trybe

Banco de Dados SQL

Bloco 21 - Aula 21.2

Roadmap da Aula

- INNER JOIN;
- LEFT E RIGHT JOIN;
- SELF JOIN;
- ATIVIDADE PRÁTICA;
- DESAFIOS;

CASO DE USO

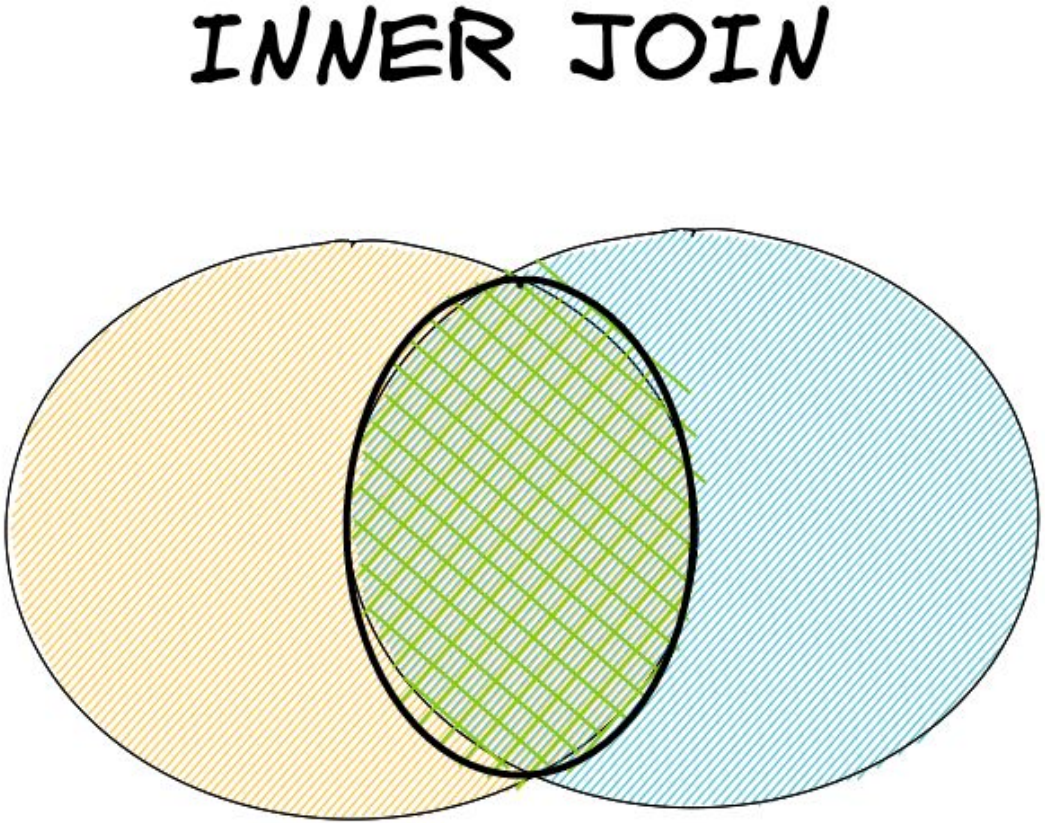
Pessoas_Colaboradoras

Id	Nome	Função
1	Renatão	P. Especialista
2	Coruja	P. Instrutora
3	Ítalo	P. Instrutora

Habilidades

Id	Titulo	Pessoa_Colaboradora_Id
1	React	1
2	Node	1
3	Node	2
4	Node	3

INNER JOIN



INNER JOIN

Select * from Pessoas_Colaboradodas INNER JOIN Habilidades

Id	Nome	Função	Id	Título	Pessoa_Colaboradora_Id
1	Renatão	P. Especialista	1	React	1
1	Renatão	P. Especialista	2	Node	1
1	Renatão	P. Especialista	3	Node	2
1	Renatão	P. Especialista	4	Node	3
2	Coruja	P. Instrutora	1	React	1
2	Coruja	P. Instrutora	2	Node	1
2	Coruja	P. Instrutora	3	Node	2
2	Coruja	P. Instrutora	4	Node	3
3	Ítalo	P. Instrutora	1	React	1
3	Ítalo	P. Instrutora	2	Node	1
3	Ítalo	P. Instrutora	3	Node	2
3	Ítalo	P. Instrutora	4	Node	3

INNER JOIN ... ON ...

```
Select * from Pessoas_Colaboradodas INNER JOIN Habilidades
on Pessoas_Colaboradodas.id = Habilidades.Pessoa_Colaboradora_id
```

Id	Nome	Função	Id	Título	Pessoa_Colaboradora_Id
1	Renatão	P. Especialista	1	React	1
1	Renatão	P. Especialista	2	Node	1
1	Renatão	P. Especialista	3	Node	2
1	Renatão	P. Especialista	4	Node	3
2	Coruja	P. Instrutora	1	React	1
2	Coruja	P. Instrutora	2	Node	1
2	Coruja	P. Instrutora	3	Node	2
2	Coruja	P. Instrutora	4	Node	3
3	Ítalo	P. Instrutora	1	React	1
3	Ítalo	P. Instrutora	2	Node	1
3	Ítalo	P. Instrutora	3	Node	2
3	Ítalo	P. Instrutora	4	Node	3

Id	Nome	Função	Id	Título	Pessoa_Colaboradora_Id
1	Renatão	P. Especialista	1	React	1
1	Renatão	P. Especialista	2	Node	1
2	Coruja	P. Instrutora	3	Node	2
3	Ítalo	P. Instrutora	4	Node	3

INNER JOIN ... ON ...

Id	Nome	Função	Id	Título	Pessoa_Colaboradora_Id
1	Renatão	P. Especialista	1	React	1
1	Renatão	P. Especialista	2	Node	1
2	Coruja	P. Instrutora	3	Node	2
3	Ítalo	P. Instrutora	4	Node	3

CASO DE USO PARA LEFT JOIN E RIGHT JOIN

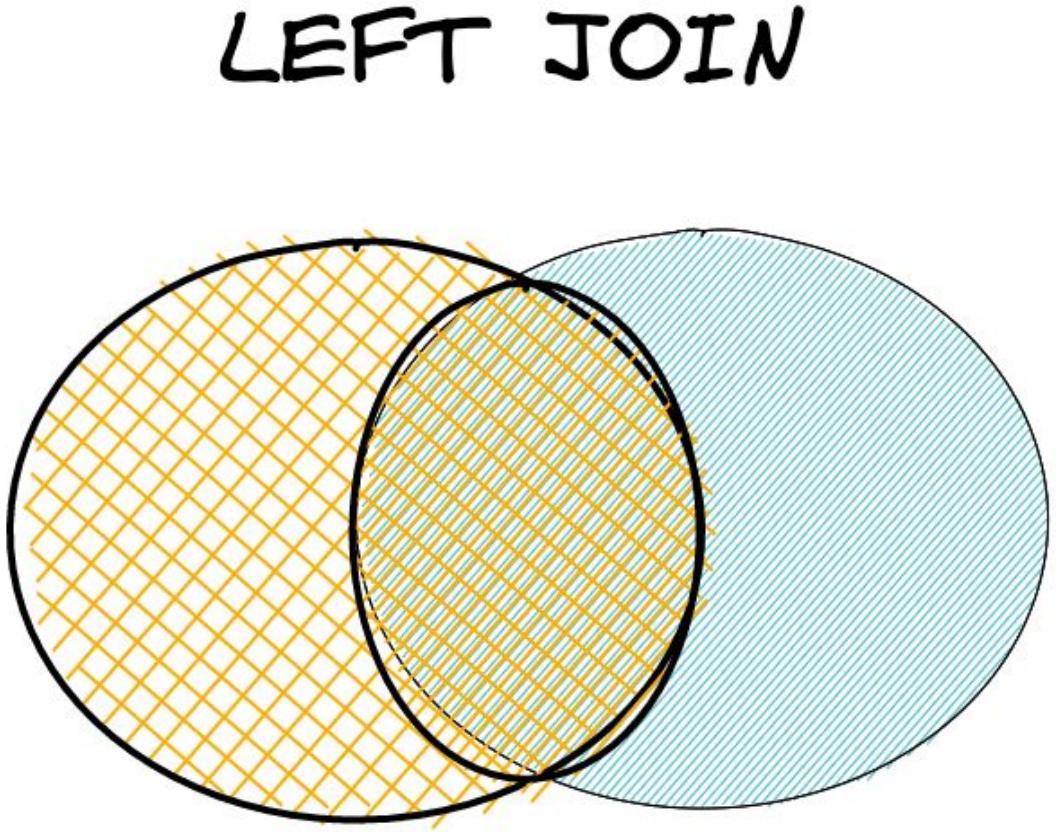
Pessoas_Colaboradoras

Id	Nome	Função
1	Renatão	P. Especialista
2	Coruja	P. Instrutora
3	Ítalo	P. Instrutora
4	Pedro	P. Instrutora

Habilidades

Id	Titulo	Pessoa_Colaboradora_Id
1	React	1
2	Node	1
3	Node	2
4	Node	3
5	C#	NULL

LEFT JOIN

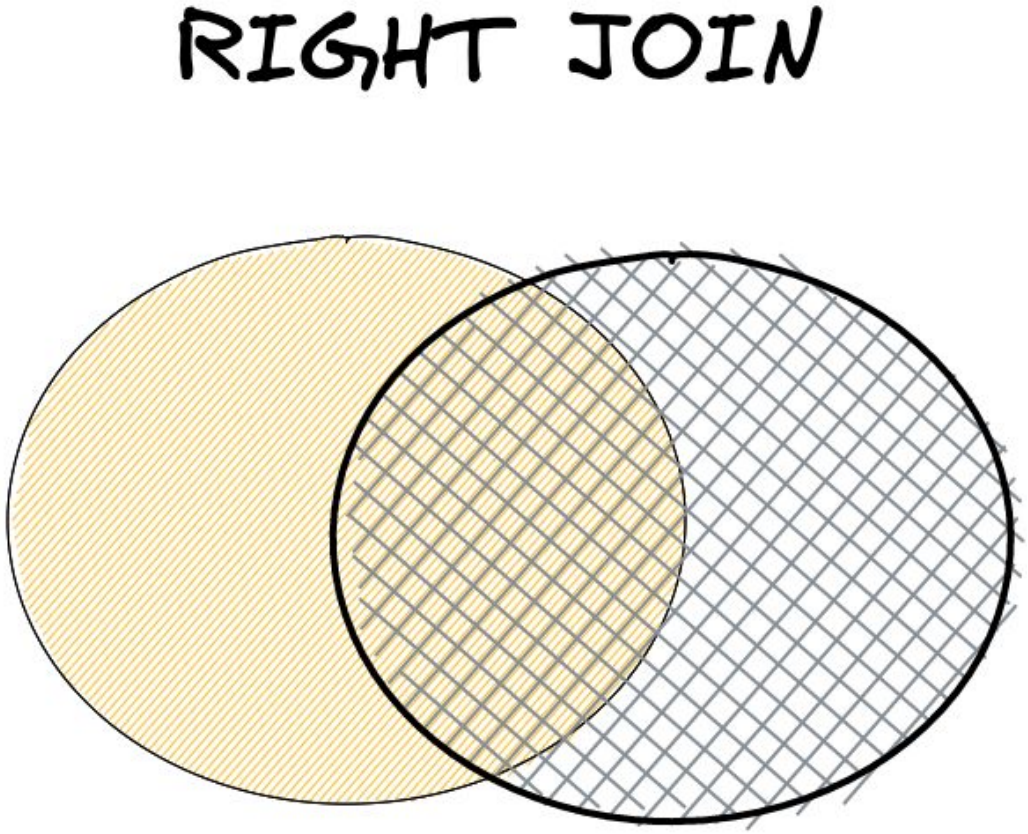


LEFT JOIN

```
Select * from Pessoas_Colaboradodas LEFT JOIN Habilidades
on Pessoas_Colaboradodas.id = Habilidades.Pessoa_Colaboradora_id
```

Id	Nome	Função	Id	Título	Pessoa_Colaboradora_Id
1	Renatão	P. Especialista	1	React	1
1	Renatão	P. Especialista	2	Node	1
2	Coruja	P. Instrutora	3	Node	2
3	Ítalo	P. Instrutora	4	Node	3
4	Pedro	P. Instrutora	NULL	NULL	NULL

RIGHT JOIN



RIGHT JOIN

```
Select * from Pessoas_Colaboradodas RIGHT JOIN Habilidades
on Pessoas_Colaboradodas.id = Habilidades.Pessoa_Colaboradora_id
```

Id	Nome	Função	Id	Título	Pessoa_Colaboradora_Id
1	Renatão	P. Especialista	1	React	1
1	Renatão	P. Especialista	2	Node	1
2	Coruja	P. Instrutora	3	Node	2
3	Ítalo	P. Instrutora	4	Node	3
NULL	NULL	NULL	5	C#	NULL

Desafios

BD PET

```sql
DROP DATABASE IF EXISTS `db`;
CREATE DATABASE `db`;
USE `db`;

CREATE TABLE `person` (
  `id` INT NOT NULL
AUTO_INCREMENT,
  `name` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`id`)
);

CREATE TABLE `pet` (
  `id` INT NOT NULL
AUTO_INCREMENT,
  `person_id` INT NULL,
  `type` VARCHAR(100),
  `name` VARCHAR(100),
  PRIMARY KEY (`id`),
  FOREIGN KEY (`person_id`)
REFERENCES `person` (`id`)
);
```

```sql
INSERT INTO `person` (`id`,
`name`) VALUES
  (1, 'John'),
  (2, 'Sarah'),
  (3, 'Rachel'),
  (4, 'Sam');

INSERT INTO `pet` (`id`,
`person_id`, `type`, `name`)
VALUES
  (1, 1, 'goldfish', 'Fishy'),
  (2, 1, 'goldfish', 'Nemo'),
  (3, 1, 'dog', 'Fido'),
  (4, 2, 'cat', 'Kitty'),
  (5, 2, 'bird', 'Feathers'),
  (6, 3, 'chinchilla', 'Fuzzy'),
  (7, NULL, 'iguana', 'Scales');
```

INNER JOIN

1. **Busque por todas pessoas e todos os pets que estejam associados**

BD PET

INNER JOIN

1. **Busque por todas pessoas e todos os pets que estejam associados**

```
SELECT person.name, pet.id, pet.person_id , pet.name
FROM person
INNER JOIN pet ON person.id = pet.person_id;
```

BD PET

INNER JOIN

2. Busque todos os nomes das pessoas que possuem pedidos realizados e as pessoas funcionárias que fizeram a operação

****BD STORE SAMPLE****

INNER JOIN

2. Busque todos os nomes das pessoas que possuem pedidos realizados e as pessoas funcionárias que fizeram a operação

```
SELECT Orders.OrderID, Customers.CustomerName, Employees.FirstName AS
'EmployeeFirstName'

FROM Orders

INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

****BD STORE SAMPLE****

INNER JOIN

3. Busque todos os nomes das pessoas que possuem pedidos realizados e as pessoas funcionárias que fizeram a operação e a empresa associada para o envio do produto

****BD STORE SAMPLE****

INNER JOIN

3. Busque todos os nomes das pessoas que possuem pedidos realizados e as pessoas funcionárias que fizeram a operação e a empresa associada para o envio do produto

```
SELECT Orders.OrderID, Customers.CustomerName, Employees.FirstName AS
'EmployeeFirstName', Shippers.ShipperName
FROM Orders
INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID;
```

BD STORE SAMPLE

INNER JOIN

4. Retorne o nome do produto, o valor, unidade e quem é o fornecedor associado e a categoria que ele se encontra

****BD STORE SAMPLE****

INNER JOIN

4. Retorne o nome do produto, o valor, unidade e quem é o fornecedor associado e a categoria que ele se encontra

```
SELECT Products.ProductName, Products.Unit, Products.Price,
Categories.CategoryName, Suppliers.SupplierName
FROM Products
INNER JOIN Categories ON Products.CategoryID = Categories.CategoryID
INNER JOIN Suppliers ON Products.SupplierID =  Suppliers.SupplierID;
```

BD STORE SAMPLE

INNER JOIN

5. Retorne o nome do produto, o valor, unidade e quem é o fornecedor associado e a categoria que ele se encontra agrupado por categoria e por fornecedor

****BD STORE SAMPLE****

INNER JOIN

5. Retorne o nome do produto, o valor, unidade e quem é o fornecedor associado e a categoria que ele se encontra agrupado por categoria e por fornecedor

```
SELECT Products.ProductName, Products.Unit, Products.Price,
Categories.CategoryName, Suppliers.SupplierName, COUNT(Products.ProductID)
FROM Products
INNER JOIN Categories ON Products.CategoryID = Categories.CategoryID
INNER JOIN Suppliers ON Products.SupplierID = Suppliers.SupplierID
GROUP BY Categories.CategoryName, Suppliers.SupplierName
```

****BD STORE SAMPLE****

LEFT JOIN

6. Busque por todas pessoas e faça a relação com seus pets, trazendo inclusive as pessoas que não possuem pets associados

****BD PET****

LEFT JOIN

6. Busque por todas pessoas e faça a relação com seus pets, trazendo inclusive as pessoas que não possuem pets associados

```
SELECT person.id AS 'ID_PERSON_PK', person.name AS 'PERSON NAME',
pet.person_id AS 'ID_PERSON_FK', pet.name AS 'PET NAME', pet.id AS
'ID_PET_PK'

FROM db.person
LEFT JOIN db.pet ON person.id = pet.person_id;
```

BD PET

LEFT JOIN

7. Retorne o CustomerName e OrderID de todas os customers mesmo que nunca tenha sido feita uma compra

****BD STORE SAMPLE****

LEFT JOIN

7. Retorne o CustomerName e OrderID de todas os customers mesmo que nunca tenha sido feita uma compra

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```

****BD STORE SAMPLE****

RIGHT JOIN

8. Busque por todos os pets e faça a relação com as pessoas, trazendo inclusive pets que não possuem pessoas associadas

BD PET

RIGHT JOIN

8. Busque por todos os pets e faça a relação com as pessoas, trazendo inclusive pets que não possuem pessoas associadas

```
SELECT person.id AS 'ID_PERSON_PK', person.name AS 'PERSON NAME',
pet.person_id AS 'ID_PERSON_FK', pet.name AS 'PET NAME', pet.id AS
'ID_PET_PK'

FROM db.person
RIGHT JOIN db.pet ON person.id = pet.person_id;
```

BD PET

RIGHT JOIN

9. Retorne o OrderID, Employees LastName e FistName de TODOS os employees, mesmo que não haja nenhuma venda (order)

****BD STORE SAMPLE****

RIGHT JOIN

9. Retorne o OrderID, Employees LastName e FistName de TODOS os employees, mesmo que não haja nenhuma venda (order)

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```

****BD STORE SAMPLE****

SELF JOIN

10. Retorne o CustomerName e City de todos os customers quais são da mesma cidade

****BD STORE SAMPLE****

SELF JOIN

10. Retorne o CustomerName e City de todos os customers quais são da mesma cidade

```
SELECT A.CustomerName AS CustomerName1, B.CustomerName AS
CustomerName2, A.City
FROM Customers A, Customers B
WHERE A.CustomerID <> B.CustomerID
AND A.City = B.City
ORDER BY A.City;
```

BD STORE SAMPLE

FULL JOIN

11. Retorne todos os dados das tabelas orders e customers e sua interligação

BD STORE SAMPLE

FULL JOIN

11. Retorne todos os dados das tabelas orders e customers e sua interligação

```sql
SELECT *
FROM customers
LEFT OUTER JOIN orders ON customers.customerid = orders.customerid
UNION
SELECT *
FROM orders
RIGHT OUTER JOIN customers ON customers.customerid = orders.customerid
```

BD STORE SAMPLE

DESAFIO FINAL

12. Retorne o nome do produto, a quantidade da compra, nome da pessoa compradora, o nome completo da pessoa funcionária e a distribuidora de todas as compras.

****BD STORE SAMPLE****

DESAFIO FINAL

BD STORE SAMPLE

12. Retorne o nome do produto, a quantidade da compra, nome da pessoa compradora, o nome completo da pessoa funcionária e a distribuidora de todas as compras.

```
SELECT products.productname, order_details.quantity,
customers.customername,
concat(employees.firstname, ' ', employees.lastname),
shippers.shippername
FROM `store-sample`.order_details
INNER JOIN products ON products.productid = order_details.productid
INNER JOIN orders ON orders.orderid = order_details.orderid
INNER JOIN customers ON orders.customerid = customers.customerid
INNER JOIN employees ON orders.employeeid = employees.employeeid
INNER JOIN shippers ON orders.shipperid = shippers.shipperid
```

EXTRAS

```
CREATE DATABASE IF NOT EXISTS db;

USE db;

CREATE TABLE friends (friend_id INT, friend_name VARCHAR(100));
CREATE TABLE pets (pet_id INT,  owner_id INT,  pet_type VARCHAR(100),
pet_name VARCHAR(100));
```

BD PET_FRIEND

EXTRAS

```sql
INSERT INTO friends VALUES(1, 'John');

INSERT INTO friends VALUES(2, 'Sarah');

INSERT INTO friends VALUES(3, 'Rachel');

INSERT INTO friends VALUES(4, 'Sam');

INSERT INTO pets VALUES(1, 1, 'goldfish', 'Fishy');

INSERT INTO pets VALUES(2, 1, 'goldfish', 'Nemo');

INSERT INTO pets VALUES(3, 1, 'dog', 'Fido');

INSERT INTO pets VALUES(4, 2, 'cat', 'Kitty');

INSERT INTO pets VALUES(5, 2, 'bird', 'Feathers');

INSERT INTO pets VALUES(6, 3, 'chinchilla', 'Fuzzy');
INSERT INTO pets VALUES(7, NULL, 'iguana', 'Scales');
```

****BD PET_FRIEND****

DESAFIO EXTRA 01

- **Retorne a lista de endereços trazendo os dados do logradouro (address), bairro (district) e o nome da cidade (city) de todos os endereços da cidade com id 449.**

```sql
SELECT A.address, A.district, A.city_id, C.city
FROM sakila.address as A
INNER JOIN sakila.city AS C ON A.city_id = C.city_id
WHERE A.city_id = 449;
```

****BD SAKILA****

DESAFIO EXTRA 02

- **Busque todos friends, mesmo que não hajam pets associados**

```
SELECT F.friend_name, P.pet_id, P.owner_id, P.pet_name
FROM friends AS F
LEFT JOIN pets AS P
ON F.friend_id = P.owner_id;
```

****BD PET_FRIEND****

DESAFIO EXTRA 03

- **Busque todos pets, mesmo que não hajam friends associados**

```
SELECT F.friend_id, F.friend_name, P.pet_id,
P.owner_id, P.pet_name
FROM friends AS F
RIGHT JOIN pets AS P ON F.friend_id = P.owner_id;
```

BD PET_FRIEND