awari

SQL Intermediário II

Data Analytics | Aula 7

Subqueries

Definição

- Subqueries ou subconsultas são consultas dentro de outras consultas
- Retorna/Utiliza dados de múltiplas tabelas em uma mesma consulta
- É possível implementar subqueries como filtros ou como colunas

Exemplo

SELECT * FROM PESSOAS;

ID	NOME	IDADE	FILHOS
1	Eliana	20	0
2	Miguel	39	3
3	Raquel	36	2
4	João	22	1
5	Marta	48	2

SELECT * FROM COMPRAS;

ID	ID_PESSOA	DATA_COMPRA	VALOR
1	1	2018-09-19	10.99
2	2	2018-09-26	16.50
3	3	2018-10-24	18.00
4	3	2018-11-06	10.99
5	6	2018-12-20	17.99

Exemplo 1 - Subquery como coluna - Parte 1

```
SELECT ID, NOME,
        (SELECT COUNT(*)
        FROM COMPRAS
        WHERE ID_PESSOA = ID) AS TOTAL_COMPRAS
FROM PESSOAS;
```

Qual é a coluna **ID**?
COMPRAS ou **PESSOAS**?

Exemplo 1 - Subquery como coluna - Parte 2

```
SELECT ID, NOME
, (SELECT COUNT(*)
FROM COMPRAS
WHERE COMPRAS.ID_PESSOA = PESSOAS.ID) AS TOTAL_COMPRAS
FROM PESSOAS;
```

ID	NOME	TOTAL_COMPRAS
1	Eliana	1
2	Miguel	1
3	Raquel	2
4	João	NULL
5	Marta	NULL

Exemplo 1 - Subquery como coluna - Parte 3

```
SELECT ID, NOME
, (SELECT COUNT(*)
FROM COMPRAS AS T2
WHERE T2.ID_PESSOA = T1.ID) AS TOTAL_COMPRAS
FROM PESSOAS AS T1;
```

ID	NOME	TOTAL_COMPRAS
1	Eliana	1
2	Miguel	1
3	Raquel	2
4	João	NULL
5	Marta	NULL

Exemplo 2 - Subquery como filtro - Parte 1

SELECT * FROM COMPRAS;

ID	ID_PESSOA	DATA_COMPRA	VALOR
1	1	2018-09-19	10.99
2	2	2018-09-26	16.50
3	3	2018-10-24	18.00
4	3	2018-11-06	10.99
5	6	2018-12-20	17.99

1, 2, 3, 6

Exemplo 2 - Subquery como filtro - Parte 2

```
SELECT ID, NOME
FROM PESSOAS
WHERE ID IN (1, 2, 3, 6);
```

ID	NOME
1	Eliana
2	Miguel
3	Raquel

```
SELECT
        DISTINCT ID_PESSOA
FROM COMPRAS;
```

ID_PESSOA
1
2
3
6

```
SELECT ID, NOME
FROM PESSOAS
WHERE ID IN (SELECT DISTINCT
ID_PESSOA FROM COMPRAS);
```

ID	NOME
1	Eliana
2	Miguel
3	Raquel

Importante

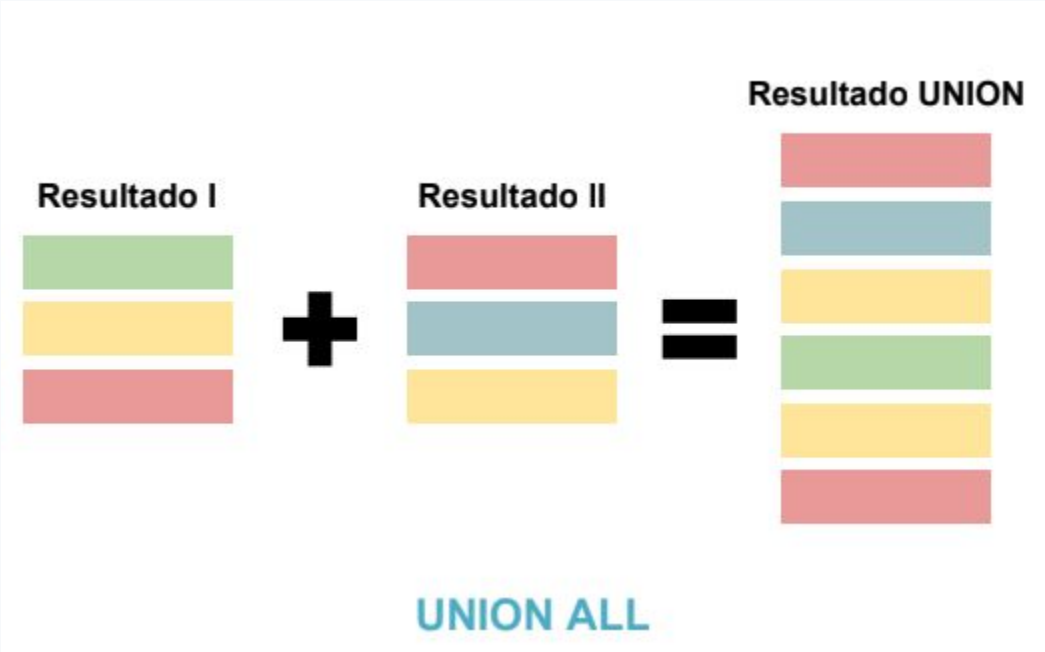
- Não existe limite de quantidade de subconsultas
- É possível escrever dezenas de subconsultas
- Consultas, dentro de consultas, dentro de...
- **Subqueries interferem no desempenho**
- Muitos níveis ou subconsultas podem afetar o tempo de resposta
- Muitas subconsultas pode dificultar a manutenção do código

UNION e UNION ALL

Definição

- Combina linhas do resultado de duas ou mais consultas
- Os resultados devem possuir:
  - O mesmo número de colunas
  - Colunas com tipos similares na mesma ordem

Tipos de Union



Exemplo - Union All

SELECT * FROM PESSOAS;

ID	NOME	IDADE	FILHOS
1	Eliana	20	0
2	Miguel	39	3
3	Raquel	36	2
4	João	22	1
5	Marta	48	2

Exemplo - Union All

SELECT * FROM COMPRAS;

ID	ID_PESSOA	DATA_COMPRA	VALOR
1	1	2018-09-19	10.99
2	2	2018-09-26	16.50
3	3	2018-10-24	18.00
4	3	2018-11-06	10.99
5	6	2018-12-20	17.99

Exemplo - Union All

SELECT ID_PESSOA FROM COMPRAS;

ID_PESSOA
1
2
3
3
6

SELECT ID FROM PESSOAS;

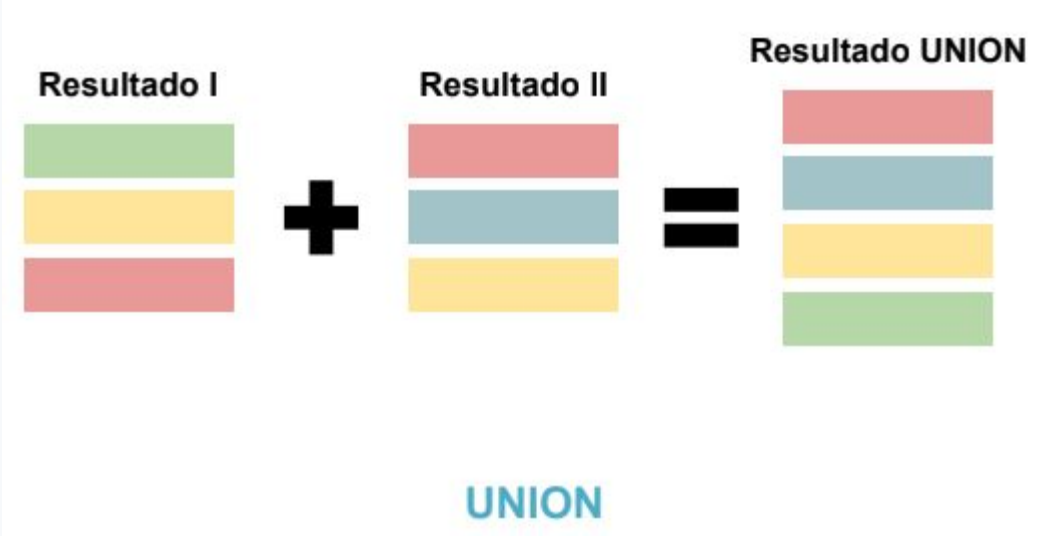
ID
1
2
3
4
5

Exemplo - Union All

```
SELECT ID_PESSOA FROM COMPRAS
UNION ALL
SELECT ID FROM PESSOAS;
```

ID_PESSOA
1
2
3
3
6
1
2
3
4
5

Tipos de Union



Exemplo - Union

```
SELECT ID_PESSOA FROM COMPRAS
UNION
SELECT ID FROM PESSOAS;
```

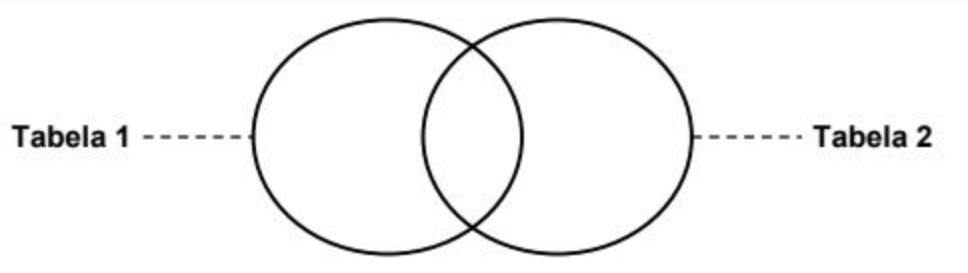
ID_PESSOA
1
2
3
6
4
5

JOINs

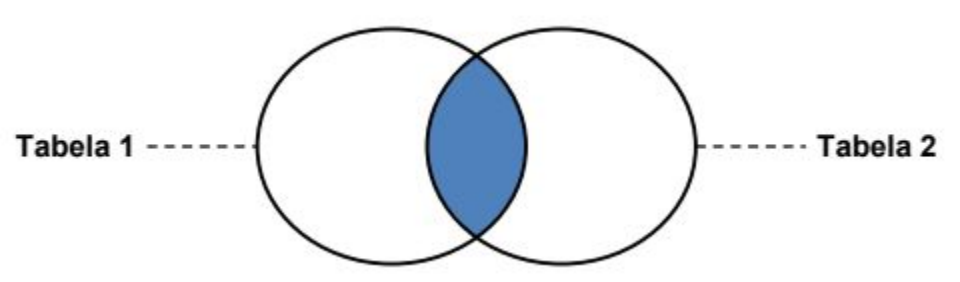
Definição

- Combina linhas de duas ou mais tabelas.
- A relação é baseada nas colunas.
- Apesar de muito útil, processar JOINS sobre grandes volumes de dados tem **custo computacional muito alto.**

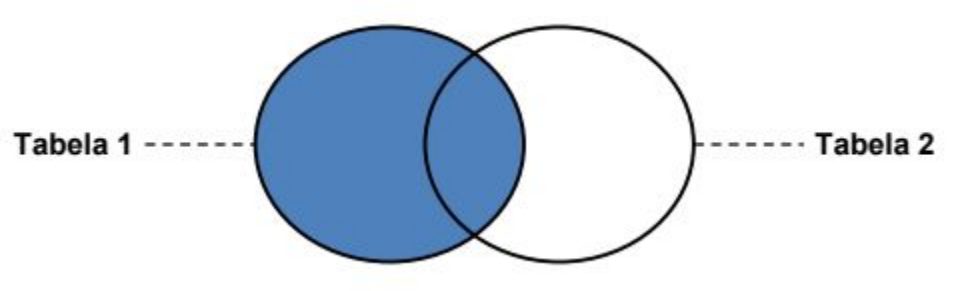
Formas de Join



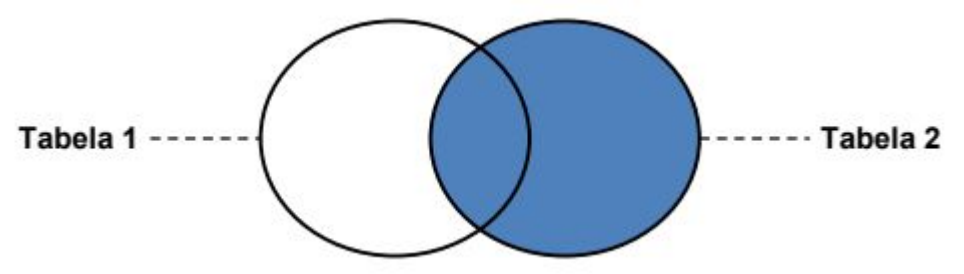
INNER JOIN



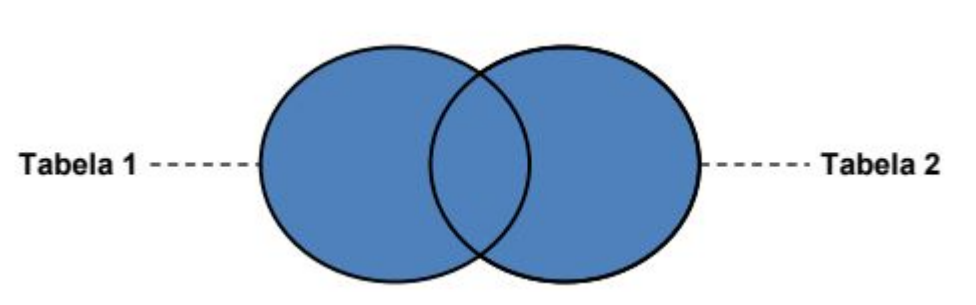
LEFT JOIN



RIGHT JOIN



FULL OUTER JOIN



Tabelas para o Exemplo

Tabela **alimento**

codigo_tipo_alimento	nome
null	Tomate
2	Rúcula
1	Maçã
3	Salgadinho
2	Alface
1	Laranja
4	Guarana
1	Banana

Tabela **tipoalimento**

codigo	nome
2	Verdura
1	Fruta
3	Industrializado
5	Legume
4	Bebida

SCRIPT de Criação das Tabelas

```
DROP TABLE IF EXISTS `awari-data-analytics-2023.datasets_awari.alimento`;
DROP TABLE IF EXISTS `awari-data-analytics-2023.datasets_awari.tipoalimento`;

CREATE TABLE `awari-data-analytics-2023.datasets_awari.tipoalimento` (
    codigo integer,
    nome string
);

CREATE TABLE `awari-data-analytics-2023.datasets_awari.alimento`(
   codigo_tipo_alimento integer,
   nome string
);
```

SCRIPT de Inserção de Dados na tabela “tipoalimento”

```
INSERT INTO `awari-data-analytics-2023.datasets_awari.tipoalimento`(codigo, nome) VALUES (1,'Fruta');
INSERT INTO `awari-data-analytics-2023.datasets_awari.tipoalimento`(codigo, nome) VALUES (2,'Verdura');
INSERT INTO `awari-data-analytics-2023.datasets_awari.tipoalimento`(codigo, nome) VALUES (3,'Industrializado');
INSERT INTO `awari-data-analytics-2023.datasets_awari.tipoalimento`(codigo, nome) VALUES (4,'Bebida');
INSERT INTO `awari-data-analytics-2023.datasets_awari.tipoalimento`(codigo, nome) VALUES (5,'Legume');
```

SCRIPT de Inserção de Dados na tabela “alimento”

```
INSERT INTO `awari-data-analytics-2023.datasets_awari.alimento`(nome, codigo_tipo_alimento) VALUES ('Guarana', 4);
INSERT INTO `awari-data-analytics-2023.datasets_awari.alimento`(nome, codigo_tipo_alimento) VALUES ('Laranja', 1);
INSERT INTO `awari-data-analytics-2023.datasets_awari.alimento`(nome, codigo_tipo_alimento) VALUES ('Maçã', 1);
INSERT INTO `awari-data-analytics-2023.datasets_awari.alimento`(nome, codigo_tipo_alimento) VALUES ('Salgadinho', 3);
INSERT INTO `awari-data-analytics-2023.datasets_awari.alimento`(nome, codigo_tipo_alimento) VALUES ('Alface', 2);
INSERT INTO `awari-data-analytics-2023.datasets_awari.alimento`(nome, codigo_tipo_alimento) VALUES ('Tomate', NULL);
INSERT INTO `awari-data-analytics-2023.datasets_awari.alimento`(nome, codigo_tipo_alimento) VALUES ('Rúcula', 2);
INSERT INTO `awari-data-analytics-2023.datasets_awari.alimento`(nome, codigo_tipo_alimento) VALUES ('Banana', 1);
```

Exemplo - INNER JOIN

```
SELECT *
FROM Alimento a
INNER JOIN TipoAlimento b
ON b.codigo = a.codigo_tipo_alimento
```

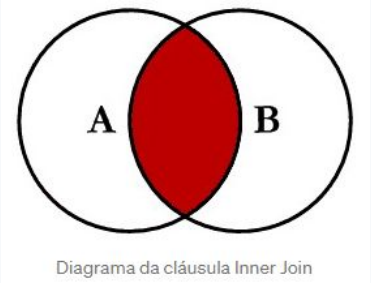


Diagrama da cláusula Inner Join

codigo_tipo_alimento	nome	codigo	nome_1
1	Laranja	1	Fruta
1	Maçã	1	Fruta
1	Banana	1	Fruta
2	Rúcula	2	Verdura
2	Alface	2	Verdura
4	Guarana	4	Bebida
3	Salgadinho	3	Industrializado

Explicação - INNER JOIN

A cláusula *Inner Join* tem o objetivo de encontrar todos os registros que possuem **algo** em comum entre as duas tabelas (esse **algo** é a ligação entre os campos ***codigo*** e ***codigo_tipo_alimento***). Podemos reparar que todos os registros listados nessa consulta possuem essa ligação, ou seja, essa consulta solicita ao banco de dados:

> *Retorne todos os registros da tabela **Alimento** que estão contidos na tabela **TipoAlimento**, e utiliza os campos **codigo** e **codigo_tipo_alimento** para criar a referência/ligação.*

O banco de dados analisa toda a tabela ***Alimento***, verificando cada registro. Vamos utilizar o "Guaraná" como exemplo:

Ao receber a instrução, o banco de dados verifica o registro "Guaraná" e valida o campo ***codigo_tipo_alimento*** (na tabela ele tem valor igual a 4). Após, procura na tabela *TipoAlimento* se possui algum registro com esse código (4), se encontrar ele traz na consulta, caso contrário temos o que chamamos de furo de banco de dados, mas isso é outro assunto.

Repare que no script foi utilizado a técnica de *alias,* onde damos um apelido para a tabela, dessa forma a tabela *Alimento* ficou com o apelido **a**, e a tabela *TipoAlimento* com o apelido **b**.

Exemplo - LEFT JOIN

```
SELECT *
FROM Alimento a
LEFT JOIN TipoAlimento ta
ON ta.codigo = a.codigo_tipo_alimento
```

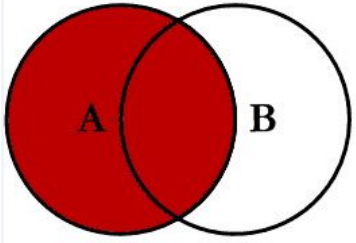


Diagrama da cláusula Left Join

codigo_tipo_alimento	nome	codigo	nome_1
null	Tomate	*null*	*null*
1	Laranja	1	Fruta
1	Maçã	1	Fruta
1	Banana	1	Fruta
2	Rúcula	2	Verdura
2	Alface	2	Verdura
3	Salgadinho	3	Industrializado
4	Guarana	4	Bebida

Explicação - LEFT JOIN

A cláusula *Left Join* retorna **todos** os registros da tabela da esquerda (tabela A - *alimento*), e também os registros que possuem ligação com os registros da tabela da direita (tabela B - *tipoalimento*). Podemos verificar que retornaram todos os registros da consulta *Inner Join,* mais um único registro, o “tomate” (em destaque), ele é o único registro da tabela da esquerda que possui valor **nulo** no campo *codigo_tipo_alimento*.

Exemplo - RIGHT JOIN

```
SELECT *
FROM Alimento a
RIGHT JOIN TipoAlimento ta
ON ta.codigo = a.codigo_tipo_alimento
```

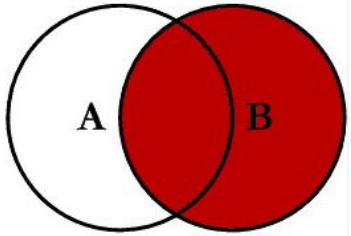


Diagrama da cláusula Right Join

codigo_tipo_alimento	nome	codigo	nome_1
1	Laranja	1	Fruta
1	Maçã	1	Fruta
1	Banana	1	Fruta
null	*null*	5	Legume
2	Rúcula	2	Verdura
2	Alface	2	Verdura
4	Guarana	4	Bebida
3	Salgadinho	3	Industrializado

Explicação - RIGHT JOIN

Esta cláusula é muito similar à anterior, mas ao invés da esquerda, agora olhando para a direita. A cláusula *Right Join* retorna **todos** os registros da tabela da direita, e também os registros que possuem ligação com os registros da tabela da esquerda.

Como no caso anterior, retornou todos os registros da consulta *Inner Join* e mais um único registro o “legume” (também em destaque), ele é o único registro da tabela da direita, porém nesse caso em particular, o “legume” é o único registro da tabela da direita que não possui nenhum vínculo com a tabela da esquerda.

Exemplo - FULL OUTER JOIN

```
SELECT *
FROM Alimento a
FULL OUTER JOIN TipoAlimento ta
ON ta.codigo = a.codigo_tipo_alimento
```

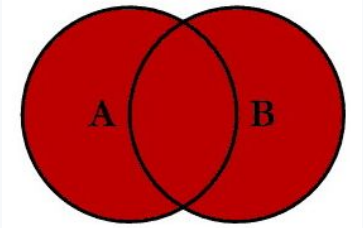


Diagrama da cláusula Full Outer Join

Linha	codigo_tipo_alimento	nome	codigo	nome_1
1	2	Rúcula	2	Verdura
2	2	Alface	2	Verdura
3	4	Guarana	4	Bebida
4	1	Laranja	1	Fruta
5	1	Maçã	1	Fruta
6	1	Banana	1	Fruta
7	*null*	Tomate	*null*	*null*
8	*null*	*null*	5	Legume
9	3	Salgadinho	3	Industrializado

Explicação - FULL OUTER JOIN

O diagrama dessa consulta fala por si só, essa cláusula irá retornar **tudo** de **todos,** sem exceção.
Repare que temos uma união das três últimas consultas, retornando todos os registros.
Não destaquei o resultado da consulta esperando que você analise a tabela e verifique que está tudo ali, “tomate”, “legume” e todo o resto.

Casos Extremos

Apesar das consultas acima serem amplamente usadas e compreenderem boa parte no que diz respeito a *joins*, ainda sim existem casos extremos em que queremos validar se um registro está somente em uma tabela e não está na outra.

Embora raramente aconteça, é interessante aprender e entender, caso um dia seja necessário aplicar.

Casos Extremos - LEFT JOIN

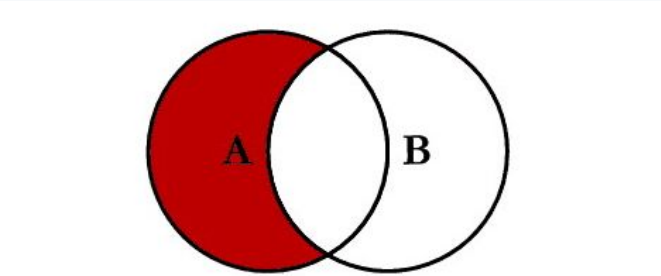


Diagrama da cláusula Left Join removendo os registros da tabela da direita

```
SELECT *
FROM Alimento a
LEFT JOIN TipoAlimento ta ON ta.codigo = a.codigo_tipo_alimento
WHERE ta.codigo is null
```

Linha	codigo_tipo_alimento	nome	codigo	nome_1
1	*null*	Tomate	*null*	*null*

Casos Extremos - LEFT JOIN

Analisando a consulta **anterior**, a primeira ideia que me vem na cabeça é a seguinte:

Por que não realizar uma consulta somente na tabela Alimento onde o campo codigo_tipo_alimento é igual a nulo?

Temos que nos lembrar que estamos falando de *joins*, ou seja, precisamos por algum motivo (regras de negócio) envolver outras tabelas. Essa consulta pergunta ao banco:

Quais registros destas duas tabelas estão somente na tabela da esquerda?

Isso pode ser usual em alguma validação para se certificar de que os registros estão apenas em uma tabela e não estão na outra, ou por qualquer outro motivo que julgar necessário validar os dados. De qualquer forma, essa consulta é possível e funciona.

Casos Extremos - RIGHT JOIN

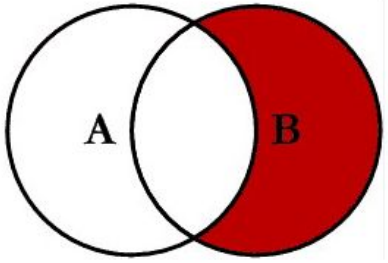


Diagrama da cláusula Right Join removendo os registros da tabela da esquerda

```
SELECT *
FROM Alimento a
RIGHT JOIN TipoAlimento ta ON ta.codigo = a.codigo_tipo_alimento
WHERE a.codigo_tipo_alimento is null
```

Linha	codigo_tipo_alimento	nome	codigo	nome_1
1	*null*	*null*	5	Legume

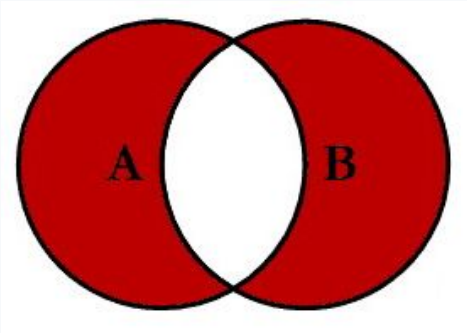
Casos Extremos - RIGHT JOIN

Aqui segue a mesma ideia da consulta anterior, mas olhando para a tabela da direita.

Podemos perceber que o único elemento que retornou foi o “legume”.

É interessante pensar que podemos fazer uma consulta para buscar dados de duas tabelas e retornar registros somente de uma.

Casos Extremos - FULL OUTER JOIN




```
SELECT *
FROM Alimento a
FULL OUTER JOIN TipoAlimento ta ON ta.codigo = a.codigo_tipo_alimento
WHERE ta.codigo is null or a.codigo_tipo_alimento is null
```

Linha	codigo_tipo_alimento	nome	codigo	nome_1
1	*null*	Tomate	*null*	*null*
2	*null*	*null*	5	Legume

Casos Extremos - FULL OUTER JOIN

Essa consulta é muito interessante e dentro do mundo das exceções talvez seja a mais usada.

Ela faz exatamente o oposto da cláusula *Inner Join*, por que ela retorna tudo o que existe em uma tabela mas não está contido na outra e vice-e-versa.

awari