



**CWI SOFTWARE**

módulo 2 | banco de dados

**Crescer 2016**



## 4 - Comandos SQL – Joins

André Luís Nunes

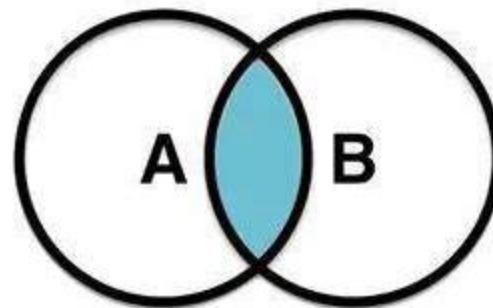
## RELACIONAMENTOS: tipos

Para consultar informações de mais de uma tabela é preciso relacioná-las, para isso devemos utilizar o seguinte:

- **Inner Join:** permite relacionar todos os registros de uma tabela comparando com outra tabela.
- **Outer (right) Join:** força o retorno de registros de uma tabela, mesmo que não exista um registro correspondente na outra.
- **Sub-query:** permite relacionar a existência (ou negação) de registros com outra consulta. Normalmente utilizada com EXISTS.
- **Cross join:** produto cartesiano de uma consulta.

# RELACIONAMENTO: inner join

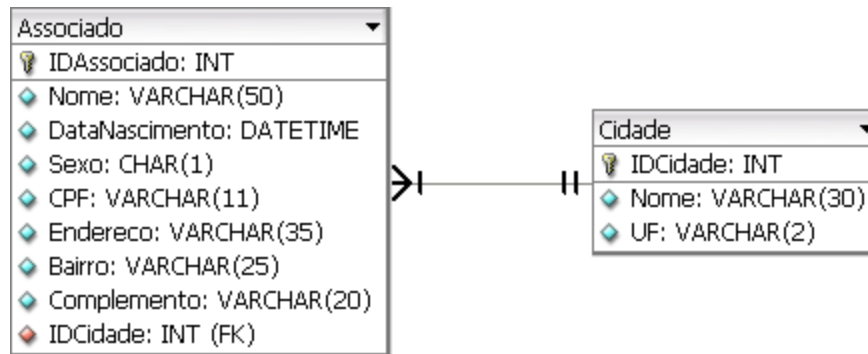
Relacionamento mais comum:



```
SELECT <fields list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```

# RELACIONAMENTO: inner join

Relacionamento mais comum.



- » Quando uma consulta é realizada sobre mais de uma tabela, relacionando-os com joins é necessário especificar a qual tabela pertence a coluna.
- » Se o campo possuir o mesmo nome em mais de uma tabela ocorrerá erro se não for informada a tabela.

# RELACIONAMENTO:

## inner join

Sintaxe do Comando:

```
Select A.<coluna>, D.<coluna>
  From <tabelaPrincipal> A
  Inner Join <tabelaSecundaria> D
            on A.<IDTabelaD> = D.<IDTabelaD>
  Where <condicao>
  Group by <coluna>
  Having <condicao>
  Order by <coluna>
```

# RELACIONAMENTO: inner join

Recuperando todos os associados que tenham Cidade cadastrada:

```
Select a.Nome as NomeAssociado,  
       c.Nome as NomeCidade  
From Associado a  
     INNER JOIN Cidade c ON c.IDCidade = a.IDCidade
```

Inner join explícito

Outra forma de escrever esta consulta é através do Where, as tabelas são adicionadas na cláusula WHERE (separadas por vírgula) e os relacionamentos são escritos no WHERE ("misturados" com demais filtros da pesquisa):

```
Select a.Nome as NomeAssociado,  
       c.Nome as NomeCidade  
From Associado a, Cidade c  
     WHERE c.IDCidade = a.IDCidade
```

Inner join implícito

# Banco de dados | relacional (SQL)

## Relação de conjuntos de dados

Cliente

Cidade



# Banco de dados | relacional (SQL)

## Relação de conjuntos de dados

### Cliente

IDCliente	Nome
1	Pedro
2	Maria
3	Julia
10	Carlos
15	Antônio

### Cidade

IDCidade	Nome	UF
1	Porto Alegre	RS
2	São Paulo	SP
3	Sapucaia do Sul	RS

# Banco de dados | relacional (SQL)

## Relação de conjuntos de dados

### Cliente

IDCliente	Nome
1	Pedro
2	Maria
3	Julia
10	Carlos
15	Antônio

Chave primária (PK)

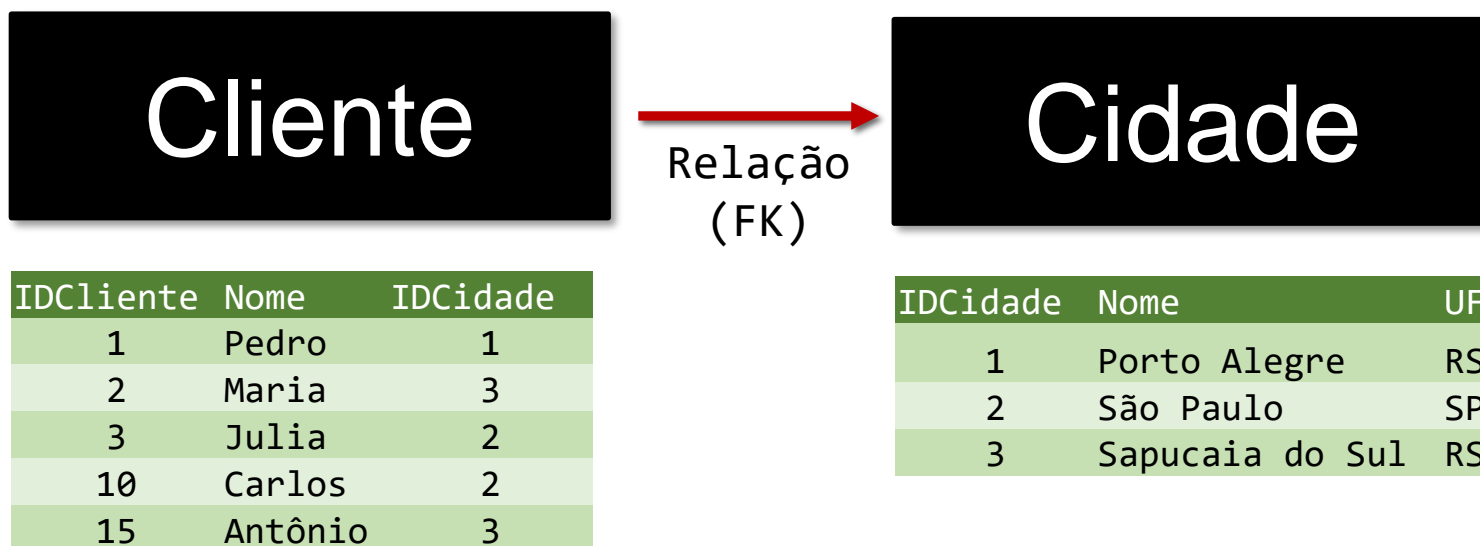
### Cidade

IDCidade	Nome	UF
1	Porto Alegre	RS
2	São Paulo	SP
3	Sapucaia do Sul	RS

Chave primária (PK)

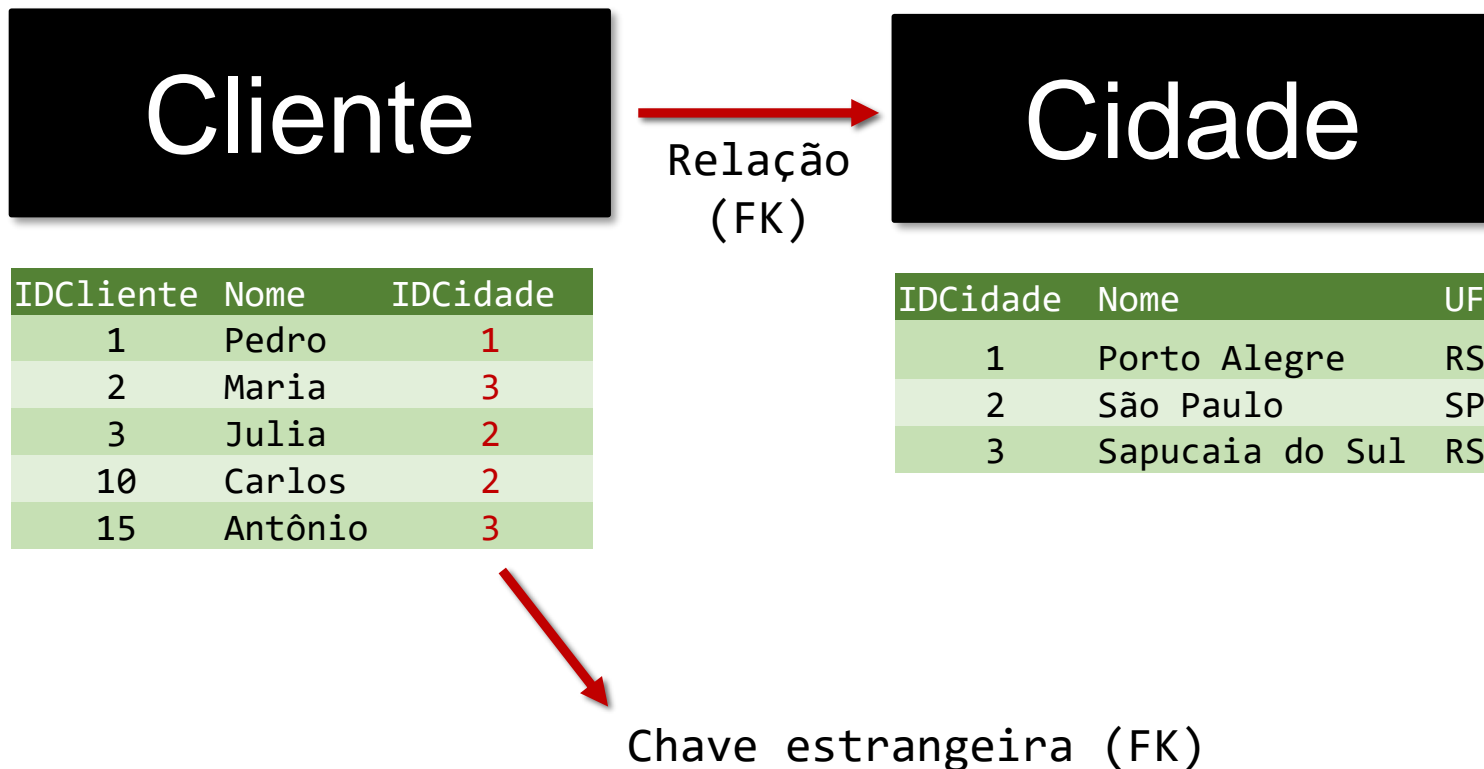
# Banco de dados | relacional (SQL)

## Relação de conjuntos de dados



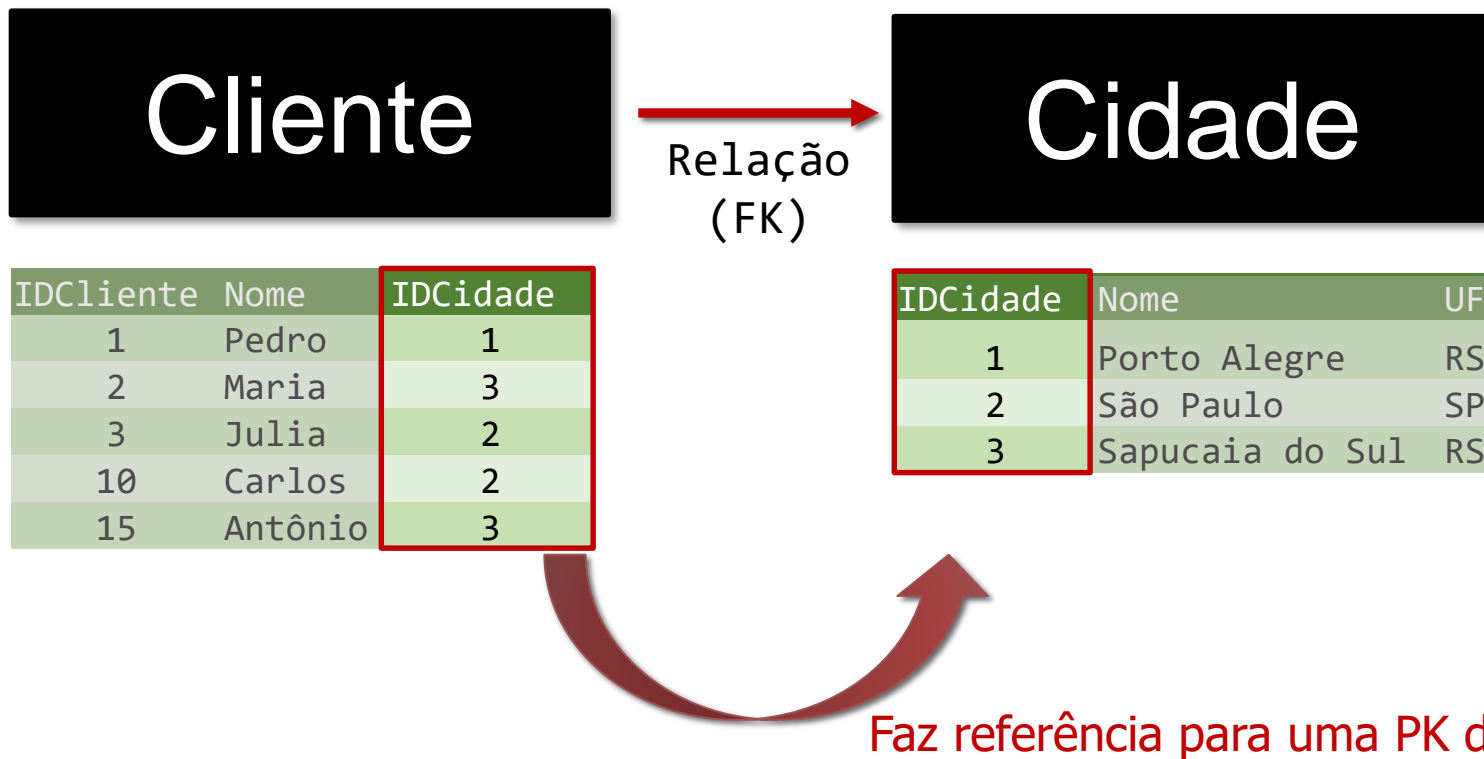
# Banco de dados | relacional (SQL)

## Relação de conjuntos de dados



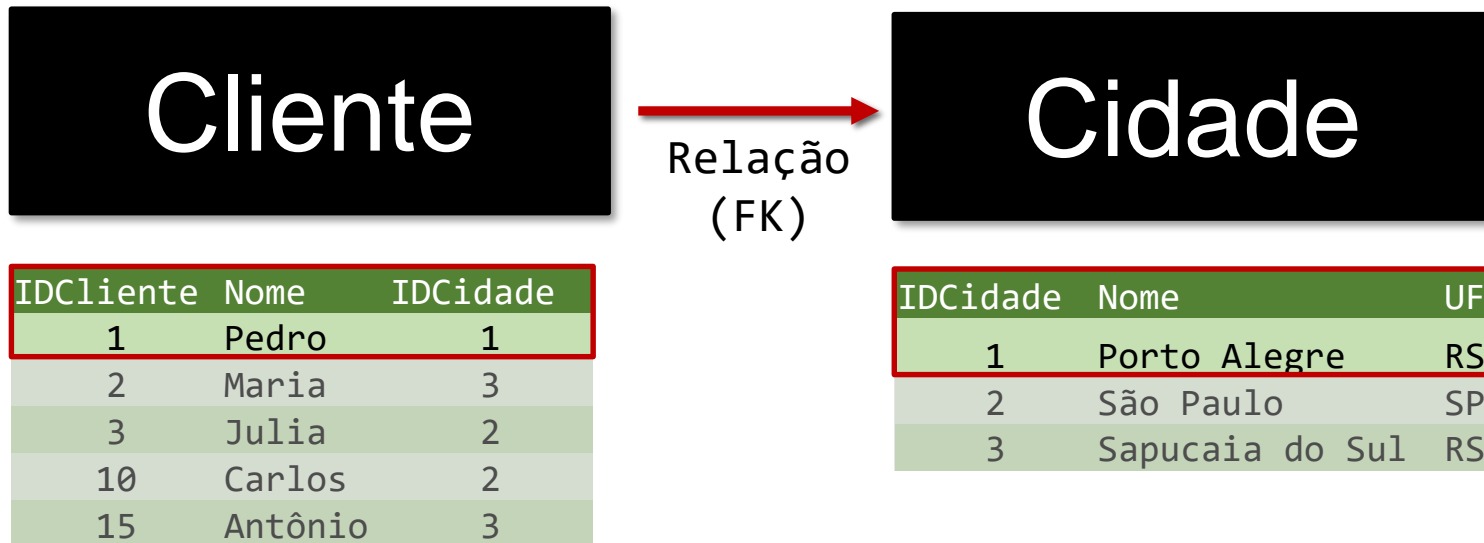
# Banco de dados | relacional (SQL)

## Relação de conjuntos de dados



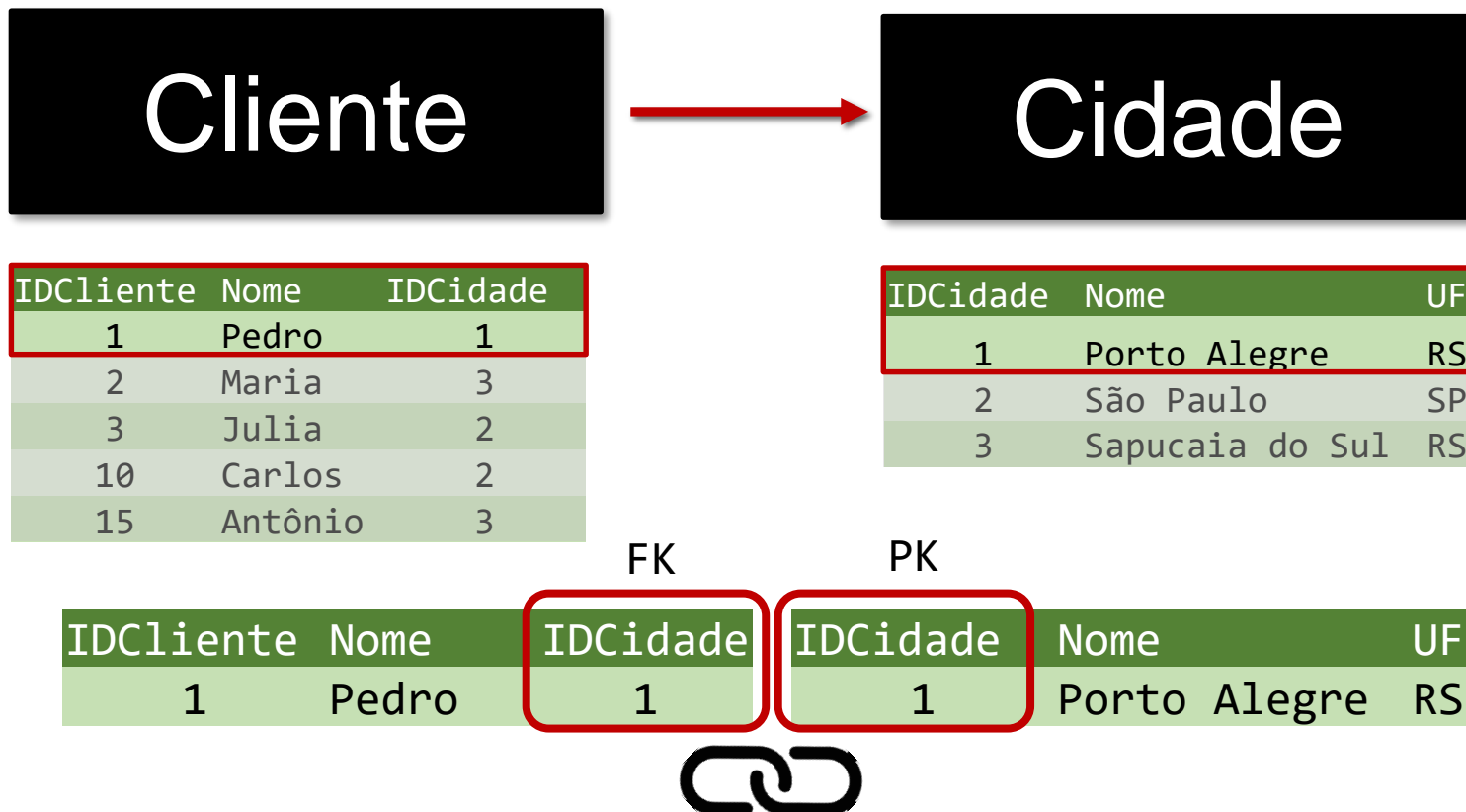
# Banco de dados | relacional (SQL)

## Relação de conjuntos de dados



# Banco de dados | relacional (SQL)

## Relação de conjuntos de dados

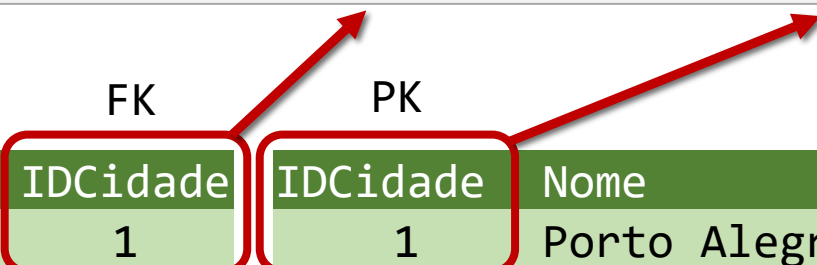


# Banco de dados | relacional (SQL)

## Relacionando 2 tabelas (joins)

```
select cliente.IDCliente,  
       cliente.Nome as nome_cliente,  
       cliente.IDCidade,  
       cidade.IDCidade,  
       cidade.Nome as nome_cidade,  
       cidade.UF  
from   cliente  
       inner join cidade on cliente.idcidade = cidade.idcidade;
```

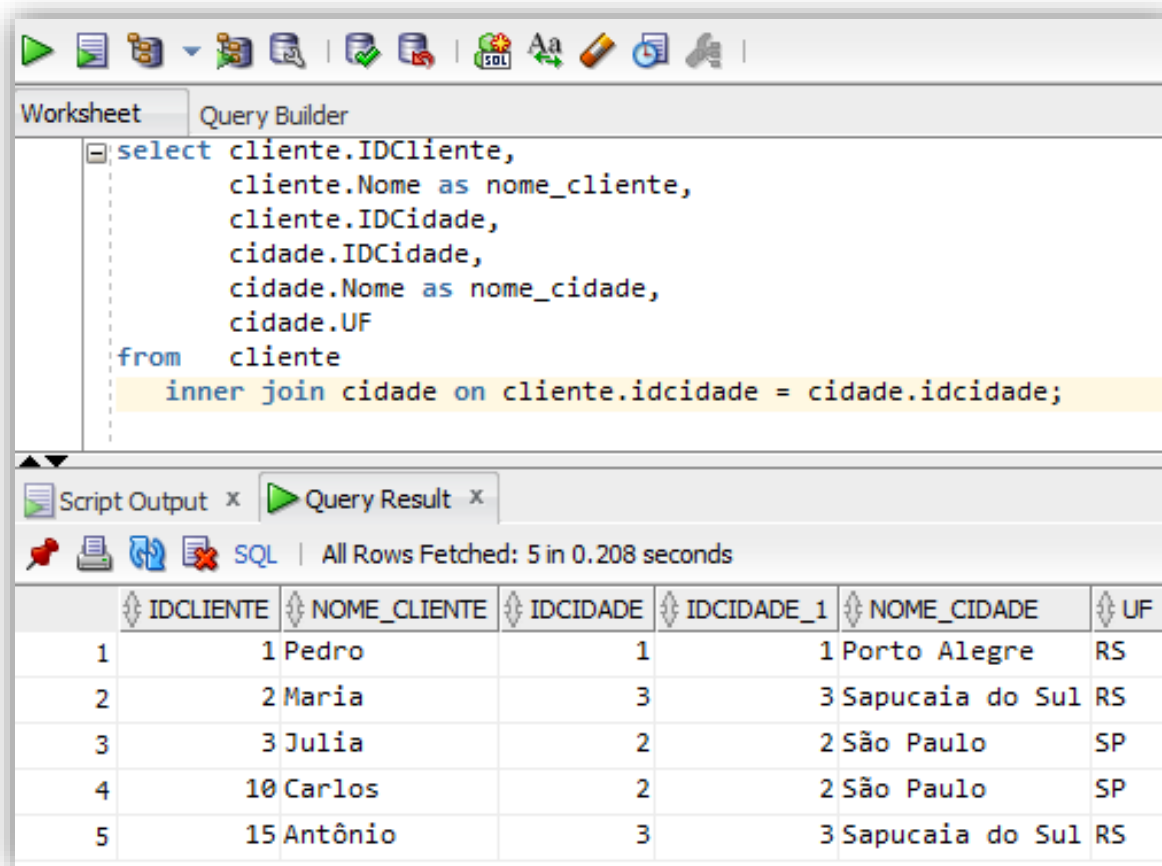
IDCliente	Nome	IDCidade	IDCidade	Nome	UF
1	Pedro	1	1	Porto Alegre	RS





# Banco de dados | relacional (SQL)

## Relação de conjuntos de dados



The screenshot displays a SQL Query Builder interface. The top toolbar includes icons for running queries, saving, and other database operations. The main area is divided into a 'Worksheet' tab and a 'Query Builder' tab. The 'Query Builder' tab shows a SQL query:

```
select cliente.IDCliente,
       cliente.Nome as nome_cliente,
       cliente.IDCidade,
       cidade.IDCidade,
       cidade.Nome as nome_cidade,
       cidade.UF
from   cliente
       inner join cidade on cliente.idcidade = cidade.idcidade;
```

Below the query editor, there are tabs for 'Script Output' and 'Query Result'. The 'Query Result' tab is active, showing a table with 5 rows and 7 columns. The status bar indicates 'All Rows Fetched: 5 in 0.208 seconds'.

	IDCLIENTE	NOME_CLIENTE	IDCIDADE	IDCIDADE_1	NOME_CIDADE	UF
1	1	Pedro	1	1	Porto Alegre	RS
2	2	Maria	3	3	Sapucaia do Sul	RS
3	3	Julia	2	2	São Paulo	SP
4	10	Carlos	2	2	São Paulo	SP
5	15	Antônio	3	3	Sapucaia do Sul	RS

# Banco de dados | relacional (SQL)

## Join (junção)

Cliente		
IDCliente	Nome	IDCidade
1	Pedro	1
2	Maria	3
3	Julia	2
10	Carlos	2
15	Antônio	3

Cidade		
IDCidade	Nome	UF
1	Porto Alegre	RS
2	São Paulo	SP
3	Sapucaia do Sul	RS

Para cada registro da tabela "Cliente" é feita uma busca na tabela "Cidade", conforme o IDCidade.

# Banco de dados | relacional (SQL)

## Join (junção)

Cliente		
IDCliente	Nome	IDCidade
1	Pedro	1
2	Maria	3
3	Julia	2
10	Carlos	2
15	Antônio	3

Cidade		
IDCidade	Nome	UF
1	Porto Alegre	RS
2	São Paulo	SP
3	Sapucaia do Sul	RS

Para cada registro da tabela "Cliente" é feita uma busca na tabela "Cidade", conforme o IDCidade.

# Banco de dados | relacional (SQL)

## Join (junção)

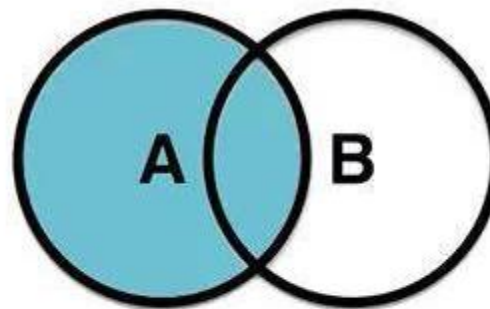
Cliente		
IDCliente	Nome	IDCidade
1	Pedro	1
2	Maria	3
3	Julia	2
10	Carlos	2
15	Antônio	3

Cidade		
IDCidade	Nome	UF
1	Porto Alegre	RS
2	São Paulo	SP
3	Sapucaia do Sul	RS

Para cada registro da tabela "Cliente" é feita uma busca na tabela "Cidade", conforme o IDCidade.

# RELACIONAMENTO: left (outer) join

Recuperando todos os dados de uma fonte, mesmo que não exista em outra (relacionamento opcional):



```
SELECT <fields list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key
```

# RELACIONAMENTO: left (outer) join

Recuperando todos os dados de uma fonte, mesmo que não exista em outra:

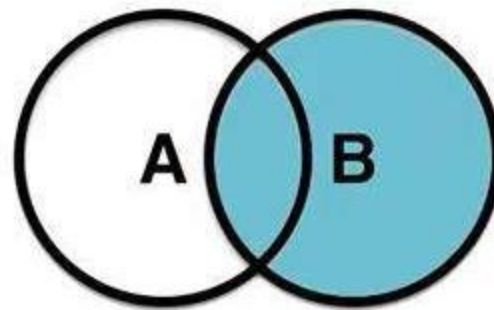
```
Select a.Nome as NomeAssociado,  
       c.Nome as NomeCidade  
From Associado a  
     LEFT JOIN Cidade c ON c.IDCidade = a.IDCidade
```

» Mesmo que exista algum registro na tabela Associado com IDCidade nulo este será exibido:

```
Select a.Nome as NomeAssociado,  
       c.Nome as NomeCidade  
From Associado a, Cidade c  
Where c.IDCidade *= a.IDCidade
```

# RELACIONAMENTO: Right (outer) join

Recuperando todos os dados de uma fonte, mesmo que não exista em outra:



```
SELECT <fields list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key
```

## RELACIONAMENTO: Right (outer) join

Recuperando todos os dados de uma fonte, mesmo que não exista em outra:

```
Select a.Nome as NomeAssociado,  
       c.Nome as NomeCidade  
From Associado a  
      RIGHT JOIN Cidade c ON c.IDCidade = a.IDCidade
```

» Mesmo que exista algum registro na tabela Cidade não tenha registro em Associado:

```
Select a.Nome as NomeAssociado,  
       c.Nome as NomeCidade  
From Associado a, Cidade c  
Where c.IDCidade =* a.IDCidade
```



# RELACIONAMENTO: Self join (auto-relacionamento)

Selecionar com a própria tabela.

É necessário adicionar a tabela novamente, da mesma que forma que um INNER JOIN (ou left join conforme a necessidade).

```
Select e.NomeEmpregado as NomeEmpregado,  
       g.NomeEmpregado as NomeGerente  
From Empregado e  
     INNER JOIN Empregado g ON e.IDGerente = g.IDEmpregado
```

» Serão exibidos somente os empregados que tenham gerente.

```
Select e.Nome as NomeEmpregado,  
       g.Nome as NomeGerente  
From Empregado e, Empregado g  
Where e.IDGerente = g.IDEmpregado
```

# RELACIONAMENTO: cross join

É o resultado de uma consulta onde as tabelas não são relacionadas. Com isso para cada linha de uma tabela A todos os registros da tabela B serão exibidos.

A		B		Resultado			
Col1	Col2	Col_1	Col_2	Col1	Col2	Col_1	Col_2
ABC	10	ABB	102	ABC	10	ABB	102
BCD	20	ACC	304	BCD	20	ABB	102
		ADD	508	ABC	10	ABB	102
				BCD	20	ACC	304
				ABC	10	ACC	304
				BCD	20	ACC	304

```
Select a.IDAssociado,
       a.Nome,
       a.IDCidade IDCidadeEmp,
       c.IDCidade,
       c.Nome
From Associado a, Cidade c
go
```

**ATENÇÃO: NUNCA ESQUEÇA DE RELACIONAR TODAS AS TABELAS EM UMA CONSULTA.**

# RELACIONAMENTO: sub-queries - exists

## (com join)

É uma query dentro de outra. Pode ser aplicado nos comandos Insert, Update e Delete também.

No **EXISTS** é necessário **relacionar** a consulta interna (subquery) com a consulta principal:

```
Select IDCidade, Nome  
  From Cidade e  
 Where EXISTS (Select 1  
                From   Associado a  
                Where  a.IDCidade = e.IDCidade)
```

» Exibirá todas as cidades que tenham associado relacionado.

# RELACIONAMENTO: sub-queries - sem join

Sub-query sem relacionar com a principal: IN, = (igual), != (diferente), e outros.

» Esta consulta apresenta desempenho inferior ao usado com EXISTS.

```
Select IDCidade, Nome  
  From Cidade e  
 Where IDCidade IN (Select IDCidade  
                   From Associado)
```

» Exibe a cidade de menor IDCidade do estado de SP:

```
Select IDCidade, Nome  
  From Cidade e  
 Where IDCidade = (Select MIN(IDCidade)  
                   From Cidade  
                   Where UF = 'SP')
```

Para utilizar o comparador de igualdade é preciso garantir que retornará somente 1 registro na sub-query.

# RELACIONAMENTO: sub-queries - no select

É possível utilizar uma consulta como coluna, na cláusula SELECT.

» Esta consulta exibirá o nome da cidade:

```
Select Nome,  
      (Select Nome  
       From   Cidade c  
       Where  c.IDCidade = a.IDCidade) as NomeCidade  
From Associado a  
go
```

Para utilizar uma sub-query no Select devemos garantir que a consulte não retorne mais de 1 linha, e que apenas 1 (uma) coluna,

# RELACIONAMENTO: sub-queries - no from

É possível utilizar uma consulta como fonte de dados, na cláusula FROM.

```
Select a.Nome, city.Nome as NomeCidade
  From Associado a
  Inner join (Select IDCidade, (Nome + '-' + UF) as Nome
             From Cidade
             Where UF in ('SC', 'PR', 'RS') ) as city
    on city.IDCidade = a.IDCidade
go
```

# RELACIONAMENTO: union

É a união de duas (ou mais) consultas, obrigatoriamente o número de colunas e tipos devem ser iguais em ambas as consultas. Existe duas formas:

- **UNION:** elimina as linhas duplicadas do resultado final (mais lento);
- **UNION ALL:** não elimina as linhas duplicadas (mais rápido).

A		B		Resultado	
Col1	Col2	Col1	Col2	Col1	Col2
ABC	10	ABB	102	ABC	10
BCD	20	ACC	304	BCD	20
		ADD	508	ABB	102
				ACC	304
				ADD	508

# RELACIONAMENTO: union

Retornando em uma consulta todos os nomes das tabelas Empregado e Associado.

```
Select Nome  
From    Associado  
UNION ALL  
Select NomeEmpregado  
From    Empregado  
go
```



# RELACIONAMENTOS



## mãos à obra

Inserção de registros, execute os comandos abaixo

- Atualizando IDCidade

```
update associado  
set idcidade = 1  
where idassociado = 1;
```

```
update associado  
set idcidade = 32  
where idassociado = 3;
```