ADMINISTRAÇÃO DE BANCO DE DADOS

Linguagem SQL – Agregações e conjuntos

Prof. Ms. Helton Souza Lima

SECTRAS

ADS

Funções de agregação

- SQL fornece 5 funções embutidas
- **COUNT**: retorna o número de tuplas ou valores especificados em uma consulta (query)
- **SUM**: retorna a soma dos valores de uma coluna
- **AVG**: retorna a média dos valores de uma coluna
- **MAX**: retorna o maior valor de uma coluna
- **MIN**: retorna o menor valor de uma coluna
- Essas funções são usadas após a cláusula SELECT ou após cláusula HAVING (será vista depois)

Funções de agregação

- Qual o gasto total com salários dos empregados?
- **SELECT SUM**(salario) **FROM** EMPREGADOS
- Qual o maior salário recebido entre os empregados e o nome do respectivo empregado?
- **SELECT** e.nome, e.salario **FROM** empregados e where e.salario = (**SELECT MAX**(f.salario) **FROM** empregados f)

EMPREGADOS				
Matrícula	**Nome**	**Endereço**	**Função**	**Salário**
100	Ana	R. Pedro I, 12, A. Branco	Secretária	500,00
250	Pedro	R. J. Silva, 24, Liberdade	Engenheiro	2500,00
108	André	R. Itália, 33, B. Nações	Técnico	950,00
210	Paulo	R. Pará, 98, B. Estados	Engenheiro	2500,00

Funções de agregação

- Quantos empregados existem?
- **SELECT COUNT**(*) **FROM** EMPREGADOS
- Quantos salários distintos existem na empresa?
- **SELECT COUNT**(distinct salario) **FROM** empregados

EMPREGADOS				
Matrícula	**Nome**	**Endereço**	**Função**	**Salário**
100	Ana	R. Pedro I, 12, A. Branco	Secretária	500,00
250	Pedro	R. J. Silva, 24, Liberdade	Engenheiro	2500,00
108	André	R. Itália, 33, B. Nações	Técnico	950,00
210	Paulo	R. Pará, 98, B. Estados	Engenheiro	2500,00

Funções de agregação

- Obtenha uma lista com o nome do empregado e a quantidade de dependentes

```
select nome , (select count(*) from dependentes d where d.mat_titular = e.matricula) as qtd
from empregados e
```

NOME	QTD
Ana	2
Pedro	1
André	1
Paulo	0

EMPREGADOS				
Matrícula	**Nome**	**Endereço**	**Função**	**Salário**
100	Ana	R. Pedro I, 12, A. Branco	Secretária	500,00
250	Pedro	R. J. Silva, 24, Liberdade	Engenheiro	2500,00
108	André	R. Itália, 33, B. Nações	Técnico	950,00
210	Paulo	R. Pará, 98, B. Estados	Engenheiro	2500,00

DEPENDENTES			
Código	**Nome**	**Matrícula Titular**	**Relação**
1	Sophia	100	Filha
2	Henrique	100	Filho
3	Isabela	250	Filha
4	Beatriz	108	Filha

Funções de agregação

- Obter o nome dos empregados que tenham 2 ou mais dependentes

```
select e.nome from empregados e where (
        select count(*) from dependentes d where d.mat_titular = e.matricula
) >= 2
```

NOME
Ana

EMPREGADOS				
Matrícula	**Nome**	**Endereço**	**Função**	**Salário**
100	Ana	R. Pedro I, 12, A. Branco	Secretária	500,00
250	Pedro	R. J. Silva, 24, Liberdade	Engenheiro	2500,00
108	André	R. Itália, 33, B. Nações	Técnico	950,00
210	Paulo	R. Pará, 98, B. Estados	Engenheiro	2500,00

DEPENDENTES			
Código	**Nome**	**Matrícula Titular**	**Relação**
1	Sophia	100	Filha
2	Henrique	100	Filho
3	Isabela	250	Filha
4	Beatriz	108	Filha

Cláusula GROUP BY

- Cláusula utilizada para agrupar tuplas com determinados valores iguais
- Ao realizar o agrupamento, não é possível selecionar tuplas individualmente
- A montagem da cláusula **SELECT** precisa acompanhar os dados utilizados no **GROUP BY**
- O agrupamento e as funções que podem ser utilizadas são aplicadas após a cláusula WHERE da consulta

Cláusula GROUP BY

- Indique quantos empregados existem para cada função da empresa
  - select funcao, count(*) as qtd from empregados group by funcao
- Indique qual a média salarial para cada função da empresa
  - select funcao, avg(salario) as media_salarial from empregados group by funcao

EMPREGADOS				
Matrícula	**Nome**	**Endereço**	**Função**	**Salário**
100	Ana	R. Pedro I, 12, A. Branco	Secretária	500,00
250	Pedro	R. J. Silva, 24, Liberdade	Engenheiro	2500,00
108	André	R. Itália, 33, B. Nações	Técnico	950,00
210	Paulo	R. Pará, 98, B. Estados	Engenheiro	2500,00

Cláusula GROUP BY

- Pode ser utilizada em conjunto com a cláusula HAVING
- Indique quantos empregados existem para cada função da empresa desde que existam pelo menos dois empregados
    - select funcao, count(*) as qtd from empregados group by funcao having count(*) >= 2

EMPREGADOS				
Matrícula	**Nome**	**Endereço**	**Função**	**Salário**
100	Ana	R. Pedro I, 12, A. Branco	Secretária	500,00
250	Pedro	R. J. Silva, 24, Liberdade	Engenheiro	2500,00
108	André	R. Itália, 33, B. Nações	Técnico	950,00
210	Paulo	R. Pará, 98, B. Estados	Engenheiro	2500,00

Cláusula ORDER BY

- Cláusula utilizada para ordenar os resultados de acordo com um campo ou uma sequência de campos
- Pode ser usado para ordenação ascendente (ASC) ou descendente (DESC)
- O padrão é ASC
- ORDER BY atua nos campos definidos após o SELECT

Cláusula ORDER BY

- select * from empregados order by funcao
- select * from empregados order by nome
- select * from empregados order by funcao, nome

EMPREGADOS				
Matrícula	**Nome**	**Endereço**	**Função**	**Salário**
100	Ana	R. Pedro I, 12, A. Branco	Secretária	500,00
250	Pedro	R. J. Silva, 24, Liberdade	Engenheiro	2500,00
108	André	R. Itália, 33, B. Nações	Técnico	950,00
210	Paulo	R. Pará, 98, B. Estados	Engenheiro	2500,00

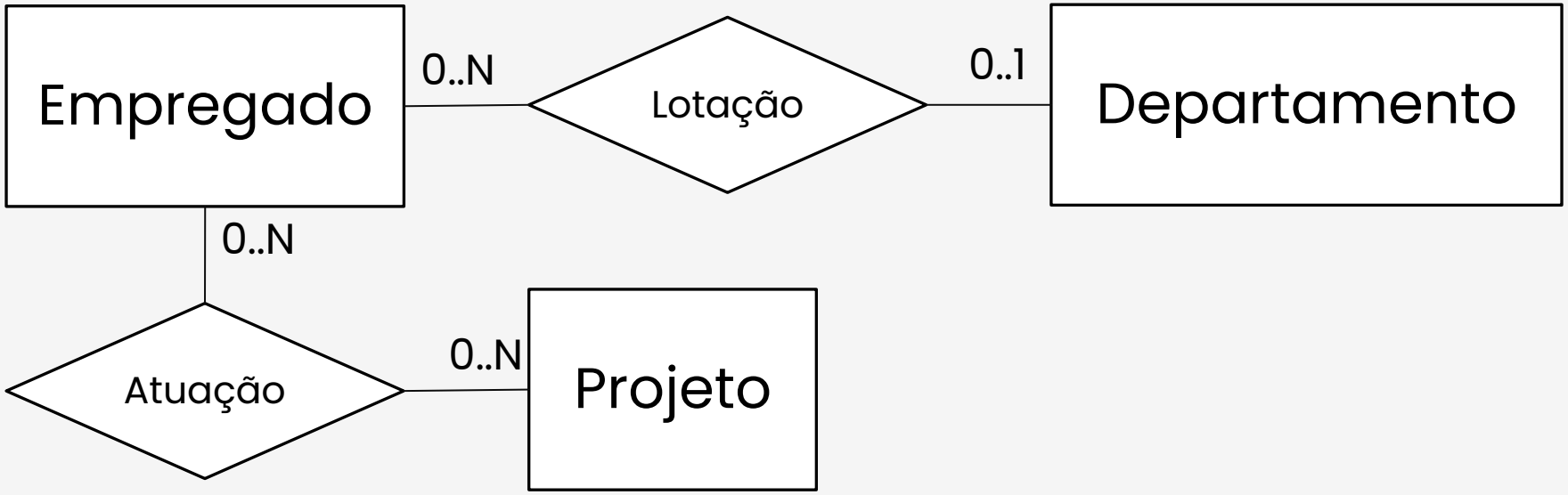
Álgebra relacional

- Álgebra relacional é o conjunto básico de **operações** para o modelo relacional (ELMASRI, NAVATHE)
- Uma sequência de operações da álgebra relacional forma uma **expressão** da álgebra relacional
- O resultado de uma expressão da álgebra relacional é uma **nova** relação
- Divididas em dois grupos
    - Operações de **Seleção**, **Projeção** e **Junção**
    - Operações de **conjunto**
    - Funções de **agregação**

Operações de conjunto

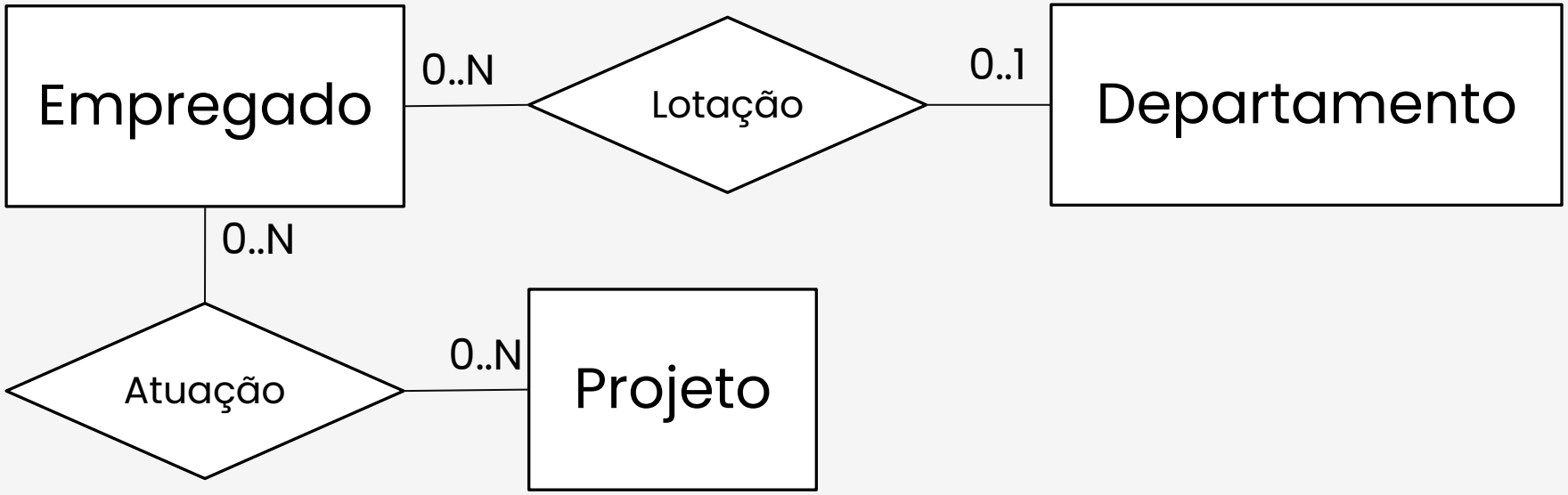
- As operações de conjunto **UNION**, **INTERSECT** e **EXCEPT** correspondem às operações da álgebra tradicional $\cup$, $\cap$, $-$, respectivamente.
- Cada uma dessas operações elimina automaticamente duplicatas.
- Para manter as duplicatas use: **UNION ALL**, **INTERSECT ALL** e **EXCEPT ALL**
- Suponha que uma tupla ocorre *m* vezes em R e *n* vezes em S, então, ela ocorre:
  - **m + n** vezes em R **union all** S
  - **min(m , n)** vezes em R **intersect all** S
  - **max(0, m-n)** vezes em R **except all** S

Operações de conjunto - UNION



- Informe o nome de todos os empregados que atuaram no Departamento de Inovação ou que participaram do projeto Hackathon

Operações de conjunto



- **SELECT** emp.nome **FROM** EMPREGADOS emp, DEPARTAMENTOS dep **WHERE** emp.depto = dep.código AND dep.nome = ‘Inovação’
- UNION
- **SELECT** emp.nome **FROM** EMPREGADOS emp, ATUACOES_PROJETOS att, PROJETOS prj **WHERE** emp.matricula = att.empregado AND att.projeto = prj.codigo AND prj.nome = ‘Hackathon’

Operações de conjunto

SELECT emp.nome **FROM** empregados emp, departamentos dep **WHERE** emp.depto = dep.codigo **AND** dep.nome = ‘Inovação’

UNION

SELECT emp.nome **FROM** empregados emp, atuacoes_projetos att, projetos prj **WHERE** emp.matricula = att.empregado **AND** att.projeto = prj.codigo **AND** prj.nome = ‘Hackathon’

- Reescrevendo a consulta com a cláusula IN

SELECT distinct emp.nome **FROM** empregados emp **WHERE** emp.depto **IN** (**SELECT** dep.codigo **FROM** departamentos dep **WHERE** dep.nome = ‘Inovação’)

OR

emp.matricula **IN** (**SELECT** att.empregado **FROM** atuacoes_projetos att, projetos prj **WHERE** att.projeto = prj.codigo **AND** prj.nome = ‘Hackathon’)

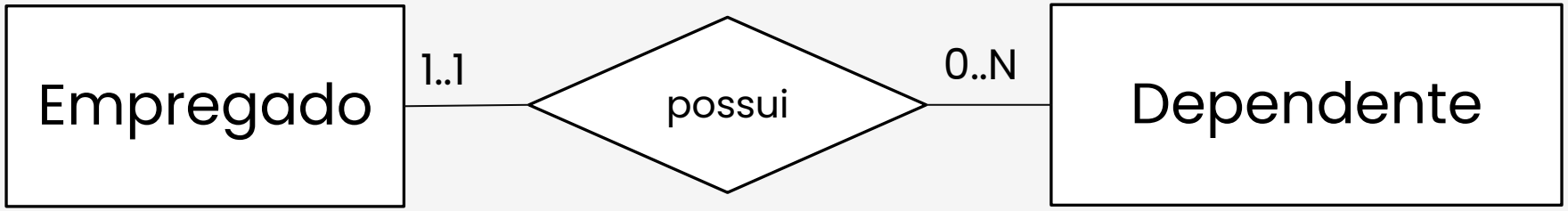
Operações de conjunto

- Mesma idéia para INTERSECT e EXCEPT
- No Oracle EXCEPT é **MINUS**

Comando EXISTS

- Usada para verificar se o resultado de uma consulta aninhada é vazia ou não. É sempre usado em conjunto com uma consulta aninhada
- Resulta no valor TRUE se a subconsulta (ou subquery) é não vazio

Comando EXISTS



EMPREGADOS				
Matrícula	**Nome**	**Endereço**	**Função**	**Salário**
100	Ana	R. Pedro I, 12, A. Branco	Secretária	500,00
250	Pedro	R. J. Silva, 24, Liberdade	Engenheiro	2500,00
108	André	R. Itália, 33, B. Nações	Técnico	950,00
210	Paulo	R. Pará, 98, B. Estados	Engenheiro	2500,00

```
CREATE TABLE EMPREGADOS ( matricula number(10), nome varchar2(50) not null, endereco varchar2(250), funcao varchar2(50), salario number(10,2), constraint pk_emp primary key (matricula) )

INSERT INTO EMPREGADOS(matricula, nome, endereco, funcao, salario) values (100, 'Ana','R. Pedro I, 12, A. Branco','Secretária', 500) ;
INSERT INTO EMPREGADOS(matricula, nome, endereco, funcao, salario) values (250, 'Pedro','R. J. Silva, 24, Liberdade','Engenheiro', 1500);
INSERT INTO EMPREGADOS(matricula, nome, endereco, funcao, salario) values (108, 'André','R. Itália, 33, B. Nações','Técnico', 950);
INSERT INTO EMPREGADOS(matricula, nome, endereco, funcao, salario) values (210, 'Paulo','R. Pará, 98, B. Estados','Engenheiro', 1810);
INSERT INTO EMPREGADOS(matricula, nome, endereco, funcao, salario) values (105, 'Sônia','R. Oliveira, 76, A. Branco','Engenheiro', 2500)
```

DEPENDENTES			
Código	**Nome**	**Matrícula Titular**	**Relação**
1	Sophia	100	Filha
2	Henrique	100	Filho
3	Isabela	250	Filha
4	Beatriz	108	Filha

```
CREATE TABLE dependentes ( codigo number(10), nome varchar2(100), mat_titular number(10), relacao varchar2(30), constraint pk_dep primary key (codigo), constraint fk_titular foreign key (mat_titular) references empregados(matricula) )

INSERT INTO DEPENDENTES(codigo, nome, mat_titular, relacao) values (1, 'Sophia',100,'Filha');
INSERT INTO DEPENDENTES(codigo, nome, mat_titular, relacao) values (2, 'Henrique',100,'Filho');
INSERT INTO DEPENDENTES(codigo, nome, mat_titular, relacao) values (3, 'Isabela',250,'Filha');
INSERT INTO DEPENDENTES(codigo, nome, mat_titular, relacao) values (4, 'Beatriz',108,'Filha')
```

Comando EXISTS

EMPREGADOS				
Matrícula	**Nome**	**Endereço**	**Função**	**Salário**
100	Ana	R. Pedro I, 12, A. Branco	Secretária	500,00
250	Pedro	R. J. Silva, 24, Liberdade	Engenheiro	2500,00
108	André	R. Itália, 33, B. Nações	Técnico	950,00
210	Paulo	R. Pará, 98, B. Estados	Engenheiro	2500,00

DEPENDENTES			
Código	**Nome**	**Matrícula Titular**	**Relação**
1	Sophia	100	Filha
2	Henrique	100	Filho
3	Isabela	250	Filha
4	Beatriz	108	Filha

- Obter o nome de todos os empregados que possuem pelo menos um dependente

Comando EXISTS

EMPREGADOS				
Matrícula	**Nome**	**Endereço**	**Função**	**Salário**
100	Ana	R. Pedro I, 12, A. Branco	Secretária	500,00
250	Pedro	R. J. Silva, 24, Liberdade	Engenheiro	2500,00
108	André	R. Itália, 33, B. Nações	Técnico	950,00
210	Paulo	R. Pará, 98, B. Estados	Engenheiro	2500,00

DEPENDENTES			
Código	**Nome**	**Matrícula Titular**	**Relação**
1	Sophia	100	Filha
2	Henrique	100	Filho
3	Isabela	250	Filha
4	Beatriz	108	Filha

```
select e.nome from empregados e where exists (
    select * from dependentes d where e.matricula = d.mat_titular
)
```

- Resultado:

NOME
Ana
André
Pedro

- Agora faça: obter o nome dos empregados que não possuem dependentes.
  - Usar NOT EXISTS

Comando EXISTS

EMPREGADOS				
Matrícula	**Nome**	**Endereço**	**Função**	**Salário**
100	Ana	R. Pedro I, 12, A. Branco	Secretária	500,00
250	Pedro	R. J. Silva, 24, Liberdade	Engenheiro	2500,00
108	André	R. Itália, 33, B. Nações	Técnico	950,00
210	Paulo	R. Pará, 98, B. Estados	Engenheiro	2500,00

DEPENDENTES			
Código	**Nome**	**Matrícula Titular**	**Relação**
1	Sophia	100	Filha
2	Henrique	100	Filho
3	Isabela	250	Filha
4	Beatriz	108	Filha

```
select e.nome from empregados e where exists (
    select * from dependentes d where e.matricula = d.mat_titular
)
```

- Resultado:

NOME
Ana
André
Pedro

- Agora faça: obter o nome dos empregados que não possuem dependentes.
  - Usar NOT EXISTS