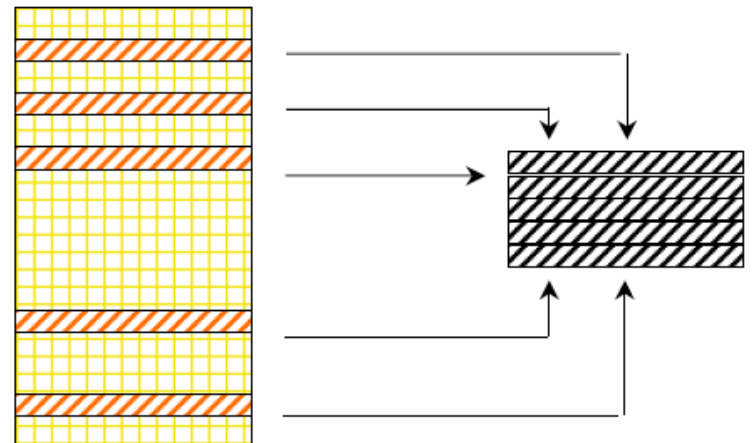


# Álgebra Relacional

# Seleção ( $\sigma$ )

- Retorna tuplas que satisfazem uma condição
- Age como um filtro que mantém somente as tuplas que satisfazem a condição
  - Ex.: selecione os funcionários com salário maior que 500
- O resultado:
  - é uma relação que contém as tuplas que satisfazem a condição
  - Possui os mesmos atributos da relação de entrada



# Seleção ( $\sigma$ )

- Sintaxe:  **$\sigma$  <condição de seleção> (<R>)**
- Sigma( **$\sigma$** ): é o símbolo que representa a seleção
- <condição de seleção> é uma expressão booleana que envolve literais e valores de atributos da relação
  - CLAUSULAS:  
<nome do atributo> <operador de comparação> <valor constante> OU  
<nome do atributo> <operador de comparação> <nome do atributo>
    - Nome do atributo: é um atributo de R;
    - Operador de comparação: =, <, <=, >, >=, <>
    - Valor constante: é um valor do domínio do atributo
  - Podem ser ligadas pelos operadores AND, OR e NOT
- <R> é o nome de uma relação ou uma *expressão da álgebra relacional* de onde as tuplas serão buscadas

# Seleção ( $\sigma$ ) - Exemplo

- Buscar os dados dos *empregados* que estão com salário menor que 2.000,00

$$\sigma_{\text{salario} < 2000} (\text{Empregado})$$

## Empregado

codEmp	Nome	Salario	idade	codDep
200	Pedro	3.000,00	45	001
201	Paulo	2.200,00	43	001
202	Maria	2.500,00	38	001
203	Ana	1.800,00	25	002

## Resultado

codEmp	Nome	Salario	idade	codDep
203	Ana	1.800,00	25	002

# Seleção ( $\sigma$ ) - Exemplo

- Buscar os dados dos empregados com salario maior que 2000 e com menos 45 anos

$\sigma_{\text{salario} > 2000 \text{ AND idade} < 45}$  (Empregado)

## Empregado

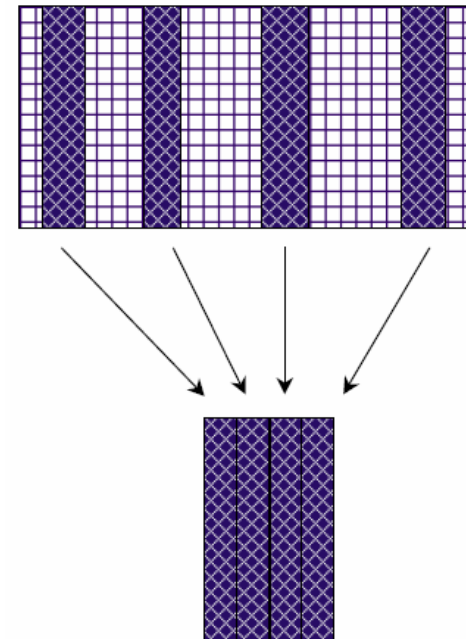
codEmp	Nome	Salario	idade	codDep
200	Pedro	3.000,00	45	001
201	Paulo	2.200,00	43	001
202	Maria	2.500,00	38	001
203	Ana	1.800,00	25	002

## Resultado

codEmp	Nome	Salario	idade	codDep
201	Paulo	2.200,00	43	001
202	Maria	2.500,00	38	001

# Projeção ( $\pi$ )

- Retorna um ou mais **atributos** de interesse
- O resultado é uma relação que contém apenas as colunas selecionadas.



\* Elimina duplicatas

# Projeção ( $\pi$ )

- Sintaxe:

**$\pi$  <lista de atributos> (<R>)**

onde:

- **<lista de atributos>** é uma lista que contém nomes de colunas de uma ou mais relações.
- **<R>** é o nome da relação ou uma expressão da álgebra relacional de onde a lista de atributos será buscada

# Projeção ( $\pi$ ) – Exemplo

- Buscar o nome e a idade de todos os empregados

**Empregado**      $\pi$  nome, idade (Empregado)

<b>codEmp</b>	<b>Nome</b>	<b>Salario</b>	<b>idade</b>	<b>codDep</b>
200	Pedro	3.000,00	45	001
201	Paulo	2.200,00	43	001
202	Maria	2.500,00	38	001
203	Ana	1.800,00	25	002

**Resultado**

<b>Nome</b>	<b>idade</b>
Pedro	45
Paulo	43
Maria	38
Ana	25



# Projeção e Seleção

- Operadores diferentes podem ser aninhados
  - Exemplo: Buscar o nome e o salario dos empregados com mais de 40 anos

$\pi_{\text{nome, salario}} (\sigma_{\text{idade} > 40} (\text{Empregado}))$

## Empregado

codEmp	Nome	Salario	idade	codDep
200	Pedro	3.000,00	45	001
201	Paulo	2.200,00	43	001
202	Maria	2.500,00	38	001
203	Ana	1.800,00	25	002

## Resultado

Nome	Salario
Pedro	3.000,00
Paulo	2.200,00

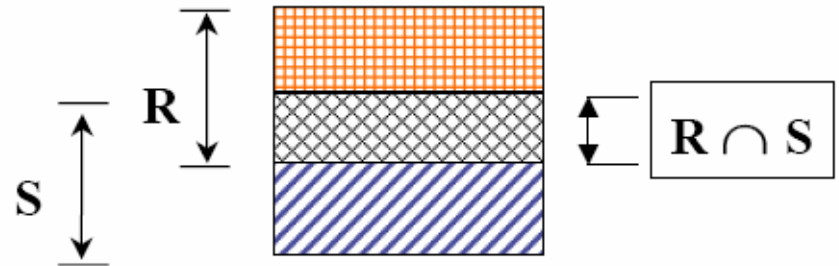
Operações

# Operações - Teoria dos Conjuntos

- A álgebra relacional utiliza 4 operadores da teoria dos conjuntos:
  - **União, Intersecção, Diferença e Produto Cartesiano**
- Todos os operadores utilizam ao menos **DUAS** relações
- As relações devem ser compatíveis:
  - possuir o mesmo número de atributos
  - o **domínio** da i-ésima coluna de uma relação deve ser idêntico ao domínio da i-ésima coluna da outra relação
- Quando os nomes dos atributos forem diferentes, adota-se a convenção de usar os nomes dos atributos da primeira relação

# Intersecção ( $\cap$ )

- Retorna uma relação com as tuplas comuns a R e S
- Notação:  $R \cap S$



R

x	y	z
1	1	1
1	2	2
2	2	3
3	1	1

S

x	y	z
1	1	1
1	2	1
3	1	1

$R \cap S$

x	y	z
1	1	1
3	1	1

# Intersecção ( $\cap$ ) - Exemplo

- buscar o nome e CPF dos funcionários de Porto Alegre que estão internados como pacientes
  - Médico (CRM, nome, idade, cidade, especialidade, #númeroA)
  - Paciente (RG, nome, idade, cidade, doença)
  - Funcionário (RG, nome, idade, cidade, salário)

$\pi$  nome, rg (Funcionario)  $\cap$   $\pi$  nome, rg ( $\sigma$  cidade = 'Porto Alegre '  
(Paciente))

# União ( $\cup$ )

- Requer que as duas relações fornecidas como argumento tenham o mesmo esquema.
- Resulta em uma nova relação, com o mesmo esquema, cujo conjunto de linhas é a união dos conjuntos de linhas das relações dadas como argumento.
- Retorna a união das tuplas de duas relações R e S
- Eliminação automática de duplicatas
- Notação:  $R \cup S$

R

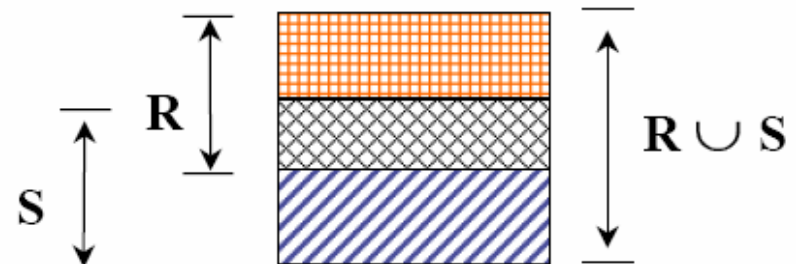
x	y	z
1	1	1
1	2	2
2	2	3
3	1	1

S

x	y	z
1	1	1
1	2	1
1	2	3

$R \cup S$

x	y	z
1	1	1
1	2	1
1	2	2
1	2	3
2	2	3
3	1	1



# União ( $\cup$ ) - Exemplo

- buscar o nome e o CPF dos médicos e dos pacientes cadastrados no hospital
  - Médico (CRM, rg, nome, idade, cidade, especialidade, #*númeroA*)
  - Paciente (RG, nome, idade, cidade, doença)

$$\pi \text{ nome, rg (Medico)} \cup \pi \text{ nome, rg (Paciente)}$$

# Diferença (-)

- Requer que as duas relações fornecidas como argumento tenham o mesmo esquema.
- Resulta em uma nova relação, com o mesmo esquema, cujo conjunto de linhas é o conjunto de linhas da primeira relação menos as linhas existentes na segunda.



# Diferença (-)

- Retorna as tuplas presentes em R e ausentes em S
- Notação:  $R - S$

**R**

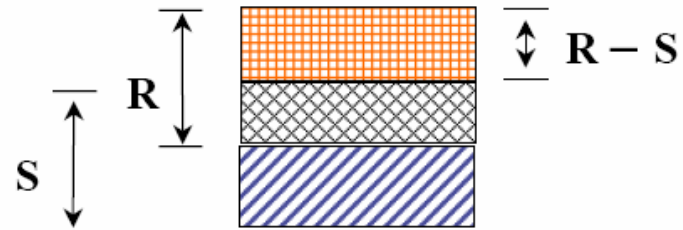
x	y	z
1	1	1
1	2	2
2	2	3
3	1	1

**S**

x	y	z
1	1	1
1	2	1
3	1	1

**R-S**

x	y	z
1	2	2
2	2	3



# Diferença (-) - Exemplo

- buscar o número dos ambulatórios onde nenhum médico dá atendimento
  - Médico (CRM, nome, idade, cidade, especialidade, #*númeroA*)
  - Ambulatorio (numeroA, nome, andar)

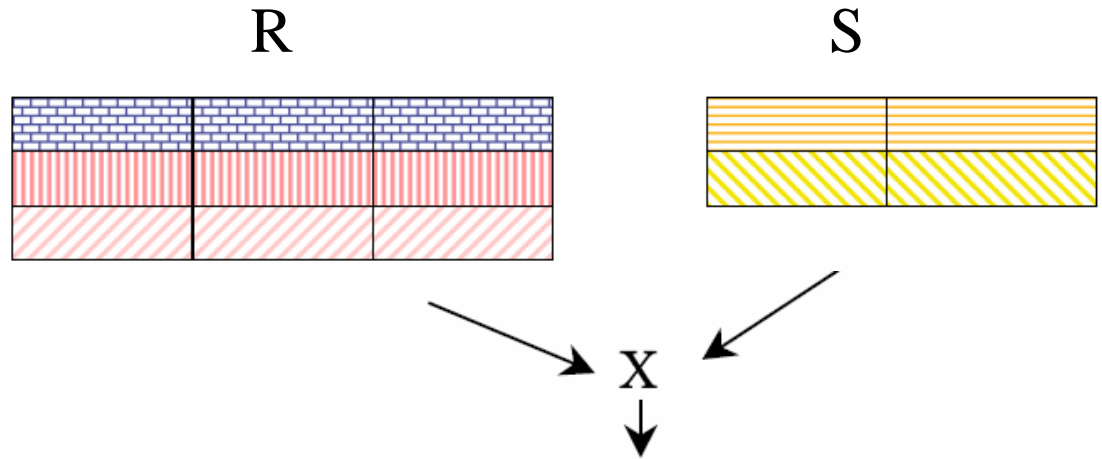
$$\pi \text{ numeroA (Ambulatorio)} \text{ } \bar{-} \text{ } \pi \text{ numeroA (Medico)}$$

# Produto Cartesiano (x)

- Retorna todas as combinações de tuplas de duas relações R e S
- O resultado é uma relação cujas tuplas são a combinação das tuplas das relações R e S, tomando-se uma tupla de R e concatenando-a com uma tupla de S
- Notação:
  - $R \times S$

# Produto Cartesiano (x)

Total de atributos do produto cartesiano =  
num. atributos de R +  
num. atributos de S



Número de tuplas do produto cartesiano = num. tuplas de R x num tuplas de R


# Produto Cartesiano (x)

- Exemplo:

R			S	
<i>x</i>	<i>y</i>	<i>z</i>	<i>w</i>	<i>y</i>
1	1	1	1	1
2	2	2	2	2
3	3	3		

<i>x</i>	$R_1.y$	<i>z</i>	<i>w</i>	$R_2.y$
1	1	1	1	1
1	1	1	2	2
2	2	2	1	1
2	2	2	2	2
3	3	3	1	1
3	3	3	2	2

# Produto Cartesiano - Exemplo

- buscar o nome dos médicos que têm consulta marcada e as datas das suas consultas
  - Médico (CRM, nome, idade, cidade, especialidade, #*númeroA*)
  - Consulta (#CRM, #RG, data, hora)

$\pi$  medico.nome, consulta.data ( $\sigma$  medico.CRM=consulta.CRM  
(Medico x Consulta))

# Produto Cartesiano - Exemplo

- buscar, para as consultas marcadas para o período da manhã (7hs-12hs), o nome do médico, o nome do paciente e a data da consulta

$\pi$  medico.nome, paciente.nome, consulta.data

$(\sigma \text{ consulta.hora} \geq 7 \text{ AND } \text{consulta.hora} \leq 12) \text{ AND}$

$\text{medico.CRM} = \text{consulta.CRM} \text{ AND } \text{consulta.RG} = \text{paciente.RG}$

$(\text{Medico x Consulta x Paciente})$

# Junção

- Retorna a combinação de tuplas de duas relações R e S que satisfazem um predicado
  - É a Seleção combinada com Produto Cartesiano
- Como esta é uma operação muito comum, foi criada para simplificar a sequência de operações necessárias para a realização de uma consulta.
- Sintaxe:
  - $\langle \text{Relação } S \rangle \bowtie \langle \text{critério} \rangle \langle \text{Relação } R \rangle$
  - onde:
    - $\langle \text{relação} \rangle$  é o nome de uma tabela ou uma expressão de álgebra relacional que resulta em uma tabela
    - $\langle \text{critério} \rangle$  é uma expressão booleana envolvendo literais e valores de atributos das duas tabelas.



# Junção - Exemplo

- buscar o número dos ambulatórios e o nome dos médicos que atendem neles

$\pi$   
ambulatorio.numeroA, medico.nome   Ambulatorio    $\bowtie$    (ambulatorio.numeroA=medico.numeroA)   Medico

Ambulatório (númeroA, andar, capacidade)

Médico (CRM, nome, idade, cidade, especialidade, #*númeroA*)

Paciente (RG, nome, idade, cidade, doença)

Consulta (#CRM, #RG, data, hora)

Funcionário (RG, nome, idade, cidade, salário)

# Junção Natural

- Junção na qual  $\bowtie$  é uma igualdade predefinida entre todos os atributos de mesmo nome presentes em duas relações R e S (atributos de junção). Estes atributos só aparecem uma vez no resultado
- Notação:

- $\langle \text{relação} \rangle R \bowtie \langle \text{relação} \rangle S$

Onde:

$\langle \text{relação} \rangle$  é o nome de uma tabela ou uma expressão de álgebra relacional que resulta em uma tabela

# Junção Natural - Exemplo

Exemplo: buscar o número e nome do ambulatório onde o médico atende

$\pi$  medico.numeroA, ambulatorio.nome    Medico  $\bowtie$  Ambulatorio

Ambulatório (numeroA, andar, capacidade, nome)

Médico (CRM, nome, idade, cidade, especialidade, #numeroA)

Paciente (RG, nome, idade, cidade, doença)

Consulta (#CRM, #RG, data, hora)

Funcionário (RG, nome, idade, cidade, salário)

# Junções Externas (outer joins)

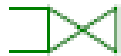
- Junção na qual as tuplas de uma ou ambas as relações que **não são combinadas** são mesmo assim preservadas no resultado
- Tipos:
  - junção externa à esquerda (left [outer] join)
  - junção externa à direita (right [outer] join)
  - junção externa completa (full [outer] join)

# Junção Externa à esquerda

- tuplas da relação à esquerda são preservadas

- Notação:

• R  $\bowtie$  S



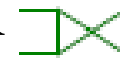
R

$x$	$y$	$z$
1	1	1
2	1	2
3	3	3
5	5	5

S

$x$	$a$	$b$
1	1	1
2	1	2
4	4	4

R



S

$x$	$y$	$z$	$a$	$b$
1	1	1	1	1
2	1	2	1	2
3	3	3		
5	5	5		

# Junção Externa à esquerda

- Exemplo: buscar os dados de todos os médicos, e para aqueles que têm consultas marcadas, mostrar os dados de suas consultas

Médico  Consulta  
(medico.CRM=consulta.CRM)

Ambulatório (númeroA, andar, capacidade)

Médico (CRM, nome, idade, cidade, especialidade, #*númeroA*)

Paciente (RG, nome, idade, cidade, doença)

Consulta (#CRM, #RG, data, hora)

Funcionário (RG, nome, idade, cidade, salário)

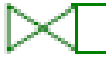
# Junção externa à direita

- tuplas da relação à direita são preservadas

- Notação



R			S		
<i>x</i>	<i>y</i>	<i>z</i>	<i>x</i>	<i>a</i>	<i>b</i>
1	1	1	1	1	1
2	1	2	2	1	2
3	3	3	4	4	4
5	5	5			

R  S

<i>x</i>	<i>y</i>	<i>z</i>	<i>a</i>	<i>b</i>
1	1	1	1	1
2	1	2	1	2
4			4	4

# Divisão

- Útil para responder questões como: “encontre os pacientes que consultaram com **TODOS** os médicos”
- Sintaxe:  $R : S$
- Os nomes das colunas de  $S$  devem estar contidos em  $R$
- A relação resultante tem como nomes das colunas aquelas que aparecem em  $R$  mas não aparecem em  $S$
- Para que uma linha apareça no resultado, é necessário que a sua concatenação com cada linha de  $R$  apareça também em  $S$ .



# Divisão

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

S

A
a1
a2
a3

$T \leftarrow R \div S$

T

B
b1
b4

R-consultas  
S-Medicos

# Divisão

- buscar o RG dos pacientes que têm consultas marcadas com todos os médicos

$\pi_{RG, crm}(\text{Consulta}) : \pi_{crm}(\text{Médico})$

Ambulatório (númeroA, andar, capacidade)

Médico (CRM, nome, idade, cidade, especialidade, #*númeroA*)

Paciente (RG, nome, idade, cidade, doença)

Consulta (#CRM, #RG, data, hora)

Funcionário (RG, nome, idade, cidade, salário)

Ambulatório (númeroA, andar, capacidade)

Médico (CRM, rg, nome, idade, cidade, especialidade, #*númeroA*)

Paciente (RG, nome, idade, cidade, doença)

Consulta (#CRM, #RG, data, hora)

Funcionário (RG, nome, idade, cidade, salário)

- 1) buscar o nome dos médicos que atenderam todos os pacientes
- 2) buscar o andar dos ambulatórios nos quais todos os médicos ortopedistas dão atendimento
- 3) buscar os dados de todos os médicos e, para aqueles que têm consultas marcadas, mostrar o nome do médico e o RG do paciente
- 4) buscar os números de todos os ambulatórios e, para aqueles ambulatórios nos quais médicos dão atendimento, exibir o CRM e o nome dos médicos associados
- 5) mostrar em uma relação o RG e nome de todos os pacientes e de todos os médicos, apresentando estes dados de forma relacionada para aqueles que possuem consultas marcadas

SQL (Structure Query Language)

# SQL

- Linguagem para:
  - Definição de dados: criação das estruturas
    - Data Definition Language (DDL)
  - Manipulação de dados: atualização e consultas
    - Data Manipulation Language (DML)

# Manipulação de Dados

- Define operações de manipulação de dados
  - C (INSERT) – Create – Criar
  - R (SELECT) – Read – Ler
  - U (UPDATE) – Update – Atualizar
  - D (DELETE) – Delete – Apagar
- Instruções declarativas
  - manipulação de conjuntos
  - especifica-se o *que fazer* e não *como fazer*

# Inserções, Alterações e Exclusões

# SQL – Insert

- Inserção de dados

```
INSERT INTO nome_tabela [(lista_atributos)]  
VALUES (lista_valores_atributos)  
        [, (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, 'Betim');
```



# SQL – Inserção a partir de outra tabela

- Inserção de dados

**Permite inserir em uma tabela a partir de outra tabela  
A nova tabela terá os mesmos atributos, com os mesmos domínios**

- Exemplos

```
INSERT into cliente as  
SELECT * from funcionario
```

# SQL – Update

- Alteração de dados

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

- Exemplos

```
UPDATE Medico  
SET cidade = 'Belo Horizonte'
```

```
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 < > 'Contagem'
```

# Exercícios

- Inserir 3 médicos na tabela de médicos
- Cadastrar 4 ambulatórios, com numeroA sendo 1,2,3 e 4
- Cadastrar 2 pacientes
- Cadastrar 3 consultas
- Alterar o numero do ambulatório de todos os médicos para 3
- Alterar o nome do paciente 1 para Pedro da Silva

Consultas: SELECT

# Estrutura Básica

- Uma consulta em SQL tem a seguinte forma:

**select**  $A_1, A_2, \dots, A_n$   
**from**  $r_1, r_2, \dots, r_m$   
**where**  $P$

- $A_i$  representa um atributo
  - $R_i$  representa uma tabela
  - $P$  é um predicado
- 
- Esta consulta é equivalente a uma expressão da Álgebra Relacional
- 
- O resultado de uma consulta SQL é sempre uma tabela

$$\Pi_{A_1, A_2, \dots, A_n} (\sigma_P(r_1 \times r_2 \times \dots \times r_m))$$

## Estrutura Básica:

**SELECT** lista de atributos desejados

**FROM** uma ou mais tabelas

**WHERE** com restrições sobre atributos

- Exemplo: encontre o nome e o salário dos funcionarios da relação *funcionário*

**SELECT** *nome, salario*

**FROM** *funcionario*

- Equivalente a operação de PROJEÇÃO na Álgebra

$\Pi_{nome, salario} (funcionario)$

# Distinct

- O SQL permite duplicatas em relações e resultados em consultas
- Para eliminar duplatas, usa-se a cláusula DISTINCT depois do SELECT

Exemplo: **SELECT** **distinct** *nome*  
**FROM** *funcionario*



## A cláusula \*

- O asterisco na cláusula SELECT denota TODOS OS ATRIBUTOS

```
SELECT *  
FROM funcionario
```

- Expressões aritméticas podem ser usadas na cláusula SELECT  
+, −, \*, /

- Exemplo: 

```
SELECT nome, salario + 200  
FROM funcionario
```

# A cláusula FROM

- Equivale a operação de Produto Cartesiano da Álgebra
- Lista as relações envolvidas na consulta
- Exemplo: `SELECT *`  
`FROM funcionario, departamento`

# A cláusula FROM

- Quando mais de uma tabela é utilizada é necessário dar um apelido para elas que deve ser utilizado para diferenciar atributos iguais
- Exemplo: 

```
SELECT f.*  
FROM funcionario f, departamento d  
WHERE f.codDepto = d.codDepto
```

# A cláusula WHERE

- A cláusula **where** especifica as condições que o resultado precisa satisfazer
- Corresponde ao predicado de seleção da álgebra

- Exemplo: **SELECT** nome, salario  
**FROM** *funcionario*  
**WHERE** *salario > 2000*

- operadores AND, OR e NOT podem ser usados
- Exemplo: **SELECT** nome, salario  
**FROM** *funcionario*  
**WHERE** *salario > 2000 AND idade < 30*

# Renomeando atributos

- Renomeação de atributos

*old-name as new-name*

- Exemplo: **SELECT** *nome as nomeCliente, (salario+200) as comissao*  
**FROM** *funcionario*

# Operações com Strings

- O SQL permite comparar strings com o operador *like*
- Pode ser combinado com outros caracteres
  - % compara substrings
- Exemplo 1: encontre o nome dos funcionarios cujos nomes iniciam com “Pedro”

```
select nome  
from funcionario  
where nome like 'Pedro%'
```

- Exemplo 2: encontre o nome dos funcionarios cujos nomes contém “Pedro” no nome

```
select nome  
from funcionario  
where nome like '%Pedro%'
```

# Operações de Conjunto

- Envolvem ao menos 2 tabelas
- Interseção e União: elimina automaticamente repetições
  - Relações precisam ser compatíveis (mesmo número de atributos)
  - Union ALL e intersects ALL preserva duplicatas

- Encontre os clientes que tenham empréstimos e contas

**(select nome from conta) intersect (select nome from emprestimo)**

**(select nome from conta) union (select nome from emprestimo)**

# Ordenando tuplas com *Order By*

- Exemplo: Liste em ordem alfabética os funcionarios que trabalham no departamento financeiro

```
select distinct funcionario.nome  
from funcionario, departamento  
where funcionario.codDepto=departamento.codDepto AND  
       departamento.nome='financeiro'  
order by funcionario.nome
```

- Order by pode ser em ordem descendente
  - Exemplo: **order by** *nome desc*



# Funções de Agregação

- Operam sobre múltiplos valores de uma coluna da tabela e retornam um valor

**avg:** média

**min:** valor mínimo

**max:** valor máximo

**sum:** soma de valores

**count:** número de valores

# Funções de Agregação

- Exemplos:

- Encontre o número de tuplas da relação CLIENTE

```
select count(*)  
FROM cliente
```

- Encontre a soma dos salarios dos funcionarios

```
select SUM(salario)  
FROM funcionario
```

# Funções de Agregação e Group By

- Encontre o total de funcionarios de cada departamento

```
select d.nome, count(f.*) as numeroFuncionarios
```

```
FROM funcionario f, departamento d
```

```
WHERE f.codDeppto=d.codDeppto
```

```
GROUP BY d.nome
```

# Funções de Agregação e Having

- A função HAVING é utilizada para aplicar condições sobre **grupos** e não sobre uma única tupla
- Exemplo: Quais são os departamentos onde a soma dos salários dos funcionários ultrapassa 50.000

```
select d.nome, sum(f.salario)  
  
  from funcionario f, departamento d  
  
 where f.codDepto=d.codDepto  
  
      group by d.nome  
      having (salario) > 50.000
```

# Consultas Aninhadas

- Uma **subconsulta select-from-where** está aninhada dentro de outra consulta
- Exemplo: Selecione os clientes que são funcionários

```
select nomeCliente  
from cliente  
where nomeCliente in (select nomeFuncionario  
                       from funcionario)
```

# Valores nulos

- Consulta sobre valores inexistentes
- Exemplo: Encontre os funcionarios que não possuem carteira de habilitação
  - **select** *nome*  
**from** *funcionario*  
**where** *carteiraHabilitacao* **is null**
- OBS: cuidado que valores nulos em operações matemáticas podem dar problemas

# SQL e Álgebra

Álgebra	SQL
$\pi_{\text{nome}} ($ $(\text{Médicos} \theta X$ $\theta = \text{Médicos.codm} = \text{Consultas.codm}$ $(\pi_{\text{codm}} (\sigma_{\text{data} = '16/10/18'} (\text{Consultas}))) ) )$	Select nome From Médicos Where codm in (select codm from Consultas where data = '16/10/18')
$(\pi_{\text{CPF}} (\text{Funcionários})) \text{ — } (\pi_{\text{CPF}} (\text{Pacientes}))$	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)

# Exercícios

## Dado o Esquema Relacional:

- Ambulatório (númeroA, andar, capacidade)
- Médico (CRM, nome, idade, cidade, especialidade, #númeroA)
- Paciente (RG, nome, idade, cidade, doença)
- Consulta (#CRM, #RG, data, hora)
- Funcionário (RG, nome, idade, cidade, salário)

## Escrever a expressão em algebra relacional e em linguagem SQL:

- 1) buscar os dados dos pacientes que estão com dengue
- 2) buscar os dados dos médicos cardiologistas com mais de 44 anos
- 3) buscar os dados das consultas, exceto aquelas marcadas para os médicos com CRM 4656 e 1879
- 4) buscar os dados dos ambulatórios do quarto andar que ou tenham capacidade igual a 50 ou tenham número superior a 10



# Exercícios

- 5) buscar o nome e a especialidade de todos os médicos
- 6) buscar os números dos ambulatórios, exceto aqueles do segundo e quarto andares, que suportam mais de 50 pacientes
- 7) buscar o nome dos médicos que têm consulta marcada e as datas das suas consultas
- 8) buscar o número e a capacidade dos ambulatórios do quinto andar e o nome dos médicos que atendem neles
- 9) buscar o nome dos médicos e o nome dos seus pacientes com consulta marcada, assim como a data destas consultas
- 10) buscar os nomes dos médicos ortopedistas com consultas marcadas para o período da manhã (7hs-12hs) do dia 19/06/18
- 11) buscar os nomes dos pacientes, com consultas marcadas para os médicos João Carlos Santos ou Maria Souza, que estão com pneumonia

# Exercícios

- 12) buscar os nomes dos médicos e pacientes cadastrados no hospital
- 13) buscar os nomes e idade dos médicos, pacientes e funcionários que residem em Ribeirão das Neves
- 14) buscar os nomes e RGs dos funcionários que recebem salários abaixo de R\$ 3000,00 e que não estão internados como pacientes
- 15) buscar os números dos ambulatórios onde nenhum médico dá atendimento
- 16) buscar os nomes e RGs dos funcionários que estão internados como pacientes