

2.3 Linguagens relacionais (continuação)

SQL – Structured Query Language

1) "Query block"

```
Select <lista de atributos>  
From <lista de relações>  
Where <expressão lógica>
```

- Um "query block" permite a implementação das operações de selecção, projecção e junção da álgebra relacional.
- Um query não especifica a ordem pela qual as operações são executadas.

2) Considere o esquema relacional:

```
Departamento(DepNum, Nome, Local)  
Empregado(EmpNum, Nome, Categoria, Salario, DepNum)  
Projecto(ProjNum, Designacao, Fundos)  
Atribuicao(EmpNum,ProjNum, Funcao)
```

3) Projecção

{Operação que permite seleccionar tuplos de uma tabela. }

- *O query seguinte constrói uma tabela de números de departamento e categorias de empregados:*

```
Select DepNum, Categoria  
From Empregado
```

Nota: Não elimina "linhas" repetidas

```
Select Distinct DepNum, Categoria  
From Empregado
```

Nota: Elimina linhas repetidas (pode consumir muito tempo)

```
{  $\Pi$  <DepNum, Categoria> (Empregado) }
```

- *É possível **ordenar** a tabela resultado de um query block:*

```
Select Nome, DepNum  
From Empregado  
Order By Nome
```

Ordenação por ordem crescente/decrescente:

```
Select Nome, DepNum  
From Empregado  
Order By Nome ASC, DepNum DESC
```

4) Restrição

{Operação que permite seleccionar tuplos de uma tabela que satisfazem uma dada condição. }

- *Seleccção de todos os empregados que são programadores:*

```
{  $\sigma$  < Categoria="Programador" > (Empregado) }
```

```
Select *  
From empregado  
Where Categoria = "Programador"
```

Nota: Select * → Selecciona todos os atributos.

- **Seleccção, projecção e ordenação**

i) *Nomes dos empregados que são programadores:*

```
Select Nome  
From Empregado  
Where Categoria = "Programador"  
Order By Nome
```

ii) *Nomes dos empregados que são programadores e têm salário superior a 2000€.*

{ Π <Nome> (σ < Categoria="Programador" and Salário > 2000> (Empregado)) }

```
Select Nome  
From empregado  
Where Categoria = "Programador"  
And Salario > 2000
```

iii) *Empregados que trabalham no departamento 7 ou 9.*

{ σ < DepNum = 7 or DepNum= 9 > (Empregado) }

```
Select *  
From empregado  
Where DepNum = 7 or DepNum = 9
```

Equivalente a:

```
Select *  
From empregado  
Where DepNum In (7, 9)
```

Operadores:

=, <>, >, >=, <, <=
And, Or, Not
In
Contains (não é standard)

Exercício:

Qual é o resultado do query seguinte ?

Select Nome
From Empregado

Where DepNum In (Select DepNum
From Departamento
Where Local = "Lisboa")

R: Nomes dos empregados que pertencem a departamentos localizados em Lisboa.

5) Junção

- *Obter uma listagem com o nome dos empregados e a localização dos respectivos departamentos.*

{ Π <Empregado.Nome, Departamento.Local >
(Empregado \bowtie <DepNum=DepNum > Departamento) }

O símbolo " \bowtie " representa \bowtie

Select Empregado.Nome, Departamento.Local
From Empregado, Departamento
Where Empregado.DepNum = Departamento.DepNum

Exercício (ver página 62):

- Qual dos queries seguintes permite fornecer: "Nomes dos programadores e respectivos departamentos se estes estão localizados em Lisboa"?

(1) Select E.Nome, D.Nome
 From Empregado E, Departamento D
 Where E.Categoria = "Programador"
 And D.Local = "Lisboa"

Ou

(2) Select E.Nome, D.Nome
 From Empregado E, Departamento D
 Where E.Categoria = "Programador"
 And D.Local = "Lisboa"
 And E.DepNum = D.DepNum

R: O segundo query. (Porquê ?)

Supondo,

<u>EmpDep</u>	Nome	Categoria	Salário	DepNum
1	E1	Programador	1000	5
2	E2	Programador	1000	6
3	E3	Analista	2000	7

<u>DepNum</u>	Nome	Local
5	D1	Lisboa
6	D2	Porto
7	D3	Lisboa

O resultado do query (1) é:

<u>E.Nome</u>	D.nome
E1	Lisboa
E1	Lisboa
E2 !?	Lisboa
E2 !?	Lisboa

O resultado do query (2) é:

<u>E.Nome</u>	D.nome
E1	Lisboa

6) Produto Cartesiano

Select *
From Empregado, Departamento

7) União, Intersecção e Diferença

União → Union

Intersecção → Intersect

Diferença → Minus (ou Except)

Os operandos têm que ser compatíveis:

- . têm que ter o mesmo grau, i.e., o mesmo número de colunas;
- . colunas correspondentes têm de ter o mesmo domínio.

- *Números dos departamentos que não têm empregados:*

{ $\Pi_{\langle \text{DepNum} \rangle}(\text{Departamento}) - \Pi_{\langle \text{DepNum} \rangle}(\text{Empregado})$ }

Select DepNum
From Departamento

Minus

Select DepNum
From Empregado

- *Números dos programadores que trabalham num projecto:*

$$\{ \Pi_{\langle \text{EmpNum} \rangle} (\sigma_{\langle \text{Categoria} = \text{"Programador"} \rangle} (\text{Empregado})) \cap \Pi_{\langle \text{EmpNum} \rangle} (\text{Atribuicao}) \}$$

```
Select EmpNum
From Empregado
Where Categoria = "Programador"
```

Intersect

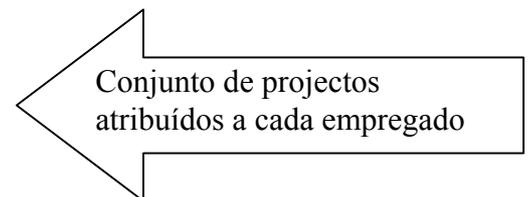
```
Select EmpNum
From Atribuicao
```

8) Divisão

- *Obter uma tabela com os números de empregado, atribuídos a todos os projectos com fundos superiores a um milhão de euros.*

$$\{ (\Pi_{\langle \text{ProjNum} \rangle} (\text{Atribuicao})) \div (\Pi_{\langle \text{ProjNum} \rangle} (\sigma_{\langle \text{Fundos} > 1000000 \rangle} (\text{Projecto}))) \}$$

```
Select Distinct EmpNum
From Atribuicao X
Where ( Select ProjNum
        From Atribuicao
        Where EmpNum = X.EmpNum)
```



Contains

```
(Select ProjNum
From Projecto
Where Fundos > 1000000)
```

Notas:

- X é uma variável de contexto cujo âmbito é a relação atribuição.
- R1 Contains R2 é verdade se o conjunto de tuplos de R2 é um subconjunto de R1.

Exercício:

Considere a base de dados exemplo (pág. 32)

- *Quais os números das obras que receberam fornecimentos de ambos os materiais cimento e areia?*

Select Unique N_Obra

From Fornecimento F

Where (Select N_Material

From Fornecimento

Where N_Obra= F.N_Obra)

Contains

(Select N_Material

From Obra

Where Nome_Material = "Areia" Or Nome_Material ="Cimento")

Em álgebra relacional?

9) Funções Standard

AVG → Valor médio
SUM → Soma de valores
COUNT
MAX
MIN

- *Qual o salário médio dos programadores?*

```
Select AVG(Salario)
From Empregado
Where Categoria = "Programador"
```

- *Número de funções diferentes que o empregado 128 executa em projectos:*

```
Select COUNT(Distinct Funcao)
From Atribuicao
Where EmpNum = 128
```

Nota: Count(*) → Número de tuplos que satisfaz a clausula Where.

Actualizações:

A SQL não é só uma linguagem de query.

É possível também inserir, eliminar ou modificar tuplos.

10) Inserção

- *Inserir o empregado 843 com o nome José e a categoria Programador:*

Insert Into Empregado (EmpNum, Nome, Categoria)

Values (843,'José','Programador')

- . Os atributos não especificados assumem o valor Null.
- . Se são especificados valores para todos os atributos não é necessário referir os seus nomes

Supondo que além das relações definidas em (2) existe também a relação:

Candidatos (EmpNum, Nome, Categoria, Salário, DepNum)

O comando

Insert into Empregado

Select EmpNum, Nome, Categoria, Salário*1.1 , DepNum

From Candidatos

Where Categoria In (“Programador” , “Analista”)

insere na relação empregado tuplos seleccionados da relação Candidatos.

11)Eliminação

- *Eliminação do tuplo do empregado 843:*

Delete From Empregado

Where EmpNum = 843

- *Eliminar todos os empregados cujo departamento está localizado em "Lisboa":*

```
Delete From Empregado
Where DepNum In( Select DepNum
From Departamento
Where Local = "Lisboa")
```

12) Actualização

- *Aumentar o salário (em 40%) do empregado 843:*

```
Update Empregado
Set Salario = Salario * 1.4
Where EmpNum = 843
```

- *Aumentar o salário (em 20%) aos empregados do projecto 10 :*

```
Update Empregado
Set Salario = Salario * 1.2
Where EmpNum In ( Select EmpNum
From Atribuicao
Where ProjNum = 10)
```