

Aula 8

Professor:

Claudio Esperança

Consultas SQL

Conteúdo:

- Junções e Produtos Cartesianos
- Ordenação
- Agrupamentos
- Funções sobre grupos
- Combinando `insert` e `select`

Novamente, o comando Select

- O comando **select** é usado para realizar consultas ao banco de dados
- Normalmente, uma consulta tem a forma

select valores **from** tabelas **where** condição

onde:

- ▶ *valores* indica o que se quer obter das tabelas:
 - * : indica que a tabela resultante contém todas as colunas das *tabelas* listadas no comando **select**
 - *expr1, ...exprN* : indica que as colunas da tabela-resultado são valores resultantes da avaliação das expressões
- ▶ *tabelas* : nomes das tabelas a serem consultadas
 - Se há mais de uma tabela, a consulta é um produto cartesiano ou uma *junção* (veremos adiante)
- ▶ *condição* é uma expressão booleana que restringe as linhas serem consideradas

Exemplo

- Consultar todos as entradas da tabela cliente:
- ▶ `select * from cliente`

<u>id</u>	<u>nome</u>	<u>endereco</u>
1	Carlos Pereira	Rua do Lavradio, 51 apto 903
2	Geraldo da Silva	Av Chile, 88
3	Maria Xavier da Silva	Av Chile, 88
4	Antonio da Costa Cavalcanti	Praca Sao Marcos, 5
5	Francisca Emmanuel Xerxes	Rua da Bica, 102 apto 901
6	Jose Marcos do Nascimento	Rua Rui Barbosa, 95 apto 201

Abrir mysql

Exemplo

- Consultar o nome de todos os clientes que possuem sobrenome Pereira
 - ▶ `select nome from cliente where nome like "%Pereira"`

nome
Carlos Pereira

- ▶ Observe o uso do operador de strings `like`
 - O segundo operando é um *padrão de semelhança*
 - O caracter % num padrão significa "pode ser substituído por 0 ou mais caracteres quaisquer"

Abrir mysql

Produtos Cartesianos e Junções

- São consultas que envolvem mais de uma tabela
- Um produto cartesiano consiste em relacionar todas as possíveis combinações entre linhas de todas as tabelas
 - ▶ Por exemplo, o produto cartesiano de duas tabelas com 3 e 4 linhas, respectivamente, resulta num produto cartesiano de $3 \times 4 = 12$ linhas
 - ▶ É uma operação infreqüente pois tende a computar resultados com muitas linhas
- Numa junção, as diversas tabelas são relacionadas por alguma propriedade comum
 - ▶ É freqüentemente usada em consultas envolvendo relacionamentos
 - ▶ Pode-se pensar numa junção como a *restrição* de um produto cartesiano

Exemplo

T1

<u>a</u>	<u>b</u>
x	1
y	2
z	3

T2

<u>c</u>	<u>d</u>
k	2
l	5
m	1



Exemplo

Produto cartesiano

```
select * from T1, T2
```

T1

<u>a</u>	<u>b</u>
x	1
y	2
z	3

T2

<u>c</u>	<u>d</u>
k	2
l	5
m	1

<u>a</u>	<u>b</u>	<u>c</u>	<u>d</u>
x	1	k	2
y	2	k	2
z	3	k	2
x	1	1	5
y	2	1	5
z	3	1	5
x	1	m	1
y	2	m	1
z	3	m	1



Exemplo

Produto cartesiano

```
select * from T1, T2
```

T1

<u>a</u>	<u>b</u>
x	1
y	2
z	3

T2

<u>c</u>	<u>d</u>
k	2
l	5
m	1

Junção

```
select * from T1, T2  
where b = d
```

<u>a</u>	<u>b</u>	<u>c</u>	<u>d</u>
x	1	k	2
y	2	k	2
z	3	k	2
x	1	1	5
y	2	1	5
z	3	1	5
x	1	m	1
y	2	m	1
z	3	m	1

<u>a</u>	<u>b</u>	<u>c</u>	<u>d</u>
y	2	k	2
x	1	m	1



Exemplo

- Para sabermos o número das contas de todos os clientes
 - ▶ `select nome, numconta from cliente, possui where id=idcliente`

<u>nome</u>	<u>numconta</u>
Carlos Pereira	1
Carlos Pereira	2
Geraldo da Silva	3
Maria Xavier da Silva	3
Maria Xavier da Silva	4
Antonio da Costa Cavalcanti	5
Francisca Emmanuel Xerxes	6
Francisca Emmanuel Xerxes	7
Jose Marcos do Nascimento	8

Abrir mysql

Exemplo

- Para sabermos o saldo de cada conta de cada cliente
 - ▶ `select nome, numconta, saldo from cliente, possui, conta where id=idcliente and numconta=numero`

<u>nome</u>	<u>numconta</u>	<u>saldo</u>
Carlos Pereira	1	2550.21
Carlos Pereira	2	30015.12
Geraldo da Silva	3	-210.23
Maria Xavier da Silva	3	-210.23
Maria Xavier da Silva	4	9199.11
Antonio da Costa Cavalcanti	5	2.99
Francisca Emmanuel Xerxes	6	-3002.00
Francisca Emmanuel Xerxes	7	873.04
Jose Marcos do Nascimento	8	1888.44

Abrir mysql

Qualificando atributos

- É possível haver atributos em tabelas diferentes com mesmo nome
- Para evitar a confusão quando tais tabelas são usadas numa mesma consulta pode-se qualificar o nome do atributo com o nome da tabela usando a notação *tabela.atributo*
- Mesmo quando não é necessário, a qualificação pode tornar a consulta mais legível
 - ▶ Por exemplo, a consulta apresentada anteriormente poderia ter sido escrita:
 - ▶

```
select nome, numconta, saldo
      from cliente, possui, conta
     where cliente.id = possui.idcliente
       and possui.numconta = conta.numero
```

Junções Internas

- Os exemplos de junções que vimos até agora são chamados de *junções internas*
 - ▶ Todas as tabelas envolvidas têm que satisfazer a condição da junção
- O SQL tem uma sintaxe alternativa para junções internas:
 - ▶ Ao invés de escrever

```
select atributos from tabela1, tabela2  
where condição
```
 - ▶ Podemos escrever

```
select atributos from tabela1  
inner join tabela2 on condição
```

Exemplo

- `select nome, numconta from cliente
inner join possui on id=idcliente`

<u>nome</u>	<u>numconta</u>
Carlos Pereira	1
Carlos Pereira	2
Geraldo da Silva	3
Maria Xavier da Silva	3
Maria Xavier da Silva	4
Antonio da Costa Cavalcanti	5
Francisca Emmanuel Xerxes	6
Francisca Emmanuel Xerxes	7
Jose Marcos do Nascimento	8

Abrir mysql

Exemplo

- ```
select nome, numconta, saldo
from cliente
inner join possui on id = idcliente
inner join conta on numconta=numero
```

| <u>nome</u>                 | <u>numconta</u> | <u>saldo</u> |
|-----------------------------|-----------------|--------------|
| Carlos Pereira              | 1               | 2550.21      |
| Carlos Pereira              | 2               | 30015.12     |
| Geraldo da Silva            | 3               | -210.23      |
| Maria Xavier da Silva       | 3               | -210.23      |
| Maria Xavier da Silva       | 4               | 9199.11      |
| Antonio da Costa Cavalcanti | 5               | 2.99         |
| Francisca Emmanuel Xerxes   | 6               | -3002.00     |
| Francisca Emmanuel Xerxes   | 7               | 873.04       |
| Jose Marcos do Nascimento   | 8               | 1888.44      |

Abrir mysql

# Junções Externas

- Nas *junções externas*, as linhas das tabelas envolvidas não necessariamente satisfazem à condição da junção
- A principal aplicação desse tipo de operação corresponde a consultar, para cada linha de uma tabela, quais as linhas uma segunda tabela satisfazem uma determinada condição
  - ▶ Se não há linhas correspondentes na segunda tabela para uma linha da primeira, então esta é emparelhada com *valores nulos*
  - ▶ Em SQL tais consultas são escritas:

```
select atributos from tabela1
left join tabela2 on condição
```
  - ▶ Para emparelhar linhas de *tabela2* com linhas de *tabela1* usa-se

```
select atributos from tabela1
right join tabela2 on condição
```

# Exemplo

T1

| <u>a</u> | <u>b</u> |
|----------|----------|
| x        | 1        |
| y        | 2        |
| z        | 3        |

`select a,c from T1      select a,c from T1  
left join T2 on b=d      right join T2 on b=d`

| <u>a</u> | <u>c</u>    |
|----------|-------------|
| x        | m           |
| y        | k           |
| z        | <i>NULL</i> |

| <u>a</u>    | <u>c</u> |
|-------------|----------|
| y           | k        |
| <i>NULL</i> | l        |
| x           | m        |

T2

| <u>c</u> | <u>d</u> |
|----------|----------|
| k        | 2        |
| l        | 5        |
| m        | 1        |

# Junções Internas vs Externas

- Observe que junções internas e externas só dão resultados diferentes quando há linhas não emparelhadas.
- A consulta para descobrir as contas de cada cliente poderia também ser feita com um left (ou right) join:

```
select nome, numconta from cliente
left join possui on id=idcliente
```

| <u>nome</u>                 | <u>numconta</u> |
|-----------------------------|-----------------|
| Carlos Pereira              | 1               |
| Carlos Pereira              | 2               |
| Geraldo da Silva            | 3               |
| Maria Xavier da Silva       | 3               |
| Maria Xavier da Silva       | 4               |
| Antonio da Costa Cavalcanti | 5               |
| Francisca Emmanuel Xerxes   | 6               |
| Francisca Emmanuel Xerxes   | 7               |
| Jose Marcos do Nascimento   | 8               |

Abrir mysql

# Auto-Junções

- Às vezes é útil fazer junções entre uma tabela e ela mesma
- Para podermos usar a mesma tabela duas (ou mais) vezes na cláusula `from`, temos que usar *apelidos*

► Forma geral:

`from tabela1 apelido1, ... tabelaN apelidoN`

► Por exemplo, para sabermos quais pares de clientes moram no mesmo endereço, podemos escrever:

```
select c1.nome, c2.nome
from cliente c1, cliente c2
where c1.endereco=c2.endereco
and c1.nome<c2.nome
```

| nome             | nome                  |
|------------------|-----------------------|
| Geraldo da Silva | Maria Xavier da Silva |

Abrir mysql

# Ordenando as linhas

- Como o resultado de um select é sempre uma tabela, pode fazer sentido querermos ordenar as linhas
- A cláusula **order by** do comando **select** permite especificar uma ou mais chaves de ordenação. A forma geral é:
  - ▶ **order by coluna1, coluna2, ... coluna N**
  - ▶ Significa que o resultado deverá ser ordenado primariamente por *coluna1*
    - Se duas ou mais linhas têm valores de *coluna1* idênticos, então a ordem entre elas é estabelecida por *coluna2*
      - O processo se repete com *coluna3* sendo usada para ordenar valores identicos de *coluna2*, etc.
  - ▶ A cláusula **order by** deve vir após a cláusula **where**, se esta existir

# Exemplos

- `select nome, endereco from cliente order by nome`

| <u>nome</u>                 | <u>endereco</u>              |
|-----------------------------|------------------------------|
| Antonio da Costa Cavalcanti | Praca Sao Marcos, 5          |
| Carlos Pereira              | Rua do Lavradio, 51 apto 903 |
| Francisca Emmanuel Xerxes   | Rua da Bica, 102 apto 901    |
| Geraldo da Silva            | Av Chile, 88                 |
| Jose Marcos do Nascimento   | Rua Rui Barbosa, 95 apto 201 |
| Maria Xavier da Silva       | Av Chile, 88                 |

- `select nome, endereco from cliente  
where endereco like "Rua%" order by endereço`

| <u>nome</u>               | <u>endereco</u>              |
|---------------------------|------------------------------|
| Francisca Emmanuel Xerxes | Rua da Bica, 102 apto 901    |
| Carlos Pereira            | Rua do Lavradio, 51 apto 903 |
| Jose Marcos do Nascimento | Rua Rui Barbosa, 95 apto 201 |

Abrir mysql

# Agrupamento

- Ao invés de ordenar, pode-se querer agrupar linhas segundo alguma característica comum
- O agrupamento de linhas é realizado pela cláusula `group by atributo`
- Resultado contém apenas *uma linha* de cada grupo
- Se o grupo contém valores diferentes para uma coluna, apenas um deles fará parte do resultado
  - ▶ O valor escolhido pode ser qualquer um

# Exemplo

- `select nome, numconta from cliente, possui  
where id = idcliente  
group by nome`

| <u>nome</u>                 | <u>numconta</u> |
|-----------------------------|-----------------|
| Antonio da Costa Cavalcanti | 5               |
| Carlos Pereira              | 1               |
| Francisca Emmanuel Xerxes   | 6               |
| Geraldo da Silva            | 3               |
| Jose Marcos do Nascimento   | 8               |
| Maria Xavier da Silva       | 3               |

Abrir mysql

# Funções sobre grupos

- A cláusula `group by` é especialmente útil combinada com funções especiais sobre os grupos formados:
  - ▶ `count(*)` : número de elementos no grupo
  - ▶ `avg(atrib)` : média entre os valores de *atrib* no grupo
  - ▶ `max(atrib)` : máximo entre os valores de *atrib* no grupo
  - ▶ `min(atrib)` : mínimo entre os valores de *atrib* no grupo
  - ▶ `sum(atrib)` : soma de todos os valores de *atrib* no grupo

# Exemplo

- Para consultar quantas contas cada cliente possui:

```
select nome, count(*) from cliente, possui
where id = idcliente
group by nome
```

| <u>nome</u>                 | <u>count( * )</u> |
|-----------------------------|-------------------|
| Antonio da Costa Cavalcanti | 1                 |
| Carlos Pereira              | 2                 |
| Francisca Emmanuel Xerxes   | 2                 |
| Geraldo da Silva            | 1                 |
| Jose Marcos do Nascimento   | 1                 |
| Maria Xavier da Silva       | 2                 |

Abrir mysql

# Exemplo

- Consultar o saldo médio entre todas as contas de cada cliente:

```
select nome, avg(saldo)
from cliente, possui, conta
where id = idcliente and numconta=numero
group by nome
```

| <u>nome</u>                 | <u>avg ( saldo )</u> |
|-----------------------------|----------------------|
| Antonio da Costa Cavalcanti | 2.990000             |
| Carlos Pereira              | 16282.665000         |
| Francisca Emmanuel Xerxes   | -1064.480000         |
| Geraldo da Silva            | -210.230000          |
| Jose Marcos do Nascimento   | 1888.440000          |
| Maria Xavier da Silva       | 4494.440000          |

Abrir mysql

# Funções sobre grupos

- No caso particular de uma função sobre grupos ser aplicada em consultas sem a cláusula `group by`, assume-se um único grupo constituído de toda a tabela
- Por exemplo:
  - ▶ Para saber quantas linhas tem uma tabela T, pode-se usar:
    - `select count(*) from T`
  - ▶ Para saber o saldo médio de todas as contas:
    - `select avg(saldo) from conta`

# Selecionando grupos com Having

- A cláusula **having** *condição* pode ser usada para especificar uma condição sobre grupos
  - ▶ Apenas aqueles grupos que satisfazem *condição* serão selecionados
  - ▶ A *condição* normalmente envolve funções sobre grupos
  - ▶ Note que a cláusula **having** é diferente da cláusula **where**, que especifica condições sobre *linhas*

# Exemplo

- Consultar o saldo médio entre todas as contas de cada cliente que têm mais de uma conta

```
select nome, avg(saldo)
from cliente, possui, conta
where id = idcliente and numconta=numero
group by nome
having count(*)>1
```

| <u>nome</u>               | <u>avg ( saldo )</u> |
|---------------------------|----------------------|
| Carlos Pereira            | 16282.665000         |
| Francisca Emmanuel Xerxes | -1064.480000         |
| Maria Xavier da Silva     | 4494.440000          |

Abrir mysql

# O operador distinct

- Serve para assegurar que apenas um exemplar de cada linha fará parte do resultado
  - ▶ Em outras palavras, o operador **distinct** serve para eliminar linhas duplicadas
- Uso:
  - ▶ **select distinct atrib from ...**
- O mesmo efeito do operador **distinct** pode ser obtido com a cláusula **group by**

# Exemplo

T

| <u>a</u> | <u>b</u> |
|----------|----------|
| x        | 1        |
| y        | 2        |
| z        | 3        |
| x        | 4        |
| z        | 5        |

```
select distinct a
from T
```

ou

```
select a from T
group by a
```

| <u>a</u> |
|----------|
| x        |
| y        |
| z        |

# A cláusula limit

- Serve para restringir o número de linhas do resultado
- A forma geral é `limit n`, onde n é o número máximo de linhas requeridas
- Vantajoso para diminuir o tempo de processamento de consultas que potencialmente retornam resultados extensos
- Por exemplo: para listar dois saldos positivos da tabela conta, podemos usar

```
▶ select saldo from conta
 where saldo>0
 limit 2
```

| <u>saldo</u> |
|--------------|
| 2550.21      |
| 30015.12     |

Abrir mysql

# Usando select com insert

- A tabela resultante de um select pode ser usada para fornecer dados para serem inseridos em outra tabela
- A forma geral é:

```
insert into tabela (atrib1, ... atribN)
select expr1, ... exprN from ...
```

- ▶ A lista de *atributos* no comando **insert** tem que ter tantas colunas quantas as *expressões* do comando **select**

# Exemplo

- Por exemplo, podemos criar uma tabela `cliente_saldo` com atributos nome e saldo onde queremos guardar o saldo médio de todas as contas de cada cliente:

```
▶ create table cliente_saldo
 (nome varchar(50), saldo decimal (16,2))

▶ insert into cliente_saldo (nome,saldo)
 select nome, avg(saldo)
 from cliente, possui, conta
 where id = idcliente and numconta=numero
 group by nome
```

Abrir mysql

# Exemplo

- `select * from cliente_saldo`

| <u>nome</u>                 | <u>saldo</u> |
|-----------------------------|--------------|
| Antonio da Costa Cavalcanti | 2.99         |
| Carlos Pereira              | 16282.67     |
| Francisca Emmanuel Xerxes   | -1064.48     |
| Geraldo da Silva            | -210.23      |
| Jose Marcos do Nascimento   | 1888.44      |
| Maria Xavier da Silva       | 4494.44      |

Abrir mysql