Bancos de Dados

CIC0097

Prof. Pedro Garcia Freitas

https://pedrogarcia.gitlab.io/

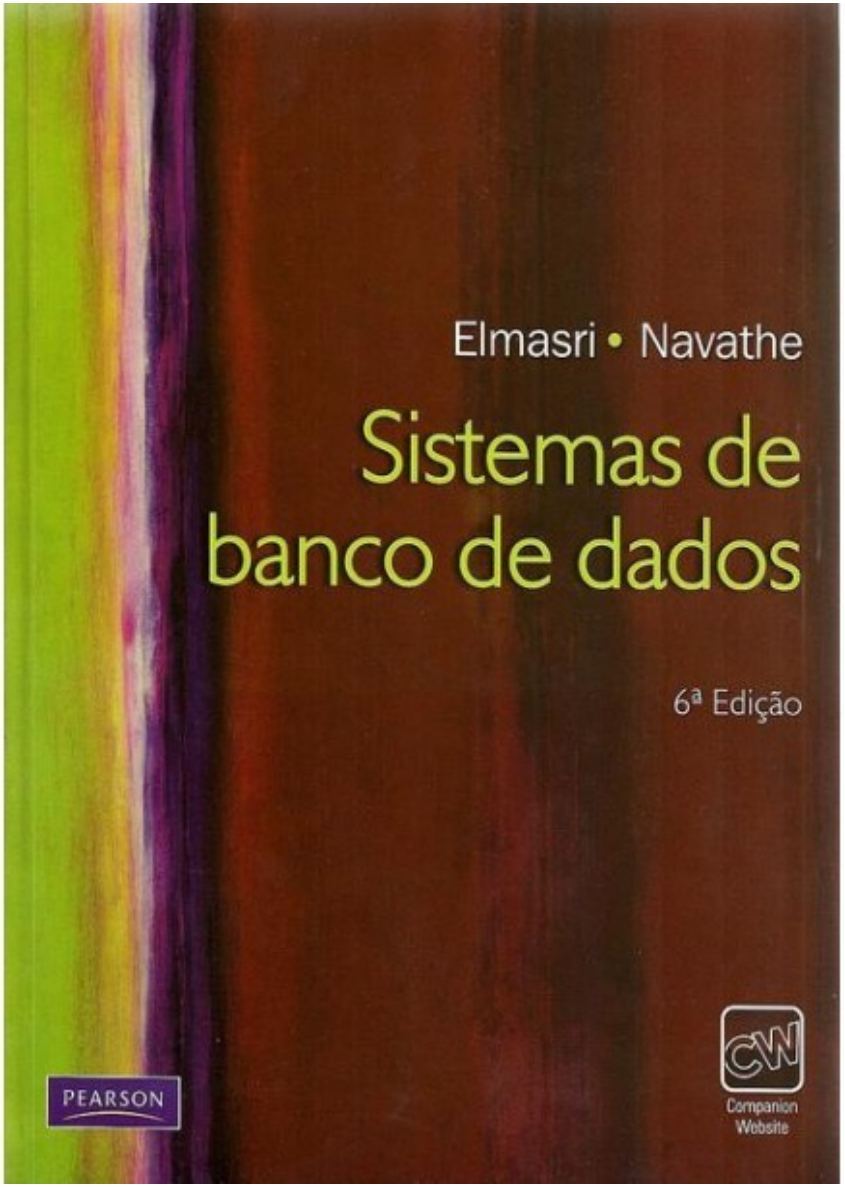
pedro.garcia@unb.br

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciências da Computação

Módulo 15

Linguagem de Consulta Estruturada Parte 4: Subconsultas e Junções (*joins*)

CIC0097/2023.1
T1/T2

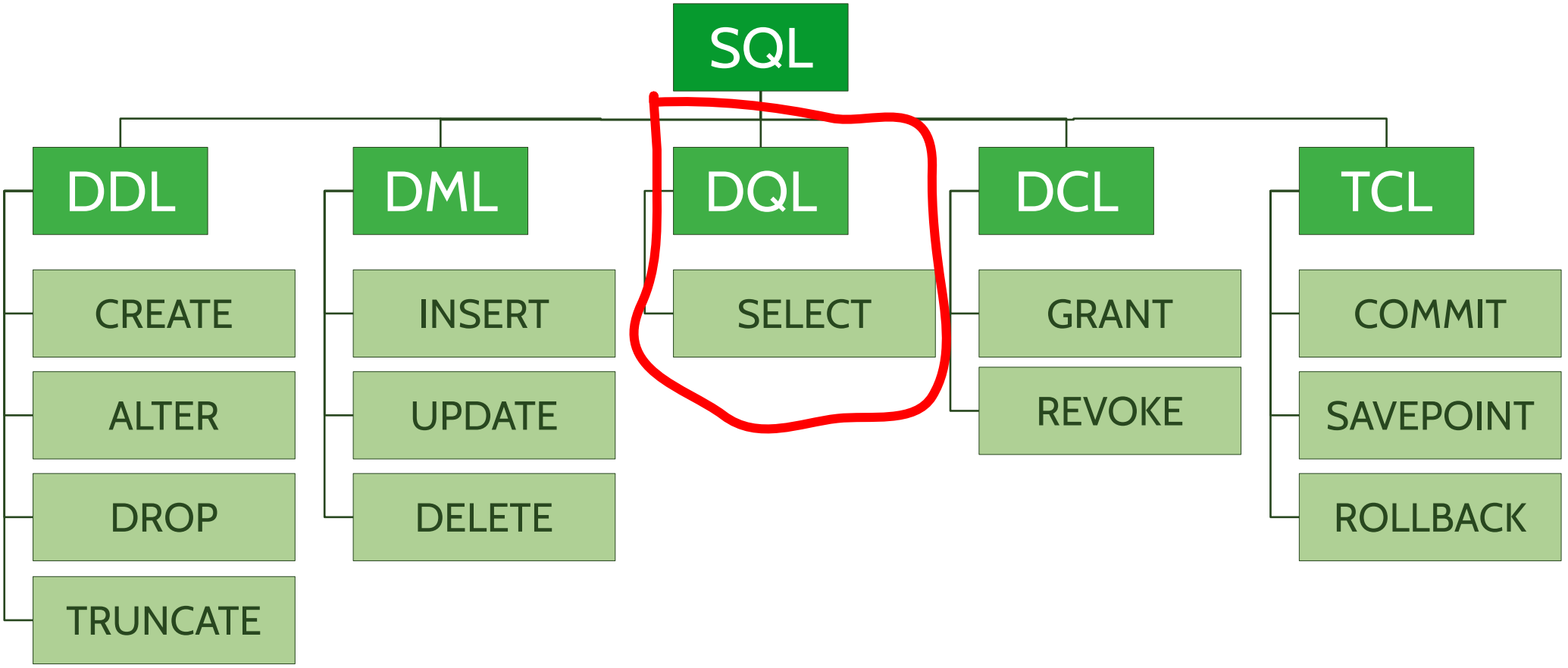


Esta aula se baseia no Capítulo 5 (SQL básica) do Elmasri e Navathe (6ª Edição).

1. Objetivos

Esta aula continua a apresentação da linguagem SQL a partir dos comandos DQL (subset da DML na perspectiva do Elmasri e Navathe) e trata das subconsultas.

2. Consultas de SQL mais complexas



Fonte: https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/

2. Consultas de SQL mais complexas

- Tal qual vimos na álgebra relacional, a busca por alguma informações pode envolver o uso do resultado de outra consulta.
- Esses casos são resolvidos por **subconsultas** (também chamadas de **consultas aninhadas**)

2. Consultas de SQL mais complexas

Subconsultas são blocos **SELECT–FROM–WHERE** especificados dentro de uma cláusula **WHERE** de outro bloco.

```
SELECT <lista de atributos>
FROM <lista de relações>
WHERE (SELECT <lista de atributos>
       FROM <lista de relações>
       WHERE <condição>);
```

2. Consultas de SQL mais complexas

Exemplo 1: Considerando o esquema textual abaixo, recupere todos os servidores que possuem dois ou mais dependentes.

- **SERVIDOR(cpf,nome,sobrenome,endereco,dt_nasc,salario, sexo,fk_cpf_supervisor,fk_dnumero)**
- **DEPENDENTE(fk_s_cpf,nome,dt_nasc,sexo,relacionamento)**
- **DEPARTAMENTO(numero, nome, fk_cpf_gerente, dtinicio)**
- **LOCALIZACOES(fk_dnumero, localizacao)**
- **PROJETO(numero, nome, localizacao, dnumero)**
- **TRABALHA_EM(fk_pnumero,fk_s_cpf,horas)**

2. Consultas de SQL mais complexas

Exemplo 1: Considerando o esquema textual abaixo, recupere todos os servidores que possuem dois ou mais dependentes.

- **SERVIDOR(cpf,nome,sobrenome,endereco,dt_nasc,salario, sexo,fk_cpf_supervisor,fk_dnumero)**
- **DEPENDENTE(fk_s_cpf,nome,dt_nasc,sexo,relacionamento)**
- **DEPARTAMENTO(numero, nome, fk_cpf_gerente, dtinicio)**
- **LOCALIZACOES(fk_dnumero, localizacao)**
- **PROJETO(numero, nome, localizacao, dnumero)**
- **TRABALHA_EM(fk_pnumero,fk_s_cpf,horas)**

2. Consultas de SQL mais complexas

Exemplo 1: Considerando o esquema textual abaixo, recupere o nome de todos os servidores que possuem dois ou mais dependentes.

```
SELECT nome
FROM Servidores AS S
WHERE (SELECT COUNT(*)
       FROM Dependente AS D
       WHERE S.cpf=D.fk_s_cpf) >= 2;
```

2. Consultas de SQL mais complexas

Exemplo 1: Consi
abaixo, recupere
que possuem do

A consulta retorna uma única tupla com um valor inteiro (referente à contagem de cada linha de S).

```
SELECT nome
FROM Servidores AS S
WHERE (SELECT COUNT(*)
       FROM Dependente AS D
       WHERE S.cpf=D.fk_s_cpf) >= 2;
```

2. Consultas de SQL mais complexas

cpf	nome	...
3	Claudia	...
4	Jorge	...
5	Moacir	...
7	Caio	...

fk_s_cpf	nome	...
3	Amanda	
4	Fabio	
5	Alan	
3	Henrique	
4	Pedro	
3	Claudia	

COUNT para **S.cpf=3**

3

>= 2 **True**

2. Consultas de SQL mais complexas

cpf	nome	...
3	Claudia	...
4	Jorge	...
5	Moacir	...
7	Caio	...

fk_s_cpf	nome	...
3	Amanda	
4	Fabio	
5	Alan	
3	Henrique	
4	Pedro	
3	Claudia	

`COUNT` para `S.cpf=4`

2

`>= 2` `True`

2. Consultas de SQL mais complexas

cpf	nome	...
3	Claudia	...
4	Jorge	...
5	Moacir	...
7	Caio	...

fk_s_cpf	nome	...
3	Amanda	
4	Fabio	
5	Alan	
3	Henrique	
4	Pedro	
3	Claudia	

COUNT para **S.cpf=5**

1

>= 2 **False**

2. Consultas de SQL mais complexas

cpf	nome	...
3	Claudia	...
4	Jorge	...
5	Moacir	...
7	Caio	...

fk_s_cpf	nome	...
3	Amanda	
4	Fabio	
5	Alan	
3	Henrique	
4	Pedro	
3	Claudia	

`COUNT` para `S.cpf=7`

0

`>= 2` `False`

2. Consultas de SQL mais complexas

cpf	nome
3	Claudia
4	Jorge
5	Moacir
7	Caio

COUNT para **S . cpf=3**

3

>= 2 **True**

COUNT para **S . cpf=4**

2

>= 2 **True**

COUNT para **S . cpf=5**

1

>= 2 **False**

COUNT para **S . cpf=7**

0

>= 2 **False**

2. Consultas de SQL mais complexas

```
SELECT nome
FROM Servidores AS S
WHERE (SELECT COUNT(*)
       FROM Dependente AS D
       WHERE S.cpf=D.fk_s_cpf) >= 2;
```

nome
Claudia
Jorge

2. Consultas de SQL mais complexas

Exemplo **2**: Considerando o esquema textual abaixo, recupere o nome dos servidores que possuem um dependente do mesmo sexo dele.

- **SERVIDOR(cpf,nome,sobrenome,endereco,dt_nasc,salario, sexo,fk_cpf_supervisor,fk_dnumero)**
- **DEPENDENTE(fk_s_cpf,nome,dt_nasc,sexo,relacionamento)**
- **DEPARTAMENTO(numero, nome, fk_cpf_gerente, dtinicio)**
- **LOCALIZACOES(fk_dnumero, localizacao)**
- **PROJETO(numero, nome, localizacao, dnumero)**
- **TRABALHA_EM(fk_pnumero,fk_s_cpf,horas)**

2. Consultas de SQL mais complexas

Exemplo 2: Considerando o esquema textual abaixo, recupere o nome dos servidores que possuem um dependente do mesmo sexo dele.

```
SELECT nome
FROM Servidores AS S
WHERE S.cpf IN(
   SELECT D.fk_s_cpf
   FROM Dependente AS D
   WHERE S.cpf=D.fk_s_cpf AND S.sexo=D.sexo
);
```

2. Consultas de SQL mais complexas

cpf	nome	sexo	...
3	Claudia	F	...
4	Jorge	M	...
5	Moacir	M	...
7	Caio	M	...

fk_s_cpf	nome	sexo	...
3	Amanda	F	...
4	Fabio	M	...
5	Alan	M	...
3	Henrique	M	...
4	Pedro	M	...
3	Claudia	F	...

2. Consultas de SQL mais complexas

cpf	nome	sexo	...
3	Claudia	F	...
4	Jorge	M	...
5	Moacir	M	...
7	Caio	M	...

fk_s_cpf	nome	sexo	...
3	Amanda	F	...
4	Fabio	M	...
5	Alan	M	...
3	Henrique	M	...
4	Pedro	M	...
3	Claudia	F	...

2. Consultas de SQL mais complexas

cpf	nome	sexo	...
3	Claudia	F	...
4	Jorge	M	...
5	Moacir	M	...
7	Caio	M	...

fk_s_cpf	nome	sexo	...
3	Amanda	F	...
4	Fabio	M	...
5	Alan	M	...
3	Henrique	M	...
4	Pedro	M	...
3	Claudia	F	...

2. Consultas de SQL mais complexas

```
SELECT nome
FROM Servidores AS S
WHERE S.cpf IN(
   SELECT D.fk_s_cpf
   FROM Dependente AS D
   WHERE S.cpf=D.fk_s_cpf AND S.sexo=D.sexo);
```

nome
Claudia
Jorge
Moacir

2. Consultas de SQL mais complexas

Exemplo **3**: Considerando o esquema textual abaixo, recupere o **nome dos servidores** que possuem um dependente com o **mesmo nome e mesmo sexo** que o dele.

- **SERVIDOR(cpf,nome,sobrenome,endereco,dt_nasc,salario, sexo,fk_cpf_supervisor,fk_dnumero)**
- **DEPENDENTE(fk_s_cpf,nome,dt_nasc,sexo,relacionamento)**
- **DEPARTAMENTO(numero, nome, fk_cpf_gerente, dtinicio)**
- **LOCALIZACOES(fk_dnumero, localizacao)**
- **PROJETO(numero, nome, localizacao, dnumero)**
- **TRABALHA_EM(fk_pnumero,fk_s_cpf,horas)**

2. Consultas de SQL mais complexas

Exemplo **3**: Considerando o esquema textual abaixo, recupere o **nome dos servidores** que possuem um dependente com o **mesmo nome e mesmo sexo** que o dele.

```
SELECT nome
FROM Servidores AS S
WHERE EXISTS (
   SELECT * FROM Dependente AS D
   WHERE S.cpf=D.fk_s_cpf
      AND S.nome=D.nome AND S.sexo=D.sexo);
```

2. Consultas de SQL mais complexas

cpf	nome	...
3	Claudia	...
4	Jorge	...
5	Moacir	...
7	Caio	...

fk_s_cpf	nome	...
3	Amanda	
4	Fabio	
5	Alan	
3	Henrique	
4	Pedro	
3	Claudia	

EXISTS para **S.cpf=3**

TRUE

EXISTS para **S.cpf=4**

FALSE

EXISTS para **S.cpf=5**

FALSE

2. Consultas de SQL mais complexas

Exemplo **4**: Recupere todos os servidores que **não possuem** dependentes.

2. Consultas de SQL mais complexas

Exemplo **4**: Recupere todos os servidores que **não possuem** dependentes.

```
SELECT nome
FROM Servidores AS S
WHERE NOT EXISTS (
   SELECT * FROM Dependente AS D
   WHERE S.cpf=D.fk_s_cpf);
```

2. Consultas de SQL mais complexas

cpf	nome	sexo	...
3	Claudia	F	...
4	Jorge	M	...
5	Moacir	M	...
7	Caio	M	...

fk_s_cpf	nome	sexo	...
3	Amanda	F	...
4	Fabio	M	...
5	Alan	M	...
3	Henrique	M	...
4	Pedro	M	...
3	Claudia	F	...

`NOT EXISTS` para `S.cpf=3 : Falso`

2. Consultas de SQL mais complexas

cpf	nome	sexo	...
3	Claudia	F	...
4	Jorge	M	...
5	Moacir	M	...
7	Caio	M	...

fk_s_cpf	nome	sexo	...
3	Amanda	F	...
4	Fabio	M	...
5	Alan	M	...
3	Henrique	M	...
4	Pedro	M	...
3	Claudia	F	...

`NOT EXISTS` para `S.cpf=4` : `Falso`

2. Consultas de SQL mais complexas

cpf	nome	sexo	...
3	Claudia	F	...
4	Jorge	M	...
5	Moacir	M	...
7	Caio	M	...

fk_s_cpf	nome	sexo	...
3	Amanda	F	...
4	Fabio	M	...
5	Alan	M	...
3	Henrique	M	...
4	Pedro	M	...
3	Claudia	F	...

`NOT EXISTS` para `S.cpf=5 : Falso`

2. Consultas de SQL mais complexas

cpf	nome	sexo	...
3	Claudia	F	...
4	Jorge	M	...
5	Moacir	M	...
7	Caio	M	...

fk_s_cpf	nome	sexo	...
3	Amanda	F	...
4	Fabio	M	...
5	Alan	M	...
3	Henrique	M	...
4	Pedro	M	...
3	Claudia	F	...

NOT EXISTS para **S.cpf=5 : True**

Não existe linhas na subconsulta (em Dependentes) em que a condição exista!

3. Comando UPDATE com consultas

Subconsultas podem ser usadas dentro de uma cláusula **WHERE** de um bloco **UPDATE**.

```
UPDATE <nome da tabela>
SET <lista de atributos>
WHERE (SELECT <lista de atributos>
       FROM <lista de relações>
       WHERE <condição>);
```

3. Comando UPDATE com consultas

Exemplo **5**: Forneça o aumento de 10% do salário para todos os servidores cujo departamento seja gerenciado pelo servidor cujo CPF é 3.

- **SERVIDOR(cpf,nome,sobrenome,endereco,dt_nasc,salario, sexo,fk_cpf_supervisor,fk_dnumero)**
- **DEPENDENTE(fk_s_cpf,nome,dt_nasc,sexo,relacionamento)**
- **DEPARTAMENTO(numero, nome, fk_cpf_gerente, dtinicio)**
- **LOCALIZACOES(fk_dnumero, localizacao)**
- **PROJETO(numero, nome, localizacao, dnumero)**
- **TRABALHA_EM(fk_pnumero,fk_s_cpf,horas)**

3. Comando UPDATE com consultas

Exemplo **5**: Forneça o aumento de 10% do salário para todos os servidores cujo departamento seja gerenciado pelo servidor cujo CPF é 3.

- **SERVIDOR(cpf,nome,sobrenome,endereco,dt_nasc,salario, sexo,fk_cpf_supervisor,fk_dnumero)**
- DEPENDENTE(fk_s_cpf,nome,dt_nasc,sexo,relacionamento)
- **DEPARTAMENTO(numero, nome, fk_cpf_gerente, dtinicio)**
- LOCALIZACOES(fk_dnumero, localizacao)
- PROJETO(numero, nome, localizacao, dnumero)
- TRABALHA_EM(fk_pnumero,fk_s_cpf,horas)

3. Comando UPDATE com consultas

Exemplo **5**: Forneça o aumento de 10% do salário para todos os servidores cujo departamento seja gerenciado pelo servidor cujo CPF é 3.

- **SERVIDOR(cpf,nome,sobrenome,endereco,dt_nasc,salario, sexo,fk_cpf_supervisor,fk_dnumero)**
- DEPENDENTE(fk_s_cpf,nome,dt_nasc,sexo,relacionamento)
- **DEPARTAMENTO(numero, nome, fk_cpf_gerente, dtinicio)**
- LOCALIZACOES(fk_dnumero, localizacao)
- PROJETO(numero, nome, localizacao, dnumero)
- TRABALHA_EM(fk_pnumero,fk_s_cpf,horas)

3. Comando UPDATE com consultas

Exemplo **5**: Forneça o aumento de 10% do salário para todos os servidores cujo departamento seja gerenciado pelo servidor cujo CPF é 3.

- **SERVIDOR(cpf,nome,sobrenome,endereco,dt_nasc,salario, sexo,fk_cpf_supervisor,fk_dnumero)**
- **DEPENDENTE(fk_s_cpf,nome,dt_nasc,sexo,relacionamento)**
- **DEPARTAMENTO(numero, nome, fk_cpf_gerente, dtinicio)**
- **LOCALIZACOES(fk_dnumero, localizacao)**
- **PROJETO(numero, nome, localizacao, dnumero)**
- **TRABALHA_EM(fk_pnumero,fk_s_cpf,horas)**

3. Comando UPDATE com consultas

```
UPDATE Servidor
SET salario=1.1*salario
WHERE fk_dnumero IN (
   SELECT numero
   FROM Departamento
   WHERE fk_cpf_gerente=3);
```

4. Comando DELETE com consultas

De maneira semelhante, linhas de uma tabela podem ser removidas via **WHERE** e uma subconsulta.

```
DELETE <nome da tabela>
WHERE (SELECT <lista de atributos>
       FROM <lista de relações>
       WHERE <condição>);
```

4. Comando DELETE com consultas

```
DELETE Servidor
WHERE fk_dnumero IN (
   SELECT numero
   FROM Departamento
   WHERE fk_cpf_gerente=3);
```

5. O comando SQL `JOIN`

O **`JOIN`** na linguagem SQL funciona como a junção da algebra relacional. Sua função é combinar linhas de diferentes tabelas de acordo com as condições existentes entre as colunas dessas tabelas.

5. O comando SQL JOIN

Ao todo, a linguagem SQL padrão define cinco tipos diferentes de cláusula **JOIN**. São elas:

- **INNER JOIN**;
- **RIGHT JOIN**;
- **LEFT JOIN**;
- **FULL JOIN**;
- **CROSS JOIN**.

5. O comando SQL JOIN

Para ilustrar o funcionamento do exemplos a seguir, considere as tabelas **A** e **B**, onde ambas possuem apenas a chave primária **id** e o atributo nome.

```
CREATE TABLE A(
   id    INT NOT NULL,
   nome VARCHAR(50) NULL,
   PRIMARY KEY(id)
);
```

```
CREATE TABLE B(
   id    INT NOT NULL,
   nome VARCHAR(50) NULL,
   PRIMARY KEY(id)
);
```

5. O comando SQL JOIN

Tuplas de teste:

```
INSERT INTO A VALUES (1, 'Fernanda');
INSERT INTO A VALUES (2, 'Mafalda');
INSERT INTO A VALUES (3, 'Luiz');
INSERT INTO A VALUES (4, 'Fernando');

INSERT INTO B(id, nome) VALUES (1, 'Carlos');
INSERT INTO B(id, nome) VALUES (2, 'Manoel');
INSERT INTO B(id, nome) VALUES (3, 'Luiz');
INSERT INTO B(id, nome) VALUES (4, 'Fernando');
```

5. O comando SQL JOIN

A título de curiosidade, considere a listagem de todas linhas das tabelas A e B:

```
SELECT * FROM A WHERE 1
```

id	nome
1	Fernanda
2	Mafalda
3	Luiz
4	Fernando

```
SELECT * FROM B
```

id	nome
1	Carlos
2	Manoel
3	Luiz
4	Fernando

5. O comando SQL JOIN

5.1. CROSS JOIN

```
SELECT * FROM A CROSS JOIN B;
SELECT * FROM A,B;
SELECT * FROM A,B WHERE 1;
```

id	nome	id	nome
1	Fernanda	1	Carlos
2	Mafalda	1	Carlos
3	Luiz	1	Carlos
4	Fernando	1	Carlos
1	Fernanda	2	Manoel
2	Mafalda	2	Manoel
3	Luiz	2	Manoel
4	Fernando	2	Manoel
1	Fernanda	3	Luiz
2	Mafalda	3	Luiz
3	Luiz	3	Luiz
4	Fernando	3	Luiz
1	Fernanda	4	Fernando
2	Mafalda	4	Fernando
3	Luiz	4	Fernando
4	Fernando	4	Fernando

5. O comando SQL JOIN

5.1. CROSS JOIN

```
SELECT * FROM A CROSS JOIN B;
SELECT * FROM A,B;
SELECT * FROM A,B WHERE 1;
```

Com o comando SQL **CROSS JOIN** é possível fazer um produto cartesiano entre as tabelas. Isso significa que para cada linha da tabela esquerda ele vai retornar todas as linhas da tabela direita, ou vice-versa.

id	nome	id	nome
1	Fernanda	1	Carlos
2	Mafalda	1	Carlos
3	Luiz	1	Carlos
4	Fernando	1	Carlos
1	Fernanda	2	Manoel
2	Mafalda	2	Manoel
3	Luiz	2	Manoel
4	Fernando	2	Manoel
1	Fernanda	3	Luiz
2	Mafalda	3	Luiz
3	Luiz	3	Luiz
4	Fernando	3	Luiz
1	Fernanda	4	Fernando
2	Mafalda	4	Fernando
3	Luiz	4	Fernando
4	Fernando	4	Fernando

5. O comando SQL JOIN

5.2. INNER JOIN

```
SELECT * FROM A INNER JOIN B;
```

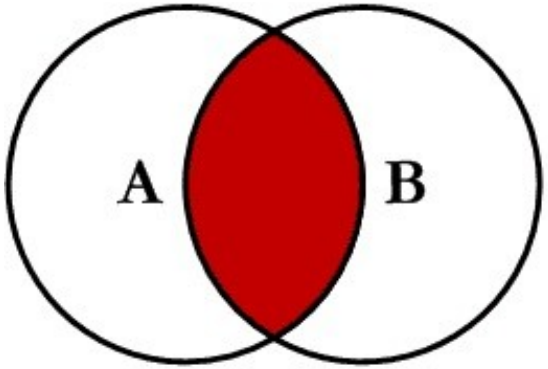
id	nome	id	nome
1	Fernanda	1	Carlos
2	Mafalda	1	Carlos
3	Luiz	1	Carlos
4	Fernando	1	Carlos
1	Fernanda	2	Manoel
2	Mafalda	2	Manoel
3	Luiz	2	Manoel
4	Fernando	2	Manoel
1	Fernanda	3	Luiz
2	Mafalda	3	Luiz
3	Luiz	3	Luiz
4	Fernando	3	Luiz
1	Fernanda	4	Fernando
2	Mafalda	4	Fernando
3	Luiz	4	Fernando
4	Fernando	4	Fernando

5. O comando SQL JOIN

5.2. INNER JOIN

```
SELECT * FROM A INNER JOIN B ON A.nome=B.nome;
```

id	nome	id	nome
3	Luiz	3	Luiz
4	Fernando	4	Fernando

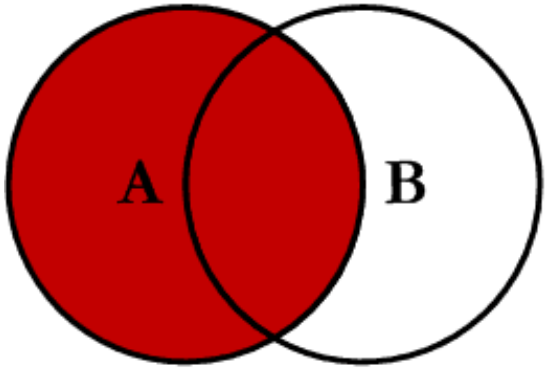


5. O comando SQL JOIN

5.3. LEFT JOIN

```
SELECT * FROM A LEFT JOIN B ON A.nome=B.nome;
```

id	nome	id	nome
3	Luiz	3	Luiz
4	Fernando	4	Fernando
1	Fernanda	*NULL*	*NULL*
2	Mafalda	*NULL*	*NULL*

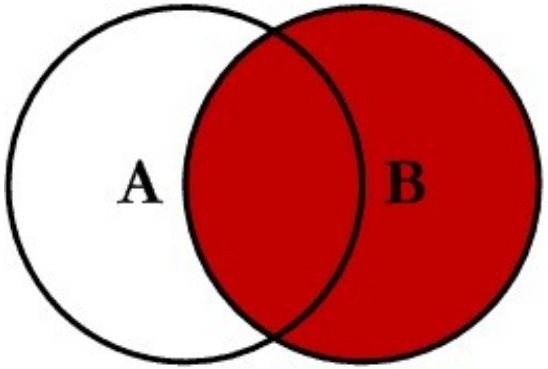


5. O comando SQL JOIN

5.4. RIGHT JOIN

```
SELECT * FROM A RIGHT JOIN B ON A.nome=B.nome;
```

id	nome	id	nome
3	Luiz	3	Luiz
4	Fernando	4	Fernando
NULL	*NULL*	1	Carlos
NULL	*NULL*	2	Manoel



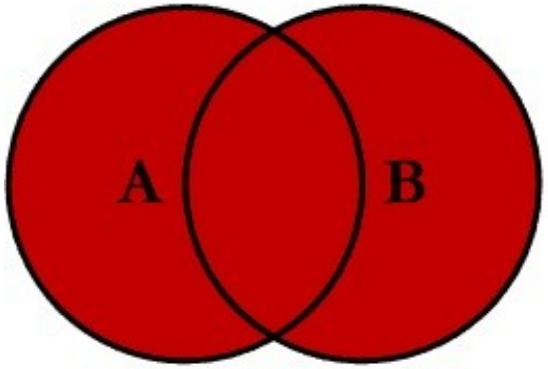
5. O comando SQL JOIN

5.5. OUTER JOIN

O **OUTER JOIN** (também conhecido por **FULL OUTER JOIN** ou **FULL JOIN**) tem como resultado **todos** os registros que estão na tabela A e todos os registros da tabela B.

```
SELECT * FROM A FULL OUTER JOIN B ON A.nome=B.nome;
```

id	nome	id	nome
3	Luiz	3	Luiz
4	Fernando	4	Fernando
NULL	*NULL*	1	Carlos
NULL	*NULL*	2	Manoel
1	Fernanda	*NULL*	*NULL*
2	Mafalda	*NULL*	*NULL*



OBS: O MySQL/MariaDB não define a operação com as palavras chaves **OUTER JOIN**. Para se realizar essa operação deve-se fazer a **UNION** do **LEFT JOIN** com o **RIGHT JOIN**!

5. C

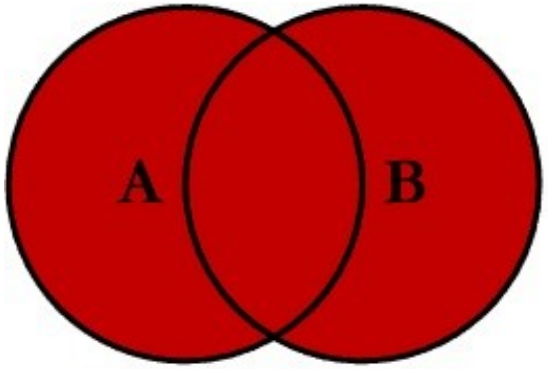
5.5.

O ou

JOIN ou **FULL** ...no resultado **todos** os registros que estão na tabe... os os registros da tabela B.

```
SELECT * FROM A F... OUTER JOIN B ON A.nome=B.nome;
```

id	nome	id	nome
3	Luiz	3	Luiz
4	Fernando	4	Fernando
NULL	*NULL*	1	Carlos
NULL	*NULL*	2	Manoel
1	Fernanda	*NULL*	*NULL*
2	Mafalda	*NULL*	*NULL*



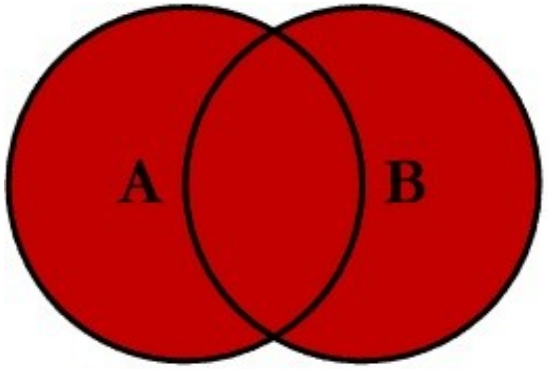
```
SELECT * FROM A RIGHT JOIN B ON A.nome=B.nome
UNION
SELECT * FROM A LEFT JOIN B ON A.nome=B.nome
```

https://dev.mysql.com/doc/refman/8.0/en/outer-join-simplification.html

JOIN ou FULL JOIN) tem como res registros que estão na tabela A e todos os regi bela B.

```
SELECT * FROM A FULL OUTER JO ON A.nome=B.nome;
```

id	nome	id	nome
3	Luiz	3	Luiz
4	Fernando	4	Fernando
NULL	*NULL*	1	Carlos
NULL	*NULL*	2	Manoel
1	Fernanda	*NULL*	*NULL*
2	Mafalda	*NULL*	*NULL*

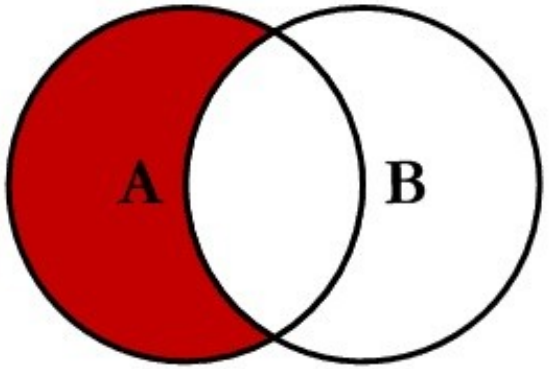


5. O comando SQL JOIN

5.6. LEFT EXCLUDING JOIN

```
SELECT * FROM A LEFT JOIN B ON A.nome=B.nome
WHERE B.nome IS NULL
```

id	nome	id	nome
1	Fernanda	*NULL*	*NULL*
2	Mafalda	*NULL*	*NULL*

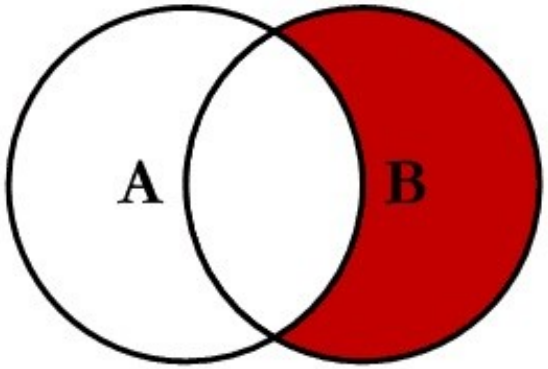


5. O comando SQL JOIN

5.7. RIGHT EXCLUDING JOIN

```
SELECT * FROM A RIGHT JOIN B ON A.nome=B.nome
WHERE A.nome IS NULL
```

id	nome	id	nome
NULL	*NULL*	1	Carlos
NULL	*NULL*	2	Manoel

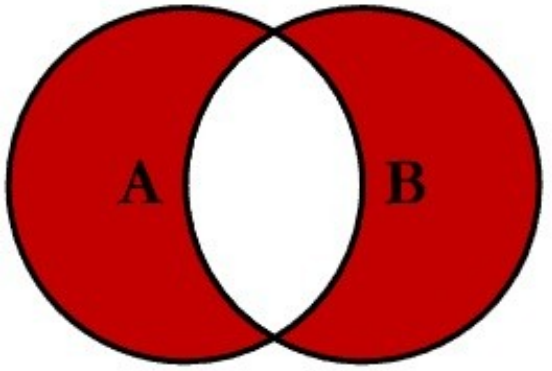


5. O comando SQL JOIN

5.8. OUTER EXCLUDING JOIN

```
SELECT * FROM A FULL OUTER JOIN B ON A.nome=B.nome
WHERE A.nome IS NULL OR B.nome IS NULL
```

id	nome	id	nome
1	Fernanda	*NULL*	*NULL*
2	Mafalda	*NULL*	*NULL*
NULL	*NULL*	1	Carlos
NULL	*NULL*	2	Manoel



https://dev.mysql.com/doc/refman/8.0/en/outer-join-simplification.html

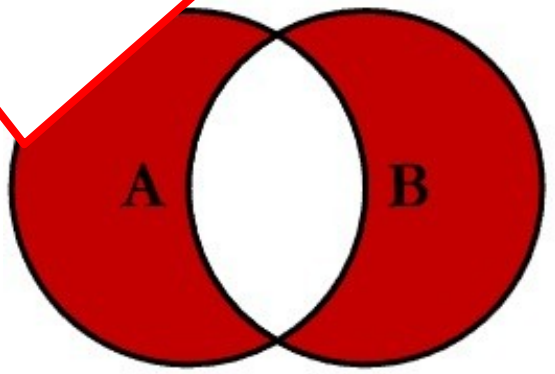
5

5.

Semelhante ao caso do **`FULL OUTER JOIN`**, o MySQL também não define **`OUTER EXCLUDING JOIN`**.

```
SE
WH
```

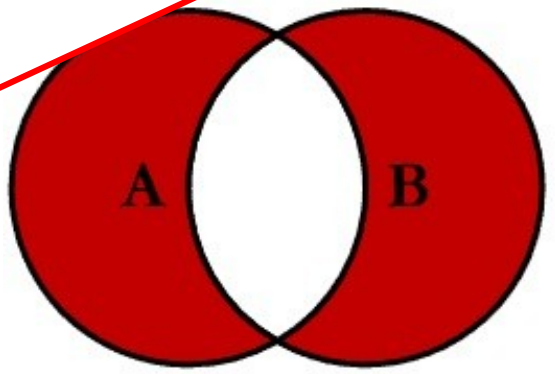
id	nome	id	nome
1	Fernanda	*NULL*	*NULL*
2	Mafalda	*NULL*	*NULL*
NULL	*NULL*	1	Carlos
NULL	*NULL*	2	Manoel



https://dev.mysql.com/doc/refman/8.0/en/outer-join-simplification.html

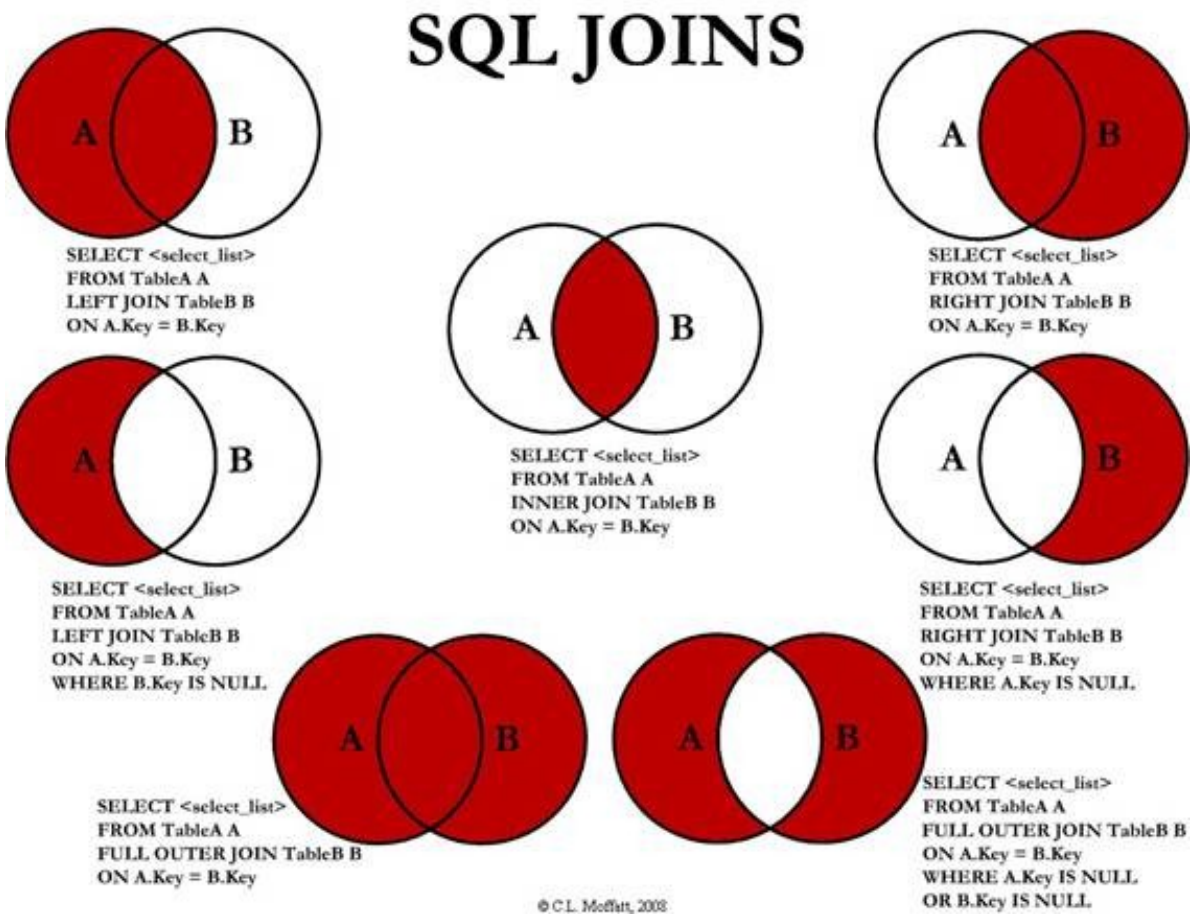
```
SELECT * FROM A LEFT JOIN B ON A.nome=B.nome
WHERE B.nome IS NULL
UNION
SELECT * FROM A RIGHT JOIN B ON A.nome=B.nome
WHERE A.nome IS NULL
```

id	nome	id	nome
1	Fernanda	*NULL*	*NULL*
2	Mafalda	*NULL*	*NULL*
NULL	*NULL*	1	Carlos
NULL	*NULL*	2	Manoel



https://dev.mysql.com/doc/refman/8.0/en/outer-join-simplification.html

5. O comando SQL JOIN



6. Observação

- Para apagar todo conteúdo de uma tabela:
  - `DELETE FROM A;`
  - `TRUNCATE TABLE B;`

6. Observaçã

- Para apagar tod

  - **DELETE FROM A;**
  - **TRUNCATE TABLE B;**

- O comando **DELETE** é usado para excluir registros específicos de uma tabela.
- O comando **TRUNCATE** é usado para excluir todos os dados da tabela.

6. Observaçã

- Para apagar tod

  - **DELETE FROM A;**
  - **TRUNCATE TABLE B;**

- **DELETE**: É um comando DML.
- **TRUNCATE** É um comando DDL.

6.

- O comando **DELETE** adquire o bloqueio em cada registro excluído; portanto, requer mais bloqueios e recursos.
- O comando **TRUNCATE** requer menos bloqueios e recursos antes de excluir a página de dados, pois adquire o bloqueio na página de dados.

- P

- **DELETE FROM**

- **TRUNCATE TABLE B;**

6.

- **DELETE**: Você não pode restaurar os dados excluídos após a execução deste comando.
- **TRUNCATE**: Você pode restaurar os dados usando o comando **COMMIT** ou **ROLLBACK**.

- P

- **DELETE FROM**
- **TRUNCATE TABLE B;**

6. Obse

- Para apa

  - **DELETE FROM A**

  - **TRUNCATE TABLE B;**

- A instrução **DELETE** exclui os registros e não interfere na identidade da tabela.
- A instrução **TRUNCATE** não exclui a estrutura da tabela, mas redefine a identidade da tabela.

Este material está disponível em: https://github.com/pedrogarciafreitas/slides-bd-2023.1

Dúvidas?