Engenharia de Computação
UTFPR - Câmpus Apucarana

Disciplina: Banco de Dados

Aula de SQL

Professor: Wendel Góes

SQL (Structured Query Language)

- Linguagem comercial para BD relacional
  - padrão ISO desde a década de 80
    - SQL-1 (86); SQL-2 (92); SQL-3 (99)
  - não é apenas uma linguagem de consulta!
    - como o nome sugere...
- Base Formal
  - álgebra relacional e cálculo relacional

SQL (Structured Query Language

- Funcionalidades principais
  - definição (DDL) e manipulação (DML) de dados
  - definição de visões e autorizações de acesso
  - definição de restrições de integridade
  - definição de transações
  - comandos para embutimento em LPs

SQL - DDL

- **Criação de um BD**
  - SQL padrão não oferece tal comando
    - BDs são criados via ferramentas do SGBD
  - alguns SGBDs (SQL Server, DB2, MySQL) oferecem este comando
    - create database *nome_BD*
    - drop database *nome_BD*

SQL - DDL

- Comandos para definição de esquemas
  - create table
    - define a estrutura da tabela, suas restrições de integridade e cria uma tabela vazia
  - alter table
    - modifica a definição de uma tabela (I / E / A atributos; I / E RIs)
  - drop table
    - remove uma tabela com todas as suas tuplas

SQL - *Create Table*

CREATE TABLE *nome_tabela* (
nome_atributo_1 tipo_1 [[NOT]NULL][UNIQUE]
[{, *nome_atributo_n tipo_n*}]
[, **PRIMARY KEY** (*nome(s)_atributo(s)*)]
[{, **FOREIGN KEY** (*nome_atributo*)
REFERENCES *nome_tabela*}]
)

- Principais tipos de dados do MySQL
  - *integer, smallint, numeric(tamanho[,nro_casas_decimais]), char(tamanho), varchar(tamanho), date, time, datetime, ...*
  - formato para data e hora
    - "DD-MM-YYYY"

Exemplos de Criação de Tabela

```
CREATE TABLE Ambulatorios (
                    nroa                integer,
                    andar               numeric(3) NOT NULL,
                    capacidade      smallint,
                    PRIMARY KEY(nroa)
)

CREATE TABLE Medicos (
                codm                integer,
                nome                varchar(40) NOT NULL,
                idade           smallint    NOT NULL,
   especialidade    char(20),
   CPF              numeric(11) UNIQUE,
   cidade           varchar(30),
   nroa             integer,
   PRIMARY KEY(codm),
   FOREIGN KEY(nroa) REFERENCES Ambulatorios
)
```

SQL – *Alter Table*

ALTER TABLE *nome_tabela*
ADD [COLUMN] *nome_atributo_1 tipo_1 [{RIs}]*
  [{, *nome_atributo_n tipo_n [{RIs}]*}]
|
MODIFY [COLUMN] *nome_atributo_1 tipo_1 [{RIs}]*
  [{, *nome_atributo_n tipo_n [{RIs}]*}]
|
DROP COLUMN *nome_atributo_1*
  [{, *nome_atributo_n* }]
|
ADD CONSTRAINT *nome_RI_1 def_RI_1*
  [{, *nome_RI_n def_RI_n*}]
|
DROP CONSTRAINT *nome_RI_1*
  [{, *nome_RI_n*}]
|
[**ADD**|**DROP**] [**PRIMARY KEY** ...|**FOREIGN KEY** ...]

Exemplos de Alteração de Tabelas

```
ALTER TABLE Ambulatórios
   ADD nome VARCHAR(30)

ALTER TABLE Médicos DROP PRIMARY KEY

ALTER TABLE Pacientes DROP COLUMN doenca, DROP COLUMN
    cidade

ALTER TABLE Funcionários
ADD FOREIGN KEY(nroa)REFERENCES Ambulatórios

ALTER TABLE Funcionarios
   ADD constraint fk_nroa
     FOREIGN KEY(nroa) REFERENCES Ambulatorios
```

SQL - Índices

- Definidos sobre atributos para acelerar consultas a dados
- Índices são definidos automaticamente para chaves primárias
- Operações

CREATE [UNIQUE] INDEX *nome_índice* **ON** *nome_tabela* (*nome_atributo_1*[{, *nome_atributo_n* }])

DROP INDEX *nome_índice* **ON** *nome_tabela*

- Exemplos

CREATE UNIQUE INDEX indPac_CPF ON Pacientes (CPF)
DROP INDEX indPac_CPF ON Pacientes

SQL – DML

- Define operações de manipulação de dados
  - I (INSERT)
  - A (UPDATE)
  - E (DELETE)
  - C (SELECT)

- Instruções declarativas
  - manipulação de conjuntos
  - especifica-se o *que fazer* e não *como fazer*

SQL – DML

- **Inserção de dados**

  INSERT INTO *nome_tabela* [(*lista_atributos*)]
  VALUES [(*lista_valores_atributos*)]

- Exemplos

  INSERT INTO Ambulatorios VALUES (1, 1, 30)

  INSERT INTO Medicos
  (codm, nome, idade, especialidade, CPF, cidade)
  VALUES (4, ‘Carlos’, 28,’ortopedia’, 11000110000,
  ‘Joinville’);

SQL - DML

- **Alteração de dados**

UPDATE *nome_tabela*
SET *nome_atributo_1 = Valor*
[{, *nome_atributo_n = Valor*}]
[**WHERE** *condição*]

- Exemplos

UPDATE Medicos
SET cidade = 'Florianopolis'

UPDATE Ambulatorios
SET capacidade = capacidade + 5, andar = 3
WHERE nroa = 2

SQL – DML

- **Exclusão de dados**

DELETE FROM *nome_tabela*
[**WHERE** *condição*]

- Exemplos

DELETE FROM Ambulatorios

DELETE FROM Medicos
    WHERE especialidade = ‘cardiologia’
or cidade < > ‘Florianopolis’

SQL – Consultas Básicas

- **Consulta a dados de uma tabela**

select *lista_atributos*
from *tabela*
[**where** *condição*]

Consulta a uma Tabela

- Exemplos

CONSULTAS SQL
Select * From Pacientes
Select * From Pacientes Where idade > 18
Select CPF, nome From Pacientes
Select CPF, nome From Pacientes Where idade > 18

Comando SELECT

- Facilidades para projeção de informações
  - Não há eliminação de duplicatas no Select
    - tabela ≡ coleção
  - retorno de valores calculados
    - uso de operadores aritméticos (+,-,*,/)
  - invocação de funções de agregação
    - COUNT (contador de ocorrências [de um atributo])
    - MAX / MIN (valores máximo / mínimo de um atributo)
    - SUM (somador de valores de um atributo)
    - AVG (média de valores de um atributo)

Comando SELECT

- **Eliminação de duplicatas**

  select [distinct] *lista_atributos*

  ...

- Exemplo
  - buscar as especialidades dos médicos

```
select distinct especialidade
from Médicos
```

Comando SELECT

- Retorno de valores calculados - Exemplos
  - quantos grupos de 5 leitos podem ser formados em cada ambulatório?

select nroa, capacidade/5 as grupos5

from Ambulatórios

→ qual o salário líquido dos funcionários (desc. 10%)?

select CPF, salário - (salário * 0.1) as líquido from Funcionários

Comando SELECT

- Retorno de valores calculados - Exemplo

select nroa, capacidade/5 as grupos5
from Ambulatórios

nroa	grupos5
1	8.0000
2	10.0000
3	8.0000
4	5.0000
5	11.0000

Comando SELECT

- **Função COUNT - Exemplos**
  - informar o total de médicos ortopedistas

```
select count(*) as TotalOrtopedistas
from Médicos
where especialidade = ´ortopedia´
```

  - total de médicos que atendem em ambulatórios

```
select count(nroa) as Total
from Médicos
```

não conta nulos

Comando SELECT

- **Função COUNT - Exemplos**
  - informar o total de médicos ortopedistas

```
select count(*) as TotalOrtopedistas
from Médicos
where especialidade = ´ortopedia´
```

```
mysql> select count(*) as TotalOrtopedistas from
Medicos where especialidade = 'ortopedia';
+-------------------+
| TotalOrtopedistas |
+-------------------+
|                 2 |
+-------------------+
1 row in set (0,01 sec)
```

Comando SELECT

- **Função COUNT - Exemplos**

  - total de médicos que atendem em ambulatórios

```
select count(nroa) as Total
from Médicos
```

não conta nulos

```
mysql> select count(nroa) as Total from medicos;
+-------+
| Total |
+-------+
|     4 |
+-------+
1 row in set (0,00 sec)
```

Comando SELECT

- **Função SUM - Exemplo**
  - informar a capacidade total dos ambulatórios do primeiro andar

```
select sum(capacidade) as TotalAndar1
from Ambulatórios
where andar = 1
```

Comando SELECT

- **Função SUM - Exemplo**
  - informar a capacidade total dos ambulatórios do primeiro andar

```
mysql> select sum(capacidade) as TotalAndar1 from
 Ambulatorios where andar = 1;
+-------------+
| TotalAndar1 |
+-------------+
|          50 |
+-------------+
1 row in set (0,00 sec)
```

Comando SELECT

- **Função AVG - Exemplo**
  - informar a média de idade dos pacientes de Florianópolis

```
select avg(idade) as MediaPacFpolis
from Pacientes
where cidade = ´Florianópolis´
```

Comando SELECT

- **Função AVG - Exemplo**
  - informar a média de idade dos pacientes de Florianópolis

```
select avg(idade) as MediaPacFpolis
from Pacientes
where cidade = ´Florianópolis´
```

```
mysql> select avg(idade) as MediaPacFpolis from
Pacientes where cidade = 'Florianopolis';
+----------------+
| MediaPacFpolis |
+----------------+
|        20.0000 |
+----------------+
1 row in set (0,00 sec)
```

Comando SELECT

- **Funções MAX / MIN - Exemplo**
  - informar o menor e o maior salário pagos aos Funcionários do departamento pessoal com mais de 50 anos

```
select min(salário) as mínimo,
     max(salário) as máximo
    from Funcionários
    where depto = ´Pessoal´
     and idade > 50
```

Comando SELECT

- **Funções MAX / MIN - Exemplo**
  - informar o menor e o maior salário pagos aos Funcionários do departamento pessoal com mais de 50 anos

```
select min(salário) as mínimo,
     max(salário) as máximo
    from Funcionários
    where depto = ´Pessoal´
     and idade > 50
```

Comando SELECT

- **Funções de Agregação com distinct**
  - valores duplicados não são computados
  - exemplos

```
select count(distinct especialidade)
 from Médicos

select avg(distinct salário)
 from Funcionários
```

Comando SELECT

- **Observação sobre as funções de agregação**
  - não podem ser combinadas a outros atributos da tabela no resultado da consulta

```
select andar, COUNT (andar)
from Ambulatórios
```

Cláusula WHERE

- Facilidades para seleção de dados
  - busca por padrões
    - cláusula [NOT] LIKE
  - teste de existência de valores nulos
    - cláusula IS [NOT] NULL
  - busca por intervalos de valores
    - cláusula [NOT] BETWEEN *valor1* AND *valor2*
  - teste de pertinência elemento-conjunto
    - cláusula [NOT] IN

Cláusula WHERE

- **Busca por padrões**

```
where atributo like ´padrão´
% : casa com qq cadeia de caracteres
´_´ : casa com um único caractere
[a-f] : casa com qq caractere entre
        ´a´ e ´f´ (SQL-Server)
```

- Exemplos
  - buscar CPF e nome dos médicos com inicial M

```
select CPF, nome
from Médicos
    where nome like ´M%´
```

Cláusula WHERE

- Exemplos
  - buscar nomes de pacientes cujo CPF termina com 20000 ou 30000

```
select nome
from Pacientes
where CPF like '%20000‘
or CPF like '%30000‘
```

- **Observações**
  - em alguns dialetos SQL, ´*´ é usado invés de ´%´
  - não é possível testar padrões em atributos *datetime* (SQL-Server)

Cláusula WHERE

- **Teste de valores nulos - Exemplo**
  - buscar o CPF e o nome dos médicos que não dão atendimento em ambulatórios

```
select CPF, nome
from Médicos
where nroa is null
```

Cláusula WHERE

- **Busca por intervalos de valores - Exemplo**

  - buscar os dados das consultas marcadas para o período da tarde

```
select *
from Consultas
    where hora between ´14:00´ and ´18:00´
```

Cláusula WHERE

- **Teste de pertinência elemento-conjunto - Exemplo**
  - buscar os dados das médicos ortopedistas, traumatologistas e cardiologistas de Florianópolis

```
      select *
from Médicos
where cidade = ´Florianópolis´
and especialidade in (´cardiologia´,
                      ´traumatologia´,
                      ´cardiologia´)
```

União de Tabelas

- Implementa a união da álgebra relacional
  - exige tabelas compatíveis

SQL
consultaSQL1 `union` *consultaSQL2*

- Exemplo
  - buscar o nome e o CPF dos médicos e pacientes

```
select CPF, nome
from Médicos
  union
   select CPF, nome
   from Pacientes
```

SQL - DML

- **Consultas envolvendo mais de uma tabela**

```
select lista_atributos
from tabela1, ..., tabelam
[where condição]
```

Exemplos

CONSULTA COM VÁRIAS TABELAS

```
Select *
From Pacientes, Consultas
```

```
Select CPF, nome, data
From Pacientes,Consultas
Where hora > '12:00'
and Pacientes.codp = Consultas.codp
```

```
Select m2.nome
From Médicos m1, Médicos m2
Where m1.nome = 'João'
and m1.especialidade = m2.especialidade
```

Produto cartesiano

SELECT * FROM Aluno, Materia

Aluno	
Matricula	Nome
101	Ana
102	Alessandra
103	Adriana
104	Luísa
105	Rodrigo

Materia	
Id	Nome
1	Português
2	Inglês
3	História
4	Geografia
5	Biologia

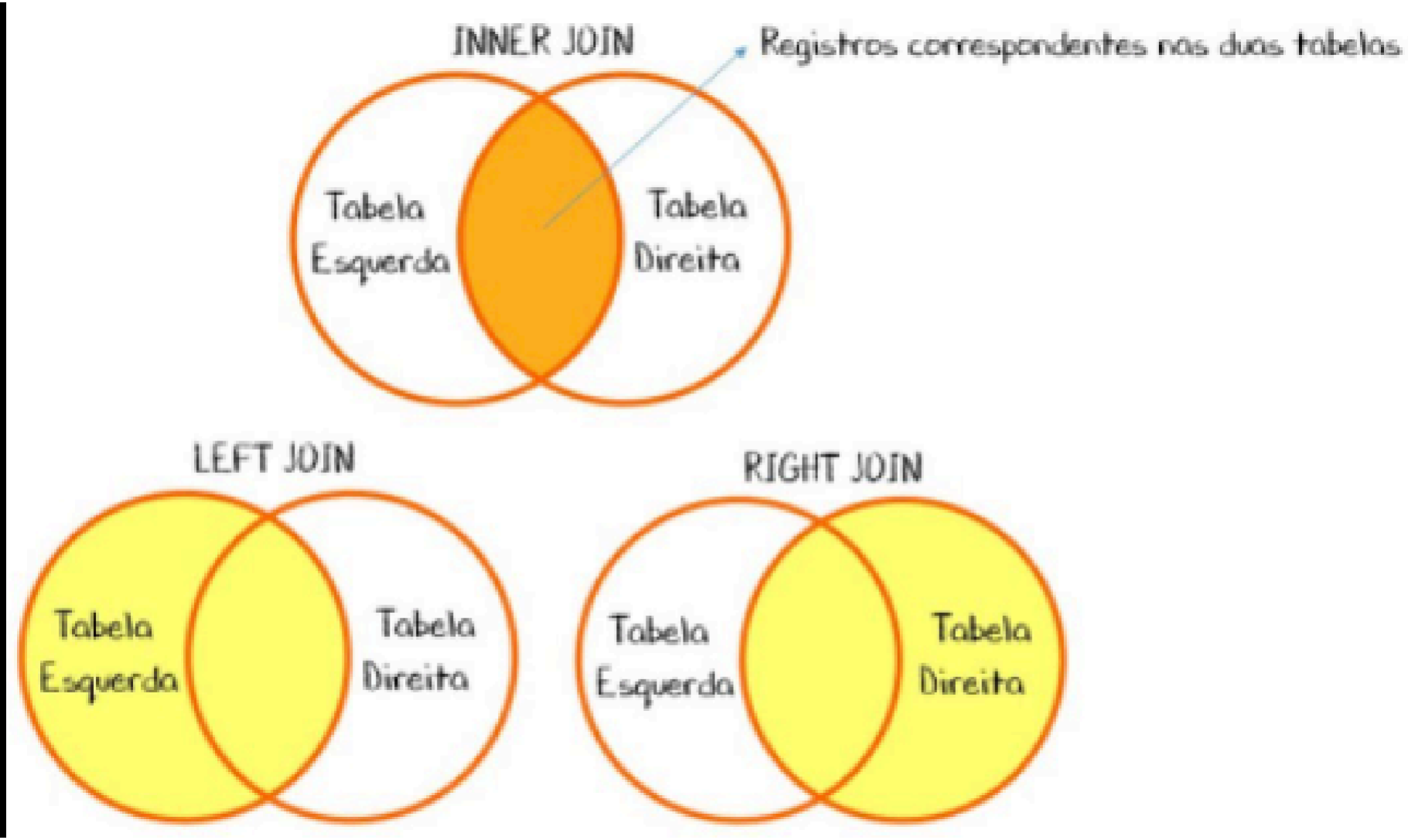
Colunas de Aluno		Colunas de Materia	
Matricula	Nome	Id	Nome
101	Ana	1	Português
101	Ana	2	Inglês
101	Ana	3	História
101	Ana	4	Geografia
101	Ana	5	Biologia
102	Alessandra	1	Português
102	Alessandra	2	Inglês
102	Alessandra	3	História
102	Alessandra	4	Geografia
102	Alessandra	5	Biologia
103	Adriana	1	Português
103	Adriana	2	Inglês
103	Adriana	3	História
103	Adriana	4	Geografia
103	Adriana	5	Biologia
104	Luísa	1	Português
104	Luísa	2	Inglês
104	Luísa	3	História
104	Luísa	4	Geografia
104	Luísa	5	Biologia
105	Rodrigo	1	Português
105	Rodrigo	2	Inglês
105	Rodrigo	3	História
105	Rodrigo	4	Geografia
105	Rodrigo	5	Biologia

Junção

- **Sintaxe**

```
select lista_atributos
from tabela1 [inner] join tabela2 on
  condição_junção [join tabela3 on ...]
[where condição]
```

Junção


INNER JOIN
Registros correspondentes nas duas tabelas
Tabela Esquerda
Tabela Direita
LEFT JOIN
Tabela Esquerda
Tabela Direita
RIGHT JOIN
Tabela Esquerda
Tabela Direita

Exemplos

CONSULTA COM JUNÇÃO DE TABELAS

```
Select *
From Pacientes join Consultas on
Pacientes.codp = Consultas.codp
```

```
Select nome
From Médicos join
Consultas on Médicos.codm =
Consultas.codm
Where data = '2006/11/13'
```

Exemplos

CONSULTA COM JUNÇÃO DE TABELAS

```
Select *
From Pacientes join Consultas on
Pacientes.codp = Consultas.codp
```

```
mysql> Select * From Pacientes inner join Consultas on Pacientes.codp = Consultas.codp;
+------+-------------+-------+-------+--------------+----------+------+------+------------+----------+
| codp | CPF         | nome  | idade | cidade       | doenca   | codm | codp | data       | hora     |
+------+-------------+-------+-------+--------------+----------+------+------+------------+----------+
|    1 | 20000200000 | Ana   |    20 | Florianopolis | gripe    |    1 |    1 | 2006-06-12 | 14:00:00 |
|    4 | 11000110000 | Carlos |    28 | Joinville    | sarampo  |    1 |    4 | 2006-06-13 | 10:00:00 |
|    1 | 20000200000 | Ana   |    20 | Florianopolis | gripe    |    2 |    1 | 2006-06-13 | 09:00:00 |
|    2 | 20000220000 | Paulo |    24 | Ilhota       | fratura  |    2 |    2 | 2006-06-13 | 11:00:00 |
|    3 | 22000200000 | Lucia |    30 | Biguacu      | tendinite |    2 |    3 | 2006-06-14 | 14:00:00 |
|    4 | 11000110000 | Carlos |    28 | Joinville    | sarampo  |    2 |    4 | 2006-06-14 | 17:00:00 |
|    1 | 20000200000 | Ana   |    20 | Florianopolis | gripe    |    3 |    1 | 2006-06-19 | 18:00:00 |
|    3 | 22000200000 | Lucia |    30 | Biguacu      | tendinite |    3 |    3 | 2006-06-12 | 10:00:00 |
|    4 | 11000110000 | Carlos |    28 | Joinville    | sarampo  |    3 |    4 | 2006-06-19 | 13:00:00 |
|    4 | 11000110000 | Carlos |    28 | Joinville    | sarampo  |    4 |    4 | 2006-06-20 | 13:00:00 |
|    4 | 11000110000 | Carlos |    28 | Joinville    | sarampo  |    4 |    4 | 2006-06-22 | 19:30:00 |
+------+-------------+-------+-------+--------------+----------+------+------+------------+----------+
11 rows in set (0,00 sec)
```

Junção Natural

- Sintaxe

```
select lista_atributos
from tabela1 natural join tabela2
[natural join tabela3 ...]
[where condição]
```

Exemplos

SQL
Select * From Pacientes natural join Consultas
Select nome From Médicos natural join Consultas Where data = '2006/11/13'

Exemplos

SQL

```
Select *
From Pacientes natural join
Consultas
```

```
mysql> Select *
    -> From Pacientes natural join Consultas;
+------+------------+--------+-------+--------------+----------+------+------------+----------+
| codp | CPF        | nome   | idade | cidade       | doenca   | codm | data       | hora     |
+------+------------+--------+-------+--------------+----------+------+------------+----------+
|    1 | 20000200000 | Ana    |    20 | Florianopolis | gripe    |    1 | 2006-06-12 | 14:00:00 |
|    4 | 11000110000 | Carlos |    28 | Joinville     | sarampo  |    1 | 2006-06-13 | 10:00:00 |
|    1 | 20000200000 | Ana    |    20 | Florianopolis | gripe    |    2 | 2006-06-13 | 09:00:00 |
|    2 | 20000220000 | Paulo  |    24 | Ilhota        | fratura  |    2 | 2006-06-13 | 11:00:00 |
|    3 | 22000200000 | Lucia  |    30 | Biguacu       | tendinite |    2 | 2006-06-14 | 14:00:00 |
|    4 | 11000110000 | Carlos |    28 | Joinville     | sarampo  |    2 | 2006-06-14 | 17:00:00 |
|    1 | 20000200000 | Ana    |    20 | Florianopolis | gripe    |    3 | 2006-06-19 | 18:00:00 |
|    3 | 22000200000 | Lucia  |    30 | Biguacu       | tendinite |    3 | 2006-06-12 | 10:00:00 |
|    4 | 11000110000 | Carlos |    28 | Joinville     | sarampo  |    3 | 2006-06-19 | 13:00:00 |
|    4 | 11000110000 | Carlos |    28 | Joinville     | sarampo  |    4 | 2006-06-20 | 13:00:00 |
|    4 | 11000110000 | Carlos |    28 | Joinville     | sarampo  |    4 | 2006-06-22 | 19:30:00 |
+------+------------+--------+-------+--------------+----------+------+------------+----------+
11 rows in set (0,00 sec)
```

Junções Externas (Não Naturais)

- **Sintaxe**

```
select lista_atributos
from tabela1 left|right|full [outer] join
  tabela2 on condição_junção
  [join tabela3 on ...]
[where condição]
```

Exemplos

SQL
Select * From Pacientes left join Consultas on Pacientes.codp = Consultas.codp
Select nome From Médicos right join Consultas on Médicos.codm = Consultas.codm Where data = '2003-05-13'

Left join

```
mysql> select * from pacientes;
+------+------------+--------+-------+--------------+-----------+
| codp | CPF        | nome   | idade | cidade       | doenca    |
+------+------------+--------+-------+--------------+-----------+
|    1 | 20000200000 | Ana    |    20 | Florianopolis | gripe     |
|    2 | 20000220000 | Paulo  |    24 | Ilhota       | fratura   |
|    3 | 22000200000 | Lucia  |    30 | Biguacu      | tendinite |
|    4 | 11000110000 | Carlos |    28 | Joinville    | sarampo   |
+------+------------+--------+-------+--------------+-----------+

mysql> select * from consultas;
+------+------+------------+----------+
| codm | codp | data       | hora     |
+------+------+------------+----------+
|    1 |    1 | 2006-06-12 | 14:00:00 |
|    1 |    4 | 2006-06-13 | 10:00:00 |
|    2 |    1 | 2006-06-13 | 09:00:00 |
|    2 |    2 | 2006-06-13 | 11:00:00 |
|    2 |    3 | 2006-06-14 | 14:00:00 |
|    2 |    4 | 2006-06-14 | 17:00:00 |
|    3 |    1 | 2006-06-19 | 18:00:00 |
|    3 |    3 | 2006-06-12 | 10:00:00 |
|    3 |    4 | 2006-06-19 | 13:00:00 |
|    4 |    4 | 2006-06-20 | 13:00:00 |
|    4 |    4 | 2006-06-22 | 19:30:00 |
+------+------+------------+----------+
11 rows in set (0,00 sec)
```

Exemplos

SQL

```
Select *
From Pacientes left join Consultas on
Pacientes.codp = Consultas.codp
```

```
mysql> Select * From Pacientes left join Consultas on Pacientes.codp = Consultas.codp;
+------+-------------+--------+-------+--------------+-----------+------+------+------------+----------+
| codp | CPF         | nome   | idade | cidade       | doenca    | codm | codp | data       | hora     |
+------+-------------+--------+-------+--------------+-----------+------+------+------------+----------+
|    1 | 20000200000 | Ana    |    20 | Florianopolis | gripe     |    3 |    1 | 2006-06-19 | 18:00:00 |
|    1 | 20000200000 | Ana    |    20 | Florianopolis | gripe     |    2 |    1 | 2006-06-13 | 09:00:00 |
|    1 | 20000200000 | Ana    |    20 | Florianopolis | gripe     |    1 |    1 | 2006-06-12 | 14:00:00 |
|    2 | 20000220000 | Paulo  |    24 | Ilhota       | fratura   |    2 |    2 | 2006-06-13 | 11:00:00 |
|    3 | 22000200000 | Lucia  |    30 | Biguacu      | tendinite |    3 |    3 | 2006-06-12 | 10:00:00 |
|    3 | 22000200000 | Lucia  |    30 | Biguacu      | tendinite |    2 |    3 | 2006-06-14 | 14:00:00 |
|    4 | 11000110000 | Carlos |    28 | Joinville    | sarampo   |    4 |    4 | 2006-06-22 | 19:30:00 |
|    4 | 11000110000 | Carlos |    28 | Joinville    | sarampo   |    4 |    4 | 2006-06-20 | 13:00:00 |
|    4 | 11000110000 | Carlos |    28 | Joinville    | sarampo   |    3 |    4 | 2006-06-19 | 13:00:00 |
|    4 | 11000110000 | Carlos |    28 | Joinville    | sarampo   |    2 |    4 | 2006-06-14 | 17:00:00 |
|    4 | 11000110000 | Carlos |    28 | Joinville    | sarampo   |    1 |    4 | 2006-06-13 | 10:00:00 |
+------+-------------+--------+-------+--------------+-----------+------+------+------------+----------+
11 rows in set (0,00 sec)
```

Right join

```
mysql> select * from consultas;
+------+------+------------+----------+
| codm | codp | data       | hora     |
+------+------+------------+----------+
|    1 |    1 | 2006-06-12 | 14:00:00 |
|    1 |    4 | 2006-06-13 | 10:00:00 |
|    2 |    1 | 2006-06-13 | 09:00:00 |
|    2 |    2 | 2006-06-13 | 11:00:00 |
|    2 |    3 | 2006-06-14 | 14:00:00 |
|    2 |    4 | 2006-06-14 | 17:00:00 |
|    3 |    1 | 2006-06-19 | 18:00:00 |
|    3 |    3 | 2006-06-12 | 10:00:00 |
|    3 |    4 | 2006-06-19 | 13:00:00 |
|    4 |    4 | 2006-06-20 | 13:00:00 |
|    4 |    4 | 2006-06-22 | 19:30:00 |
+------+------+------------+----------+
11 rows in set (0,00 sec)
```

codm	CPF	nome	idade	especialidade	cidade	nroa
1	10000100000	Joao	40	ortopedia	Florianopolis	1
2	10000110000	Maria	42	traumatologia	Blumenau	2
3	11000100000	Pedro	51	pediatria	São José	2
4	11000110000	Carlos	28	ortopedia	Joinville	NULL
5	11000111000	Marcia	33	neurologia	Biguacu	3

5 rows in set (0,00 sec)

Exemplos

SQL

```
Select nome
From Médicos right join
Consultas on Médicos.codm =
Consultas.codm
Where data = ‘2006-06-14’
```

```
mysql> Select nome From Medicos right join Consultas on Medicos.codm =
Consultas.codm Where data = '2006-06-14';
+-------+
| nome  |
+-------+
| Maria |
| Maria |
+-------+
2 rows in set (0,01 sec)
```

```
mysql> Select nome From Medicos left join Consultas on Medicos.codm = Consultas.
codm;
+--------+
| nome   |
+--------+
| Joao   |
| Joao   |
| Maria  |
| Maria  |
| Maria  |
| Maria  |
| Pedro  |
| Pedro  |
| Pedro  |
| Carlos |
| Carlos |
| Marcia |
+--------+
12 rows in set (0,01 sec)

mysql> Select nome From Medicos right join Consultas on Medicos.codm = Consultas
.codm;
+--------+
| nome   |
+--------+
| Joao   |
| Joao   |
| Maria  |
| Maria  |
| Maria  |
| Maria  |
| Pedro  |
| Pedro  |
| Pedro  |
| Carlos |
| Carlos |
+--------+
11 rows in set (0,00 sec)
```

Subconsultas ou Consultas Aninhadas

- Forma alternativa de especificar consultas envolvendo relacionamentos entre tabelas
- **Otimização**
  - filtragens prévias de dados na subconsulta
    - apenas tuplas/atributos de interesse são combinados com dados da(s) tabela(s) da consulta externa
- Cláusulas de subconsulta
  - *nome_atributo* [NOT] IN (*consulta_SQL*)
  - *nome_atributo* [< | <= | > | >= | < > | !=] ANY (*consulta_SQL*)
  - *nome_atributo* [< | <= | > | >= | < > | !=] ALL (*consulta_SQL*)

Subconsultas com IN

- Testam a relação de pertinência ou não-pertinência elemento-conjunto

```
select lista_atributos
from tabela1 [...]
where atributo_ou_expressão [NOT] IN
(consulta_SQL)
```

Exemplos

SQL

```
Select nome
From Médicos
Where codm in (select codm
 from Consultas
 where data = '06/11/13')
```

```
Select CPF
From Funcionários
Where CPF not in (select CPF
 from Pacientes)
```

```
Select CPF
From Médicos
Where CPF in (select CPF
 from Pacientes)
```

Subconsultas com ANY

- Permitem outras comparações do tipo elemento-conjunto
  - testa se um valor é >, <, =, ... que *algum* valor em um conjunto

```
select lista_atributos
from tabela1 [, ...]
where atributo_ou_expressão [=|<|<=|>|>=|<>|!=] ANY
(consulta_SQL)
```

Exemplos

SQL

```
Select nome
From Médicos
Where codm = any (ou in)
(select codm
 from Consultas
 where data = '06/11/13')
```

```
Select nome
From Funcionários
Where idade < any (
Select idade from Funcionários)
```

Subconsultas com ALL

- Realiza uma comparação de igualdade ou desigualdade de um elemento com todos os elementos de um conjunto

```
select lista_atributos
from tabela1 [, ...]
where atributo_ou_expressão [=|<|<=|>|>=|<>|!=]
  ALL(consulta_SQL)
```

- Não tem mapeamento para a álgebra relacional
  - não é equivalente a divisão
    - na divisão existe apenas comparação de igualdade
    - dividendo deve ter mais atributos que o divisor
    - não filtra automaticamente atributos do dividendo

Exemplos

```
Select nome
From Funcionários
Where salário > all
       (Select salário
        From Funcionários
        Where departamento = 'contábil')
```

```
Select nome
From Funcionários
Where CPF < > all (or not in)
       (Select CPF
        From Pacientes)
```

Comparações Elemento-Elemento

- Casos em que a subconsulta retorna apenas um elemento como resultado
  - cardinalidade da subconsulta = 1
  - não é utilizada nenhuma cláusula de subconsulta neste caso

```
select lista_atributos
from tabela1 [, ...]
where atributo_ou_expressão [=|<|<=|>|>=|<>|!=]
  (consulta_SQL com um único elemento)
```

Exemplos

```
Select nome
From Funcionários
Where salário >
       (Select salário
        From Funcionários
        Where CPF = 22000200002)
```

```
select nome, CPF
from Médicos
where CPF < > 10000100001
and especialidade =
       (select especialidade
        from Médicos
        where CPF = 10000100001)
```

Subconsultas com EXISTS

- Quantificador existencial do cálculo relacional
  - testa se um predicado é V ou F na subconsulta
  - para cada tupla da consulta externa a ser analisada, a subconsulta é executada

```
select lista_atributos
from tabela1 [, ...]
where [NOT] EXISTS (consulta_SQL)
```

Exemplos

SQL

```
Select nome
From Médicos m
Where exists
  (Select  *
   From Consultas
   Where data = '06/11/13'
   and codm = m.codm)
```

```
Select f.nome
From Funcionários f
Where f.depto = 'pessoal'
and not exists
  (Select  *
   From Pacientes
   Where CPF = f.CPF)
```

Exemplo

SQL

```
Select p.nome
From Pacientes p
Where not exists
  (Select  *
   From Médicos m
   Where not exists
   (Select *
    From Consultas c
    Where c.codm = m.codm
    and c.codp = p.codp))
```

Subconsulta na Cláusula FROM

- Gera uma tabela derivada a partir de uma ou mais tabelas, para uso na consulta externa
  - otimização: filtra linhas e colunas de uma tabela que são desejadas pela consulta externa

```
select lista_atributos
from (consulta_SQL) as nome_tabela_derivada
```

Exemplos

SQL

```
select Medicos.*, C.hora
from Medicos join
 (select codm, hora
  from Consultas
  where data = '06/11/13')
  as C
on Médicos.codm = C.codm
```

```
select Amb.*
from (select nroa, andar from
ambulatorios) as Amb join
 (select nroa from Medicos
  where cidade = 'Fpolis')
  as MFlo
on Amb.nroa = MFlo.nroa
```

Ordenação de Resultados

- **Cláusula ORDER BY**

```
select lista_atributos
from lista_tabelas
[where condição]
[order by nome_atributo 1 [desc] {[,
  nome_atributo n [desc]]} ]
```

- Exemplos

```
select *
from Pacientes
order by nome
```

```
select salário, nome
from Funcionários
order by salário desc, nome
```

Ordenação de Resultados

- É possível determinar a quantidade de valores ordenados a retornar

```
select ...
 limit valor1 [,valor2]
```

- Exemplos

retorna as 5 primeiras tuplas

```
select *
from Pacientes
order by nome
limit 5
```

retorna tuplas 6 a 15

```
select salário, nome
from Funcionários
order by salário desc,
nome
limit 5,10
```

Definição de Grupos

- **Cláusula GROUP BY**

  select *lista_atributos*
  from lista_*tabelas*
  [**where** *condição*]
  [**group by** *lista_atributos_agrupamento*
  [**having** *condição_para_agrupamento*]]

- **GROUP BY**
  - define grupos para combinações de valores dos atributos definidos em *lista_atributos_agrupamento*
  - apenas atributos definidos em *lista_atributos_agrupamento* podem aparecer no resultado da consulta
  - geralmente o resultado da consulta possui uma função de agregação

Definição de Grupos

- Exemplo

  select especialidade, count(*)

  from Médicos

  group by especialidade

especialidade	*Count*
ortopedia	2
pediatira	1
neurologia	1
traumatologia	3

especialidade	"grupos"					
ortopedia	**codm**	**nome**	**idade**	**RG**	**cidade**	**nroa**
	1	João	40	1000010000	Fpolis	1
	4	Carlos	28	1100011000	Joinville	
pediatria	**codm**	**nome**	**idade**	**RG**	**cidade**	**nroa**
	3	Pedro	51	1100010000	Fpolis	2
neurologia	**codm**	**nome**	**idade**	**RG**	**cidade**	**nroa**
	5	Márcia	33	1100011100	Biguaçu	3
traumatologia	**codm**	**nome**	**idade**	**RG**	**cidade**	**nroa**
	2	Maria	42	1000011000	Blumenau	2
	6	Joana	37	1111110000	Fpolis	3
	7	Mauro	53	1111000011	Blumenau	2

Definição de Grupos

- Cláusula HAVING
  - define condições para que grupos sejam formados
    - condições só podem ser definidas sobre atributos do agrupamento ou serem funções de agregação
  - existe somente associada à cláusula GROUP BY

- Exemplos

```
select especialidade, count(*)
from Médicos
group by especialidade
having count(*) > 1
```

Atualização com Consulta

- Comandos de atualização podem incluir comandos de consulta
  - necessário toda vez que a atualização deve testar relacionamentos entre tabelas

- Exemplo 1

```
delete from Consultas
where hora > ’17:00:00’
and codm in (select codm
             from Médicos
             where nome = ‘Maria’)
```

Atualização com Consulta

- **Exemplo 2**

```
update Médicos
set nroa = NULL
where not exists
 (select * from Médicos m
  where m.codm <> Médicos.codm
  and m.nroa = Médicos.nroa)
```

- **Exemplo3**

```
update Ambulatórios
set capacidade = capacidade +
        (select capacidade
         from Ambulatórios where nroa = 4)
where nroa = 2
```

Atualização com Consulta

- **Exemplo 4** (supondo MedNovos(código, nome, especialidade))

```
insert into MedNovos
   select codm, nome, especialidade
   from Médicos
   where idade < 21;
```