ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Structured Query Language

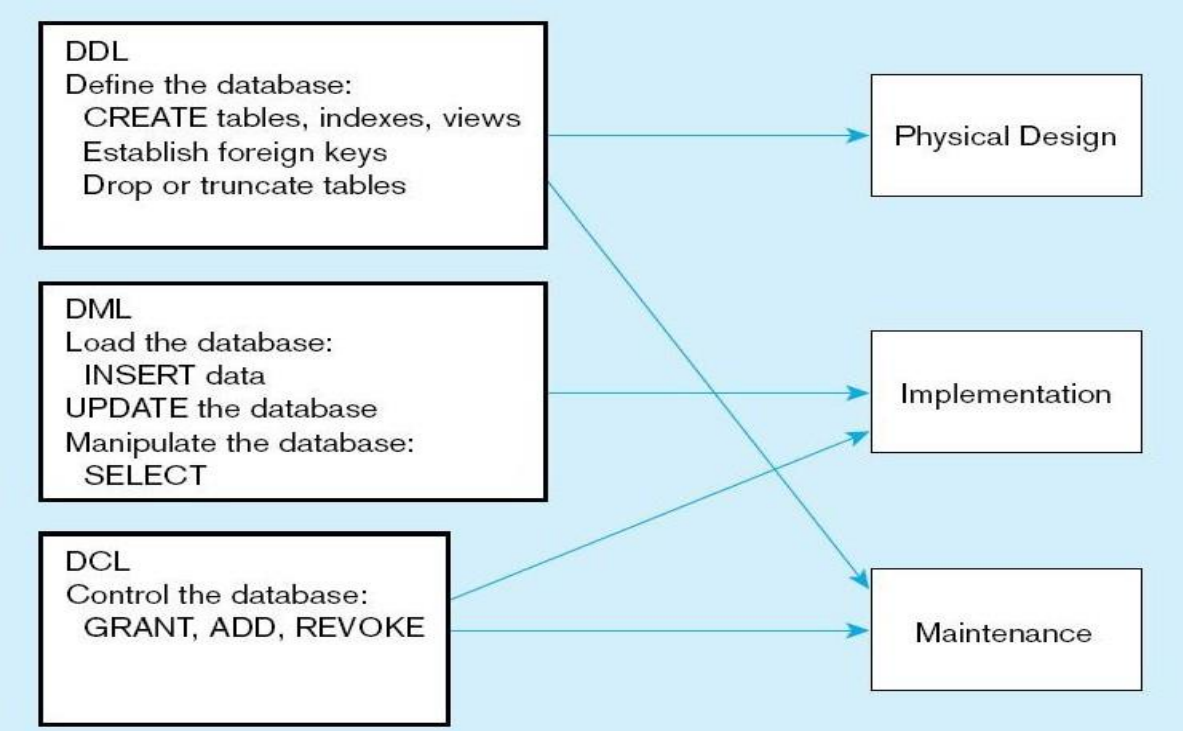
SubQueries

DEE
Departamento de Engenharia Electrotécnica

Licenciatura em Engenharia de Telecomunicações e Informática

SQL- Strutured Query Language

- SQL é mais que uma linguagem de interrogação estruturada. Inclui características para a definição da estrutura de dados, para alterar os dados de uma base de dados, e para especificar esquemas de segurança. Estas características agrupam-se do seguinte modo:



SubQueries

- *Uma* Subquery é uma instrução SELECT que está embebida numa cláusula de outra instrução SELECT.
- Elas podem ser úteis quando precisamos de selecionar linhas de uma tabela com uma condição que depende dos dados na própria tabela.
- Na maioria dos sistemas de base de dados, as subconsultas normalmente fazem parte da cláusula WHERE mas podem estar no Select, From ou Having

```
SELECT      ListaDeColunas
FROM        TabelaExterna
WHERE       Expressao Operador

                    (SELECT     ListaDeColunas
                    FROM        TabelaInterna
                    [WHERE      Expressão Operador]);
```

SubQueries

- A *subquery* (consulta interna) geralmente será executada primeiro, e a sua saída é usada para completar a condição de consulta para a consulta principal (ou externa).
- Podemos usar os operadores de comparação Aritméticos (>, < ou =) se subquery retorna **uma só linha**

```
Select  nome_empregado
From empregado
Where idade = (select max(idade)
                    from  empregado)
```

- Podemos usar os operadores de comparação do tipo IN, ANY ou ALL se a subquery retorna **várias linhas**

```
Select  nome_empregado
From empregado
Where  codDepartamento IN (select coddepartamento
                                from  Departamento)
```

Tipo de SubQueries

- Subconsulta de linha única: retorna zero ou uma linha.
- Subconsulta de várias linhas: retorna uma ou mais linhas.
- Subconsultas de múltiplas colunas: Retorna uma ou mais colunas.
- Subconsultas Não Correlacionadas: A subconsulta interna não está relacionada com a subconsulta externa
- Subconsultas Correlacionadas: Refere-se a uma ou mais colunas na instrução SQL externa. A subconsulta é conhecida como uma subconsulta correlacionada porque a subconsulta está relacionada à instrução SQL externa

NÃO CORRELACIONADAS

- SELECT Interior
  - Não depende de valores do SELECT exterior.
  - Não referência colunas do SELECT exterior.
- SELECT Interior é executado em 1º lugar (uma vez) porque o SELECT exterior é que depende do SELECT interior (executado depois).

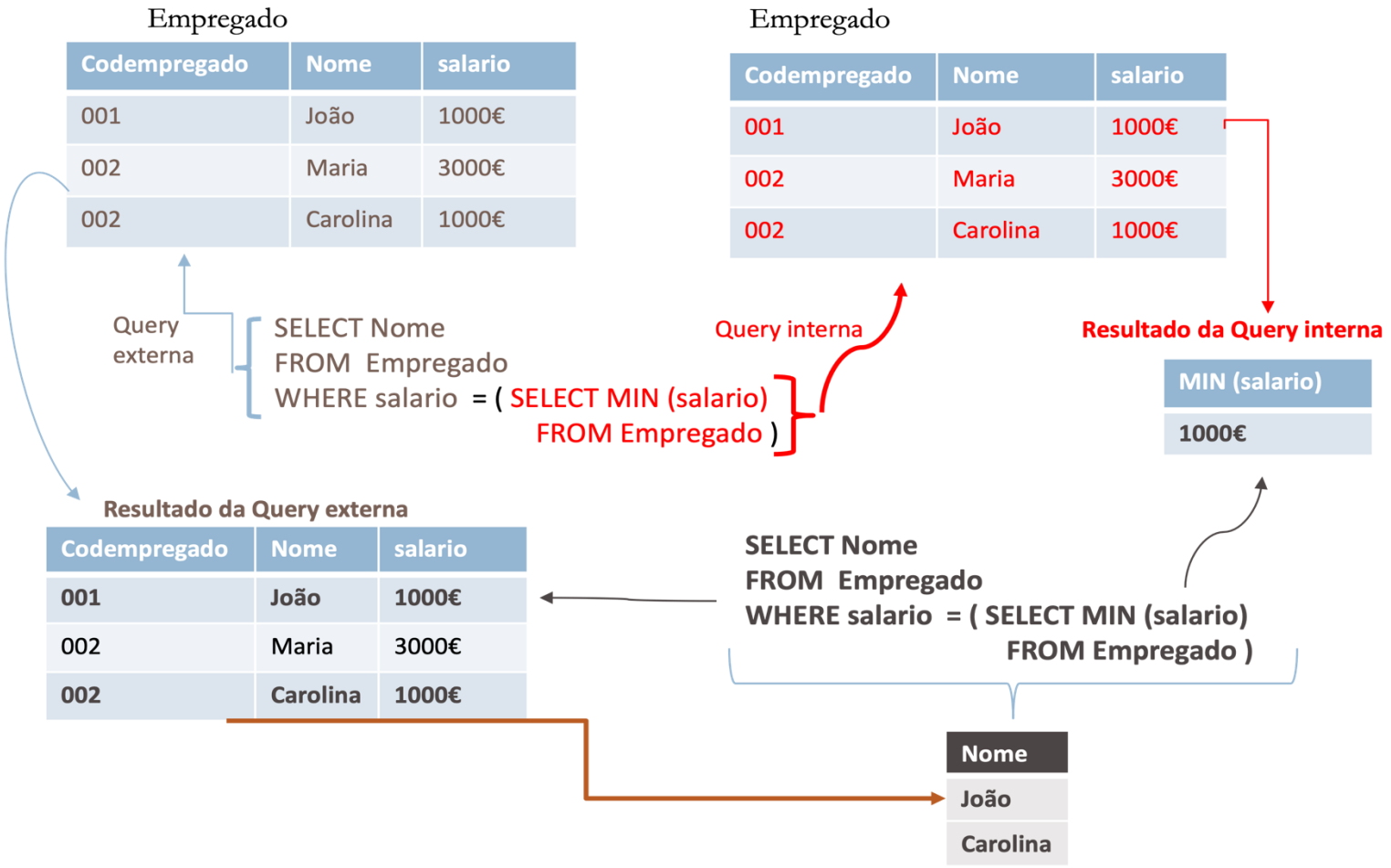
```
SELECT Nome
FROM  Empregado
WHERE salario  = ( SELECT MIN (salario)
                   FROM Empregado )
```

Operador aritmético porque subquery retorna um registo único

```
WHERE salario = MIN(salario)
```

Ilegal

NÃO CORRELACIONADAS



CORRELACIONADAS

- SELECT Interior depende de valores do SELECT exterior.
- SELECT Interior é executado tantas vezes quanto o SELECT exterior e espera por um valor do SELECT exterior.

```
SELECT Nome, Salario
FROM Empregado
WHERE Salario < ( SELECT SUM(Valor)
                  FROM Comissao WHERE Comissão.codempregado = Empregado.codempregado )
```

valor variável;
depende da linha do
select exterior

faz a ligação ao
select exterior

CORRELACIONADAS

Empregado

Codempregado	Nome	salario
001	João	1000€
002	Maria	3000€
003	Carolina	1000€

Comissão

Codempregado	data	valor
001	12/01/2017	800€
001	13/01/2017	250€
002	1/02/2018	1500€
002	2/0272018	200€
002	3/02/2018	300€
003	3/02/2018	1200€

```
SELECT Nome, Salario
FROM Empregado
WHERE Salario <  ( SELECT SUM(Valor)
                   FROM Comissao
                   WHERE Comissao.codempregado = Empregado.codempregado )
```

Codempregado	Nome	salario
001	João	1000€
002	Maria	3000€
003	Carolina	1000€

Codempregado	data	valor
001	12/01/2017	800€
001	13/01/2017	250€
002	1/02/2018	1500€
002	2/0272018	200€
002	3/02/2018	300€
003	3/02/2018	1200€

1050 1000<1050

2000 3000<2000

1200 1000<1200

Nome	salario
João	1000€
Carolina	1000€

RESULTADO FINAL

RESUMO

Tipo de SubQuery	Select Interior	Sentido de Execução	Execução do select interior
Não correlacionada	Não depende do select exterior	1º o select Interior 2º select Exterior	Uma vez
Correlacionada	Depende do select exterior	Execução do interior intercalada com o exterior	nr. vezes = ao numero de vezes do select exterior

OERADOR IN/NOT IN

- Seleciona as linhas em que os campos indicados antes do operador existam na *subquery*.
- Os campos indicados têm de ser no mesmo número dos campos retornados pela query e têm de ter domínios compatíveis.
- O operador NOT IN permite obter o resultado inverso.

```
SELECT nome FROM empregado
WHERE nr_emp NOT IN   (SELECT diretor FROM departamento);
```

- Aparecem na forma:

<attribute-name> IN (subquery) Ou

<attribute-name> NOT IN (subquery)

OERADOR ANY

DEE Departamento de Engenharia Electrotécnica

- Seleciona os resultados cujos campos indicados sejam iguais (=), maiores (>), menores(<) ou diferentes (<>) do **que pelo menos uma linh**a da subquery.
- Os campos indicados têm de ser no mesmo número dos campos retornados pela subquery e têm de ter domínios compatíveis.

=ANY é o mesmo que IN

SOME é o mesmo que ANY

```
SELECT nome, salario, nrdep FROM empregado

WHERE salario < ANY (SELECT distinct salario FROM
                                empregado where nrDep=2)
```

A Condição ANY retorna TRUE quando qualquer dos valores do resultado da subquery satisfaz a condição.

OERADOR ALL

- Seleciona os resultados cujos campos indicados sejam iguais (=), maiores (>), menores(<) ou diferentes(<>) **do que todos os** tuplos da subquery.
- Os campos indicados têm de ser no mesmo número dos campos retornados pela query e tem de ter domínios compatíveis.
- <>ALL é o mesmo que NOT IN

```
SELECT nome, salario, nrdep FROM empregado

WHERE salario > ALL (SELECT distinct salario FROM
                     empregado where nrDep=2)
```

OERADOR EXISTS/ NOT EXISTS

DEE Departamento de Engenharia Electrotécnica

- Em subqueries correlacionadas todos os operadores lógicos são aplicados, contudo usa-se o operador **EXISTS ou NOT EXISTS** para testar
  - se um valor recuperado pela consulta externa existe no conjunto de valores recuperados pela consulta interna.

- O operador EXISTS ou NOT EXISTS é usado para determinar se há dados numa lista de valores (restringe o conjunto de resultados de uma consulta externa para as linhas que atendam a subquery).
  - Verifica se resultado de subquery é ou não um conjunto vazio.

- O operador EXISTS e NOT EXISTS retorna TRUE ou FALSE, dependendo se as queries devolvem linhas ou não;

Operadores IN e EXISTS

Operador	Exemplo	Observação
IN	coluna [**NOT**] **IN** (*subquery*)	▪ Operador binário; ▪ Verifica se valor existe no resultado de uma *subquery*; ▪ Conjunto pode ser uma *subquery*.
EXISTS	[**NOT**] **EXISTS** (*subquery*)	▪ Operador unário; ▪ Verifica se resultado de *subquery* é ou não um conjunto vazio;

- **Exemplos**
  - Tabelas: Colaborador (**id**, nome, ..., salario, ..., codPostal) ; Postal (**codPostal**, ..., localidade)

```
SELECT nome, codPostal
  FROM colaborador
 WHERE codPostal IN ( SELECT codPostal
                        FROM postal
                       WHERE localidade = 'Lisboa' );
```

```
SELECT nome, codPostal
  FROM colaborador c
 WHERE EXISTS ( SELECT codPostal
                  FROM postal p
                 WHERE c.codPostal = p.codPostal AND localidade = 'Lisboa' );
```

Ordenação de Dados com subconsultas

- **NÃO se pode utilizar uma cláusula ODER BY numa subconsulta.**
  - Mantém-se a regra de que só se pode ter uma única cláusula ORDER BY para uma instrução de SELECT e, se especificada, terá de ser na última cláusula no comando SELECT.

```
SELECT nome FROM empregado
WHERE nr_emp NOT IN   (SELECT diretor FROM departamento)
ORDER BY nome;
```

```
SELECT nome FROM empregado
WHERE nr_emp NOT IN   (SELECT diretor FROM departamento
                       order by nome);
```

→ **Não é válido**

Linhas de Orientação

- a consulta interna deverá ser colocada entre parênteses e, terá de estar no lado direito da condição.
- A subconsulta não pode conter uma cláusula ORDER BY.
- A cláusula ORDER BY aparece no fim da instrução de SELECT principal.
  - **Usar uma cláusula ORDER BY na instrução SELECT principal (consulta externa), que será a última cláusula.**
- Use operadores aritméticos com subconsultas de linha única.
- Coluna múltiplas na lista de SELECT da consulta interna terão de estar pela mesma ordem que as colunas que apareçam na cláusula de condição da consulta principal. O tipo de dados e o número de coluna tem também de corresponder.
- As subconsultas são sempre executadas primeiro a partir da mais interior para menos interior no encadeamento, a não ser que sejam subconsultas correlacionadas.
- Podem-se utilizar operadores lógicos e operadores de SQL, bem como ANY e ALL.
- Se uma subconsulta (consulta interna) retornar um valor nulo para a consulta externa, a consulta externa não retornará nenhuma linha ao usar determinados operadores de comparação numa cláusula WHERE.

Linhas de Orientação

- As subconsultas podem :
  - produzir uma ou mais linhas.
  - produzir uma ou mais colunas.
  - Utilizar uma ou mais colunas.
  - utilizar GRPOUP BY ou funções de grupo.
  - Ser utilizadas em vários predicados AND e OR da mesma consulta
  - Juntar tabelas.
  - extrair dados de uma tabela diferente da tabela da consulta externa.
  - aparecer em instruções de SELECT, UPDATE; DELETE; INSERT; CREATE TABLE
  - Serem correlacionadas com uma consulta externa.

EXEMPLOS

DEE Departamento de Engenharia Electrotécnica

Utilizando uma subconsulta como uma tabela derivada

- É um conjunto de registos dentro de uma consulta que funciona como uma tabela;
- Ela toma o lugar da tabela na cláusula FROM.
- É otimizada com o resto da consulta.

```
1  SELECT empregado.*, TH.hora
2  FROM  empregado, ( SELECT   nr_emp, sum(TotalHoras) hora
3                     FROM trabalho
4                     GROUP BY nr_emp)  TH
5  WHERE  empregado.nr_emp = TH.nr_emp;
6
```

EXEMPLOS

DEE Departamento de Engenharia Electrotécnica

Utilizando uma subconsulta como uma expressão

- É executada uma vez para toda a instrução.

```
select * from trabalho
SELECT nr_emp, nome, (SELECT MIN(totalhoras)  FROM trabalho T WHERE T.nr_emp = E.nr_emp) minimohora
FROM empregado E;
```

EXEMPLOS

Utilizando uma subconsulta na clausula Having e/ou Where

- Numa cláusula HAVING/WHERE temos sempre uma condição e a subquery atua operando dentro dessa condição.

```
SELECT nrdep COUNT(*) TotalEmpregados
FROM empregado
GROUP BY nrdep
HAVING COUNT(*) = ( SELECT MAX (conta)
                    FROM ( SELECT count(*)  FROM empregado  GROUP BY  nrdep )  maximoepregados)
```

```
NRDEP TOTALEMPREGADOS
------ --------------
     2              3
```

```
SELECT nrdep COUNT(*) TotalEmpregados
FROM empregado
GROUP BY nrdep
HAVING COUNT(*) = ( SELECT MAX (conta)
                    FROM ( SELECT count(*)  FROM empregado
                           GROUP BY  nrdep )  maximoepregados)
```

```
AX(COUNT(*))
------------
           3
```

Funções de Agregação Encaixadas

DEE Departamento de Engenharia Electrotécnica

- **Funções de Agregação**
  - COUNT
  - SUM
  - AVG
  - MAX
  - MIN

- **Exemplo**
  - Tabela: Colaborador (**id**, nome, …, salario, …, codPostal)
  - Consulta: Média do total de salários dos colaboradores por código postal
  - Solução 1: *Subquery* Não-Correlacionada

```
SELECT AVG(total) média
  FROM ( SELECT SUM(salario) total
           FROM colaborador
          GROUP BY codPostal );
```

EXEMPLOS

Inscritos

Id_aluno	Id_disciplina
1090	BASDAD
1090	LPROJ
1080	BASDAD
1070	BASDAD
1060	BASDAD
1060	LPROJ

Disciplina

Id_disciplina	descricao
BASDAD	Base de Dados
LPROJ	Laboratorio projeto

Pergunta: Quais são os alunos inscritos em **todas** as disciplinas?

Id_aluno	Id_disciplina
1090	BASDAD
1090	LPROJ
1060	BASDAD
1060	LPROJ

CONSULTA - TODOS

Pergunta: Quais são os alunos inscritos em **todas** as disciplinas?

- **por Inclusão de Conjuntos - usa-se EXISTS / IN e EXCEPT**

```
SELECT I1.ID_ALUNO
FROM INSCRITOS I1
WHERE NOT EXISTS ( SELECT ID_DISCIPLINA --------- todas as disciplinas
                   FROM DISCIPLINA
                   EXCEPT
                   SELECT ID_DISCIPLINA --------- todas as disciplinas de I1
                   FROM INSCRITOS I2
                   WHERE I2.ID_ALUNO = I1.ID_ALUNO);
```

Se o aluno 1090 **(i1.id_aluno)** estiver inscrito em todas as disciplinas,

então: DISCIPLINAS – $\sigma_{ID\_ALUNO=1090}(INSCRITOS) = \emptyset$

CONSULTA - TODOS

Pergunta: Quais são os alunos inscritos em **todas** as disciplinas?

- **por Comparação de Cardinalidades – usa-se GROUP BY e COUNT()**

```
SELECT ID_ALUNO
FROM INSCRITOS
GROUP BY ID_ALUNO
HAVING COUNT(*)=( SELECT COUNT(*)
                  FROM DISCIPLINA)
```

-nr. disciplinas aluno = total disciplinas

Id_disciplina	descricao
BASDAD	Base de Dados
LPROJ	Laboratorio projeto

Id_aluno	Id_disciplina
1090	BASDAD
1090	LPROJ
1080	BASDAD
1070	BASDAD
1060	BASDAD
1060	LPROJ

CONSULTA - TODOS

Pergunta: Quais são os alunos inscritos em **todas** as disciplinas?

- **por Quantificação - usa-se EXISTS e/ou IN**
  - Reformulando a pergunta:
    - "Quais são os alunos, para os quais não existe nenhuma disciplina sem a sua inscrição?"

```
SELECT DISTINCT ID_ALUNO
FROM INSCRITOS I1
WHERE NOT EXISTS (
SELECT *
                FROM DISCIPLINA D WHERE NOT EXISTS (
    SELECT *
    FROM INSCRITOS I2
    WHERE I2.ID_ALUNO = I1.ID_ALUNO
    AND I2.ID_DISCIPLINA = D.ID_DISCIPLINA));
```

PIVOT CLAUSE

- permite escrever uma tabulação cruzada.
  - Isso significa que se pode agregar os resultados de consulta e girar as linhas em colunas.

```
SELECT first_column AS <first_column_alias>,
[pivot_value1], [pivot_value2], ... [pivot_value_n]
FROM
(<source_table>) AS <source_table_alias>
PIVOT
(
 aggregate_function(<aggregate_column>)
 FOR <pivot_column>
 IN ([pivot_value1], [pivot_value2], ... [pivot_value_n])
) AS <pivot_table_alias>;
```

employee_number	last_name	first_name	salary	dept_id
12009	Sutherland	Barbara	54000	45
34974	Yates	Fred	80000	45
34987	Erickson	Neil	42000	45
45001	Parker	Sally	57500	30
75623	Gates	Steve	65000	30

TotalSalaryByDept	30	45
TotalSalary	122500	176000

```
SELECT 'TotalSalary' AS TotalSalaryByDept,
[30], [45]
FROM
(SELECT dept_id, salary
 FROM employees) AS SourceTable
PIVOT
(
 SUM(salary)
 FOR dept_id IN ([30], [45])
) AS PivotTable;
```