

Curso Profissional: Programador/a de Informática
PSD – 10.º ano: UFCD 0814 – Programação em linguagem SQL avançada
Ficha de Trabalho 1

Ano letivo 21/22

NOTA: Todos os exercícios desta ficha devem ser executados no WampServer.

SUBQUERIES

- **Querie** identifica, normalmente, qualquer comando SQL.
- **Subquerie** é normalmente utilizado para se referenciar qualquer **SELECT** ou conjunto de comandos **SELECT** contidos dentro de um outro **SELECT**. A consulta principal irá depender da subconsulta.

Exemplos:

1. Qual o nome da Empregados com o menor salário?
Precisamos saber qual o menor salário e qual o nome da Empregados a que o salário corresponde.

- a) Menor salário:
SELECT MIN (Salario) AS menor
FROM Empregados

Menor
800

- b) Quem ganha 800:
SELECT Nome
FROM Empregados
WHERE Salario=800

Nome
Carla

Com queries encadeadas, isto é usando subqueries, seria:

```
SELECT Nome  
FROM Empregados  
WHERE Salario= (SELECT MIN (Salario) AS menor  
FROM Empregados)
```

Nome
Carla

Retorno de uma subconsulta

- Uma subconsulta de **valor único** devolve apenas um valor e pode ser usada no lugar de qualquer expressão utilizando operadores (=, <, >, <>). Sendo esse o caso do exemplo anterior (pode no entanto em alguns sgbd ser aplicado a um resultado de múltiplos valores).

1 coluna → 1 valor

WHERE A = (SELECT b...) /*Verdade se A=B*/

- Quando uma subconsulta retorna **múltiplas linhas** só pode ser usada na condição WHERE utilizando cláusulas especiais.

1 coluna → muitos valores

Where A **IN|ALL|ANY**

O operador IN aplicado a SubQuery

O operador IN quando aplicado a uma SubQuery, devolve o valor lógico VERDADE se dessa SubQuery resultar alguma linha de resultados. Caso contrário devolve FALSO. Pode ser considerado um apelido para “= ANY”.

Sintaxe:

```
SELECT ...  
FROM...  
WHERE <campo> [NOT] IN (SubQuery)
```

Exemplo: Qual o conjunto de nomes e código postal dos Empregados que vivem no Sabugal?

```
SELECT Nome, CodPostal  
From Empregados  
WHERE CodPostal IN (SELECT CodPostal  
                     FROM Postal  
                     WHERE Localidade='Sabugal')
```

nome	CodPostal
Bruno	6320
Alexandra	6320

Para responder à questão: qual o conjunto de Empregados que não vivem no Sabugal? Bastará negar a condição apresentada no exemplo anterior (NOT IN).

O operador ANY aplicado a SubQuery

Deve seguir um operador de comparação e devolve TRUE se a comparação é verdadeira para algum dos valores (atributos) na coluna resultante da subconsulta.-> FUNCIONA COMO OR

Sintaxe:

```
SELECT ...  
FROM...  
WHERE <campo> = | > | < | <> | <= | >= ANY (SubQuery)
```

Exemplo: Qual o conjunto de nomes dos Empregados com idades inferiores ao funcionário mais idoso.

```
SELECT Nome  
FROM Empregados  
WHERE idade < ANY (SELECT idade  
                   FROM Empregados)
```

Nome
Aníbal
Anita
Sousa
Xavier
Bruno
Alexandra
Carla

ALL

Deve seguir um operador de comparação e devolve TRUE se a comparação é verdadeira para todos os valores na coluna resultante da subconsulta. ->FUNCIONA COMO AND

Sintaxe: semelhante á da cláusula anterior

Exemplo: Qual o conjunto de nomes dos Empregados que têm salário superior ao gestor de marketing?

```
SELECT Nome
FROM Empregados
WHERE salario > ALL (SELECT salario
                     FROM Empregados WHERE cargo ='gestor de marketing')
```

Nome
Cruz

O operador EXISTS

É um operador que verifica se, da execução de uma SubQuery, resultou alguma linha. Este operador só pode ser utilizado para avaliar o resultado de SubQueries.

Sintaxe:

```
SELECT ...
FROM...
WHERE [NOT] EXISTS (SubQuery)
```

Exemplo: Qual o nome e a Morada completa de todas os Empregados que têm comissões a receber?

1.Resolução usando EXISTS

```
SELECT Nome, Empregados.CodPostal, Localidade
```

```
From Empregados, Postal
WHERE Empregados.CodPostal = Postal.CodPostal
AND EXISTS (SELECT CodC
            FROM Comissao)
```

Nome	CodPostal	Localidade
Ana Marques	6200	Covilhã
Lili Caneças	6200	Covilhã
Marcelo de Sousa	6200	Covilhã
Filomena Cautela	6230	Fundão
Charles Xavier	6230	Fundão
Cristina Ferreira	6320	Sabugal
Nilton	6320	Sabugal
Jorge Jesus	6230	Fundão

2.Resolução usando IN

```
SELECT Nome, Empregados. CodPostal, Localidade
From Empregados, Postal
WHERE Empregados.CodPostal = Postal.CodPostal AND
      CodC IN (SELECT CodC
              FROM Comissao)
```

3. Resolução usando a junção tradicional

```
SELECT DISTINCT Nome, CodPostal, Localidade
From Empregados, Postal, comissao
WHERE Empregados.CodPostal = Postal.CodPostal AND
      Comissao. CodC =Empregados. CodC
```

SubQueries Correlacionadas e Não Correlacionadas

- Á *SubQuery* (colocada entre os parêntesis) que depende, para o seu funcionamento, de valores da *Query* mais exterior chama-se **Query Correlacionada**. Sendo por isso necessário fazer a ligação entre os comandos select. ~~ou seja, campos da consulta principal fazem parte do critério de filtro da subconsulta.~~

(O SELECT interior depende do SELECT exterior)

- Se a SubQuery interior não depender de valores da Query exterior diz-se que é uma **Query Não Correlacionada**.

(O SELECT interior não depende do SELECT exterior)

Exemplo de Query Não Correlacionada:

O sentido da execução é de dentro para fora.

```
SELECT Nome
FROM Empregados
WHERE Salario= (SELECT MIN(Salario) AS menor
                FROM Empregados)
```




Exemplo de Query Correlacionada:

Qual o nome e salário dos Empregados cujo salário é inferior ao total das suas comissões *15?

O sentido da execução é de fora para dentro.

```
SELECT Nome, Salario
FROM Empregados P
WHERE Salario < (SELECT SUM(Valor)
                FROM Comissao C
                WHERE C.CodC=P.CodC) * 15
```



O SELECT exterior envia o salário a fim de ser comparado com o total de comissões associadas ao CodC a que pertence o salário.

Para cada Empregado “estudado” no SELECT exterior, é executado o SELECT interior calculando o total de comissões que lhe estão associadas.

NOTA: Relembra que a utilização de operadores lógicos e relacionais na comparação com o resultado de uma *SubQuery* obriga a que esta última devolva apenas uma única linha de resultado. Caso contrário obtém-se um erro.

Exemplo:

```
SELECT *
FROM Empregados
WHERE CodC < (SELECT DISTINCT CodC
               FROM Comissao
               WHERE Valor < 350)
```

CodC
2
3
4

Sentido da execução
 Resultado da SubQuery

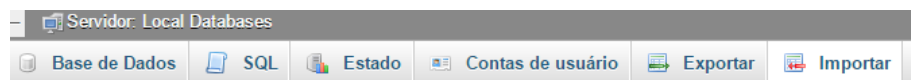
Ao executar o SELECT exterior dá ERRO!

EXERCÍCIOS

PARTE 1

Os exercícios devem ser executados e copiados para o notepad++ e a enviar para a classroom, com o nome F4-parte1.

- A-** Importa a base de dados VendasCD.SQL que se encontra na *Classroom Disciplina de PSD, no tópico UFCD 0814 - Programação em linguagem SQL avançada*.
- No MySQL seleciona o separador **Importar** e pressiona no botão **Escolher ficheiro** para selecionar o ficheiro (neste exemplo **VENDASCD_UTF8.SQL**),
Escolhe o mapa de caracteres **utf-8**:



Fazendo importação para o servidor atual

File to import:

O ficheiro pode ser comprimido (gzip, bzip2, zip) ou descomprimido.

O nome de um ficheiro comprimido tem que acabar em **[format].[compression]**. Exemplo: **.sql.zip**

Procurar no seu computador: SQLVENDAS_UTF8.SQL (Tamanho máximo: 128MB)

You may also drag and drop a file on any page.

Configurar o Mapa de Caracteres do ficheiro:

- B-** Define as chaves externas necessárias.

Tabelas: *CD, Vendas, Lojas, Localidades e Editoras*.

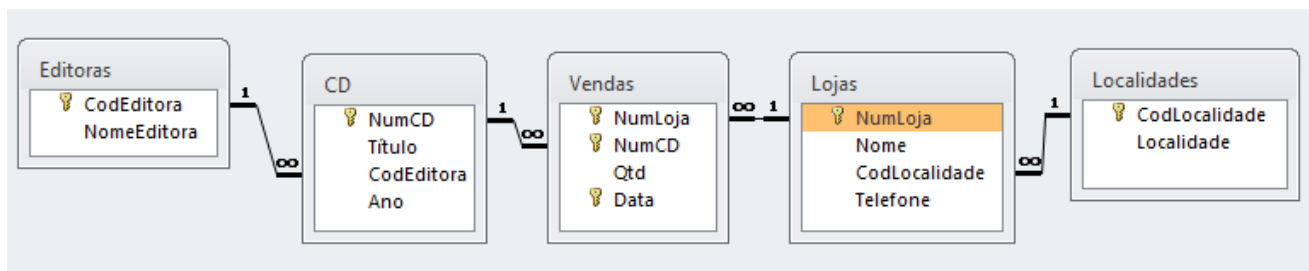


Fig. 1 - Tabelas da base de dados VendasCD

Nota: todos os códigos ("NumCD", "CodEditora", etc.) são numéricos e únicos. O campo Qtd refere-se ao número de unidades de CD's vendido.

C- Escreve as instruções SQL que permitam satisfazer os seguintes pedidos:

- 1- Mostrar o total de N° de unidades vendidas do CD com o NumCD nº 3.
- 2- Mostrar para cada CodLocalidade, o total de N° de unidades de CD Vendidas.
- 3- Mostrar o Título do CD e o Nome da Editora, ordenando os registos ascendentemente pelo Título.
- 4- Quantas lojas existem em cada Localidade (mostrar **Localidade** e **Qtd de Lojas**)?
- 5- Qual o máximo N° de unidades vendidas do CD com o **NumCD** nº 3?
- 6- Mostrar **Nome** das lojas ordenado por ordem descendente do N° de unidades do CD vendidas do **CD** com o **NumCD** nº 5 (mostrar **Nome** e **Qtd**).
- 7- Mostrar o nome das Lojas que não venderam CD's.
 - a. Usa os operadores NOT, EXISTS;
 - b. Usa os operadores NOT e IN.
- 8- Mostrar o total de CD's vendidos por nome de distrito.
- 9- Mostrar o nome das Lojas do Distrito de Aveiro.
 - a. Usa a junção de tabelas;
 - b. Usa o operador IN.
- 10- Mostrar para cada CodLocalidade os totais de N° de unidades do CD vendido maiores ou iguais a 50.
- 11- Mostrar o Nome das Lojas que não têm Telefone.
- 12- Mostrar Todos os Títulos dos CD e respetivo nome da Editora dos CD que têm no título a string "Psi" em qualquer parte do mesmo.
- 13- Mostrar **Nome** das lojas que venderam o CD com o **NumCD** nº 3 (mostrar **Nome** sem repetições).
- 14- Mostrar o total de CD's vendidos por nome de Localidade.
- 15- Mostrar o nome das Lojas da Localidade Fundão.
 - a. Usa a junção de tabelas;
 - b. Usa o operador IN.

EXERCÍCIOS

PARTE 2

1- Considera agora as tabelas da base de dados VendasCD, importada na parte 2:

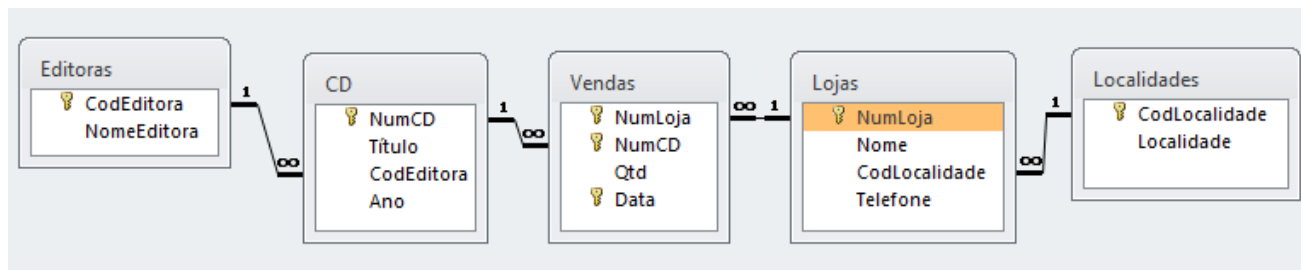


Fig. 1 - Tabelas da base de dados VendasCD

2- Cria uma consulta na BD *VendasCD* e respetivo SQL que:

- Adiciona o campo preço, com o tipo double à tabela CD;
- Atribui o preço de 10 euros a todos os CD;
- Altera os registos da tabela CD, de modo a que, todos os CD da Editora com o código 1 ou 2, tenham uma redução no preço de 5%.

3- Cria outra consulta na BD *VendasCD* e respetivo SQL que Introduza na tabela Editoras, uma linha com os valores 40 e CoviLoL, referente aos campos CodEditora e NomeEditora.

4- Cria outra consulta na BD *VendasCD* e respetivo SQL que apague da tabela Vendas os registos em que o campo Qtd esteja entre 20 e 25.

5- Cria outra consulta na BD *VendasCD* e respetivo SQL que apague da tabela CDS os registos dos Cds cuja data de venda seja anterior a 31-12-1960.

6- Cria outra consulta na BD *VendasCD* e respetivo SQL que, nas vendas, acrescente à quantidade de unidades vendidas 100 cds, sempre que se refira à loja 1 ou 3.

7- Exporta a BD e atribui-lhe o nome VendasCDF4 e envia-a também para a classroom da disciplina.

Bibliografia

Damas, L. (2005). SQL. Lisboa: FCA

Tavares, F. (2015). MySQL. Lisboa: FCA

Santos, R. L. Revisão + Visões + Sub-Consultas + JOINS.2015 [S.l.]. Disponível em:

<http://www.ricardoluis.com/wp-content/uploads/2015/08/Revisao-visoes-subconsultas-joins.pdf>

<https://www.tutorialsteacher.com/sql/all>