# DML: DATA MANIPULATION LANGUAGE

*Prof. Roberson Alves*

# DML

- Insert
- Update
- Delete
- Select

# COMANDO INSERT

# Incluindo dados em tabelas

- **Para incluir dados em tabelas, utilizamos o comando:**

**INSERT INTO tabela [(coluna [, coluna...])] VALUES (conteúdo, conteúdo,...)**

# Incluindo dados em tabelas

- **Exemplos:**

**INSERT INTO aluno (codalu, nomalu)**
**VALUES (1, ‘Fulano da Silva);**

**INSERT INTO aluno VALUES (1,**
**‘Fulano da Silva’, ‘Rua XYZ’, 1);**

# Inserindo várias linhas

- **Para inserir várias linhas, utilizamos o comando:**

**INSERT INTO tabela [(coluna [, coluna...])]**

**SELECT comando-select**

# Inserindo várias linhas

- **Exemplos:**

```
INSERT INTO copia_aluno
SELECT * FROM aluno;
```

```
INSERT INTO copia_aluno (codalu, nomalu)
SELECT codalu, nomalu FROM aluno;
```

# Inserindo várias linhas

- **Exemplos:**

```
INSERT INTO aluno (codalu, nomalu,
endalu, codnac) VALUES
(1, ‘Fulano da Silva’, ‘Rua XYZ’, 1),
(2, ‘Beltrano da Silva’, ‘Rua XYZ’, 2),
(3, ‘Fulana da Silva’, ‘Rua XYZ’, 1),
(4, ‘Ciclana da Cunha’, ‘Rua XYZ’, 2);
```

# Sintaxe PGSQL

- **Sintaxe do INSERT no PGSQL:**

https://www.postgresql.org/docs/current/sql-insert.html

# COMANDO UPDATE

# Atualizando dados em tabelas

- **Uma vez que uma linha esteja em uma tabela, pode-se querer alterar o conteúdo de uma ou mais colunas, ou até o conteúdo de uma coluna em diversas linhas. Para isso, utilizamos o comando:**

**UPDATE tabela SET coluna=conteúdo,**
**coluna=conteúdo,...**
**WHERE condição**

# Atualizando dados em tabelas

- **Exemplo:**

```
UPDATE aluno SET nomalu = 'Ciclano da Silva'
WHERE codalu = 2;

UPDATE historico
SET vlrnot = vlrnot * 1.05
WHERE codalu = 1 and coddis = 2;
```

# Atualizando dados em tabelas

- **Exemplo:**

```
UPDATE historico
SET vlrnot = vlrnot * 1.05;
```

# Sintaxe PGSQL

- **Sintaxe do UPDATE no PGSQL:**

https://www.postgresql.org/docs/current/sql-update.html

# COMANDO DELETE

# Exclusão de dados em tabelas

- **Sempre que alguma informação deixar de ser relevante para ser armazenada, pode-se excluí-la da tabela. Para isso, utilizamos o comando DELETE.**

**DELETE FROM tabela [WHERE condição];**

# Exclusão de dados em tabelas

- **Exemplo:**

```
DELETE  FROM aluno WHERE
codalu = 1;
```

```
DELETE  FROM aluno;
```

# Sintaxe PGSQL

- **Sintaxe do DELETE no PGSQL:**

https://www.postgresql.org/docs/current/sql-delete.html

# COMANDO SELECT

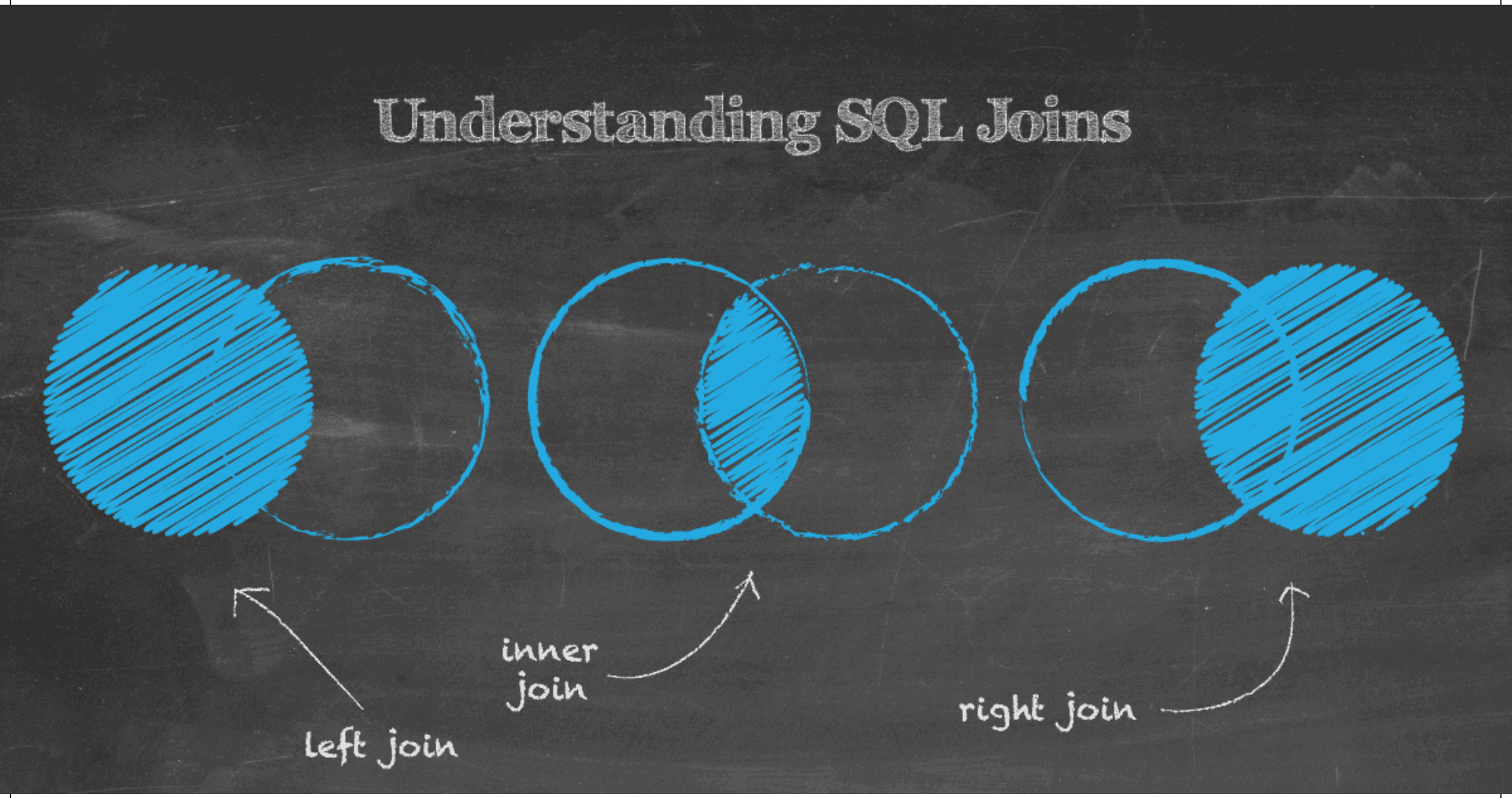
# Consultas Dados

- **Comando para consulta de dados:**

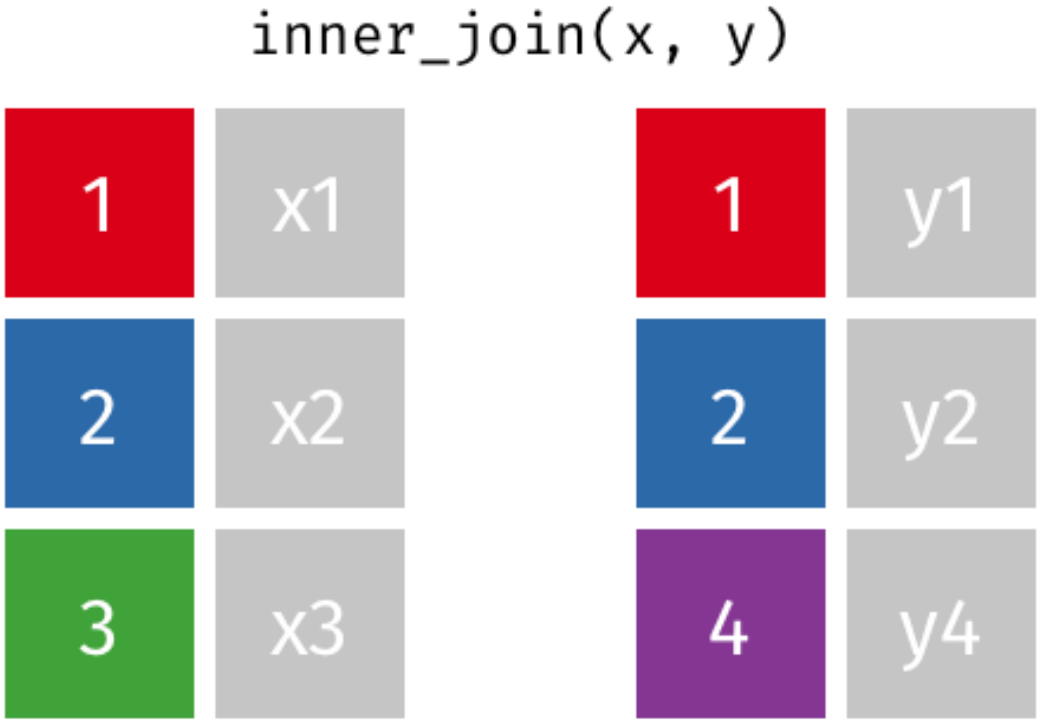
**SELECT [EXPRESSAO] [ tabela.coluna [AS] rotulo {, tabela.coluna }* ]**
**FROM** tabela {, tabela }* [[**AS**] rotulo]
[ **WHERE** condicao de busca ]
[ **GROUP BY** coluna {, coluna }* ]
[ **HAVING** condicao ]
[ **ORDER BY** coluna {, coluna }* [**ASC** | **DESC**] ];

# Consultas Dados

- **Funções Especiais auxiliam elaboração de consultas mais robustas e completas;**
- **Funções Matemáticas, Funções String, Funções de Data/Hora, Condicionais, Funções do sistema, entre outras.**


Understanding SQL Joins
left join
inner join
right join

# JUNÇÕES SQL OU JOINS



# JUNÇÕES SQL OU JOINS

- As junções são operações que permitem consultar dados de várias tabelas. Existem vários tipos de junções, dentre elas podem ser destacadas: self join, natural join, inner join, right [outer] join, left [outer] join, full [outer] join e o cross join.
- Alguns dos exemplos de junções são acompanhados de um diagrama Venn(ilustrações utilizadas para representar conjuntos, relações matemáticas ou relações lógicas).

# JUNÇÕES SQL OU JOINS

- **Regra geral para uso de JOINS:**

**Se houverem N tabelas envolvidas na consulta, são necessários N – 1 joins.**

# JUNÇÕES SQL OU JOINS

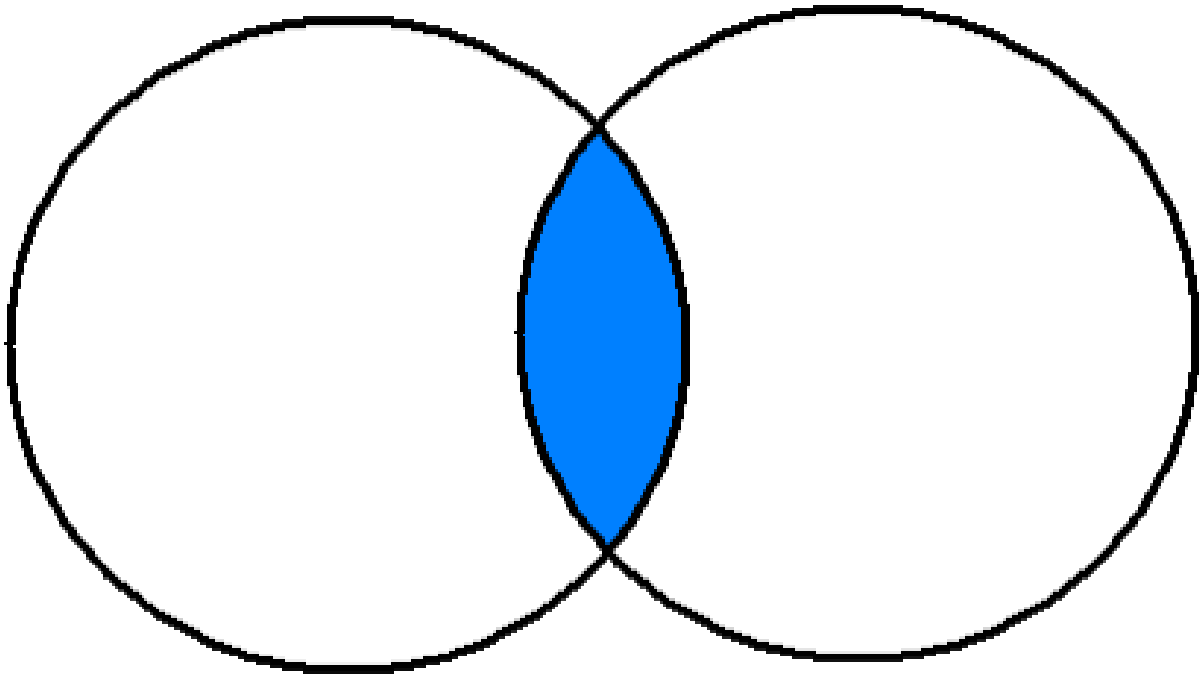
- **Esquema relacional exemplo:**

**EMPRESA(CODEMP, NOMEMP)**

**CIDADE(CODCID, NOMCID)**

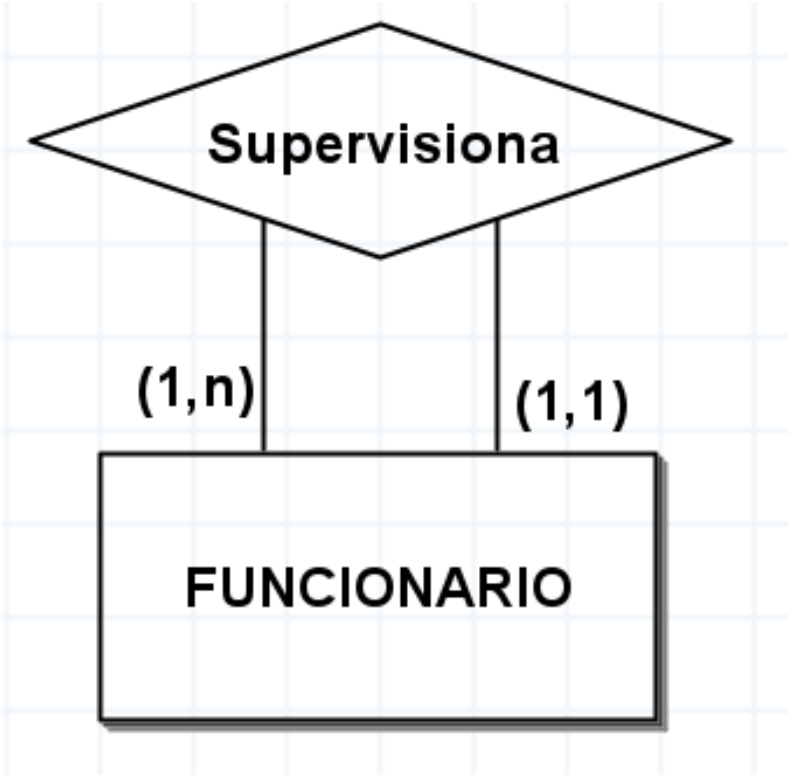
**FUNCIONARIO(NUMFUN, NOMFUN, CODEMP, NUMSUP, CODCID)**

# INNER JOIN



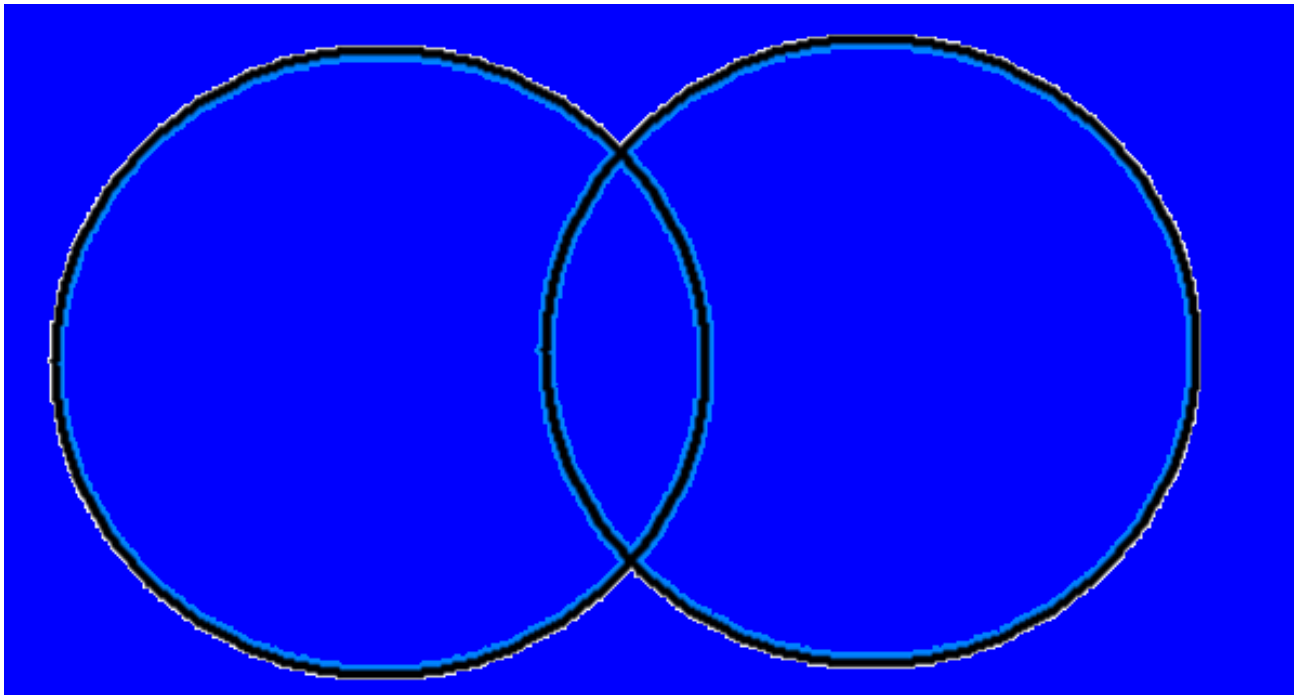
```
SELECT F.*, C.NOMCID
FROM FUNCIONARIO F
INNER JOIN CIDADE C ON F.CODCID =
C.CODCID;
```

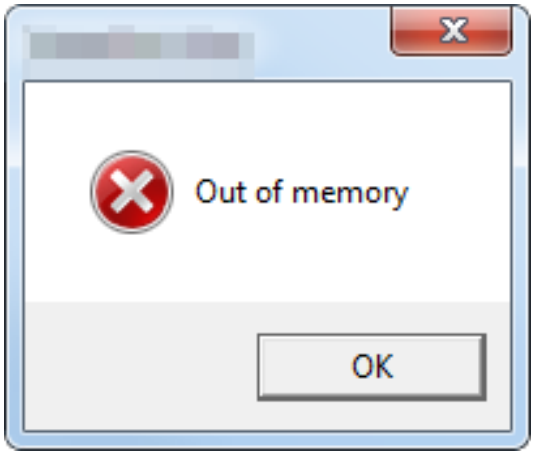
# SELF JOIN




```
SELECT F1.NOMFUN FUNC, F2.NOMFUN SUPER
FROM FUNCIONARIO F1, FUNCIONARIO F2
WHERE F1.NUMSUP = F2.NUMFUN;
```

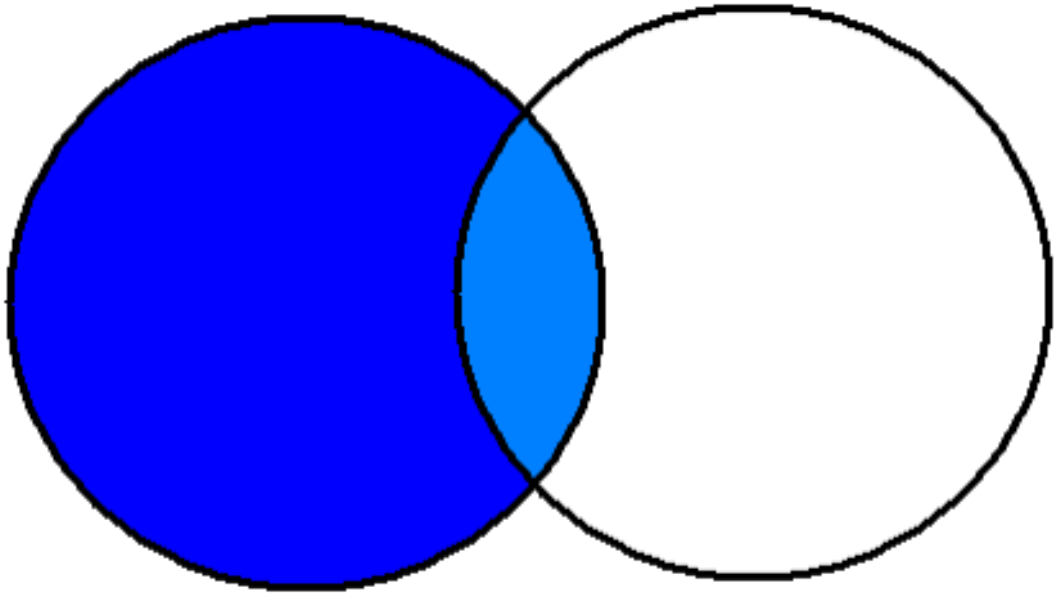
# CROSS JOIN





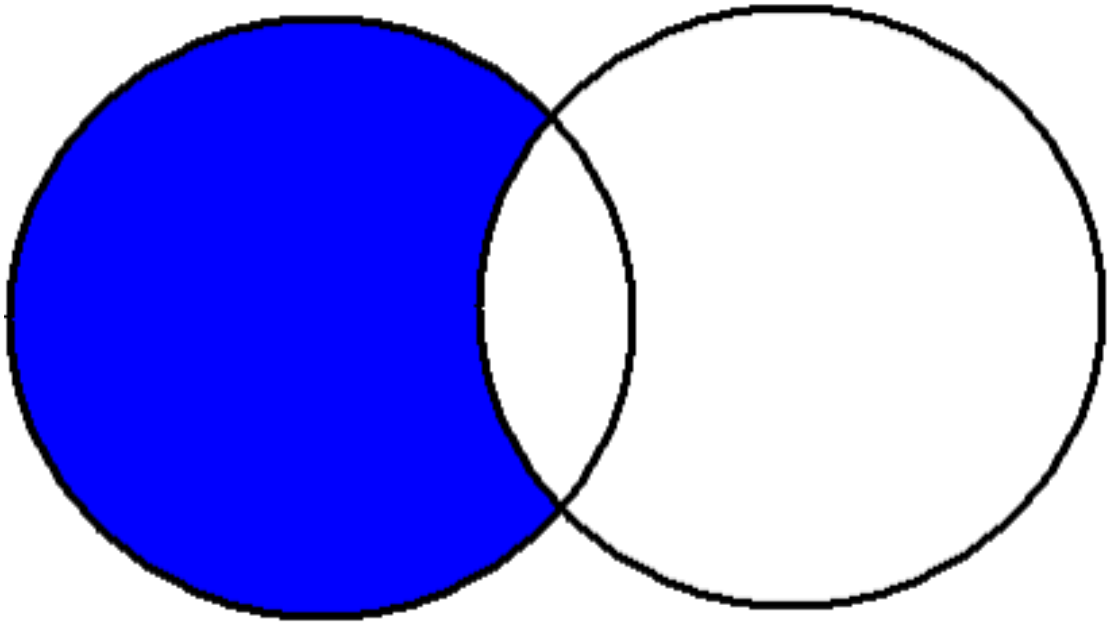

```
SELECT F.*, C.*
FROM FUNCIONARIO F
CROSS JOIN CIDADE C;
```

# LEFT [OUTER] JOIN



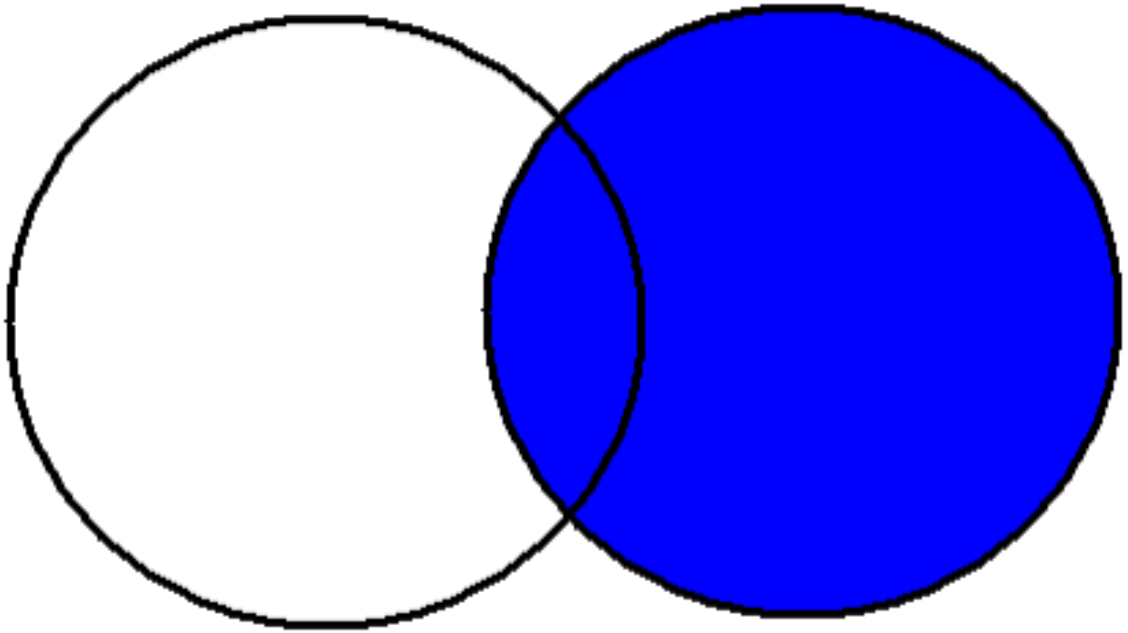
```
SELECT F.NUMFUN, F.NOMFUN, C.NOMCID
FROM FUNCIONARIO F
LEFT JOIN CIDADE C ON F.CODCID =
C.CODCID;
```

# LEFT [OUTER] JOIN



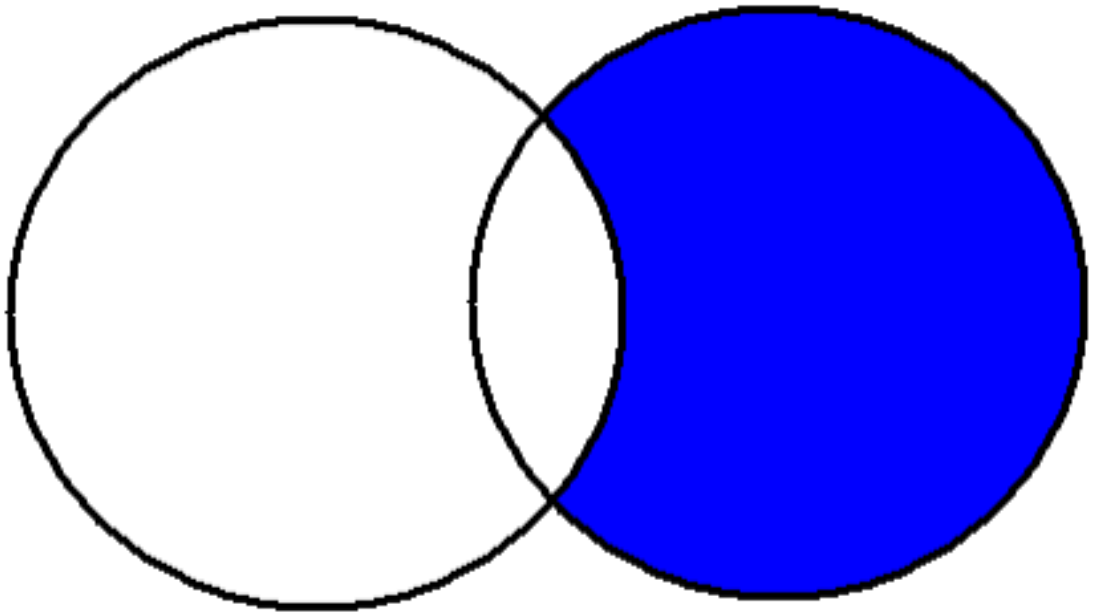
```
SELECT F.NUMFUN, F.NOMFUN, C.NOMCID
FROM FUNCIONARIO F
LEFT JOIN CIDADE C ON F.CODCID = C.CODCID
WHERE F.CODCID IS NULL;
```

# RIGHT [OUTER] JOIN



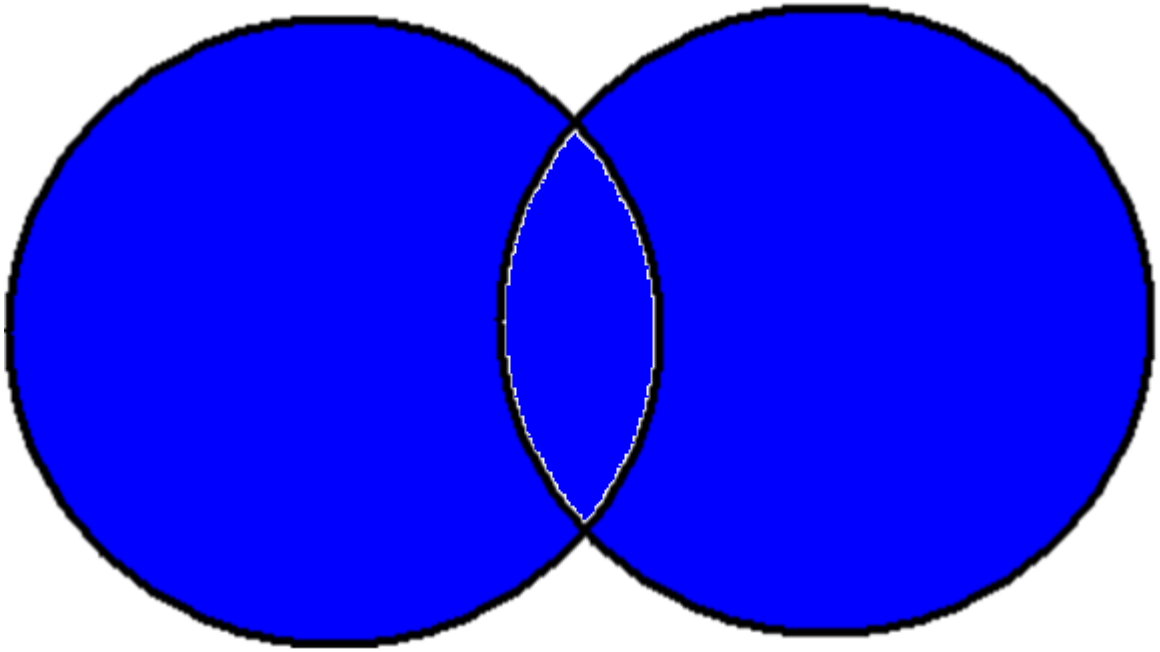
```
SELECT F.NUMFUN, F.NOMFUN, C.NOMCID
FROM FUNCIONARIO F
RIGHT JOIN CIDADE C ON F.CODCID =
C.CODCID;
```

# RIGHT [OUTER] JOIN



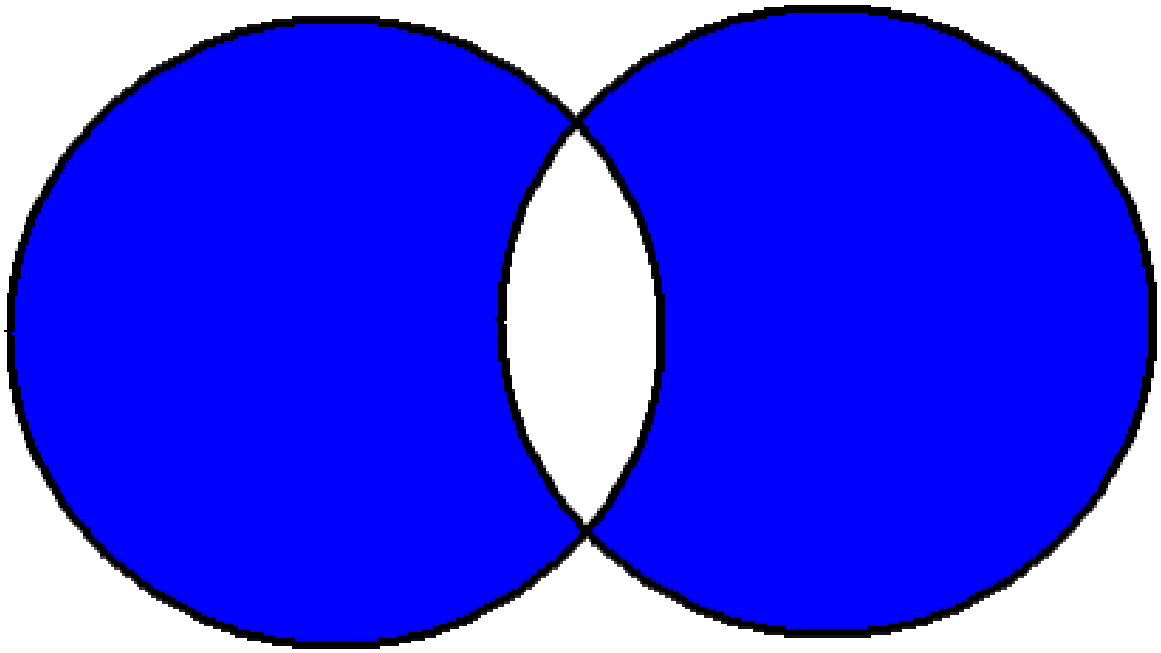
```
SELECT F.NUMFUN, F.NOMFUN, C.NOMCID
FROM FUNCIONARIO F
RIGHT JOIN CIDADE C ON F.CODCID =
C.CODCID WHERE F.CODCID IS NULL;
```

# FULL [OUTER] JOIN



```
SELECT F.NUMFUN, F.NOMFUN, C.NOMCID
FROM FUNCIONARIO F
FULL JOIN CIDADE C ON F.CODCID =
C.CODCID;
```

# FULL [OUTER] JOIN



```
SELECT F.NUMFUN, F.NOMFUN, C.NOMCID
FROM FUNCIONARIO F FULL JOIN CIDADE C
ON F.CODCID = C.CODCID WHERE F.CODCID
IS NULL;
```

# SUBQUERYS OU SUBCONSULTAS

- **Uma instrução SELECT trabalha sobre predicados lógicos que precisam ser atendidos para que determinadas tuplas sejam recuperadas. Uma *subquery* nada mais é que uma instrução SELECT embutida em outra instrução SELECT.**
- **Operadores: IN, SOME, EXISTS e ANY.**

# SUBQUERYS: EXEMPLOS

- **Exemplo de JOIN "estranho":**

```
SELECT NUMFUN, NOMFUN
FROM FUNCIONARIO
WHERE CODCID = (SELECT CODCID FROM CIDADE
WHERE CODCID = 10);
```

- **Usando IN:**

```
SELECT NUMFUN, NOMFUN
FROM FUNCIONARIO
WHERE CODCID IN (SELECT CODCID FROM CIDADE
WHERE NOMCID IN
('SMOESTE','MARAVILHA','PINHALZINHO'));
```

# DESAFIO

- Torneio intitulado “SQL CHALLENGE: DML COMMANDS”
- O torneio será realizado pelo beecrowd.com.br
- Acessar e se registrar no torneio;
- Torneio inicia em 10/08 21h e vai até 13/08 23h59min