

Q34: Qual a média de idades dos marinheiros por valor de classificação que tem pelo menos 2 marinheiros?

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

Exemplo de GROUP BY

- Qual a idade do mais novo marinheiro com mais de 18 anos para cada *rating* cuja média de idades dos marinheiros com mais de 18 anos seja superior à média de idade de todos os marinheiros?

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

Ordenação de Tuplos
ORDER BY

A cláusula **`order by`** permite ordenar tuplos

- `asc` – ascendente
- `desc` – descendente

- Listar por ordem alfabética os nomes dos marinheiros que reservaram o o barco 103

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

Valor NULL

- **arg1 = arg2, arg1 `AND` arg2, `NOT` arg**, …
  - Se arg1 ou arg2 for NULL então o resultado é nulo
- **arg1 OR arg2**
  - Só é NULL se ambos arg1 e arg2 forem NULL
  - ou seja se arg1 for NULL, o resultado é igual a arg2

Condições com NULL

- A condição **qualification** do **WHERE** filtra linhas cujo resultado seja
  - falso ou NULL
- Para testar:
  - sid **`IS NULL`**
    - True se sid for NULL
  - sid **`IS NOT NULL`**
    - True se sid não for NULL

Operadores de Agregação e o valor NULL

- **`COUNT(*)`**
  - Conta NULL
- Todos os outros ignoram valores NULL
  - `COUNT`(rating)
  - `COUNT`(`DISTINCT` rating)
  - `MAX`(rating)
  - ...
  - Se aplicados só a valores NULL, devolvem NULL

Ponto da situação

- Linguagem SQL
  - SELECT / FROM / WHERE
  - Conjuntos
  - Sub-interrogações
  - Divisão
  - Funções de Agregação
  - GROUP BY / HAVING
  - NULL
  - **Junção externa (OUTER JOIN)**
  - Vistas

Junção Natural

- Quais o números de barcos reservados para cada marinheiro?

```
SELECT S.sname, COUNT(R.bid)
FROM Sailors S, Reserves R
WHERE S.sid = R.sid
GROUP BY S.sname
```

* Não devolve marinheiros sem reservas

Junção Externa

```
SELECT S.sname, R.bid
FROM Sailors S LEFT OUTER JOIN
     Reserves R ON (S.sid = R.sid)
```

- `LEFT` indica que todas as linhas de Sailors vão aparecer
  - mesmo as que não têm reserva
  - essas são juntas com colunas de R atribuídas com valor NULL

Ponto da situação

- Linguagem SQL
  - SELECT / FROM / WHERE
  - Conjuntos
  - Sub-interrogações
  - Divisão
  - Funções de Agregação
  - GROUP BY / HAVING
  - NULL
  - Junção externa (OUTER JOIN)
  - **Vistas**
  - Autorização e Segurança

Elementos de Estudo

- Raghu Ramakrishnan, Database Management Systems, 3ª edição: Sec. 3.6

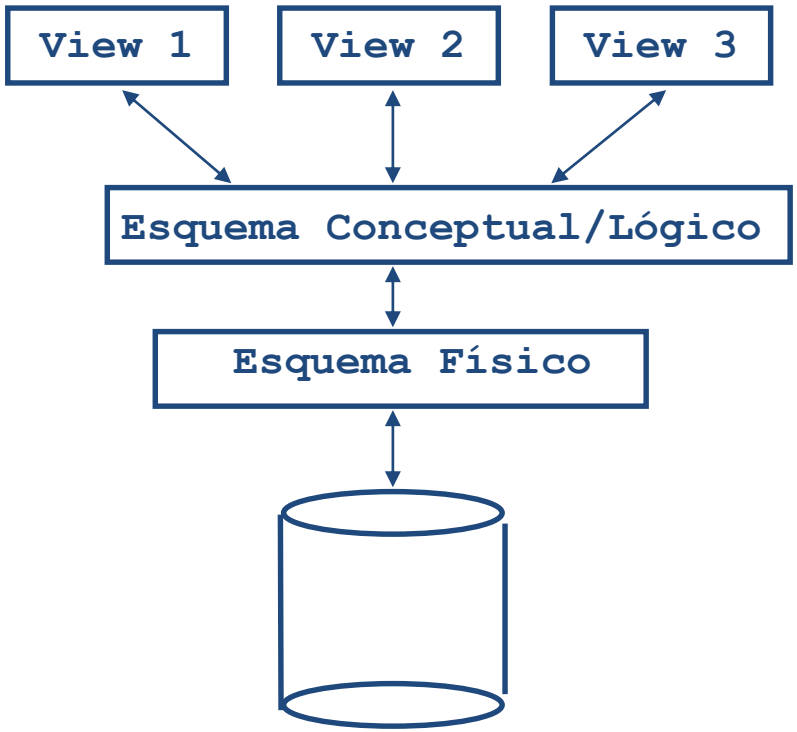
Vistas

- Uma **vista** é uma tabela “virtual”
  - Cujas linhas não são explicitamente armazenadas
  - Mas calculadas assim que necessário
  - Para restringir ou consolidar a informação

Relembrar: Níveis de Abstracção Modelo ANSI/SPARC

Muitas vistas (*views*), um só *esquema conceptual/lógico* e um só *esquema físico*.

- **Esquema externo** (***views***) descreve como os utilizadores vêm os dados
- **Esquema conceptual** projecta-se na estrutura lógica
- **Esquema Físico** descreve ficheiros (tabelas e índices)



Vistas

Definidas com base numa instrução SQL SELECT

`CREATE VIEW` *`V`* `as select`……

- Uma vez criada, pode ser usada como uma relação
  - a vista exprime uma relação em interrogações
  - mas não é o mesmo que criar uma tabela

Exemplo

```
CREATE VIEW B-Students (name, sid, course)
 AS SELECT S.sname, S.sid, E.cid
    FROM Students S, Enrolled E
    WHERE S.sid = E.studid AND E.grade = 'B'
```

name	*sid*	*course*
Jones	53666	History105
Guldu	53832	Reggae203

```
SELECT *
FROM B-Students
WHERE name = 'Jones'
```

Vistas, independência de dados e segurança

- Suporta a independência lógica dos dados do modelo relacional
- Úteis no contexto de segurança
  - Definir vistas que dão acesso, a um grupo de utilizadores, só à informação que eles estão autorizados a ver.

Actualizações sobre vistas

- SQL:1999 distingue entre vistas:
  - Cujas linhas podem ser modificadas *(**updatable views**)*
    - Uma coluna de uma vista pode ser actualizada se for obtida a partir de exactamente uma tabela base e a chave primária da tabela base estiver incluída nas colunas da vista
  - E vistas onde novas linhas podem ser inseridas *(**insertable views**)*
  - Tem de existir uma relação de um para um entre as linhas da vista e das respectivas tabelas base.

Apagar Vistas

```
DROP VIEW B-Students
```

Ponto da situação

- Linguagem SQL
  - SELECT / FROM / WHERE
  - Conjuntos
  - Sub-interrogações
  - Divisão
  - Funções de Agregação
  - GROUP BY / HAVING
  - NULL
  - Junção externa (OUTER JOIN)
  - Vistas
  - **Autorização e Segurança**

Elementos de Estudo

- Raghu Ramakrishnan, Database Management Systems, 3ª edição: Cap 21

Recordando Vistas

- Uma vista é apenas uma relação, mas realizamo-la guardando a sua definição em vez de um conjunto de tuplos

```
CREATE  VIEW  ActiveSailors (name, age, day)
  AS SELECT   S.sname, S.age, R.day
   FROM  Sailors S, Reserves R
   WHERE  S.name=R.sname  AND  S.rating>6
```

Interrogações sobre vistas

- **Expansão de vistas** - técnica para avaliação de interrogações sobre vistas
  - Referências a uma vista são substituídas pela sua definição
  - Notar como `sname` é substituído por `name` para acomodar a definição da vista

```
SELECT  A.name,  MAX ( A.day )
FROM  ActiveSailors A
GROUP BY  A.name
```

```
SELECT  A.name, MAX ( A.Day )
FROM
   ( SELECT S.sname AS name,
          S.age, R.day
     FROM  Sailors S,  Reserves R
     WHERE  S.sname=R.sname
     AND  S.rating>6 ) A
GROUP BY  A.name
```

Vistas e Segurança

- Vistas podem ser utilizadas para apresentar informação necessária (ou um sumário), escondendo detalhes das relações em que se baseiam.
  - Dada `ActiveSailors`, mas não `Sailors` ou `Reserves`, podemos encontrar os marinheiros que têm uma reserva, mas não os *bid* dos barcos que foram reservados.
- Os comandos **GRANT/REVOKE** do SQL podem ser usados para controlar o acesso a relações e vistas
- Em conjunto com a capacidade de definir vistas, fornece um mecanismo de controlo de acesso poderoso

Privilégios

- Forma de segurança mais comum em SGBDs (segurança discricionária)
  - Utilizadores têm *login* com autenticação
    - conta tem privilégios de **CREATE, DROP, SELECT, UPDATE**
  - Relações têm dono (owner)
    - dão-se privilégios de **SELECT, UPDATE, REFERENCE**, etc (normalizado em SQL-92 com comandos GRANT e REVOKE)

Privilégios: GRANT e REVOKE

- `GRANT INSERT, SELECT ON Sailors TO Horatio`
- `GRANT DELETE ON Sailors TO Yuppy WITH GRANT OPTION`
- `GRANT UPDATE (rating) ON Sailors TO Dustin`
  - Actualização restrita à coluna `rating` de `Sailors`
- `GRANT SELECT ON ActiveSailors TO Guppy, Yuppy`
  - Isto não dá acesso directo a Sailors!
- `REVOKE`: quando um privilégio é retirado a X, é também retirado a todos os que o obtiveram apenas de X

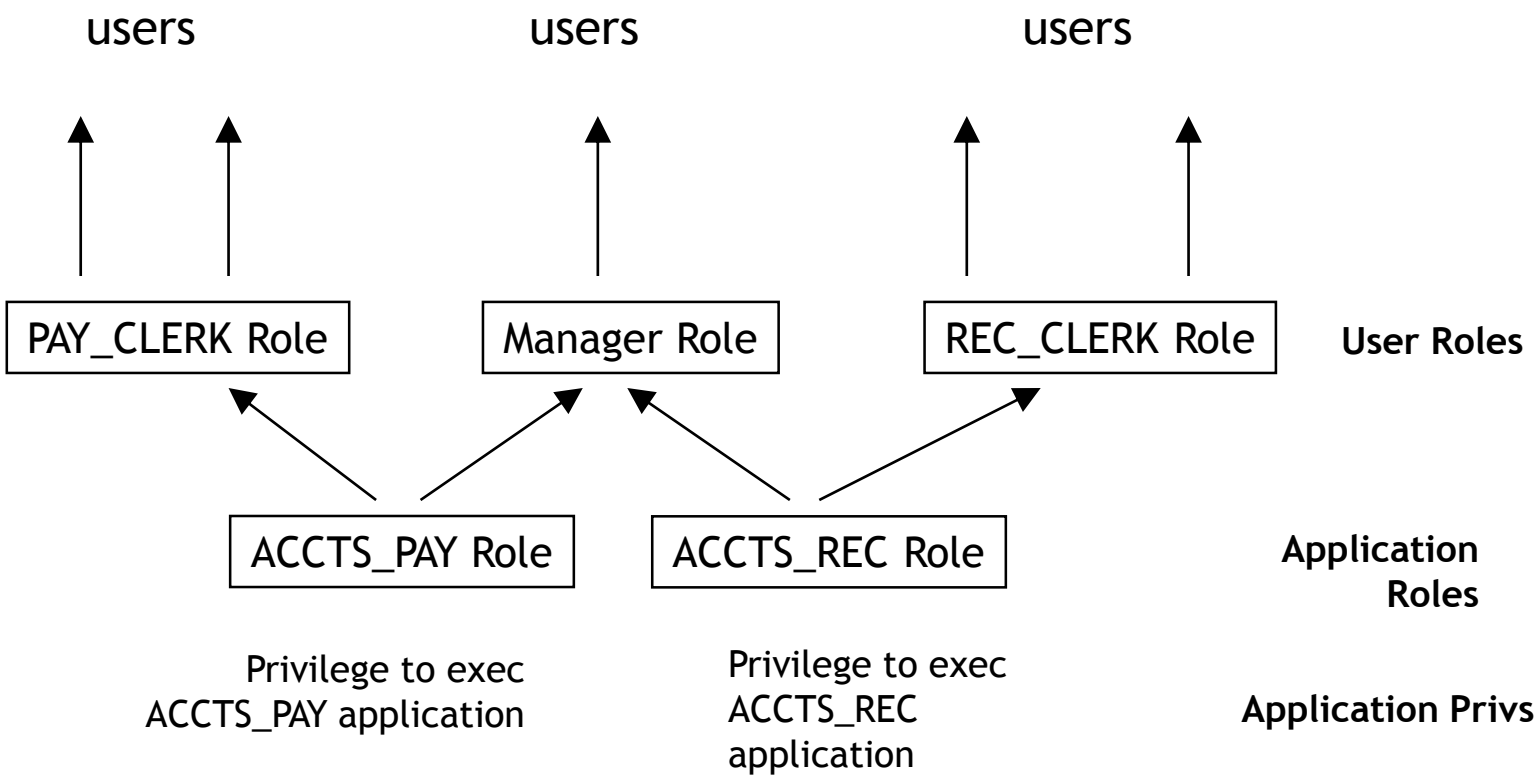
Parâmetros de GRANT

- **Privilégios**: É uma acção sobre um objecto, que pode tomar os valores ALTER, DELETE, EXECUTE, INDEX, INSERT, REFERENCES, SELECT, ou UPDATE
- **ALL PRIVILEGES**: atribui todos os privilégios acima
- ***column***: especifica uma coluna de uma tabela, na qual os privilégios são atribuídos. Só é possivel especificar colunas, quando se atribuem os privilégios INSERT, REFERENCES, ou UPDATE
- **ON:** identifica o objecto, para o qual as funcionalidades são concedidas. Se não se indicar o valor de *schema* - Oracle assume que nos estamos a referir a um objecto no nosso espaço de trabalho.
- **TO** identifica utilizadores ou papeis, para os quais o privilegio é concedido
- **PUBLIC** atribui privilégios a todos os utilizadores
- **WITH GRANT OPTION**: permite ao utilizador que recebe os privilégios, propagá-los a outros utilizadores

Segurança até ao nível de um campo!

- Podemos criar uma vista que apenas retorna um campo de um tuplo!
- De seguida, damos acesso à vista como pretendido
- Permite granularidade arbitrária do controlo
  - Especificação não é muito elegante
  - Pode ser facilitada com uma boa interface de utilizador/programador

Papéis(*Roles*)



Grupos de privilégios criados para gerir privilégios de um grupo de utilizadores ou de uma dada aplicação

Tipos de Segurança de Sistemas

- Discricionária
  - modelo de privilégios atribuídos aos utilizadores para actuar sobre objectos (modelo de controlo de acessos)
  - **o que é suportado na generalidade dos sistemas de informação**
- Mandatória
  - utilizadores e objectos classificados em hierarquia de níveis de segurança
  - política de segurança estabelece normas de acesso
  - aplicação a nível militar

Segurança Mandatória

- **Classes Típicas**
  1. Top Secret (TS)
  2. Secret (S)
  3. Confidential (C)
  4. Unclassified (U)
- **Modelo de Bell-LaPadula**: Agentes (utilizadores, programas) e Objectos (tabelas, atributos) classificados num dos níveis
- **Regras de Acesso**:
  - Um agente A só pode ler um Objecto O se Classe(A) >= Classe(O)
  - Um agente A só pode escrever um Objecto O se Classe(A) = Classe(O)

Sumário

- Linguagem SQL
- **Próxima aula: Restrições de Integridade em SQL**