
FIAP

AULA 08

SQL/DQL – SELECT JOIN

Welcome to the next evolution in higher education.

DISCIPLINA

BUILDING RELATIONAL DATABASE

PROFº DRº FRANCISCO D. L. ABREU
PROFFRANCISCO@FIAP.COM.BR

SUMÁRIO

- DQL
  - Sobre Selecionar Dados com Junção
  - Tipos de Junção (SQL Worksheet)
  - Exercício Prático

OBJETIVO

Introduzir conceitos iniciais do SQL/DQL

Aplicar os conceitos no Oracle SQL Developer

SOBRE SELECT JOIN

SOBRE

FIAP

- ***Data Query Language*** **(DQL) – Linguagem de Consulta de Dados** , expressa o comando que especifica a:
  - **CONSULTAR** 1 ou vários dados (SELECT)

- Os comandos da **DQL** viabiliza o acesso aos dados de forma compatível ao modelo de dados projetado.
- Em algumas literaturas colocam o **SELECT** dentro da **DML**

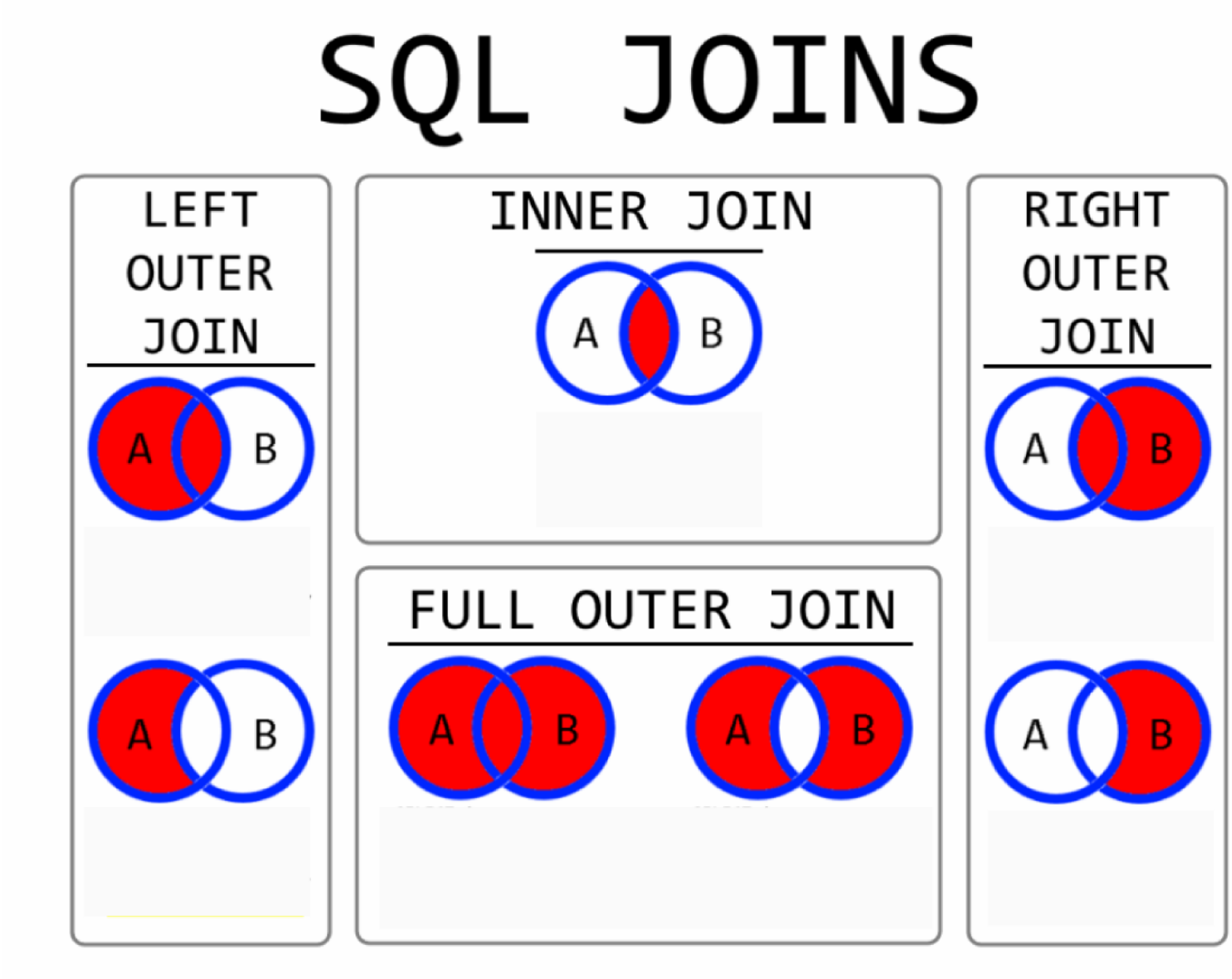
SOBRE SELECT JOIN

FIAP

- O **SELECT JOIN** permite o agrupamento de dados em tabelas separada por meio dos relacionamentos explícitos (Chave Primária com Chave Estrangeira)
    - Também é possível por relacionamentos não explícitos (ou não previstos anteriormente na modelagem conceitual)
    - Você pode unir dados a partir de quaisquer colunas entre as tabelas, desde que os tipos sejam iguais E a operação faça sentindo
- **Boas junções**:
    - As colunas de junção normalmente é a coluna de chave primária com estrangeira
    - As colunas de junção devem ser do mesmo tipo.

TIPOS DE SELECT JOIN

TIPOS DE SELECT JOIN



TIPOS DE SELECT JOIN

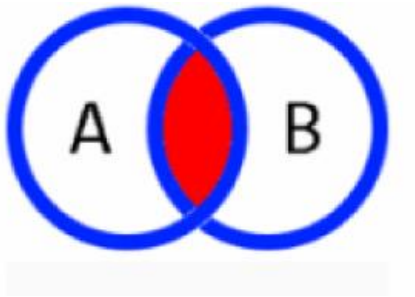
```sql
-- Cria a tabela "Clientes"
CREATE TABLE Clientes (
    ID NUMBER(5) PRIMARY KEY,
    Nome VARCHAR2(255),
    Cidade VARCHAR2(255)
);

-- Insercao de dados na tabela "Clientes"
INSERT INTO Clientes (ID, Nome, Cidade)
VALUES
    (1, 'João', 'São Paulo'),
    (2, 'Maria', 'Rio de Janeiro'),
    (3, 'Carlos', 'Belo Horizonte'),
    (4, 'Ana', 'Porto Alegre'),
    (5, 'Rafael', 'Brasília');
```

```sql
-- Cria a tabela "Pedidos"
CREATE TABLE Pedidos (
    ID NUMBER(5) PRIMARY KEY,
    Cliente_ID NUMBER(5),
    Produto VARCHAR2(255),
    FOREIGN KEY (CLIENTE_ID) REFERENCES
        Clientes (ID)
);

-- Insercao de dados na tabela "Pedidos"
INSERT INTO Pedidos (ID, Cliente_ID, Produto)
VALUES
    (101, 1, 'Celular'),
    (102, 2, 'Laptop'),
    (103, 3, 'Tablet'),
    (104, 1, 'TV'),
    (105, 4, 'Geladeira');
```

COMANDO: SELECT EQUI JOIN



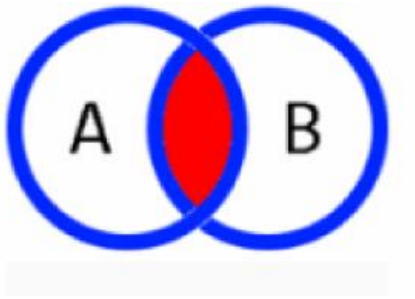
- O comando **SELECT EQUI JOIN** é utilizado para retorna apenas os registros que têm correspondências nas duas tabelas que estão sendo unidas. Ou seja, apenas os registros que satisfazem a condição de junção são retornados.

- Veja o exemplo da Sintaxe:

SINTAXE

```
SELECT * FROM CLIENTES, PEDIDOS
    WHERE CLIENTES.ID =
PEDIDOS.CLIENTE_ID;
```

COMANDO: SELECT INNER JOIN



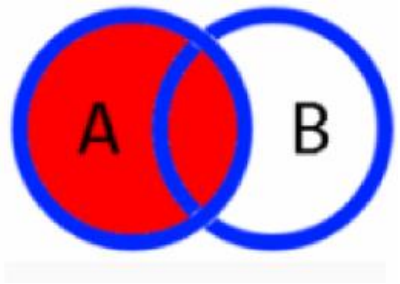
- O comando **SELECT INNER JOIN** é utilizado para retorna apenas os registros que têm correspondências nas duas tabelas que estão sendo unidas. Ou seja, apenas os registros que satisfazem a condição de junção são retornados.

- Veja o exemplo da Sintaxe:

SINTAXE

```
SELECT Clientes.Nome, Pedidos.Produto
FROM Clientes
INNER JOIN Pedidos ON Clientes.ID =
Pedidos.Cliente_ID;
```

COMANDO: SELECT LEFT JOIN



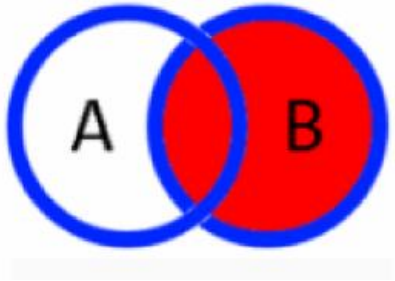
- O comando **SELECT LEFT JOIN** é utilizado para retorna todos os registros da tabela à ESQUERDA, junto com os registros correspondentes (ou similares) da tabela à DIREITA.

- Veja o exemplo da Sintaxe:

SINTAXE

```
SELECT Clientes.Nome, Pedidos.Produto
FROM Clientes
LEFT JOIN Pedidos ON Clientes.ID =
Pedidos.Cliente_ID;
```

COMANDO: SELECT RIGHT JOIN



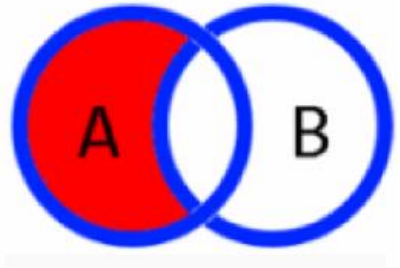
- O comando **SELECT RIGTH JOIN** é utilizado para retorna todos os registros da tabela à DIREITA, junto com os registros correspondentes (ou similares) da tabela à ESQUERDA

- Veja o exemplo da Sintaxe:

SINTAXE

```
SELECT Clientes.Nome, Pedidos.Produto
FROM Clientes
RIGHT JOIN Pedidos ON Clientes.ID =
Pedidos.Cliente_ID;
```

COMANDO: SELECT LEFT EXCLUDING JOIN



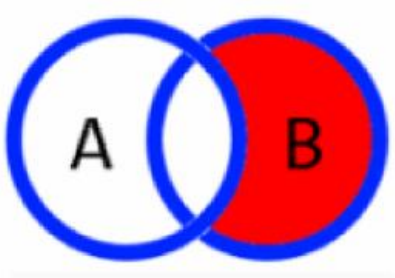
- O comando **SELECT LEFT EXCLUDING JOIN** é utilizado para retorna todos os registros da tabela à ESQUERDA, exceto os registros correspondentes (ou similares) da tabela à DIREITO

- Veja o exemplo da Sintaxe:

SINTAXE

```
SELECT Clientes.Nome, Pedidos.Produto
FROM Clientes
LEFT JOIN Pedidos ON Clientes.ID =
Pedidos.Cliente_ID
WHERE Pedidos.ID IS NULL;
```

COMANDO: SELECT RIGHT EXCLUDING JOIN



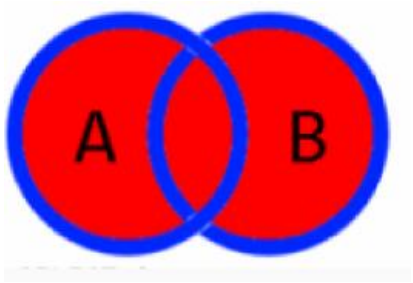
- O comando **SELECT RIGTH EXCLUDING JOIN** é utilizado para retorna todos os registros da tabela à DIREITA, exceto os registros correspondentes (ou similares) da tabela à ESQUERDA

- Veja o exemplo da Sintaxe:

SINTAXE

```
SELECT Clientes.Nome, Pedidos.Produto
FROM Clientes
RIGHT JOIN Pedidos ON Clientes.ID =
Pedidos.Cliente_ID
WHERE Clientes.ID IS NULL;
```

COMANDO: SELECT FULL JOIN



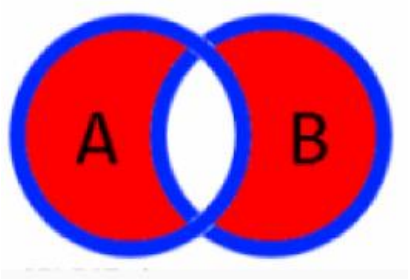
- O comando **SELECT FULL JOIN** é utilizado para retorna todos os registros diferentes da tabela à DIREITA e da tabela à ESQUERDA, além de unir aqueles que são comuns entre as duas tabelas.

- Veja o exemplo da Sintaxe:

SINTAXE

```
SELECT Clientes.Nome, Pedidos.Produto
FROM Clientes
FULL OUTER JOIN Pedidos ON Clientes.ID
= Pedidos.Cliente_ID;
```

COMANDO: SELECT OUTER JOIN



- O comando **SELECT OUTER JOIN** é utilizado para retorna todos os registros diferentes da tabela à DIREITA e da tabela à ESQUERDA, exceto aqueles que são comuns entre as duas tabelas.

- Veja o exemplo da Sintaxe:

SINTAXE

```
SELECT Clientes.Nome, Pedidos.Produto
FROM Clientes
LEFT JOIN Pedidos ON Clientes.ID =
Pedidos.Cliente_ID
WHERE Pedidos.ID IS NULL

UNION

SELECT Clientes.Nome, Pedidos.Produto
FROM Clientes
RIGHT JOIN Pedidos ON Clientes.ID =
Pedidos.Cliente_ID
WHERE Clientes.ID IS NULL;
```

SOBRE SELF JOIN

- O **SELF JOIN** (ou autojunção) é uma operação de junção em que uma tabela é combinada consigo mesma
- O "SELF JOIN" é frequentemente usado quando você tem dados em uma única tabela que estão relacionados entre si por meio de campos dentro dessa tabela
- Nesse casso, é necessário acessa a mesma tabela duas vezes, uma para recuperar os dados e a outra para buscar os dados auto-relacionados
    - **Atenção!** É necessário utilizar o comando AS (para definir apelidos) entre a tabela e colunas.

SOBRE SELF JOIN

FIAP

```sql
CREATE TABLE Funcionarios (
    ID INT PRIMARY KEY,
    Nome VARCHAR(255),
    Supervisor_ID INT
);
```

```sql
INSERT INTO Funcionarios (ID, Nome, Supervisor_ID) VALUES (1, 'João', NULL);
-- João é o chefe e não tem supervisor
INSERT INTO Funcionarios (ID, Nome, Supervisor_ID) VALUES  (2, 'Maria', 1); -
- Maria é subordinada a João
INSERT INTO Funcionarios (ID, Nome, Supervisor_ID) VALUES  (3, 'Carlos', 1);
-- Carlos é subordinado a João
INSERT INTO Funcionarios (ID, Nome, Supervisor_ID) VALUES  (4, 'Ana', 2);
 -- Ana é subordinada a Maria
INSERT INTO Funcionarios (ID, Nome, Supervisor_ID) VALUES  (5, 'Rafael', 2);
-- Rafael é subordinado a Maria
```

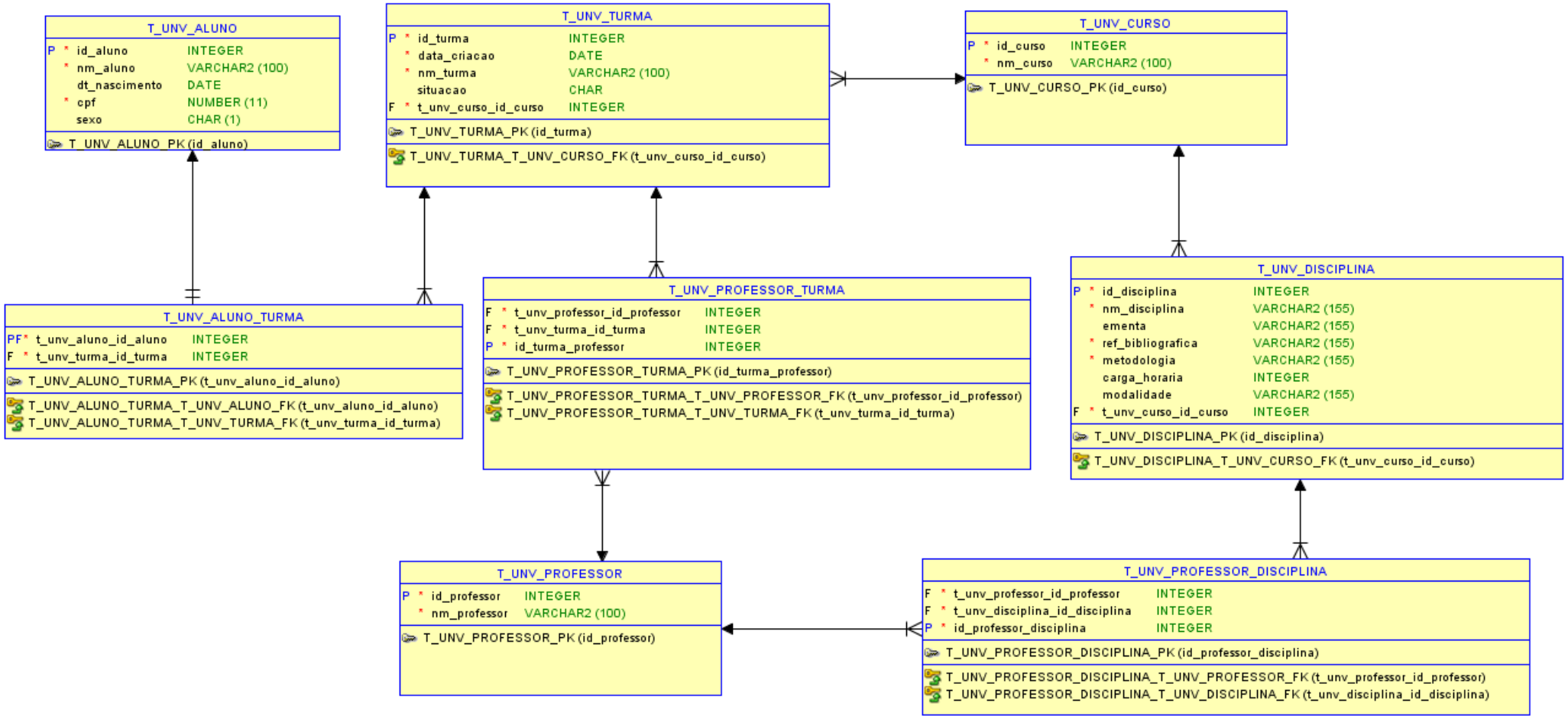
SOBRE SELF JOIN

```sql
SELECT A.Nome AS Funcionario, B.Nome AS Supervisor
FROM Funcionarios A
LEFT JOIN Funcionarios B ON A.Supervisor_ID = B.ID;
```

Atenção! É necessário utilizar o comando AS (para definir apelidos) entre a tabela e colunas

EXERCÍCIO PRÁTICO

EXERCÍCIO PRÁTICO



ALTERAÇÃO DDL

- Altere as estruturas das tabelas no SQL Developer:
  - Adicione o campo SITUACAO (CHAR(1)) na tabela ALUNO e PROFESSOR;
  - Altere o nome do campo CARGA_HORARIA para CH da tabela DISCIPLINA ;
  - Altere o nome da tabela ALUNO para DISCENTE e PROFESSOR para DOCENTE;
  - Altere o tipo das colunas EMENTA, REF_BIBLIOGRAFICA E METODOLOGIA para LONG VARCHAR;
  - Remova todas as tabelas existentes (Observe que há uma ordem correta de exclusão);

THE WAY WE ARE