Bases de Dados

- SQL – Parte II –

- Consultas encadeadas e funções de agregação -

António Sousa
antonioribeirosousa@gmail.com

Consultas encadeadas

O SQL permite que uma consulta, chamada de **consulta encadeada**, seja utilizada como parte da condição WHERE de outra consulta, chamada de **consulta externa**, sendo possível ter vários níveis de consultas encadeadas.

```
[<ATRIB> | <VAL>] [NOT] IN <CONSULTA>
[<ATRIB> | <VAL>] [= | < | <= | > | >= | <>] <CONSULTA>
[<ATRIB> | <VAL>] [NOT] [= | < | <= | > | >= | <>]ALL
<CONSULTA>
[<ATRIB> | <VAL>] [NOT] [= | < | <= | > | >= | <>]ANY
<CONSULTA>
```

(A **IN** CONS) é verdadeiro se A ∈ CONS. Caso contrário é falso.
(A **θ** CONS) é verdadeiro se CONS tiver um único tuplo com um só atributo de valor C tal que A θ C. Caso contrário é falso.
(A **θALL** CONS) é verdadeiro se A θ C para todo o C ∈ CONS. Caso contrário é falso.
(A **θANY** CONS) é verdadeiro se A θ C para algum C ∈ CONS. Caso contrário é falso.

Consultas encadeadas

- Obtenha o nome dos empregados que trabalham no departamento de Produção.

```
SELECT NomeP, NomeF
FROM EMPREGADO
WHERE NumDep = (SELECT Num
                FROM DEPARTAMENTO
                WHERE Nome = 'Produção');
```

- Obtenha o nome dos empregados cujo salário é superior ao salário de todos os empregados do departamento 4.

```
SELECT NomeP, NomeF
FROM EMPREGADO
WHERE Salário >ALL (SELECT Salário
                    FROM EMPREGADO
                    WHERE NumDep = 4);
```

Consultas encadeadas

- Obtenha o nome dos projectos nos quais o Rui Silva trabalha
ou é o gerente.

```
SELECT DISTINCT Nome
FROM PROJECTO
WHERE Num IN (SELECT NumProj
              FROM EMPREGADO, TRABALHA_EM
              WHERE NomeP = 'Rui' AND NomeF = 'Silva'
              AND NumBI = EmpBI)
        OR
        Num IN (SELECT PROJECTO.Num
              FROM EMPREGADO, DEPARTAMENTO, PROJECTO
              WHERE NomeP = 'Rui' AND NomeF = 'Silva'
              AND NumBI = GerenteBI AND
              DEPARTAMENTO.Num = PROJECTO.NumDep);
```

Consultas encadeadas

- Obtenha o número do BI dos empregados que trabalham as mesmas horas que o empregado Rui Silva (NumBI = '487563') trabalha num determinado projecto.

```
SELECT DISTINCT EmpBI
FROM TRABALHA_EM
WHERE (NumProj, Horas) IN (SELECT NumProj, Horas
                           FROM TRABALHA_EM
                           WHERE EmpBI = '487563');
```

- Obtenha o número do BI dos empregados que trabalham nos projectos 4, 5 ou 6.

```
SELECT DISTINCT EmpBI
FROM TRABALHA_EM
WHERE NumProj IN (4, 5, 6);
```

Funções de Agregação

- O SQL permite sumariar informação por utilização de funções de agregação.
  - **COUNT**(<ATRIB>)
  - **SUM**(<ATRIB>)
  - **MAX**(<ATRIB>)
  - **MIN**(<ATRIB>)
  - **AVG**(<ATRIB>)

- Os valores NULL existentes nos atributos a sumariar **não são considerados** pelas funções de agregação.

- Obtenha o valor do salário máximo, do salário mínimo, da média do valor dos salários e da soma do salário de todos os empregados.

```
SELECT
MAX(Salário), MIN(Salário),AVG(Salário), SUM(Salário)
FROM EMPREGADO;
```

Funções de Agregação

- Obtenha o número de total de empregados que trabalham no departamento de 'Marketing'

```
SELECT count( * )
FROM EMPREGADO, DEPARTAMENTO AS D
WHERE Numdep = D.Código AND Nome = 'Marketing';
```

- Obtenha o número de valores diferentes de salário.

```
SELECT COUNT(DISTINCT Salário)
FROM EMPREGADO;
```

- Obtenha o nome dos empregados com dois ou mais dependentes.

```
SELECT NomeP, NomeF
FROM EMPREGADO AS E
WHERE (SELECT COUNT(*)
        FROM DEPENDENTE
        WHERE E.NumBI = EmpBI) >= 2;
```

GROUP BY

- Permite agrupar tuplos em função de um conjunto de atributos. Em cada grupo, todos os tuplos têm o mesmo valor para os atributos de agrupamento.

```
GROUP BY <ATRIB_1>, ..., <ATRIB_N>
```

- É utilizado em conjunto com as funções de agregação. As funções de agregação são aplicadas separadamente e individualmente sobre cada grupo.
- Se existirem valores NULL nos atributos de agrupamento **é criado um grupo separado** para esse conjunto de tuplos.

- Os atributos do corpo da cláusula SELECT **devem estar em funções de agregação ou devem fazer parte dos atributos de agrupamento**.

- Obtenha o número de empregados por departamento e a respectiva média salarial.

```
SELECT NumDep, COUNT(*), AVG(Salário)
FROM EMPREGADO
GROUP BY NumDep;
```

HAVING

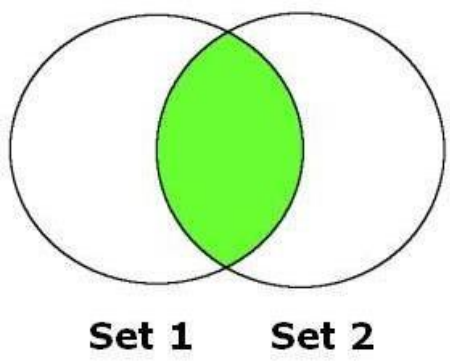
- **Clausula Having** – utilizada no GROUP BY
- A clausula Having permite definir condições para os grupos
- Permite definir condições ao agrupamento
- É utilizada em conjunto com a clausula Group By

HAVING - exemplo

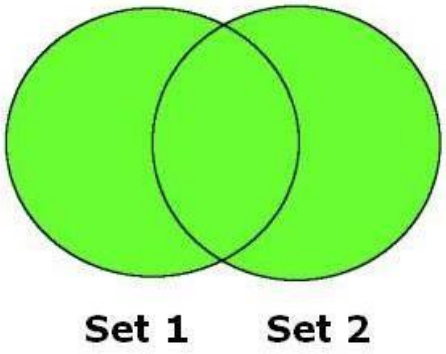
- SELECT <atributos>, <Função de agregação>
- FROM <TABELA>
- GROUP BY <atributos> HAVING <condições>

```
SELECT NumDep, COUNT(*), AVG(Salário)
FROM EMPREGADO
GROUP BY NumDep HAVING AVG(Salário)>1000;
```

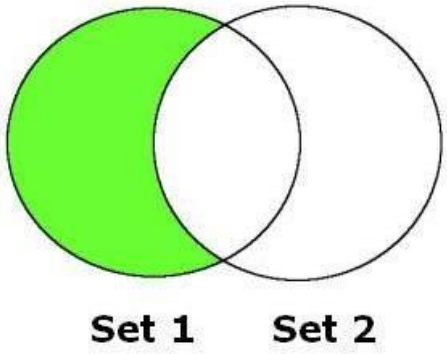
União, Interseção e Diferença



Intersect



Union
Union All



Minus

UNION

- O operador **UNION** permite combinar os resultados de duas ou mais instruções SELECT num único conjunto de resultados.
  - As linhas duplicadas são eliminadas, exceto se for utilizada a instrução UNION ALL

- Os conjuntos de resultados que são combinados utilizando UNION devem todos ter a mesma estrutura.

- Devem ter o mesmo número de colunas e as colunas de conjuntos de resultados correspondentes devem ter os tipos de dados compatíveis.

UNION

Estrutura:

```
SELECT * FROM Tabela1
UNION
SELECT * FROM Tabela2
```

Exemplo:

```
SELECT ID_Viatura FROM Viatura
UNION
SELECT Id_Viatura FROM Vendas
```

INTERSECT

- O operador **INTERSECT** permite efetuar uma interseção dos resultados de duas consultas SELECT

- O resultado desta interseção é o conjunto de todas as linhas que estão simultaneamente em ambas os resultados das duas consultas.
  - As linhas duplicadas são eliminadas, exceto se for utilizada a instrução INTERSECT ALL

- Ou seja, retorna a interseção das duas consultas

INTERSECT

Estrutura:

SELECT * FROM Tabela1

INTERSECT

SELECT * FROM Tabela2

Exemplo – Quais viaturas vendidas:

SELECT ID_Viatura FROM Viatura

INTERSECT

SELECT Id_Viatura FROM Vendas

EXCEPT (MINUS)

- O operador **EXCEPT** retorna todas as linhas presentes no resultado da Consulta*1*, mas que não estão presentes no resultado da Consulta*2*
  - As linhas duplicadas são eliminadas, exceto se for utilizada a instrução EXCEPT ALL
- Pode ser chamado de Diferença entre 2 consultas
- Em PL/SQL – Oracle e MySQL este operador designa-se **MINUS**

EXCEPT (MINUS)

Estrutura:

```
SELECT * FROM Tabela1
EXCEPT
SELECT * FROM Tabela2
```

Exemplo – Viaturas ainda não vendidas:

```
SELECT ID_Viatura FROM Viatura
EXCEPT
SELECT Id_Viatura FROM Vendas
```

LIMIT e ROWNUM

- O LIMIT em SQL Standard e o ROWNUM em PL/SQL da Oracle permitem limitar o número de resultados de uma consulta

- Exemplo:

```
SELECT * FROM Empregado LIMIT 5
```

```
SELECT * FROM Empregado WHERE
ROWNUM<=5
```

LIMIT e ROWNUM

- Exemplo: Selecionar os 3 salários mais altos dos empregados:

```
SELECT MAX(Salário)
FROM EMPREGADO
ORDER BY Salário
DESC  LIMIT 3;
```

LIMIT e ROWNUM - OFFSET

- Utilizando o OFFSET juntamente com o LIMIT
- O **OFFSET** permite definir o início das linhas que queremos obter.

- Exemplo: Queremos obter os 3º,4º e 5º salários mais altos, ou seja, saltamos os 2 primeiros:

```
SELECT
MAX(Salário)  FROM
EMPREGADO
ORDER BY Salário DESC
LIMIT 3 OFFSET 2;
```