Instituto Superior de Engenharia do Porto

Structured Query Language

Data Maniulation Language (DML) - SELECT

DEE
Departamento de Engenharia Electrotécnica

Licenciatura em Engenharia de Telecomunicações e Informática

SQL - SELECT

- O SELECT permite extrair informação de uma Base de Dados ou seja, efetuar consultas à Base de Dados.
- O resultado de uma consulta é uma relação.
- Na sua forma mais básica é preciso apenas indicar a cláusula SELECT e FROM, sendo as restantes opcionais.

```
SELECT [DISTINCT] < lista de atributos>
FROM < lista de tabelas>
[ WHERE <critério> ]
[GROUP BY <grupo_expressão>
[HAVING <critério>]]
[ORDER BY <order_expressão> ASC | DESC]
```

SQL - SELECT

- SQL permite duplicados nas relações, bem como nos resultados das consultas.
- Para forçar a eliminação de duplicados, deve-se inserir a declaração **DISTINCT** depois do select.

```
SELECT DISTINCT < lista de atributos>
  .....
```

- A cláusula SELECT pode conter expressões aritméticas envolvendo os operadores +, - , * , / e operações em constantes ou atributos dos registos.
- A cláusula SELECT também permite renomear os nomes da tabelas e os nomes dos atributos

SQL - SELECT

DEE Departamento de Engenharia Electrotécnica

- **Exemplos**

veiculo (matricula, marca modelo, kms, preço)

matricula	marca	modelo	kms	preço
23-12-QA	OPEL	ASTRA	5500	18900
12-34- QW	RENAULT	CLIO	34000	23000
23-43-TR	RENAULT	MEGANE	1000	30000
78-56-XA	OPEL	ASTRA	12000	12700

SELECT * FROM veiculo

matricula	marca	modelo	kms	preço
23-12-QA	OPEL	ASTRA	5500	18900
12-34- QW	RENAULT	CLIO	34000	23000
23-43-TR	RENAULT	MEGANE	1000	30000
78-56-XA	OPEL	ASTRA	12000	12700

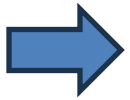
SQL - SELECT

matricula	marca	modelo	kms	preço
23-12-QA	OPEL	ASTRA	5500	18900
12-34- QW	RENAULT	CLIO	34000	23000
23-43-TR	RENAULT	MEGANE	1000	30000
78-56-XA	OPEL	ASTRA	12000	12700

- **Exemplos**

SELECT marca, modelo FROM veiculo

marca	modelo
OPEL	ASTRA
RENAULT	CLIO
RENAULT	MEGANE
OPEL	ASTRA



SELECT DISTINCT marca, modelo FROM veiculo

marca	modelo
OPEL	ASTRA
RENAULT	CLIO
RENAULT	MEGANE

SELECT matricula, marca, preço*1.20 FROM veiculo

matricula	marca	preço*1.20
23-12-QA	OPEL	22680
12-34- QW	RENAULT	27600
23-43-TR	RENAULT	36000
78-56-XA	OPEL	15240

SQL – SELECT – RENOMEAR

matricula	marca	modelo	kms	preço
23-12-QA	OPEL	ASTRA	5500	18900
12-34- QW	RENAULT	CLIO	34000	23000
23-43-TR	RENAULT	MEGANE	1000	30000
78-56-XA	OPEL	ASTRA	12000	12700

- **Exemplos**

```
SELECT matricula, marca, preço*1.20 AS novoPreço
FROM veiculo
```

matricula	marca	novoPreço
23-12-QA	OPEL	22680
12-34- QW	RENAULT	27600
23-43-TR	RENAULT	36000
78-56-XA	OPEL	15240

```
SELECT matricula, Kms AS Quilometros
FROM veiculo
```

matricula	Quilometros
23-12-QA	5500
12-34- QW	34000
23-43-TR	1000
78-56-XA	12000

SQL – SELECT - WHERE

- A cláusula **WHERE** corresponde ao predicado de seleção da álgebra relacional.
- O comando **WHERE** permite selecionar um sub-conjunto de registos de uma relação que satisfazem uma determinada condição sobre alguns atributos.
- A condição pode conter operadores de comparação (<, >, <=, <>, ...) e pode ser composta usando **AND**, **OR** e **NOT**.
- Permite também o uso de **IS NULL** ou **IS NOT NULL**

```
SELECT [DISTINCT] lista-atributos
FROM lista-tabelas
WHERE condições
```

SQL – SELECT - WHERE

matricula	marca	modelo	kms	preço
23-12-QA	OPEL	ASTRA	5500	18900
12-34- QW	RENAULT	CLIO	34000	23000
23-43-TR	RENAULT	MEGANE	1000	30000
78-56-XA	OPEL	ASTRA	12000	12700

- **Exemplos**

SELECT * FROM veiculo WHERE marca =“OPEL” OR KMS >12000;

matricula	marca	modelo	kms	preço
23-12-QA	OPEL	ASTRA	5500	18900
12-34- QW	RENAULT	CLIO	34000	23000
78-56-XA	OPEL	ASTRA	12000	12700

SELECT * FROM VEICULO WHERE marca IS NULL;

matricula	marca	modelo	kms	preço

SELECT * FROM veiculo WHERE preço/kms > 2;

matricula	marca	modelo	kms	preço
23-12-QA	OPEL	ASTRA	5500	18900

SQL – SELECT - WHERE

- Permite usar o operador **LIKE** em operações com strings.
- O operador LIKE é usado para pesquisas não exatas.
- Faz uso de dois caracteres especiais:

  _ (substitui 1 caracter)

  % (substitui qualquer sequência de caracteres)

Veiculo

matricula	marca	modelo	kms	preço
23-12-QA	OPEL	ASTRA	5500	18900
12-34- QW	RENAULT	CLIO	34000	23000
23-43-TR	RENAULT	MEGANE	1000	30000
78-56-XA	OPEL	ASTRA	12000	12700

Select * From veiculo where **marca like '__NA%'**

matricula	marca	preço
12-34- QW	RENAULT	23000
23-43-TR	RENAULT	30000

SQL – SELECT – ORDER BY

- A resposta a uma pergunta pode ser ordenada segundo uma ou mais colunas.
- Para proceder à ordenação usa-se o comando **ORDER BY** seguido das colunas que queremos ordenar.
- Por omissão as colunas são ordenadas ascendentemente.
- Podemos mudar a direção da ordenação usando as palavras **ASC** e **DESC**.

```
SELECT [DISTINCT] lista-atributos
FROM lista-tabelas
WHERE condições
GROUP BY lista-atributos-do-grupo
HAVING condições-do-grupo
ORDER BY <order_expressão> ASC | DESC]
```

SQL – SELECT – ORDER BY

matricula	marca	modelo	kms	preço
23-12-QA	OPEL	ASTRA	5500	18900
12-34- QW	RENAULT	CLIO	34000	23000
23-43-TR	RENAULT	MEGANE	1000	30000
78-56-XA	OPEL	ASTRA	12000	12700

- **Exemplos**

```
SELECT marca, kms
FROM VEICULO
WHERE   marca="OPEL"
ORDER BY  kms DESC ;
```

marca	kms
OPEL	12000
OPEL	5500

```
SELECT marca, kms FROM VEICULO
ORDER BY  marca  DESC, kms ;
```

marca	kms
RENAULT	34000
OPEL	5500
OPEL	12000

SQL – SELECT – GROUP BY AND HAVING

- O resultado da cláusula GROUP BY é uma tabela em que as linhas com valores iguais para a lista-atributos-do-grupo são agrupadas.
- A lista-atributos da cláusula SELECT consiste

  (1) numa lista de nomes e

  (2) uma lista de termos de agrupadores.
- Todas as colunas da cláusula SELECT têm de estar incluídas na lista-atributos-do-grupo.

```
SELECT [DISTINCT] lista-atributos
FROM lista-tabelas
WHERE condições
GROUP BY lista-atributos-do-grupo
HAVING condições-do-grupo
```

SQL – SELECT – GROUP BY AND HAVING

matricula	marca	modelo	kms	preço
23-12-QA	OPEL	ASTRA	5500	18900
12-34- QW	RENAULT	CLIO	34000	23000
23-43-TR	RENAULT	MEGANE	1000	30000
78-56-XA	OPEL	ASTRA	12000	12700

- A cláusula HAVING serve para filtrar os valores dos grupos.
- Elimina do resultado os grupos que não satisfazem a condição.
- As condições na cláusula HAVING são aplicadas após a formação dos grupos.

```
SELECT marca, SUM(Kms)
 FROM VEICULO
GROUP BY marca
```

marca	Sum(kms)
OPEL	17500
RENAULT	35000

```
SELECT marca, SUM(Kms)
 FROM VEICULO
GROUP BY marca
HAVING SUM(kms) < 35000
```

marca	Sum(kms)
OPEL	17500

SQL – SELECT – FUNÇÕES DE AGREGAÇÃO

- Estas funções operam nos multi-conjuntos de valores de uma coluna de uma relação e retornam um valor:
  - COUNT ([DISTINCT] A): O número de valores (únicos) na coluna A
  - SUM ([DISTINCT] A): A soma de todos os valores (únicos) de A
  - AVG ([DISTINCT] A): A média de todos os valores (únicos) de A
  - MAX (A). O valor máximo existente na coluna A
  - MIN (A). O valor mínimo existente na coluna A
- **Estas funções só podem ser usadas numa cláusula SELECT ou numa cláusula HAVING**
- **Não é possível usar diretamente os operadores de agregação na cláusula WHERE**

SQL – SELECT – FUNÇÕES DE AGREGAÇÃO

- Se um SELECT tem uma operação de agregação, então apenas pode conter colunas com operadores de agregação, excepto se tiver uma cláusula GROUP BY

```
SELECT marca, Avg(Kms)
 FROM VEICULO
```

```
SELECT Avg(Kms)
FROM VEICULO
```

```
SELECT marca, Avg(Kms)
 FROM VEICULO
GROUP BY marca
```

```
SELECT marca, Avg(Kms)
 FROM VEICULO
GROUP BY marca
 HAVING Avg(kms) <5000
```

SQL – SELECT – RESUMO

```
SELECT localidade, Count(*) Total_empregados
FROM Empregados
WHERE salario > 3000
GROUP BY localidade
HAVING Count(*) >1
ORDER BY localidade DESC
```

LOCALIDADE	TOTAL_EMPREGADOS
Lisboa	2

Tabela Empregados

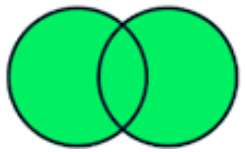
ID_EMPREGADO	NOME	ENDERECO	LOCALIDADE	SALARIO
1	Nico Seixal	Avenida da Glória 1000	Lisboa	6200
2	Ana Rita	Avenida Da Glória 1200	Lisboa	5980
3	Joana Gonçalves	Rua Antunes Almeida 114	Porto	6200
4	Paula Silva	Rua Pereira 186 1º DTO	Porto	1520
5	António Castro	Rua Direita 25 2º ESQ	Porto	2100
6	Maria Santos	Rua da Liberdade 150	Braga	1520
7	Francisco Vale	Avenida do Monte 1240	Braga	650
8	Patricia Cunha	Rua da Velhota 233 5ª ESQ	Braga	745

SQL – SELECT – OPERAÇÕES DE CONJUNTO

- As operações de conjunto Union, Intersect e Except operam em relações;
- Eliminam os registos repetidos; se desejarmos obter as repetições, devemos explicitar através da forma union all, intersect all e except all.

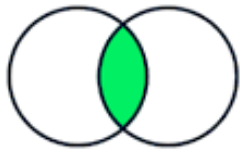
- Tem de existir compatibilidade entre os conjuntos:
  - Mesmo número de campos
  - Tipos de dados compatíveis

UNION
UNION ALL



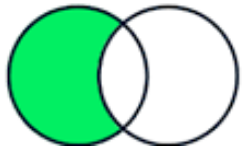
```
SELECT * FROM veiculo
UNION
SELECT * FROM Automovel
```

INTERSECT



```
SELECT * FROM veiculo
INTERSECT
SELECT * FROM Automovel
```

EXCEPT



```
SELECT * FROM veiculo
EXCEPT
SELECT * FROM Automovel
```

SQL – SELECT – OPERAÇÕES DE CONJUNTO

SINTAXE:

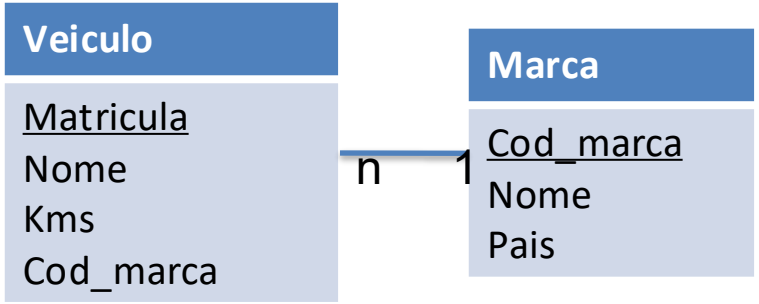
```
SELECT Column1, Column2, … ColumnN
FROM <table>
[WHERE conditions]
[GROUP BY Column(s]]
[HAVING condition(s)]
UNION
SELECT Column1, Column2, … ColumnN
FROM table
[WHERE condition(s)];
ORDER BY Column1,Column2…
```

```
SELECT column-1, column-2 …… FROM
table 1 WHERE….. INTERSECT
SELECT column-1, column-2 ……
 FROM table 2 WHERE…..
```

```
SELECT column-1, column-2 …… FROM
table 1 WHERE…..
 EXCEPT
 SELECT column-1, column-2 ……
FROM table 2 WHERE…..
```

SQL – SELECT – JOIN

- Permite-nos combinar registos de relações diferentes.
- Basta indicar as várias relações no operador FROM usando vírgulas.
- Se as tabelas tiverem atributos com o mesmo nome, iremos ter resultados ambíguos. Neste caso devemos referenciar os atributos com o nome da tabela:
- Retorna uma relação com os atributos das várias tabelas e todas as combinações possíveis de registos.



ISSO É O PRODUTO CARTESIANO.

```
SELECT veiculo.nome, marca.nome
FROM veiculo, marca;
```

ENTÃO O COMO SE FAZ O JOIN ?

SQL – SELECT – JOIN

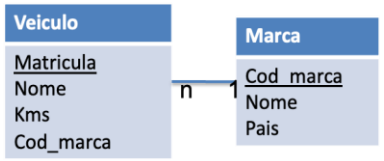
- Se usarmos o produto cartesiano conjuntamente com o operador WHERE estamos a realizar **a junção natural.**
- A condição de junção é implementada no **SQL igualando a chave estrangeira de uma tabela com a respectiva chave primária da outra.**
- Em junções que envolvam mais de duas tabelas, **deve existir pelo menos uma condição de junção** para cada uma das tabelas.

```
SELECT Veiculo.Nome, marca.nome
FROM Veiculo, marca
WHERE veiculo.cod_marca=Marca.cod_marca
```

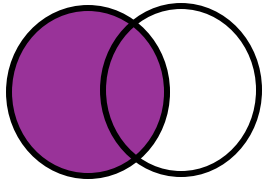
ou

```
SELECT Veiculo.nome, Marca.nome
FROM Veiculo INNER JOIN marca
ON veiculo.Cod_marca = Marca.cod_marca
```

Colocamos o nome da tabela porque o atributo nome existe em ambas as tabelas

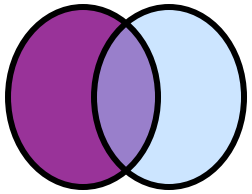


SQL – SELECT – TIPOS DE JOIN



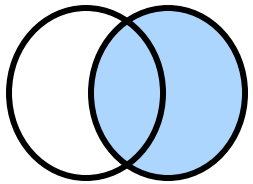
Left

Left Outer Join – além dos registos que fazem match, os registos da tabela esquerda que não existam na direita também aparecem. Quando não existe numa das tabelas os valores aparecem a NULL;



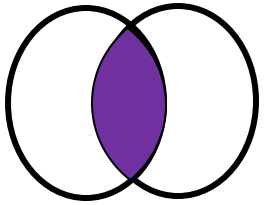
Full

Full Outer Join – aparecem todos os registos independentemente de fazerem ou não match.



Right

Rigth Outer Join – além dos registos que fazem match, aparecem todos os restantes registos da tabela direita que não existam na esquerda;



Natural join **apenas aparecem os que fazem match**. Tem o mesmo efeito que igualar ao campos de ambas as tabelas **na cláusula WHERE**

SQL – SELECT – SINTAXE JOIN

```
SELECT table1.column, table2. column
FROM table1, table2
WHERE table1. column1 = table2. column2
```

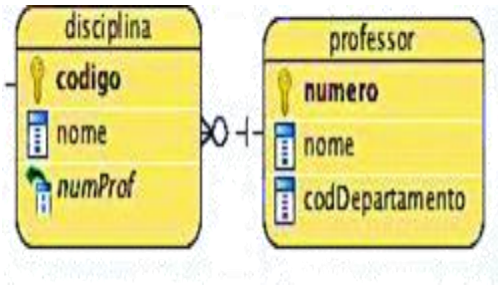
```
SELECT table1.column, table2. column
FROM table1 INNER JOIN table2
ON table1. column1= table2. column2;
```

```
SELECT   table1.column, table2.column
FROM     table1
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column1 = table2.column2)]
```

Nota: table1. column1 = table2. column2 é a condição que une (ou relaciona) as tabelas.

SQL – SELECT – JOIN

EXEMPLO: Liste o nome dos professores e o nome das disciplinas que lecionam




```
SELECT p.nomeprof, d.nome
FROM disciplina d, professor p
WHERE  d.numprof = p.numero
order by p.nomeprof;
```

	NOMEPROF	NOME
1	Carlos Mamede	Programacao
2	Gorreti Marreiros	Programacao Avancada
3	José Avelino	Estatistica
4	Paulo Proença	Eletronica Aplicada
5	Paulo Proença	Electromagnetismo
6	Paulo Proença	Engenharia de Aplicaçoes
7	Rosa Reis	Engenharia de Software
8	Rosa Reis	Bases de Dados

usando o inner join ->

```
SELECT p.nomeprof, d.nome
FROM disciplina d
inner join professor p on d.numprof = p.numero
order by p.nomeprof;
```