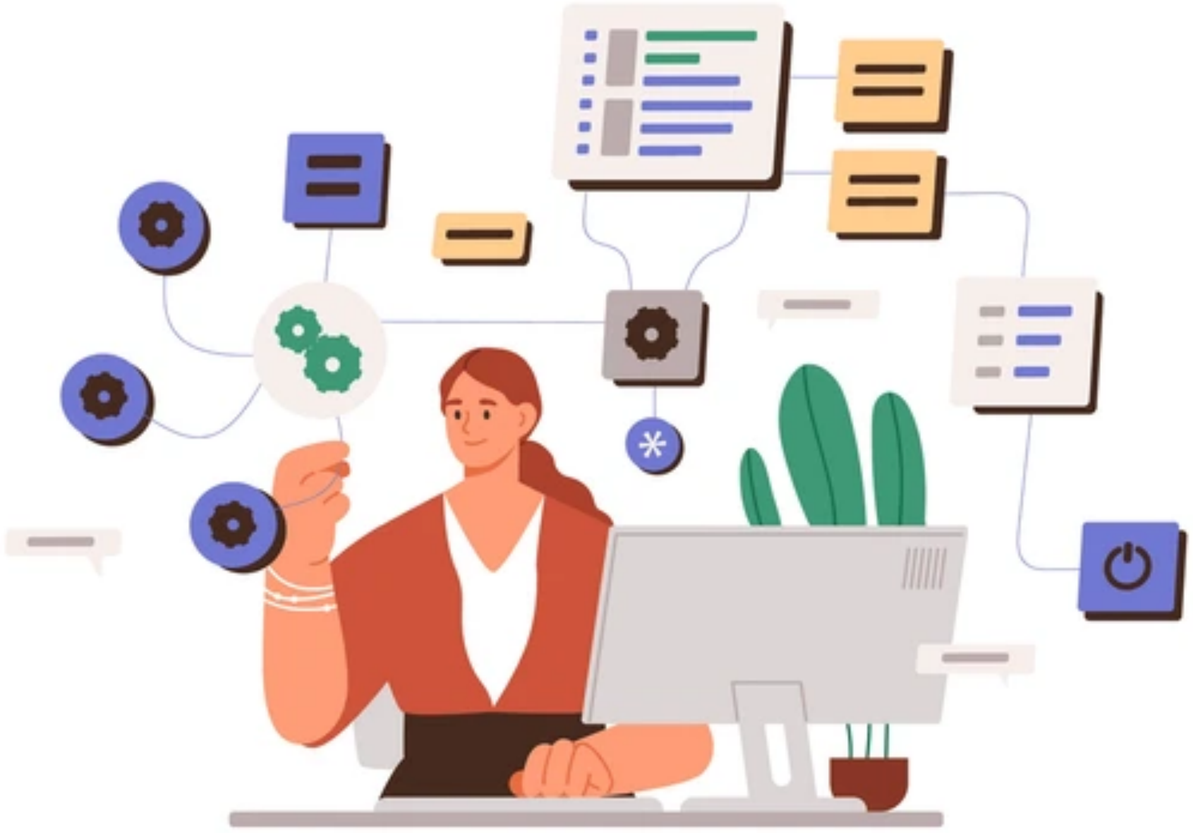


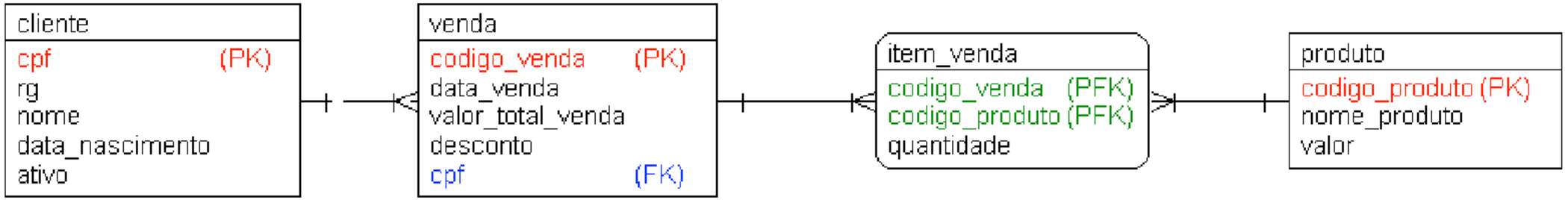
JUNÇÃO ENTRE TABELAS

Prof. Rangel Nunes

A CLÁSULA JOIN

Uma cláusula JOIN é usada para combinar linhas de duas ou mais tabelas, com base em uma coluna relacionada entre elas

IMAGINE UM BANCO DE DADOS SIMPLIFICADO DE UM CONTROLE DE ESTOQUE



E se eu quisesse implementar consultas que retornam atributos de mais de uma tabela?

Ex.: Os nomes dos clientes e o valor total das suas compras

O POSTGRESQL SUPORTA:

INNER JOIN

LEFT JOIN

RIGHT JOIN

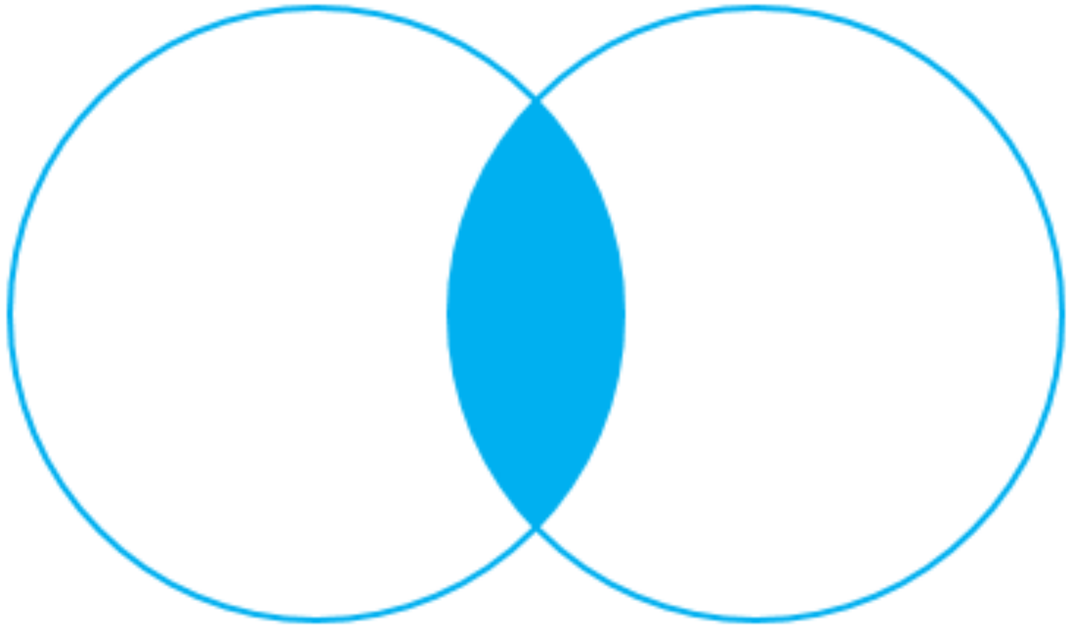
FULL OUTER JOIN

CROSS JOIN

NATURAL JOIN

E um tipo especial chamado SELF-JOIN

INNER JOIN



</ CODING SQL>

```
SELECT
  lista_de_atributos
FROM
  tabela1
INNER JOIN tabela2
  ON tabela.nome_da_coluna =
tabela.nome_da_coluna;
```

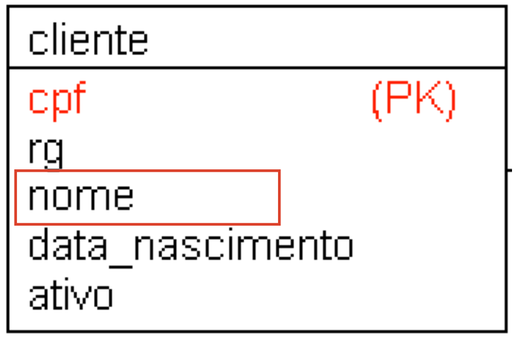
</ CODING SQL>

Quais os nomes dos clientes que já compraram nesta loja?

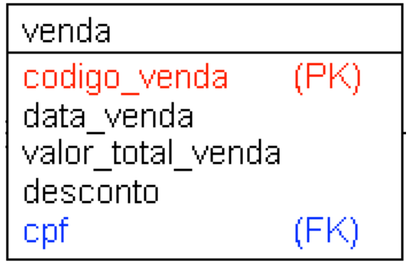
</ CODING SQL>

Duas perguntas que sempre vão ti ajudar a implementar Joins

O QUE VOCÊ QUER RETORNAR NA CONSULTA?

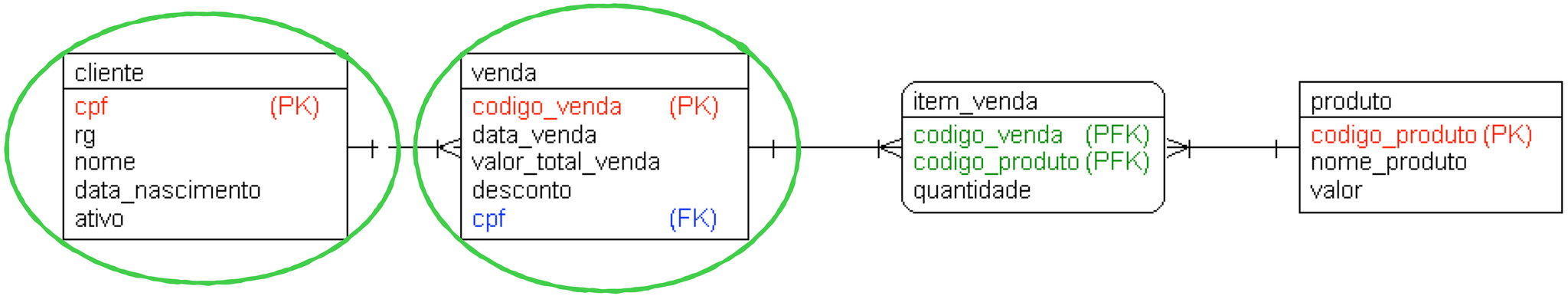


QUAL É A CONDIÇÃO?



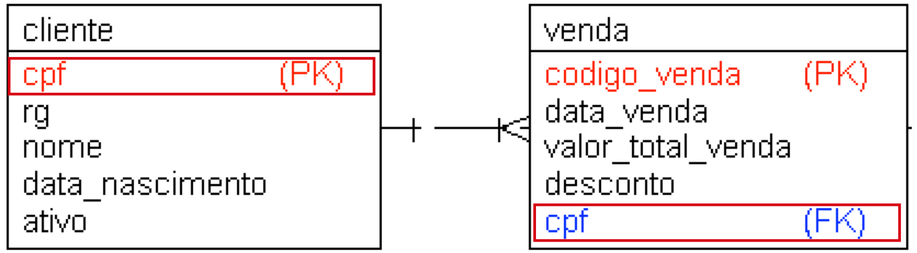
</ CODING SQL>

Com estas perguntas, sempre saberei quais tabelas devo usar no junção



</ INNER JOIN>

```
SELECT [ * | EXPRESSÃO ]
FROM TABELA1
[INNER] JOIN TABELA2
USING (ATRIBUTO_DE_JUNÇÃO)
```




```
SELECT [ * | EXPRESSÃO ]
FROM TABELA1
[INNER] JOIN TABELA2
ON (TABELA1.CHAVE_PRIMARIA = TABELA2.CHAVE_ESTRANGEIRA)
```

</ CODING SQL>

Quais os nomes dos clientes que já compraram nesta loja?

```
SELECT NOME FROM CLIENTE
INNER JOIN VENDA
USING(CPF);
```

```
SELECT NOME FROM CLIENTE
INNER JOIN VENDA
ON (CLIENTE.CPF = VENDA.CPF);
```

NATURAL JOIN

Cria uma junção implícita com base nas colunas de nomes iguais, encontradas nas tabelas envolvidas

</ CODING SQL>

```
SELECT lista_de_atributos
FROM tabela1
NATURAL [INNER, LEFT, RIGHT] JOIN tabela2;
```

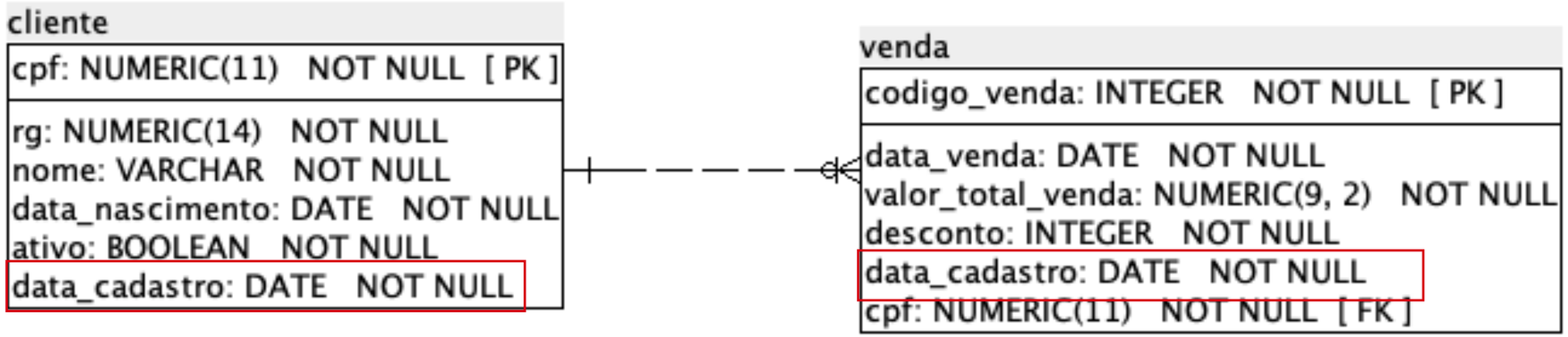
</ CODING SQL>

Quais os nomes dos clientes que já compraram nesta loja?

```
SELECT NOME
FROM CLIENTE
NATURAL JOIN VENDA;
```

RESULTADO INESPERADO!

Exemplo de quando uma junção natural pode retornar um resultado inesperado



</ CODING SQL>

Criando e povoando as tabelas…

https://drive.google.com/file/d/1kIgaqjH_m3wVDWkIQ6Q3HXNQ2tHGpcB-/view?usp=sharing

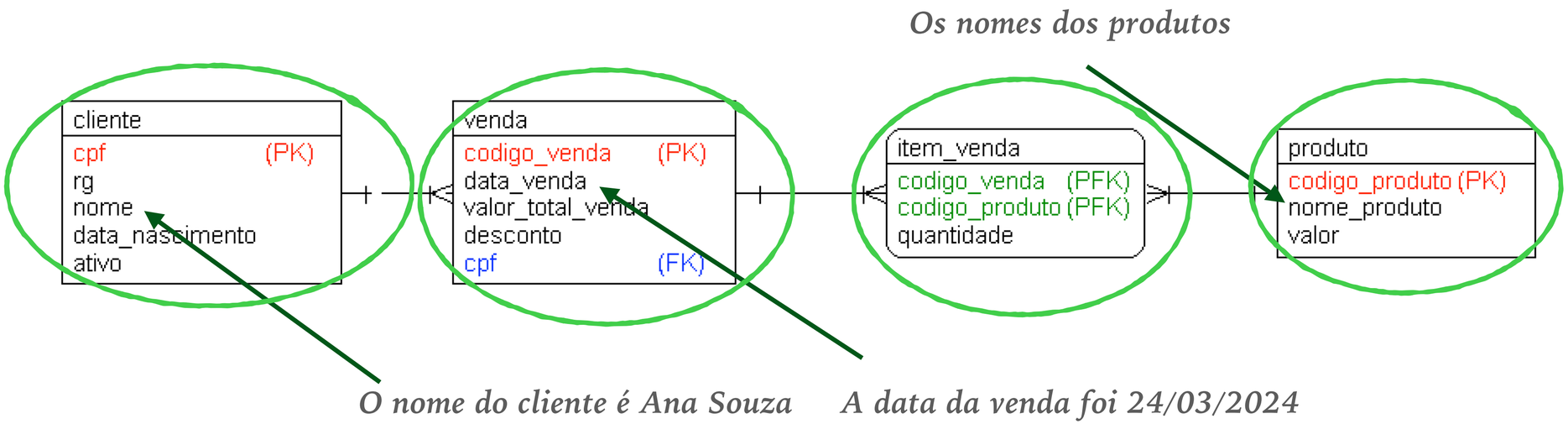
JUNÇÃO PODE CONTER CONDIÇÕES

INNER JOIN COM WHERE

INNER JOIN COM WHERE

Quais os nomes dos produtos comprados pela cliente Ana Souza, no dia 24/03/2024?

O QUE VOCÊ QUER RETORNAR NA CONSULTA?

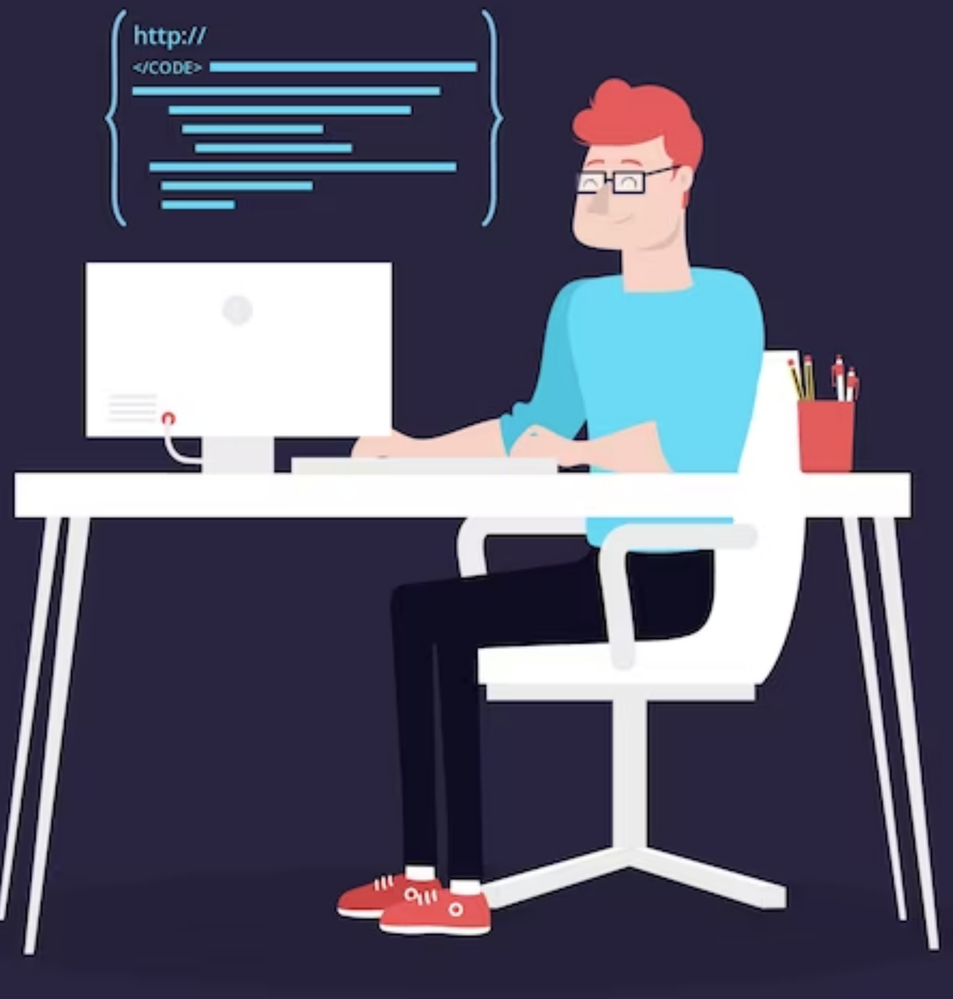


QUAL É A CONDIÇÃO?

</ CODING SQL>

Quais os nomes dos produtos comprados pela cliente Ana Souza, no dia 24/03/2024?

```
SELECT NOME_PRODUTO FROM PRODUTO
INNER JOIN ITEM_VENDA USING(CODIGO_PRODUTO)
INNER JOIN VENDA USING(CODIGO_VENDA)
INNER JOIN CLIENTE USING(CPF)
WHERE NOME = 'ANA SOUZA'
AND DATA_VENDA = '24/03/2024';
```



HANDS-ON

</ CODING SQL>

Criando e povoando o banco de dados do projeto de demandas de disciplinas

https://drive.google.com/file/d/1buWJpxXot0oSJ7qUiaq23-yMP1zyKTou/view?usp=sharing

https://drive.google.com/file/d/14P5EMeLay1_az0vYu6HS1eTFDCaobigo/view?usp=sharing

Implemente as seguintes consultas:

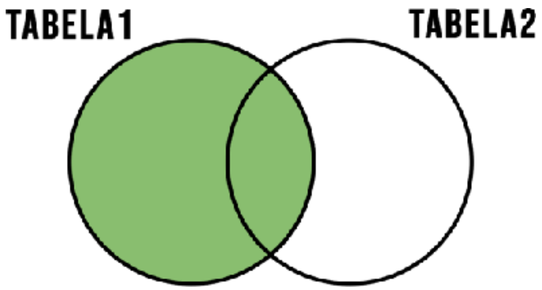
https://drive.google.com/file/d/1NndOHYRGCstp7MI-lGAZUmVEBkK8tBUb/view?usp=sharing

VAMOS PENSAR UM POUCO...

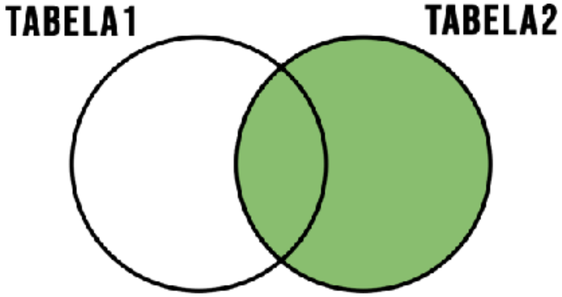
E se eu quisesse retornar do banco as matrículas e os nomes dos professores que já ministraram a disciplina da Cálculo I e que as aulas aconteciam nas segundas e nas quartas?

OUTER JOIN

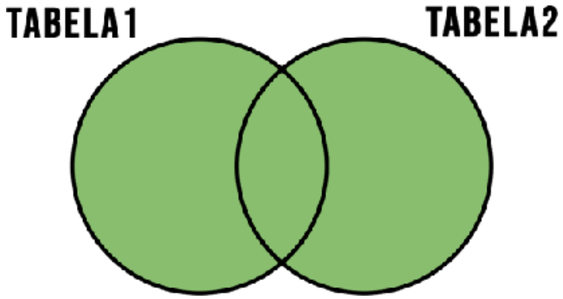
LEFT JOIN



RIGHT JOIN

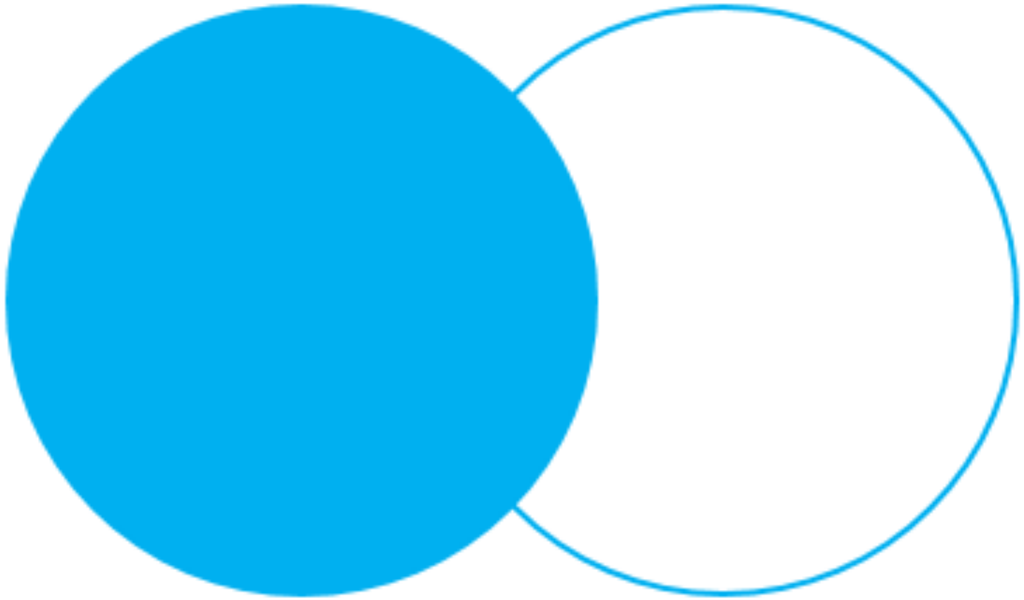


FULL JOIN



LEFT JOIN

Une uma tabela da esquerda com a tabela da direita e retorna as linhas da tabela da esquerda que podem ou não ter linhas correspondentes na tabela da direita



</ CODING SQL>

```sql
SELECT
  lista_de_atributos
FROM
  tabela1
LEFT JOIN tabela2
  ON tabela1.nome_da_coluna = tabela2.nome_da_coluna;
```

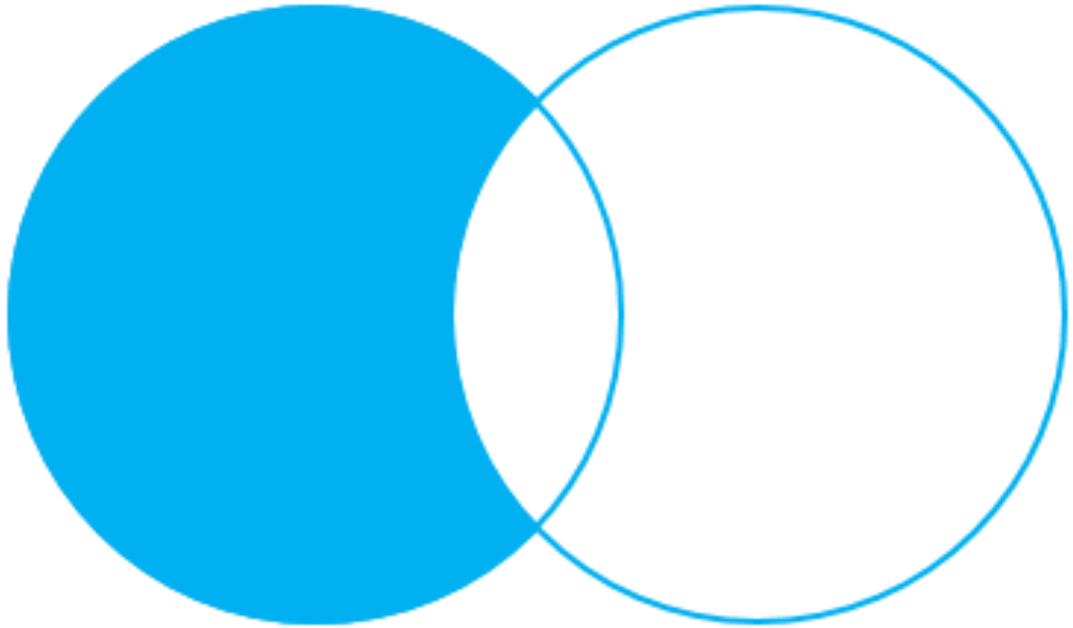
</ CODING SQL>

IMPLEMENTE UMA CONSULTA QUE RETORNE OS NOMES DOS CLIENTES E DATAS DAS COMPRAS

```
SELECT NOME
FROM CLIENTE
LEFT JOIN VENDA
USING(CPF)
```

</ CODING SQL>

E se o dono desta loja quisesse consultar todos os clientes que estão cadastrados, mas nunca fizeram compra?



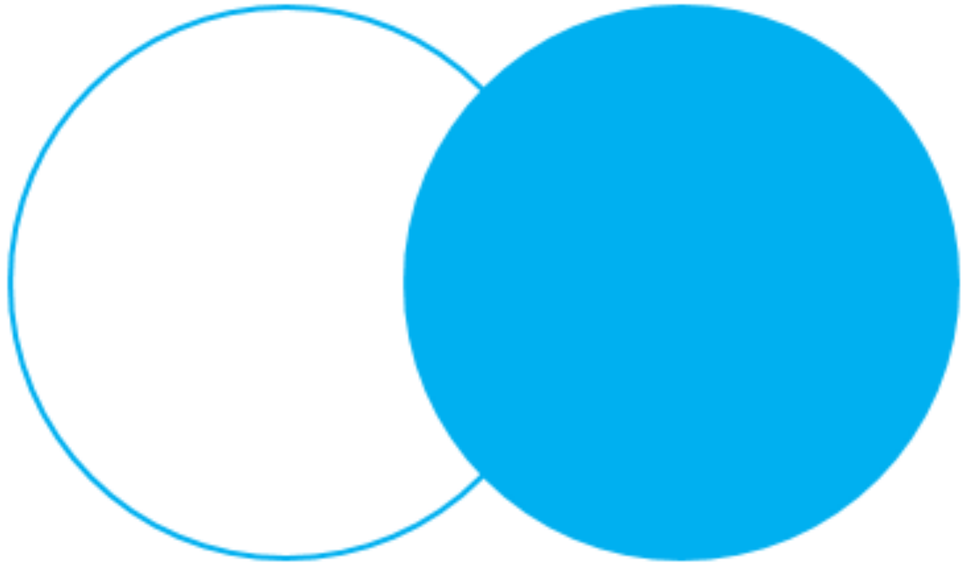
</ CODING SQL>

E se o dono desta loja quisesse consultar todos os clientes que estão cadastrados, mas nunca fizeram compra?

```
SELECT
  NOME
FROM
  CLIENTE
  LEFT JOIN VENDA USING (CPF)
WHERE
  VENDA.CODIGO_VENDA IS NULL;
```

RIGHT JOIN

Une uma tabela da direita com uma tabela da esquerda e retorna as linhas da tabela da direita que podem ou não ter linhas correspondentes na tabela da esquerda



</ CODING SQL>

```sql
SELECT
  lista_de_atributos
FROM
  tabela1
RIGHT JOIN tabela2
  ON tabela1.nome_da_coluna = tabela2.nome_da_coluna;
```

</ CODING SQL>

IMPLEMENTE UMA CONSULTA QUE RETORNE OS NOMES DOS PRODUTOS E DATAS EM QUE FORAM COMPRADOS

```
SELECT NOME_PRODUTO,DATA_VENDA
FROM VENDA
INNER JOIN ITEM_VENDA
USING(CODIGO_VENDA)
RIGHT JOIN PRODUTO
USING(CODIGO_PRODUTO)
```

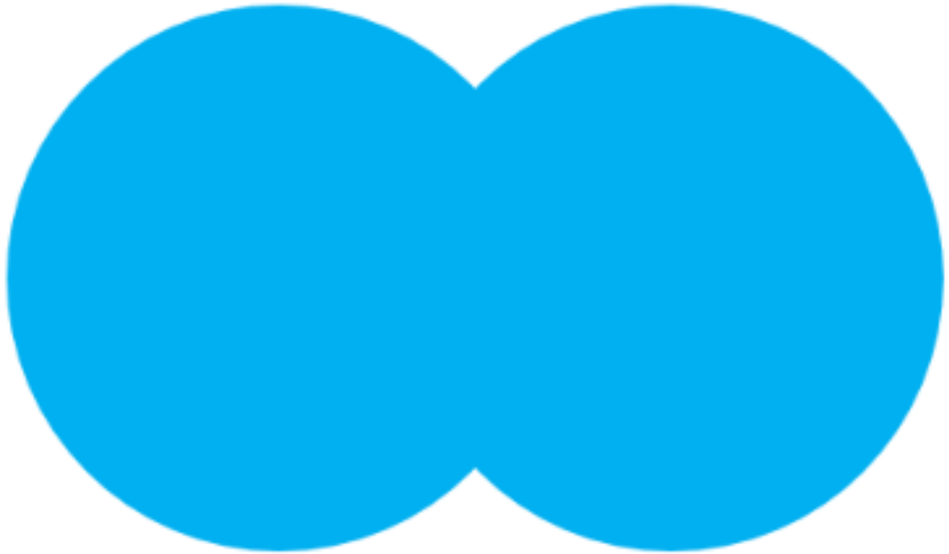
</ CODING SQL>

E se o dono desta loja quisesse consultar todos os produtos que estão cadastrados, mas nunca foram comprados?

```
SELECT NOME_PRODUTO,DATA_VENDA
FROM VENDA
INNER JOIN ITEM_VENDA
USING(CODIGO_VENDA)
RIGHT JOIN PRODUTO
USING(CODIGO_PRODUTO)
WHERE CODIGO_VENDA IS NULL
```

FULL OUTER JOIN

Combina dados de duas tabelas e retorna todas as linhas de ambas as tabelas, incluindo linhas correspondentes e não correspondentes de ambos os lados



</ CODING SQL>

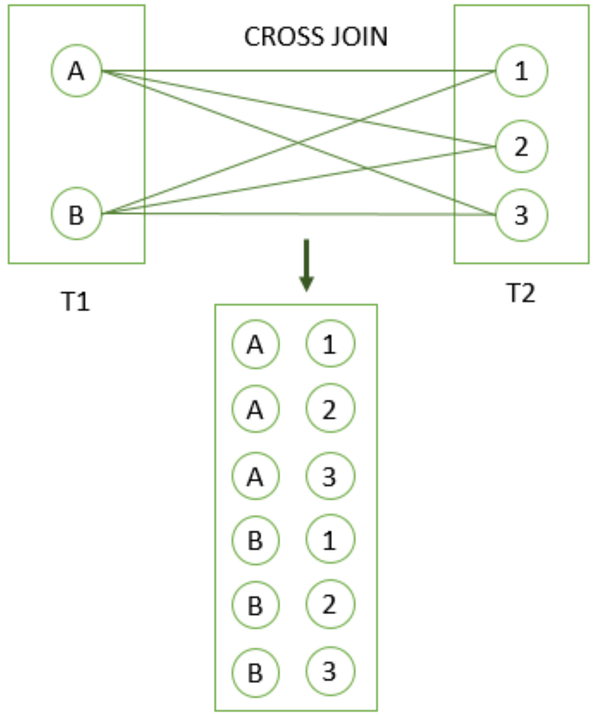
```sql
SELECT lista_de_atributos
FROM tabela1
FULL OUTER JOIN tabela2
   ON tabela1.nome_da_coluna = tabela2.nome_da_coluna;
```

</ CODING SQL>

```
SELECT NOME_PRODUTO,NOME
FROM CLIENTE
FULL JOIN VENDA USING(CPF)
FULL JOIN ITEM_VENDA USING(CODIGO_VENDA)
FULL JOIN PRODUTO
USING(CODIGO_PRODUTO)
```

CROSS JOIN

Permite produzir um produto cartesiano de linhas em duas tabelas. Isso significa que ele combina cada linha da primeira tabela com cada linha da segunda tabela



</ CODING SQL>

```sql
SELECT
  lista_de_atributos
FROM
  tabela1
CROSS JOIN tabela2;
```

Não possui um predicado de junção

```sql
SELECT
  lista_de_atributos
FROM
  tabela1,tabela2;
```

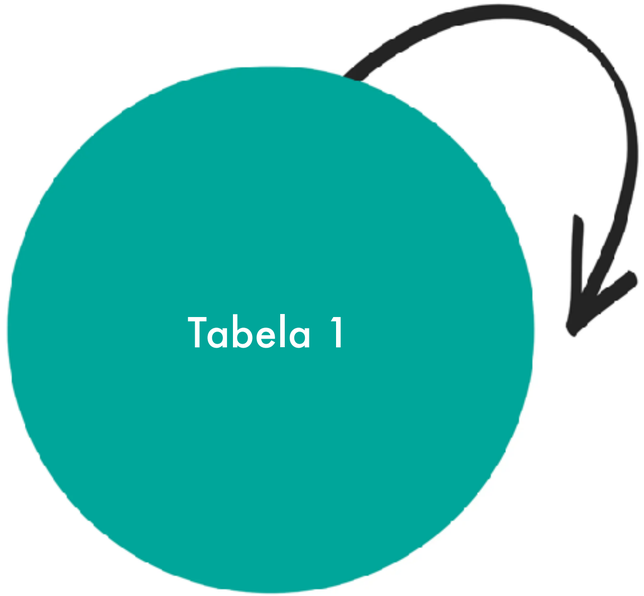
</ CODING SQL>

Considerando as tabelas funcionários e turnos de trabalho, se eu quisesse saber todas as combinações possíveis.

```
SELECT NOME, TURNO
FROM FUNCIONARIO
CROSS JOIN TURNO_DE_TRABALHO
```

SELF-JOIN

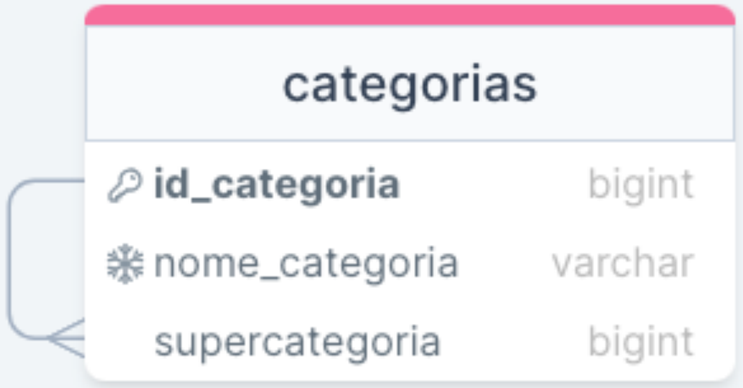
É uma junção regular que une uma tabela a si mesma



</ CODING SQL>

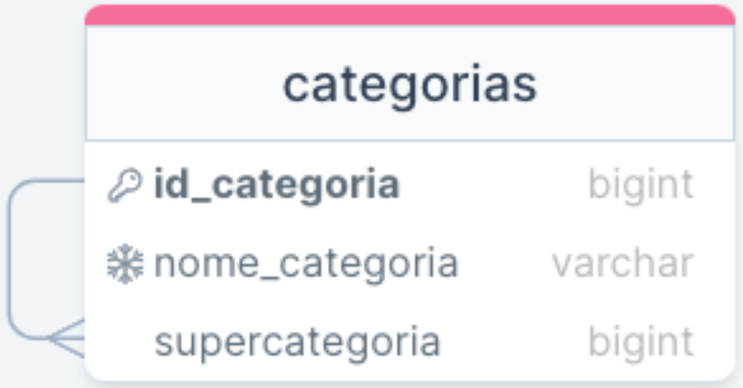
```sql
SELECT lista_de_atributos
FROM nome_da_tabela t1
INNER JOIN nome_da_tabela t2
ON predicado_de_junção;
```

</ CODING SQL>




```
CREATE TABLE CATEGORIAS(
    ID_CATEGORIA INT NOT NULL PRIMARY KEY,
    NOME_CATEGORIA VARCHAR NOT NULL,
    SUPERCATEGORIA INT,
    CONSTRAINT CATEGORIAS_FK FOREIGN KEY(SUPERCATEGORIA)
    REFERENCES CATEGORIAS(ID_CATEGORIA)
);
```

</ CODING SQL>




```
INSERT INTO CATEGORIAS VALUES(1, 'ELETRODOMÉSTICOS', NULL);
INSERT INTO CATEGORIAS VALUES(2, 'MÓVEIS', NULL);
INSERT INTO CATEGORIAS VALUES(3, 'GELADEIRA', 1);
INSERT INTO CATEGORIAS VALUES(4, 'MESA DE JANTAR', 2);
```

</ CODING SQL>

Selecione os nomes das categorias com as suas supercategorias

```
SELECT
  S.NOME_CATEGORIA || ' - ' || C.NOME_CATEGORIA AS CATEGORIAS
FROM
  CATEGORIAS S
  INNER JOIN CATEGORIAS C ON C.ID_CATEGORIA = S.SUPERCATEGORIA
ORDER BY
  C.NOME_CATEGORIA;
```