

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
- 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 doença,  
    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)  
        [, (lista_valores_atributos)]
```

MySQL



- 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'
```

Exercícios (MySQL)

1. Crie um BD com nome **Clinica**
2. Crie as seguintes tabelas neste BD, considerando que os atributos sublinhados são chaves primárias e os em itálico são chaves estrangeiras:
 - **Ambulatorios:** nroa (int), andar (numeric(3)) (não nulo), capacidade (smallint)
 - **Medicos:** codm (int), nome (varchar(40)) (não nulo), idade (smallint) (não nulo), especialidade (char(20)), CPF (numeric(11)) (único), cidade (varchar(30)), *nroa* (int)
 - **Pacientes:** codp (int), nome (varchar(40)) (não nulo), idade (smallint) (não nulo), cidade (char(30)), CPF (numeric(11)) (único), doença (varchar(40)) (não nulo)
 - **Funcionarios:** codf (int), nome (varchar(40)) (não nulo), idade (smallint), CPF (numeric(11)) (único), cidade (varchar(30)), salario (numeric(10)), cargo (varchar(20))
 - **Consultas:** codm (int), *codp* (int), data (date), hora (time)
3. Crie a coluna **nroa (int)** na tabela **Funcionarios**
4. Crie os seguintes índices:
 - Medicos: CPF (único)
 - Pacientes: doença
5. Remover o índice **doença** em Pacientes
6. Remover as colunas **cargo** e **nroa** da tabela de **Funcionarios**

Exercícios (MySQL)

Popular as tabelas:

Medicos

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

Ambulatorios

nroa	andar	capacidade
1	1	30
2	1	50
3	2	40
4	2	25
5	2	55

Pacientes

codp	nome	idade	cidade	CPF	doenca
1	Ana	20	Florianopolis	20000200000	gripe
2	Paulo	24	Palhoca	20000220000	fratura
3	Lucia	30	Biguacu	22000200000	tendinite
4	Carlos	28	Joinville	11000110000	sarampo

Consultas

codm	codp	data	hora
1	1	2006/06/12	14:00
1	4	2006/06/13	10:00
2	1	2006/06/13	9:00
2	2	2006/06/13	11:00
2	3	2006/06/14	14:00
2	4	2006/06/14	17:00
3	1	2006/06/19	18:00
3	3	2006/06/12	10:00
3	4	2006/06/19	13:00
4	4	2006/06/20	13:00
4	4	2006/06/22	19:30

Funcionarios

codf	nome	idade	cidade	salario	CPF
1	Rita	32	Sao Jose	1200	20000100000
2	Maria	55	Palhoca	1220	30000110000
3	Caio	45	Florianopolis	1100	41000100000
4	Carlos	44	Florianopolis	1200	51000110000
5	Paula	33	Florianopolis	2500	61000111000

Exercícios (MySQL)

Realizar as seguintes atualizações no BD:

- 1) O paciente Paulo mudou-se para Ilhota
- 2) A consulta do médico 1 com o paciente 4 passou para às 12:00 horas do dia 4 de Julho de 2006
- 3) A paciente Ana fez aniversário e sua doença agora é cancer
- 4) A consulta do médico Pedro (codf = 3) com o paciente Carlos (codf = 4) passou para uma hora e meia depois
- 5) O funcionário Carlos (codf = 4) deixou a clínica
- 6) As consultas marcadas após as 19 horas foram canceladas
- 7) Os pacientes com câncer ou idade inferior a 10 anos deixaram a clínica
- 8) Os médicos que residem em Biguacu e Palhoca deixaram a clínica

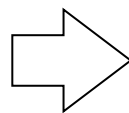
SQL – Consultas Básicas

- Consulta a dados de uma tabela

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

- Mapeamento para a álgebra relacional

```
select  $a_1, \dots, a_n$   
from  $t$   
where  $c$ 
```



```
 $\pi_{a_1, \dots, a_n} (\sigma_c(t))$ 
```


Consulta a uma Tabela

- Exemplos

Álgebra	SQL
(Pacientes)	Select * From Pacientes
$\sigma_{idade > 18}$ (Pacientes)	Select * From Pacientes Where idade > 18
$\pi_{CPF, nome}$ (Pacientes)	Select CPF, nome From Pacientes
$\pi_{CPF, nome}(\sigma_{idade > 18} \text{ (Pacientes)})$	Select CPF, nome From Pacientes Where idade > 18

Comando SELECT

- Facilidades para **projeção** de informações
 - Não há eliminação de duplicatas no **Select**
 - **tabela** \equiv 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
```

$\equiv \rho_{(nroa, grupo5)} (\pi_{nroa, capacidade/5} (Ambulatórios))$

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

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

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 SUM - Exemplo

- informar a capacidade total dos ambulatórios do primeiro andar

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

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

- 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 ambulatorios

```
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

álgebra	SQL
$relação1 \cup relação2$	<code>consultaSQL1 union consultaSQL2</code>

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

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

Exercícios

Realizar as seguintes consultas no BD:

- 1) Buscar o nome e o CPF dos médicos com menos de 40 anos ou com especialidade diferente de traumatologia
- 2) Buscar todos os dados das consultas marcadas no período da tarde após o dia 19/06/2006
- 3) Buscar o nome e a idade dos pacientes que não residem em Florianópolis
- 4) Buscar a hora das consultas marcadas antes do dia 14/06/2006 e depois do dia 20/06/2006
- 5) Buscar o nome e a idade (em meses) dos pacientes
- 6) Em quais cidades residem os funcionários?
- 7) Qual o menor e o maior salário dos funcionários da Florianópolis?
- 10) Qual o horário da última consulta marcada para o dia 13/06/2006?
- 11) Qual a média de idade dos médicos e o total de ambulatorios atendidos por eles?
- 12) Buscar o código, o nome e o salário líquido dos funcionários. O salário líquido é obtido pela diferença entre o salário cadastrado menos 20% deste mesmo salário
- 13) Buscar o nome dos funcionários que terminam com a letra “a”
- 14) Buscar o nome e CPF dos funcionários que não possuam a seqüência “00000” em seus CPFs
- 15) Buscar o nome e a especialidade dos médicos cuja segunda e a última letra de seus nomes seja a letra “o”
- 16) Buscar os códigos e nomes dos pacientes com mais de 25 anos que estão com tendinite, fratura, gripe e sarampo

SQL – DML

- Consultas envolvendo mais de uma tabela

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

- Mapeamento para a álgebra relacional

```
select a1, ..., an  
from tab1, ..., tabm  
where c
```



```
 $\pi_{a_1, \dots, a_n} (\sigma_c (tab_1 \times \dots \times tab_m))$ 
```

Exemplos

Álgebra

SQL

(Pacientes X Consultas)

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

$\pi_{\text{CPF, nome, data}} ($
 $\sigma_{\text{hora} > 12:00} (\text{Pacientes X Consultas})$
 $\wedge \text{Pacientes.codp} = \text{Consultas.codp})$

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

$\pi_{\text{m2.nome}} ($
 $\sigma_{\text{m1.nome} = \text{'Joao'} \wedge \text{m1.especialidade} =$
 $\text{m2.especialidade} ($
 $(\rho_{\text{m1}} (\text{Médicos})) \text{ X }$
 $(\rho_{\text{m2}} (\text{Médicos}))$
 $))$

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

Junção

- Sintaxe

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

- Mapeamento para a álgebra relacional

```
select  $a_1, \dots, a_n$   
from  $tab_1$  join  $tab_2$   
on  $tab_1.x > tab_2.x$   
where  $c$ 
```

$\Rightarrow \pi_{a_1, \dots, a_n} (\sigma_c (tab_1 \theta X tab_2))$
 $\theta = tab_1.x > tab_2.x$

Exemplos

Álgebra	SQL
$(\text{Pacientes } \theta \text{ X Consultas})$ $\theta = \text{Pacientes.codp} = \text{Consultas.codp}$	Select * From Pacientes join Consultas on Pacientes.codp = Consultas.codp
$\pi_{\text{nome}} (\sigma_{\text{data} = '2006/11/13'} (\text{Médicos } \theta \text{ X Consultas}))$ $\theta = \text{Médicos.codm} = \text{Consultas.codm}$	Select nome From Médicos join Consultas on Médicos.codm = Consultas.codm Where data = '2006/11/13'

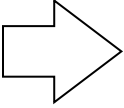
Junção Natural

- Sintaxe

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

- Maapeamento para a álgebra relacional

```
select  $a_1, \dots, a_n$   
from  $tab_1$  natural join  $tab_2$   
where  $c$ 
```



```
 $\pi_{a_1, \dots, a_n} (\sigma_c (tab_1 \bowtie tab_2))$ 
```

Exemplos

Álgebra	SQL
$(\text{Pacientes} \bowtie \text{Consultas})$	Select * From Pacientes natural join Consultas
$\pi_{\text{nome}} (\sigma_{\text{data} = '2006/11/13'} (\text{Médicos} \bowtie \text{Consultas}))$	Select nome From Médicos natural join Consultas Where data = '2006/11/13'

Junções Externas (Não Naturais)

- Sintaxe

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

- Mapeamento para a álgebra relacional

```
select  $a_1, \dots, a_n$   
from  $t_1$  left join  $t_2$   
on  $t_1.x > t_2.x$   
where  $c$ 
```

$\Rightarrow \pi_{a_1, \dots, a_n} (\sigma_c (t_1 \theta \bowtie t_2))$
 $\theta = t_1.x > t_2.x$

Exemplos

Álgebra	SQL
$(\text{Pacientes} \theta \bowtie \text{Consultas})$ $\theta = \text{Pacientes.codp} = \text{Consultas.codp}$	Select * From Pacientes left join Consultas on Pacientes.codp = Consultas.codp
$\pi_{\text{nome}} (\sigma_{\text{data} = '05/13/03'} (\text{Consultas} \theta \Join \text{Médicos}))$ $\theta = \text{Médicos.codm} = \text{Consultas.codm}$	Select nome From Médicos right join Consultas on Médicos.codm = Consultas.codm Where data = '05/13/03'

Observação: MySQL não implementa *full join*

Exercícios

Defina cada uma das seguintes buscas através de um produto, de uma junção (e de uma junção natural, quando possível). Quando necessário, utilizar junção externa:

- 1) nome e CPF dos médicos que também são pacientes do hospital
- 2) pares (código, nome) de funcionários e de médicos que residem na mesma cidade
- 3) código e nome dos pacientes com consulta marcada para horários após às 14 horas
- 4) número e andar dos ambulatórios utilizados por médicos ortopedistas
- 5) nome e CPF dos pacientes que têm consultas marcadas entre os dias 14 e 16 de junho de 2006
- 6) nome e idade dos médicos que têm consulta com a paciente Ana
- 7) código e nome dos médicos que atendem no mesmo ambulatório do médico Pedro e que possuem consultas marcadas para dia 14/06/2006
- 8) nome, CPF e idade dos pacientes que têm consultas marcadas com ortopedistas para dias anteriores ao dia 16
- 9) nome e salário dos funcionários que moram na mesma cidade do funcionário Carlos e possuem salário superior ao dele
- 10) dados de todos os ambulatórios e, para aqueles ambulatórios onde médicos dão atendimento, exibir também os seus códigos e nomes
- 11) CPF e nome de todos os médicos e, para aqueles médicos com consultas marcadas, exibir os CPFs e nomes dos seus pacientes e as datas das consultas

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)
```

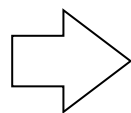
- Mapeamento para a álgebra relacional

select a_1, \dots, a_n

from t_1

where $c \text{ IN}$

(**select** x **from** t_2
 where $d > 5$)



$\pi_{a_1, \dots, a_n} (t_1 \theta X (\pi_x (\sigma_{d > 5} (t_2))))$
 $\theta = t_1.c = t_2.x$

Exemplos

Álgebra	SQL
$\pi_{\text{nome}} ($ $(\text{Médicos} \theta X$ $\theta = \text{Médicos.codm} = \text{Consultas.codm}$ $(\pi_{\text{codm}} (\sigma_{\text{data} = '06/11/13'} (\text{Consultas})))$ $))$	Select nome From Médicos Where codm in (select codm from Consultas where data = '06/11/13')
$(\pi_{\text{CPF}} (\text{Funcionários})) \text{ — } (\pi_{\text{CPF}} (\text{Pacientes}))$	Select CPF From Funcionários Where CPF not in (select CPF from Pacientes)
$(\pi_{\text{CPF}} (\text{Médicos})) \cap (\pi_{\text{CPF}} (\text{Pacientes}))$	Select CPF From Médicos Where CPF in (select CPF from Pacientes)

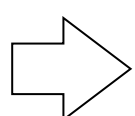
Subconsultas com ANY

- 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)
```

- Mapeamento para a álgebra relacional

```
select a1, ..., an
from t1
where c > ANY
      (select x from t2
       where d > 5)
```



$\pi_{a_1, \dots, a_n} (t_1 \theta X (\pi_x (\sigma_{d > 5} (t_2))))$
 $\theta = t_1.c > t_2.x$

Exemplos

Álgebra	SQL
$\pi_{\text{nome}} ($ $(\text{Médicos} \theta X$ $\theta = \text{Médicos.codm} = \text{Consultas.codm}$ $(\pi_{\text{codm}} (\sigma_{\text{data} = '06/11/13'}$ $(\text{Consultas})))))$	Select nome From Médicos Where codm = any (ou in) (select codm from Consultas where data = '06/11/13')
$\pi_{\text{Funcionários.idade}} ($ $((\pi_{\text{idade}} (\text{Funcionários})) \theta X$ $\theta = \text{Funcionários.idade} < f2.idade$ $(\pi_{\text{idade}} (\rho_{f2} (\text{Funcionários})))$	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)
```

- Mapeamento para o cálculo relacional

select a_1, \dots, a_n

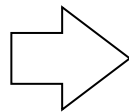
from T_1

where **EXISTS**

(select * from T_2

where $d > 5$

and $T_2.x = T_1.c$)



$\{t_1.a_1, \dots, t_1.a_n \mid t_1 \in T_1 \wedge \exists t_2 \in T_2$
 $(t_2.d > 5 \wedge t_2.x = t_1.c) \}$

Exemplos

Cálculo	SQL
$\{m.nome \mid m \in \text{Médicos} \wedge \exists c \in \text{Consultas} (c.data = '06/11/13' \wedge c.codm = m.codm)\}$	Select nome From Médicos <i>m</i> Where exists (Select * From Consultas Where data = '06/11/13' and codm = <i>m.codm</i>)
$\{f.nome \mid f \in \text{Funcionários} \wedge f.depto = 'pessoal' \wedge \neg \exists p \in \text{Pacientes} (p.CPF = f.CPF)\}$	Select f.nome From Funcionários f Where f.depto = 'pessoal' and not exists (Select * From Pacientes Where CPF = f.CPF)

Exemplo

Cálculo	SQL
$\{p.\text{nome} \mid p \in \text{Pacientes} \wedge$ $\neg \exists m \in \text{Médicos}$ $(\neg \exists c \in \text{Consultas}$ $(c.\text{codm} = m.\text{codm} \wedge$ $p.\text{codp} = c.\text{codp}))\}$	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))

Exercícios

Resolva o que se pede utilizando subconsultas IN:

- 1) nome e CPF dos médicos que também são pacientes do hospital
- 2) código e nome dos pacientes com consulta marcada para horários após às 14 horas
- 3) nome e idade dos médicos que têm consulta com a paciente Ana
- 4) número e andar dos ambulatorios onde nenhum médico dá atendimento
- 5) nome, CPF e idade dos pacientes que têm consultas marcadas sempre para dias anteriores ao dia 16

Resolva o que se pede utilizando subconsultas ANY e/ou ALL:

- 1) números e andares de todos os ambulatorios, exceto o de menor capacidade
- 2) nome e idade dos médicos que têm consulta com a paciente Ana
- 3) nome e a idade do médico mais jovem (sem usar função MIN!)
- 4) nome e CPF dos pacientes com consultas marcadas para horários anteriores a todos os horários de consultas marcadas para o dia 12 de Novembro de 2006
- 5) nome e CPF dos médicos que não atendem em ambulatorios com capacidade superior à capacidade dos ambulatorios do segundo andar

Resolva o que se pede utilizando subconsultas EXISTS:

- 1) nome e CPF dos médicos que também são pacientes do hospital
- 2) nome e idade dos médicos que têm consulta com a paciente Ana
- 3) número do ambulatorio com a maior capacidade (sem usar função MAX!)
- 4) nome e CPF dos médicos que têm consultas marcadas com todos os pacientes
- 5) nome e CPF dos médicos ortopedistas que têm consultas marcadas com todos os pacientes de Florianópolis

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
```

- Mapeamento para a álgebra relacional

```
select a1  
from (select x  
from t1 where d > 5)  
as t2 join t3  
on t3.c = t2.x
```

$\Rightarrow \pi_{a_1} (t_3 \theta X \rho_{t_2} (\pi_x (\sigma_{d > 5} (t_1))))$
 $\theta = t_3.c = t_2.x$

Exemplos

Álgebra	SQL
$\pi_{\text{Médicos.codm}, \dots, \text{nroa}, \text{hora}} ($ $(\text{Médicos} \theta X$ $\theta = \text{Médicos.codm} = \text{C.codm}$ $\rho_C (\pi_{\text{codm}, \text{hora}} (\sigma_{\text{data} = '06/11/13'}$ $(\text{Consultas})))))$	<pre>select Medicos.*, C.hora from Medicos join (select codm, hora from Consultas where data = '06/11/13') as C on Médicos.codm = C.codm</pre>
$\pi_{\text{Amb.nroa}, \text{andar}, \text{capacidade}} ($ $\rho_{\text{Amb}} (\pi_{\text{nroa}, \text{andar}} (\text{Ambulatórios}))$ θX $\theta = \text{Amb.nroa} = \text{M_ort.nroa}$ $\rho_{\text{MFlo}} (\pi_{\text{nroa}} (\sigma_{\text{cidade} = 'Fpolis'}$ $(\text{Médicos}))))$	<pre>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</pre>

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	<i>Count</i>
ortopedia	2
pediatria	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;
```

Exercícios

Buscar o que se pede utilizando subconsultas na cláusula FROM:

- 1) todos os dados das consultas marcadas para a médica Maria
- 2) código e nome dos pacientes com consulta marcada para horários após às 14 horas
- 3) nome e cidade dos pacientes que têm consultas marcadas com ortopedistas
- 4) nome e CPF dos pacientes de Florianópolis que não têm consultas com o médico João

Buscar o que se pede utilizando ORDER BY e GROUP BY:

- 1) os dados de todos os funcionários ordenados pelo salário (decrescente) e pela idade (crescente). Buscar apenas os três primeiros funcionários nesta ordem
- 2) o nome dos médicos e o número e andar do ambulatório onde eles atendem, ordenado pelo número do ambulatório
- 3) o nome do médico e o nome dos pacientes com consulta marcada, ordenado pela data e pela hora.
- 4) idades dos médicos e o total de médicos com a mesma idade
- 5) datas e o total de consultas em cada data, para horários após às 12 hs.
- 6) andares onde existem ambulatórios e a média de capacidade por andar
- 7) andares onde existem ambulatórios cuja média de capacidade no andar seja ≥ 40
- 8) nome dos médicos que possuem mais de uma consulta marcada

Realizar as seguintes atualizações:

- 1) passar para às 19hs todas as consultas marcadas para a paciente Ana
- 2) excluir os pacientes que não possuem consultas marcadas
- 3) passar para 21/11/2006 todas as consultas do médico Pedro marcadas antes do meio-dia
- 4) o ambulatório 4 foi transferido para o mesmo andar do ambulatório 1 e sua capacidade é agora o dobro da capacidade do ambulatório de maior capacidade da clínica
- 5) o funcionário Caio (codf = 3) tornou-se médico. Sua especialidade é a mesma da médica Maria (codm = 2) e ele vai atender no mesmo ambulatório dela. Inserir Caio na tabela Médicos