

Banco de Dados



SQL Avançado – Parte 2

Prof. Rinaldo Lima
DEINFO/UFRPE

rinaldo.ufrpe@gmail.com



Conteúdo da Aula

- Gerando Agregações em SQL
- Tipos de Junções
- Subconsultas
 - Escalar, multiplas linhas e colunas
 - Correlatas
- Visões
- Índices






Revisando o Comando SELECT

```
SELECT <colunas>  
FROM <tabela>  
WHERE <condicoes>  
ORDER BY <colunas> [ASC | DESC];
```



SQL


DML
Funções de Agregação (ou
Agrupamento)



Funções sobre Conjuntos de Valores

- Máximos e Mínimos:
 - Seleciona o valor máximo ou o mínimo para um campo;
 - Uso das funções **MIN** e **MAX**:

```
SELECT [MIN | MAX ] (<coluna>);
```



Funções sobre Conjuntos de Valores

- Totalizando Colunas:
 - Somatório dos valores de um atributo;
 - Uso do comando **SUM**;
SELECT SUM (<coluna>);
- Calculando Médias:
 - Apresenta a média dos valores de um campo;
 - Uso do comando **AVG**;
SELECT AVG (<coluna>);

Funções sobre conjuntos de valores

- Crie consultas SQL para apresentar:
 - a) O maior salário entre os empregados.
 - b) O menor salário entre os empregados.
 - c) A média dos salários pagos aos empregados.
 - d) O total de salários pago aos empregados.

EMPREGADO									
PNOME	MINICIAL	UNOME	SSN	DATANASC	ENDERECO	SEXO	SALARIO	SUPERSSN	DNO

Resposta

a)

```
SELECT MAX (salario) FROM EMPREGADO;
```

b)

```
SELECT MIN (salario) FROM EMPREGADO;
```

c)

```
SELECT AVG (salario) FROM EMPREGADO;
```

d)

```
SELECT SUM (salario) FROM EMPREGADO;
```

Contagem dos Elementos de um Dataset

- Contando linhas de uma consulta:
 - Uso da função **COUNT**:
 SELECT COUNT (<coluna>)
 FROM ...
 WHERE ...;

Contando Elementos

- Crie uma consulta para :
 - a) Contar quantos empregados ganham acima de 25000

EMPREGADO

PNOME	MINICIAL	UNOME	<u>SSN</u>	DATANASC	ENDERECO	SEXO	SALARIO	SUPERSSN	DNO
-------	----------	-------	------------	----------	----------	------	---------	----------	-----





Resposta

```
SELECT COUNT (salario)
FROM EMPREGADO
WHERE
    salario > 25000;
```



Agrupando Dados

- Agrupando informações:
 - Uso do comando **GROUP BY**;
SELECT <colunas>
FROM <tabela>
WHERE <condicoes>
GROUP BY <colunas>;
 - Normalmente usado com outras funções como COUNT, AVG, SUM, etc.

Agrupando Dados

- Uso do **GROUP BY** : (cont.)
 - Qual a média dos salários dos empregados **por sexo**?

sexo	media_salario
M	37600
F	31000

```
SELECT sexo, AVG(salario) as media_salario
FROM EMPREGADO
GROUP BY sexo;
```

EMPREGADO

PNOME	MINICIAL	UNOME	SSN	DATANASC	ENDereco	SEXO	SALARIO	SUPERSSN	DNO
EMPREGADO									
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Born	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

Operações de Agrupamento

- Crie uma consulta SQL para exibir quantos dependentes cada empregado têm

DEPENDENTE				
ESSN	NOME_DEPENDENTE	SEXO	DATANASC	PARENTESCO
333445555	Alice	F	1986-04-05	FILHA
333445555	Theodore	M	1983-10-25	FILHO
333445555	Joy	F	1958-05-03	CÔNJUGE
987654321	Abner	M	1942-02-28	CÔNJUGE
123456789	Michael	M	1988-01-04	FILHO
123456789	Alice	F	1988-12-30	FILHA
123456789	Elizabeth	F	1967-05-05	CÔNJUGE

Resposta

```
SELECT essn, COUNT(*) as tot_dep
FROM DEPENDENTE
GROUP BY essn
```

ESSN	tot_dep
123456789	3
333445555	3
987654321	1

DEPENDENTE				
ESSN	NOME_DEPENDENTE	SEXO	DATANASC	PARENTESCO
333445555	Alice	F	1986-04-05	FILHA
333445555	Theodore	M	1983-10-25	FILHO
333445555	Joy	F	1958-05-03	CÔNJUGE
987654321	Abner	M	1942-02-28	CÔNJUGE
123456789	Michael	M	1988-01-04	FILHO
123456789	Alice	F	1988-12-30	FILHA
123456789	Elizabeth	F	1967-05-05	CÔNJUGE

Restringindo o Resultado da Consulta

- Podemos agrupar informações de forma condicional, isto é, selecionar entre os grupos
 - Uso do comando **HAVING**;


```
SELECT <colunas>
FROM <tabela>
WHERE <condicoes>
GROUP BY <colunas>
HAVING <condicoes>;
```


Restringindo o resultado da consulta

Liste os essn dos empregados que têm mais de um dependente, e o total de dependentes por empregado.

DEPENDENTE				
ESSN	NOME_DEPENDENTE	SEXO	DATANASC	PARENTESCO
333445555	Alice	F	1986-04-05	FILHA
333445555	Theodore	M	1983-10-25	FILHO
333445555	Joy	F	1958-05-03	CÔNJUGE
987654321	Abner	M	1942-02-28	CÔNJUGE
123456789	Michael	M	1988-01-04	FILHO
123456789	Alice	F	1988-12-30	FILHA
123456789	Elizabeth	F	1967-05-05	CÔNJUGE

Resposta

```
SELECT essn, COUNT(*) as tot_dep
FROM DEPENDENTE
GROUP BY essn
HAVING tot_dep > 1;
```

ESSN	tot_dep
333445555	3
123456789	3

DEPENDENTE				
ESSN	NOME_DEPENDENTE	SEXO	DATANASC	PARENTESCO
333445555	Alice	F	1986-04-05	FILHA
333445555	Theodore	M	1983-10-25	FILHO
333445555	Joy	F	1958-05-03	CÔNJUGE
987654321	Abner	M	1942-02-28	CÔNJUGE
123456789	Michael	M	1988-01-04	FILHO
123456789	Alice	F	1988-12-30	FILHA
123456789	Elizabeth	F	1967-05-05	CÔNJUGE

Comparando as consultas: group by com having

```
SELECT essn, COUNT(*) as tot_dep
FROM DEPENDENTE
GROUP BY essn
```

ESSN	tot_dep
123456789	3
3334455555	3
987654321	1

```
SELECT essn, COUNT(*) as tot_dep
FROM DEPENDENTE
GROUP BY essn
HAVING tot_dep > 1;
```

ESSN	tot_dep
3334455555	3
123456789	3

Outro Exemplo: Group by / Having

EXEMPLO:

Para cada projeto no qual mais de 2 empregados trabalham, encontre o número do projeto, o nome do projeto e o número de empregados que trabalham no projeto

```
SELECT Pnumber, Pname, COUNT(*)
FROM PROJECT P, WORKS_ON W
WHERE
  P.Pnumber = W.Pno
GROUP BY Pnumber, Pname
HAVING COUNT(*) > 2;
```



**Etapas da
consulta com
GROUP BY e
HAVING**

(b)

Pname	Pnumber	...	Essn	Pno	Hours
ProductX	1		123456789	1	32.5
ProductX	1		453453453	1	20.0
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
ProductZ	3		666884444	3	40.0
ProductZ	3		333445555	3	10.0
Computerization	10	...	333445555	10	10.0
Computerization	10		999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

After applying the WHERE clause but before applying HAVING

These groups are not selected by the HAVING condition of Q26.

Pname	Pnumber	...	Essn	Pno	Hours
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
Computerization	10		333445555	10	10.0
Computerization	10	...	999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

After applying the HAVING clause condition

Pname	Count (*)
ProductY	3
Computerization	3
Reorganization	3
Newbenefits	3

Result of Q26
(Pnumber not shown)



**Etapas da
consulta com
GROUP BY e
HAVING**

(b)

Pname	Pnumber	...	Essn	Pno	Hours
ProductX	1		123456789	1	32.5
ProductX	1		453453453	1	20.0
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
ProductZ	3		666884444	3	40.0
ProductZ	3		333445555	3	10.0
Computerization	10	...	333445555	10	10.0
Computerization	10		999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

After applying the WHERE clause but before applying HAVING

These groups are not selected by the HAVING condition of Q26.

Pname	Pnumber	...	Essn	Pno	Hours
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
Computerization	10		333445555	10	10.0
Computerization	10	...	999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

After applying the HAVING clause condition

Pname	Count (*)
ProductY	3
Computerization	3
Reorganization	3
Newbenefits	3

Result of Q26
(Pnumber not shown)



Junções

- INNER JOIN ou apenas JOIN
 - São incluídas na resposta somente as linhas que satisfazem à condição do JOIN;
- CROSS JOIN (produto cartesiano):
 - Incluídas cada uma das combinações de todas as linhas entre as tabelas;
- OUTER JOIN:
 - Incluídas linhas que satisfazem à condição de JOIN e as linhas restantes de uma das tabelas do JOIN.



Junções

- Sintaxe:

```
SELECT  <tabela>.<campo>,
        <tabela>.<campo>...
```

```
FROM
```

```
<tabela> [tipo de JOIN] <tabela>
```

```
ON <condicao>;
```

INNER JOIN

- Crie uma consulta que liste os nomes dos empregados e os nomes dos seus dependentes

EMPREGADO

PNAME	MINICIAL	UNOME	<u>SSN</u>	DATANASC	ENDereco	SEXO	SALARIO	SUPERSSN	DNO
-------	----------	-------	------------	----------	----------	------	---------	----------	-----

DEPENDENTE

ESSN	NOME_DEPENDENTE	SEXO	DATANASC	PARENTESCO
------	-----------------	------	----------	------------



Resposta

EMPREGADO

PNAME	MINICIAL	UNOME	<u>SSN</u>	DATANASC	ENDereco	SEXO	SALARIO	SUPERSSN	DNO
-------	----------	-------	------------	----------	----------	------	---------	----------	-----

DEPENDENTE

<u>ESSN</u>	NOME_DEPENDENTE	SEXO	DATANASC	PARENTESCO
-------------	-----------------	------	----------	------------

```
SELECT e.pnome, d.nome_dependente
FROM
  empregado e JOIN dependente d
  ON e.ssn = d.essn;
```



CROSS JOIN

- Exemplo:
 - Juntar dependentes com empregados. Mesmo que não sejam seus respectivos dependentes. (produto cartesiano)

```
SELECT  
    empregado.pnome,  
    dependente.nome_dependente  
FROM empregado CROSS JOIN dependente;
```



OUTER JOIN

- Usado para forçar a saída de todas as linhas de uma das tabelas num JOIN, mesmo quando não existe uma correspondência entre as PK e FK das tabelas envolvidas na consulta
- Muito usado na geração de relatórios



TIPOS DE OUTER JOIN

- LEFT OUTER JOIN:
 - Incluídas todas as linhas da primeira tabela citada na expressão do JOIN;
- RIGHT OUTER JOIN:
 - Incluídas todas as linhas da segunda tabela citada na expressão do JOIN;
- FULL OUTER JOIN:
 - Incluídas todas as linhas que não satisfazem a expressão do JOIN, tanto da primeira quanto da segunda tabela.



OUTER JOIN

- Exemplo:
 - Listar os empregados que tem ou não tem dependentes

```
SELECT
    empregado.pnome, dependente.nome_dependente
FROM
    empregado LEFT [OUTER] JOIN dependent
ON empregado.ssn = dependente.essn;
```

Encadeando Tabelas nos Joins

- E quando se tem várias tabelas para fazer a junção?

EMPREGADO									
PNAME	MINICIAL	UNOME	<u>SSN</u>	DATANASC	ENDERECO	SEXO	SALARIO	SUPERSSN	DNO

DEPARTAMENTO			
DNAME	DNUMERO	GERSSN	GERDATAINICIO

DEPTO_LOCALIZACOES	
DNUMERO	DLOCALIZACAO

PROJETO			
PJNOME	PNUMERO	PLOCALIZACAO	DNUM

TRABALHA_EM		
ESSN	PNO	HORAS

DEPENDENTE				
ESSN	NOME_DEPENDENTE	SEXO	DATANASC	PARENTESCO

Encadeando Tabelas nos Joins

- Calcular a média de salário dos empregados que trabalham em Houston?

EMPREGADO									
PNAME	MINICIAL	UNOME	<u>SSN</u>	DATANASC	ENDERECO	SEXO	SALARIO	SUPERSSN	DNO

PROJETO			
PJNOME	PNUMERO	PLOCALIZACAO	DNUM

TRABALHA_EM		
ESSN	PNO	HORAS



Resposta

EMPREGADO

PNOME	MINICIAL	UNOME	<u>SSN</u>	DATANASC	ENDERECO	SEXO	SALARIO	SUPERSSN	DNO
-------	----------	-------	------------	----------	----------	------	---------	----------	-----

PROJETO

PJNOME	PNUMERO	PLOCALIZACAO	DNUM
--------	---------	--------------	------

TRABALHA_EM

ESSN	PNO	HORAS
------	-----	-------

```
SELECT AVG(E.salario) AS SALARIO
FROM
  EMPREGADO E JOIN TRABALHA_EM T
    ON E.SSN = T.ESSN
JOIN PROJETO P
    ON T.PNO = P.PNUMERO
WHERE P.PLOCALIZACAO = 'Houston';
```

Subconsultas e seus Tipos

- Retornam uma linha e uma coluna (**escalar**)
 - A consulta aninhada retorna um único valor ou mais de um campo em uma mesma tupla.
- Retornam múltiplas linhas:
 - A consulta aninhada retorna várias tuplas.
- Subconsultas **correlatas**:
 - A avaliação da consulta **filha** (aninhada ou interna) é dependente da avaliação da consulta **pai** (ou externa)

Cláusulas para Subqueries com Uma Coluna

■ ANY:

- Busca por pelo menos um valor no resultado da consulta aninhada;
- *Exemplo:* **Selecionar os funcionários que ganham mais do que pelo menos 1 funcionário do departamento 5.**

```
SELECT E.nome
FROM EMPREGADO E
WHERE e.salario > ANY (SELECT salario
                           FROM EMPREGADO
                           WHERE DNO = 5);
```

Consulta Aninhada ou Subconsulta

Subconsulta -

- Selecionar os nomes dos funcionários que trabalham em projetos menos horas do que pelo menos um empregado alocado no projeto 'infra'.

EMPREGADO

PNOME	MINICIAL	UNOME	<u>SSN</u>	DATANASC	ENDERECO	SEXO	SALARIO	SUPERSSN	DNO
-------	----------	-------	------------	----------	----------	------	---------	----------	-----

PROJETO

PJNOME	<u>PNUMERO</u>	PLOCALIZACAO	DNUM
--------	----------------	--------------	------

TRABALHA_EM

<u>ESSN</u>	<u>PNO</u>	HORAS
-------------	------------	-------



Resposta

“Selecionar os nomes dos funcionários que trabalham em projetos menos horas do que pelo menos um empregado alocado no projeto ‘infra’ ”

```
SELECT e.pnome
FROM (EMPREGADO e JOIN TRABALHA_EM t
      ON e.ssn = t.essn )
WHERE t.horas < ANY (
      SELECT horas
      FROM
        TRABALHA_EM te JOIN PROJETO p
        ON te.pno = p.pnumero)
      WHERE p.pjnome = 'infra' );
```

Cláusulas para Subqueries com Uma Coluna

■ **ALL:**

- Busca por todos os valores no resultado da consulta aninhada.
- Exemplo : **Selecionar funcionários que ganham mais do que todos os funcionários do departamento 5.**

```
SELECT pnome
FROM EMPREGADO
WHERE salario > ALL
      (SELECT salario
      FROM EMPREGADO
      WHERE DNO = 5);
```

EMPREGADO

PNO	MINICIAL	UNOME	SSN	DATANASC	ENDERECO	SEXO	SALARIO	SUPERSSN	DNO
-----	----------	-------	-----	----------	----------	------	---------	----------	-----

Subconsulta Correlata

- Listar os nomes dos empregados que recebem menos do que a média de salário paga ao seu departamento?

EMPREGADO

PNome	MINICIAL	UNOME	<u>SSN</u>	DATANASC	ENDereco	SEXO	SALARIO	SUPERSSN	DNO
-------	----------	-------	------------	----------	----------	------	---------	----------	-----



Resposta

```
SELECT e.pnome, e.dno
FROM EMPREGADO e
WHERE e.salario <
      (SELECT AVG(s.salario)
       FROM EMPREGADO s
       WHERE s.dno = e.dno);
```

A consulta interna é iterada **N** vezes em função do valor da coluna fornecida pela consulta externa

Para cada departamento de um empregado (e.dno) da consulta externa, será calculado a média de salários dos empregados deste departamento



União entre Conjuntos

- Uso do comando **UNION**:
 - Empregados que trabalham nos projetos 1 ou 20:

```
(SELECT essn, FROM TRABALHA_EM WHERE pno =1)
```

UNION

```
(SELECT essn, FROM TRABALHA_EM WHERE pno =20);
```



Interseção entre Conjuntos

- Uso do comando **INTERSECT**:
 - Empregados que trabalham nos projetos 3 e 20:

```
(SELECT essn, FROM TRABALHA_EM WHERE pno =3)
```

INTERSECT

```
(SELECT essn, FROM TRABALHA_EM WHERE pno =20);
```



Diferença entre Conjuntos

- Uso do comando **EXCEPT**:
 - Empregados que não têm dependentes:

```
(SELECT essn, FROM EMPREGADO)
```

EXCEPT

```
(SELECT essn, FROM DEPENDENTE);
```



SQL

Criação de Índices

Índices (1/3)

- Estruturas que otimizam o armazenamento dos dados e a busca por eles:
 - Ao invés de se realizar uma busca de forma sequencial, a busca é feita pelo índice:

Mas cuidado com o desempenho!!!

- Cada vez que um registro é inserido ou atualizado a tabela de índices também é atualizada.
- Definição de índices requer:
 - Experiência por parte do administrador;
 - Verificação experimental na maioria dos casos.

Índices (2/3)

- Sintaxe:

- **CREATE UNIQUE INDEX** <índice>
USING <metodo_acesso >
 ON <tabela>
 (coluna)

BTREE, HASH

O método default é BTREE



Índices (3/3)

- Criando índice:

```
CREATE UNIQUE INDEX depto_indice  
ON DEPARTAMENTO (dnumero );
```

- Removendo o índice:

```
DROP INDEX depto_indice ON DEPARTAMENTO;
```



Visões de dados



View (Visão)

- Alternativa para visualizarmos dados específicos ou derivados de uma ou mais tabelas:
 - São virtuais, não ocupam espaço físico;
 - Podem ser criadas e removidas como tabelas.



Operações de Views



CRIAÇÃO

CREATE VIEW <nome> (<colunas>) AS <consulta>;

- <nome> - nome da visão;
- (<colunas>) – colunas da visão, entre vírgulas;
- <consulta> - consulta da qual serão extraídas as colunas da visão.



REMOÇÃO

DROP VIEW <nome>;



Criando Visões

- Visão de empregados para o RH:

```
CREATE VIEW EMPREGADO_RH (nome, sobrenome,
ssn, nascimento, sexo, salario) AS
SELECT pnome, unome, ssn, datanasc, sexo, salario
FROM EMPREGADO;
```

EMPREGADO

PNOME	MINICIAL	UNOME	<u>SSN</u>	DATANASC	ENDERECO	SEXO	SALARIO	SUPERSSN	DNO
-------	----------	-------	------------	----------	----------	------	---------	----------	-----



Criando Visões

- Visão de empregados experientes por departamento:

```
CREATE VIEW empregado_depto_experiente
(nome, sobrenome, ssn, nascimento, departamento) AS
SELECT pnome, unome, ssn, datanasc, dno
FROM EMPREGADO
WHERE datanasc < '31/12/1990';
```

EMPREGADO

PNOME	MINICIAL	UNOME	<u>SSN</u>	DATANASC	ENDERECO	SEXO	SALARIO	SUPERSSN	DNO
-------	----------	-------	------------	----------	----------	------	---------	----------	-----



Resumo do comando SELECT

SELECT<attribute and function list>
FROM<table list>
[**WHERE**<condition>]
[**GROUP BY**<grouping attribute(s)>]
[**HAVING** <group condition>]
[**ORDER BY**<attribute list>];