Banco de Dados Relacional

# Curso de Modelagem de Banco de Dados: entidades, relacionamentos e atributos

Finalizado em 14/05/2023

# Entrevista:

Queremos coletar os dados pessoais de nossos clientes, como se ele é pessoa física ou jurídica. No caso de PF o seu CPF e RG, e no caso de jurídica o CNPJ e IE. Além disso, queremos coletar e armazenar o seu nome, endereço, telefone e e-mail.

O produto principal do e-commerce são livros. Estes livros têm informações associadas a eles como o título, categoria, o ISBN (International Standard Book Number), o ano de publicação, o valor, a editora que publicou o livro, bem como o autor ou autora da obra.

Os livros são fornecidos por editoras. Precisamos ter guardados o telefone da editora, o nome de contato, o e-mail e no máximo 2 telefones.

Sabemos que não podemos ter o mesmo livro vindo de várias editoras. O livro é exclusivo de uma editora.

Nosso cliente pode comprar um ou mais livros através de um pedido de compra. Porém, sempre que ele faz uma compra precisamos verificar no estoque se o livro está ou não disponível antes de efetuar a operação.

# Modelo Conceitual

É o modelo de mais **alto nível**, ou seja, que esta mais próximo da realidade dos usuários.

O nível conceitual é desenvolvido com alto nível de abstração, a partir dos requisitos do sistema, extraídos na fase de levantamento de requisitos.

Versão 1 - 14/05/2023


# SGBD

* É o que gerencia o banco de dados;
* Interface para incluir, alterar ou consulta;
* Tudo que fazemos passa por ele;
* É comum que confundam o SGBD com o banco de dados.

Os mais utilizados: MySQL, OROCLE, SQLSERVER, entre outros.

Está ligado ao “Baixo Nível” – linguagem das máquinas – modelos físicos.

“Alto Nível” – está relacionado a linguagem das pessoas – modelos conceituais.

# Mini-Mundo

É um recorte do mundo real.

# Abstração

É um processo mental que permite escolher ou isolar um aspecto de coisas complexas, tem como finalidade simplificar.

• É subjetivo;

• Ignora-se os detalhes.

# Entrevista

A modelagem de dados é a base para se ter um bom projeto final do banco de dados. Uma das etapas mais importantes deste processo é a entrevista com os(as) clientes, na qual serão identificadas as regras de negócio do projeto. Quando não identificamos as necessidades do projeto, pode surgir novamente a necessidade de realizar esta etapa, gerando assim, atraso em todo o processo.

A entrevista dará todo o direcionamento ao nosso projeto. Através dela, conhecemos todos os detalhes do negócio e podemos estruturar os próximos passos. Um fator importante desse processo é saber de quem vamos colher as informações, ou seja, devemos entrevistar uma ou mais pessoas que possam passar todos os detalhes importantes do negócio.

Outro fator é escolher as perguntas ideais para definir todos os pontos-chaves do projeto. Nessa etapa, a pessoa que faz a entrevista precisa ter um conhecimento prévio sobre os pontos mais relevantes para a construção do projeto, o que possibilita coletar informações realmente essenciais para a modelagem do banco de dados.

Em resumo, a entrevista é a base para construir um projeto coerente e que atende às necessidades do(a) cliente.

# MER

Modelo Entidade de Relacionamento

* Modelo conceitual;
* Usado para descrever objetos, suas característica e como se relacionam.

# DER

Diagrama Entidade de Relacionamento

* É a reapresentação gráfica do MER;
* Muitas vezes é usado como sinônimo;
* Diagramas facilitam a comunicação entre todos.

# Entidade

* Entidade é um objeto único no mundo real;

1. Clientes de uma empresa;
2. Carros que são vendidos;
3. Departamentos de vendas;

* Pode ser abstrata ou concreta.

## Entidade Forte

Existe independentemente de outra entidade.

Uma entidade forte sempre tem a chave primária no conjunto de atributos que descreve a entidade forte. A entidade forte, por si só, possui um atributo que poderá ser utilizado como chave primária.

## Entidade Fraca

Depende da existência de outra entidade.

A entidade fraca não possui a chave primária, mas tem uma chave parcial que discrimina de maneira única as entidades fracas. A entidade fraca possui uma chave composta formada a partir da chave primária da entidade forte e chave parcial da entidade fraca.

# Chaves

Tabela

Descrição gerada automaticamente

## Chave primaria

Uma chave primária é uma regra implementada em uma coluna ou em um conjunto de colunas de forma a garantir que os valores contidos nesta (s) sejam únicos, ou seja, estes valores nunca irão se repetir.

Tabela

Descrição gerada automaticamente

Na figura a coluna ID\_CLIENTE contém a regra de chave primária de forma que estes valores identificaram cada um dos clientes cadastrados. Esta é a única coluna que possui informações que podem distinguir uma linha das demais, pois elas não se repetem. As colunas NOME, DATA\_NASC e SEXO podem possuir informações repetidas de forma que não são bons identificadores.

Podemos ter mais de uma coluna recebendo a regra de chave primária.

## Chave primaria composta

Geralmente, criamos chaves compostas quando não há um campo que, por si só, possa ser usado para identificar as linhas da tabela de forma exclusiva, necessitando que seja realizada uma combinação de dois ou mais campos para tal.

Tabela

Descrição gerada automaticamente

Na figura acima temos quatro colunas recebendo a regra de chave primária. Cada tabela só pode ter uma regra de chave primária, independente da quantidade de colunas que a compõem.

Então podemos dizer que o cliente que possui o código (id\_cliente) número 2, por exemplo, locou no dia 24 de abril 2015 os DVDs de código número 3 e 4 exatamente as 09:11. Como a regra de chave primária envolve as quatro colunas podemos garantir que um mesmo cliente nunca conseguirá locar a mesma fita na mesma data e no mesmo horário. De forma que a combinação das quatro colunas nunca se repetirão.

Neste caso, uma única coluna não foi o suficiente para identificar a tabela Aluguel, de forma que somente com a composição das quatro colunas garantimos que não hajam dados repetidos.

## Chave Estrangeira

Tabela

Descrição gerada automaticamente

Na figura acima, na tabela ALUGUEL, vemos a coluna ID\_DVD que possui uma regra de chave estrangeira, referenciando os valores de chave primária da tabela DVD. Ou seja, só conseguiremos inserir algum valor na coluna ID\_DVD da tabela ALUGUEL se este valor já previamente existir na coluna ID\_DVD da tabela DVD. Na coluna ID\_CLIENTE da tabela ALUGUEL temos outra regra de chave estrangeira, só que esta está referenciando a chave primária da tabela CLIENTE.

Desta forma, com a implementação da chave estrangeira, podemos garantir que sempre que houver um aluguel, este será feito obrigatoriamente para um cliente cadastrado e que somente um dvd cadastrado poderá ser alugado.

# Tipos de Relacionamentos

Forma

Descrição gerada automaticamente com confiança baixa

Diagrama

Descrição gerada automaticamente

Diagrama

Descrição gerada automaticamente

# Cardinalidade

Diagrama

Descrição gerada automaticamente

Exemplo: um departamento que só pode ter um colaborador

Diagrama

Descrição gerada automaticamente

Exemplo: um departamento que pode ter muitos colaboradores; um colaborador pode trabalhar um só departamento

Diagrama

Descrição gerada automaticamente

Exemplo: um pedido que pode conter vários livros; um livro pode ser vendido em diversos pedidos

# Curso de Modelagem de banco de dados relacional: modelagem lógica e física.

# Modelo Lógico de Dados (MLD)

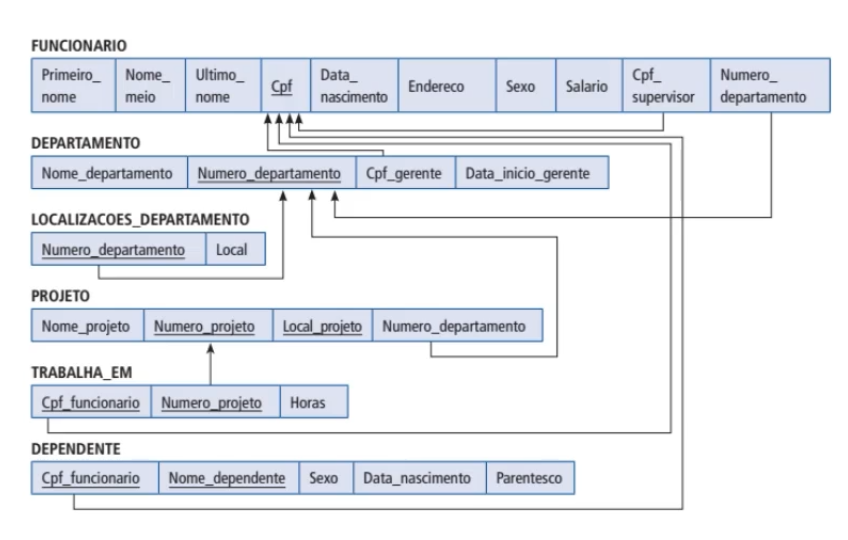
É feito com base no modelo conceitual, é o segundo passo.

Há duas formas que podemos utilizar para representar o modelo lógico: **a forma horizontal ou a forma vertical**.

Na **representação vertical**, os atributos (que mais a frente serão chamados de campos) são representados um abaixo do outro, os atributos chaves são identificados e os relacionamentos são representados através de linhas ligando uma relação à outra.

 Exemplo de modelo lógico representado na forma vertical.


Na **forma horizontal**, os atributos (campos) são apresentados um ao lado do outro, os atributos chave são sublinhados nas relações e os relacionamentos são representados através de uma seta, que sai da relação dependente e chega à relação proprietária. Já as cardinalidades não são representadas.



As duas formas são válidas, porém utilizamos a forma vertical.

# ~~ENTIDADES~~ RELAÇÕES

As entidades dentro do modelo logico são nomeadas de relações.

Alguns softwares adotam os nomes dos modelos físicos, tal qual, tabela.

# ~~ATRIBUTOS~~ CAMPOS

Os atributos agora são chamados de campos.

# Modelo Físico de Dados (MFD)

O terceiro passo é o modelo físico que se assemelha e está muito próximo da implementação do banco de dados.

# Tipos de domínios

**Nos tipos de dados numéricos temos:**

*Int* armazena valores numéricos inteiros.

*Float* armazena valores numéricos aproximados com precisão de ponto flutuante.

*Decimal* armazena valores numéricos com casas decimais, utilizando precisão.

**Nos tipos de dados string temos:**

*Varchar* armazena valores de string de tamanho variável de acordo com o seu limite.

*Char* armazena valores de string de tamanho fixo.

*Text* armazena valores de string de tamanho variável.

**Nos tipos de dados Data temos:**

*Date* armazena apenas valores de data.

*Time* armazena apenas valores de hora.

*Timestamp* armazena valores de data e hora.

# Relacionamentos

O **relacionamento identificador** é aquele na qual uma ocorrência da entidade fraca, precisa estar associada a exatamente uma ocorrência da entidade forte e não pode existir sem ela. Nesse caso, esse relacionamento é utilizado para representar uma chave primária estrangeira.

No **relacionamento não identificador**, cada ocorrência da entidade fraca pode ser identificada sem a necessidade de saber a qual ocorrência da entidade forte está associada. É utilizado para representar apenas chaves estrangeiras.

# Modelagem de banco de dados relacional: normalização

# Diretrizes informais

1. Semântica clara com esquemas fáceis de explicar;
2. Evitar informações redundantes, pois podem causar anomalias;
3. Impossibilitar valores NULL nas tuplas;
4. Atenção ao surgimento de tuplas falsas.

# Anomalias

* De inserção (causa a repetição desnecessária de dados (redundância));
* De alteração ( leva a inconsistência e aumenta o esforço para a atualização dos dados);
* De remoção (causa a perda de informação associada a um dado registro).

| **CPF** | **nome\_colaborador** | **end\_colaborador** | **cod\_depto** | **nome\_depto** |
| --- | --- | --- | --- | --- |
| 123456 | Marta | Rua das Flores | 15 | Tecnologia |
| 147896 | Robledo | Rua Pio XXI | 23 | Vendas |
| 965537 | Helena | Rua ABC | 15 | Tecnologia |

Na **Anomalia de Inserção** ao inserirmos um registro novo nessa tabela de colaboradores, precisamos adicionar também o departamento. Neste processo, pode acontecer de algum colaborador ainda não estar vinculado a um departamento ou de incluirmos um departamento que ainda não tem colaboradores.

Na **Anomalia de Alteração**, caso seja necessário alterar um departamento, precisaríamos alterar todos os registros em questão, gerando um "retrabalho" dentro da tabela. O resultado é uma inconsistência dentro do banco de dados.

Sobre a **Anomalia de Remoção**, se precisarmos remover um departamento ou até mesmo um colaborador, teremos que remover uma tupla inteira. Neste caso, aconteceria outra inconsistência que é a falta de dados. Para evitarmos todas essas anomalias, podemos usar as formas normais e, consequentemente, normalizarmos o banco de dados.

# Dependência funcional

A dependência funcional estabelece uma relação de atributos dentro da tabela.

Exemplo: CPF -> Nome

O atributo que determina o valor é chamado de Determinante. O outro atributo é chamado de Dependente.

| **CPF** | **nome\_colaborador** | **end\_colaborador** |
| --- | --- | --- |
| 123456 | Marta | Rua das Flores |
| 147896 | Robledo | Rua Pio XXI |
| 965537 | Helena | Rua ABC |
| 437627 | Marta | Rua Floriano Peixoto |

Observando o CPF "123456" notamos que ele é da Marta. Se buscarmos pelo nome Marta, aparecerão dois CPFs: "123456" e "437627". Assim, fica difícil entender e encontrar a Marta que estamos procurando. Então, o **CPF é a Chave primária**, o "nome\_colaborador" e "end\_colaborador" dependem dessa chave primária para existirem dentro da tabela.

Podemos dizer, portanto, que o CPF determina funcionalmente o nome do colaborador e também determina funcionalmente o endereço do colaborador.

# Forma normal

Para uma tabela estar em uma determinada **forma normal,** ela precisa satisfazer algumas propriedades baseadas nas **dependências funcionais**.

### A **primeira forma normal** deve satisfazer as propriedades baseadas na **dependência funcional**.

* Evitar a mistura de assuntos em uma tabela
* Não admitir repetições ou campos que tenham mais que um valor.

Procedimentos para aplicar a regra:

1. Identificar a chave primaria da tabela
2. Identificar o grupo repetitivo e removê-lo da tabela.

Dessa forma devemos criar uma bela nova com esse grupo repetitivo e usar a chave primaria que foi quebrada e usar como chave estrangeira da tabela nova com o grupo repetitivo.

### A **segunda forma normal** deve satisfazer as propriedades baseadas na **dependência funcional parcial.**

Quando uma tabela está na segunda forma normal?

* Identificar se a tabela possui chave primaria composta;
* Se está na primeira forma normal;
* Atributos não chave dependem da chave composta em sua totalidade.

Procedimentos para aplicar a regra:

1. Identificar se a tabela tem chave primaria composta;
2. Identificar os atributos que dependem parcialmente dessa chave primaria e criar uma nova tabela com eles.

### A **terceira forma normal** deve satisfazer as propriedades baseadas na **dependência transitiva**.

* Se está na segunda forma normal;
* Se nenhuma coluna possui dependência transitiva em relação a outra coluna que não participe da chave primária

É um conceito que deve ser analisado com calma, pois muitos sistemas precisam armazenar os resultados de cálculos, totais e valores.

Procedimentos para aplicar a regra:

1. Identificar todos os atributos que são funcionalmente dependentes de outros atributos não chave e removê-los.

### A **forma normal de Boyce-Codd** deve satisfazer as propriedades baseadas na **dependência funcional trivial**.

Definimos que uma tabela está em FNBC se e somente se todos os determinantes são chaves candidatas.

A superposição é quando existem chaves candidatas compostas com o atributo que se repete entre elas.

Procedimentos para aplicar a regra:

1. Identificar todos os atributos que seja determinado por outro atributo que não seja uma chave candidata;
2. Removê-los e levar para outra tabela.

### A **quarta forma normal** deve satisfazer as propriedades baseadas na **dependência Multivalorada.**

* Está na terceira forma;
* Não ter mais de um atributo multivalorado

Resumindo, na quarta forma normal não pode existir dependência multivalorada.

**ATENÇÃO**: por causa da dependência multivalorada, a quarta forma normal pode ser facilmente confundida com o relacionamento binário muitos-para-muitos. A diferença é que o relacionamento binário acontece entre tabelas, por exemplo, "clientes" pode ter vários "pedidos" e "pedidos" pode ter vários "clientes". Já a dependência multivalorada acontece dentro da tabela, onde um atributo determina um conjunto de valores.

Procedimentos para aplicar a regra:

1. Identificar se existe um multi determinante que aponte para mais de um multi dependente;
2. Identificar se existe dependência entre esses multi dependentes.

### A **quinta forma normal** deve satisfazer as propriedades baseadas na **dependência junção**.

O que é a quinta forma normal?

A Quinta Forma Normal trata de casos onde as informações podem ser recuperadas através de tabelas menores. Ela não é muito diferente da quarta forma normal, isso não existir uma constante simétrica que atue como regra de negócio dentro da tabela em questão. Portanto, se não existir essa constante simétrica e a tabela estiver na quarta forma normal, ela também estará na quinta forma normal.

Resumindo, na quinta forma normal precisamos aplicar a constante simétrica, se existir uma nova regra de negócios na tabela, e atribuir as responsabilidades a esquemas específicos, por exemplo: o vendedor vende o produto; vende para determinada empresa; e a empresa compra e vende esse determinado produto.

Assim, eliminamos a repetição de dados e dividimos a tabela em três: vendedor e produto; vendedor e empresa; empresa e produto.

### O que é Dependência Transitiva?

Ocorre quando um atributo não-chave não depende da chave primária, nem parcialmente, mas depende de outro atributo não-chave. Quando isso acontece, temos a Dependência Funcional Transitiva.

Poderíamos dizer que a diferença entre a dependência parcial e a transitiva é que na parcial o atributo depende ao menos de uma chave primária. Já na transitiva, o atributo não depende de nenhuma chave primária e sim do atributo não-chave.

* Quando há a dependência parcial, os atributos dependem parcialmente da chave primária composta.
* Quando há a dependência transitiva, pelo menos um atributo da tabela depende de outro que não seja chave primária da tabela.

### O que é uma dependência multivalorada?

Na dependência multivalorada, o atributo determina um conjunto de valores de um atributo. podemos dizer que a dependência multivalorada é uma dependência funcional.

**X multi determina Y ou que Y é multi dependente de X.**

### O que é dependência de junção?

Uma tabela é n-decomponível quando pode ser dividida em mais de duas. Logo, n é maior que dois. Esse conceito só pode ser aplicado em tabelas com mais de três atributos.

Exemplo: se uma tabela A tem três atributos {a1, a2, a3}, temos três projeções possíveis para ela, isto é, três tabelas:

A1: {a1, a2} A2: {a1, a3} A3: {a2, a3}

Dizemos haver uma dependência de junção se a tabela A puder ser recomposta através das três projeções: A1, A2 e A3. Ou seja, se pudermos dividir a tabela A em outras três projeções e pudermos recuperá-la sem perda de informações, então temos uma dependência de junção. Em resumo, utilizamos os conceitos de decomposição sem perdas e junção.

#Inserindo valores fora de ordem

INSERT INTO LIVROS

(CATEGORIA, AUTORIA, NOME\_LIVRO, EDITORA, ID\_LIVRO, PREÇO)

VALUES

('Biografia', 'Malala Yousafzai', 'Eu sou Malala' , 'Companhia das Letras', 11, 22.32),

('Biografia', 'Michelle Obama', 'Minha história','Objetiva', 12, 57.90);

# Excluindo uma tabela

DROP TABLE VENDEDORES;

#Como visualizar todas as informações de uma tabela

SELECT ID\_LIVRO AS "CODIGO DO LIVRO" FROM LIVROS

# PESQUISA ESPECIFICA

#Para definir que a parte que queremos é o final das palavras usaremos a notação %lândia, como se fosse “alguma coisa” lândia. Veja a construção do comando e o resultado.

#Lembre-se que o % é um operador que substitui qualquer valor na procura. Então, caso queira cidades que começam com A, basta usar A%. Outro exemplo seria tentar localizar as cidades que contém a sílaba ‘ma’, podemos usar o %MA% para aparecer todas as cidades que possuem ‘ma’ posicionada em qualquer lugar.

SELECT CIDADE, ESTADO

FROM MAPA

WHERE CIDADE LIKE '%LÂNDIA';

#SELEÇÃO DISTINTA

SELECT DISTINCT ID\_LIVRO FROM VENDAS

WHERE ID\_VENDEDOR = 1

ORDER BY ID\_LIVRO;

#fazer alterações no valores

UPDATE LIVROS SET PREÇO = 0.9\*PREÇO;

#caso haja bloquei por estar em modo safe

SET SQL\_SAFE\_UPDATES = 0;

#para ativar de novo

SET SQL\_SAFE\_UPDATES = 1;

#UNIR TABELAS

SELECT VENDAS.ID\_VENDEDOR, VENDEDORES.NOME\_VENDEDOR, VENDAS.QTD\_VENDIDA

FROM VENDAS, VENDEDORES

WHERE VENDAS.ID\_VENDEDOR = VENDEDORES.ID\_VENDEDOR;

#no script acima os números aparecem individualmente, desta forma para somar todas as vendas de cada vendedor usasse "SUM(a linha a ser somada)"

SELECT VENDAS.ID\_VENDEDOR, VENDEDORES.NOME\_VENDEDOR, SUM(VENDAS.QTD\_VENDIDA)

FROM VENDAS, VENDEDORES

WHERE VENDAS.ID\_VENDEDOR = VENDEDORES.ID\_VENDEDOR

GROUP BY VENDAS.ID\_VENDEDOR;

#USANDO INNER JOIN

SELECT VENDAS.ID\_VENDEDOR, VENDEDORES.NOME\_VENDEDOR, SUM(VENDAS.QTD\_VENDIDA)

FROM VENDAS INNER JOIN VENDEDORES

ON VENDAS.ID\_VENDEDOR = VENDEDORES.ID\_VENDEDOR

GROUP BY VENDAS.ID\_VENDEDOR;

##########

#MAX: a partir de um conjunto de valores é retornado o maior entre eles;

#MIN: analisa um grupo de valores e retorna o menor entre eles;

#SUM: calcula o somatório dos valores de um campo específico;

#AVG: realiza a média aritmética dos valores de uma determinada coluna; e

#COUNT: contabiliza a quantidade de linhas selecionadas.

#Para saber a maior venda feita no Clube do Livro, podemos construir o comando:

#SELECT MAX(QTD\_VENDIDA) FROM VENDAS;

#Usando o GROUP BY agrupamos os registros que serão agregados em grupos de valores permitindo que a função de agregação seja realizada para cada um desses grupos.

#Para saber, por exemplo, a maior venda por código de pessoa vendedora podemos consultar da seguinte maneira:

#SELECT MAX(QTD\_VENDIDA) FROM VENDAS

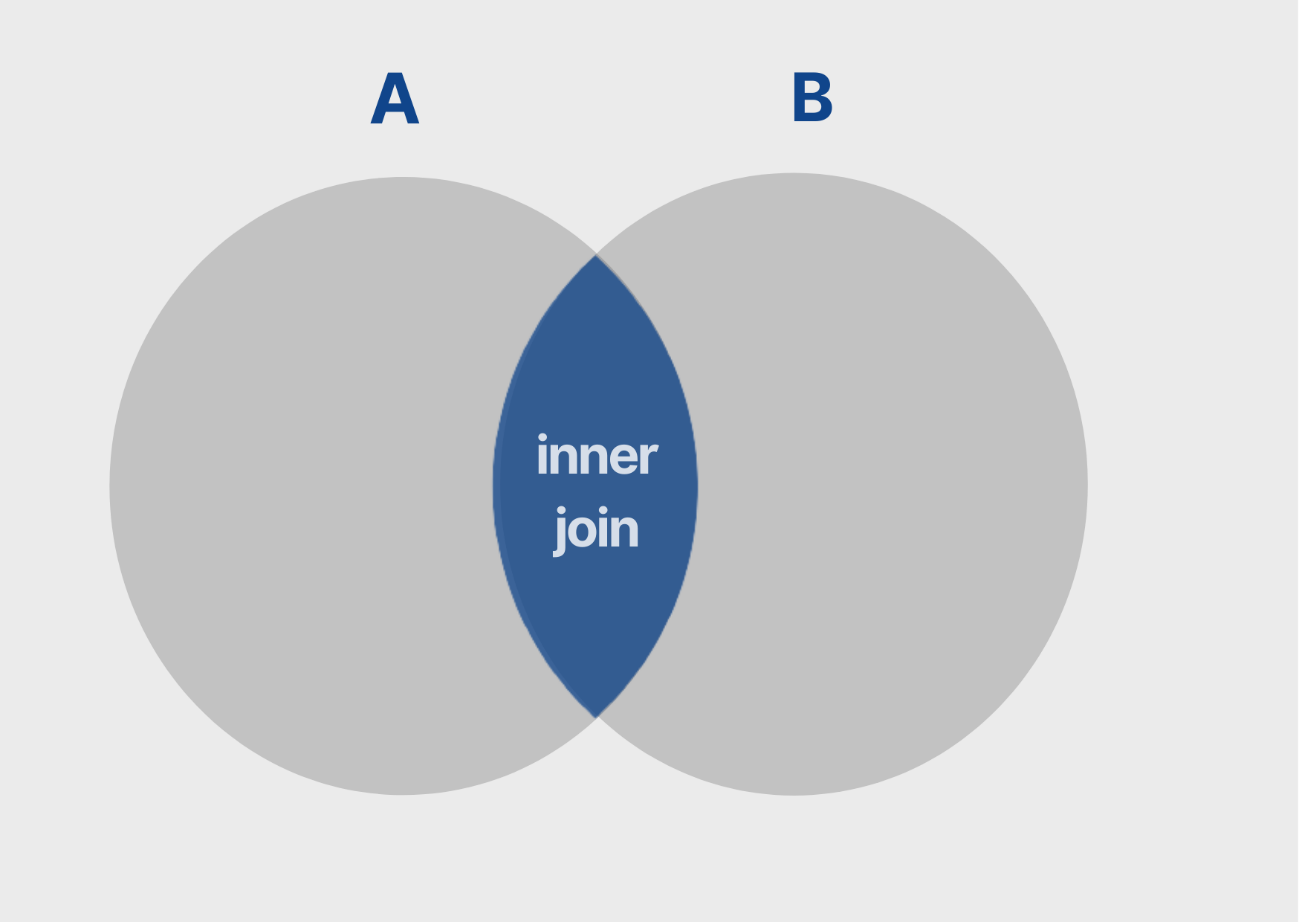
#GROUP BY(ID\_VENDEDOR);

#######

Quando temos informações em diferentes tabelas e queremos reunir uma única consulta é comum fazer **junção** entre tabelas. Para explicar a diferença entre os famosos **Joins**, é bem comum representarmos por Diagrama de Venn, onde cada círculo (conjunto) representa uma tabela do nosso modelo relacional. Confira as diferença entre as principais junções:

* **Inner Join**

O INNER JOIN é a **interseção** entre duas tabelas, ou seja, na consulta aparecerá todas as informações de um determinado campo da tabela A que também foi encontrado na tabela B.



**Para trazer a interseção das informações entre tabelas, podemos usar o comando:**

SELECT <CAMPOS>

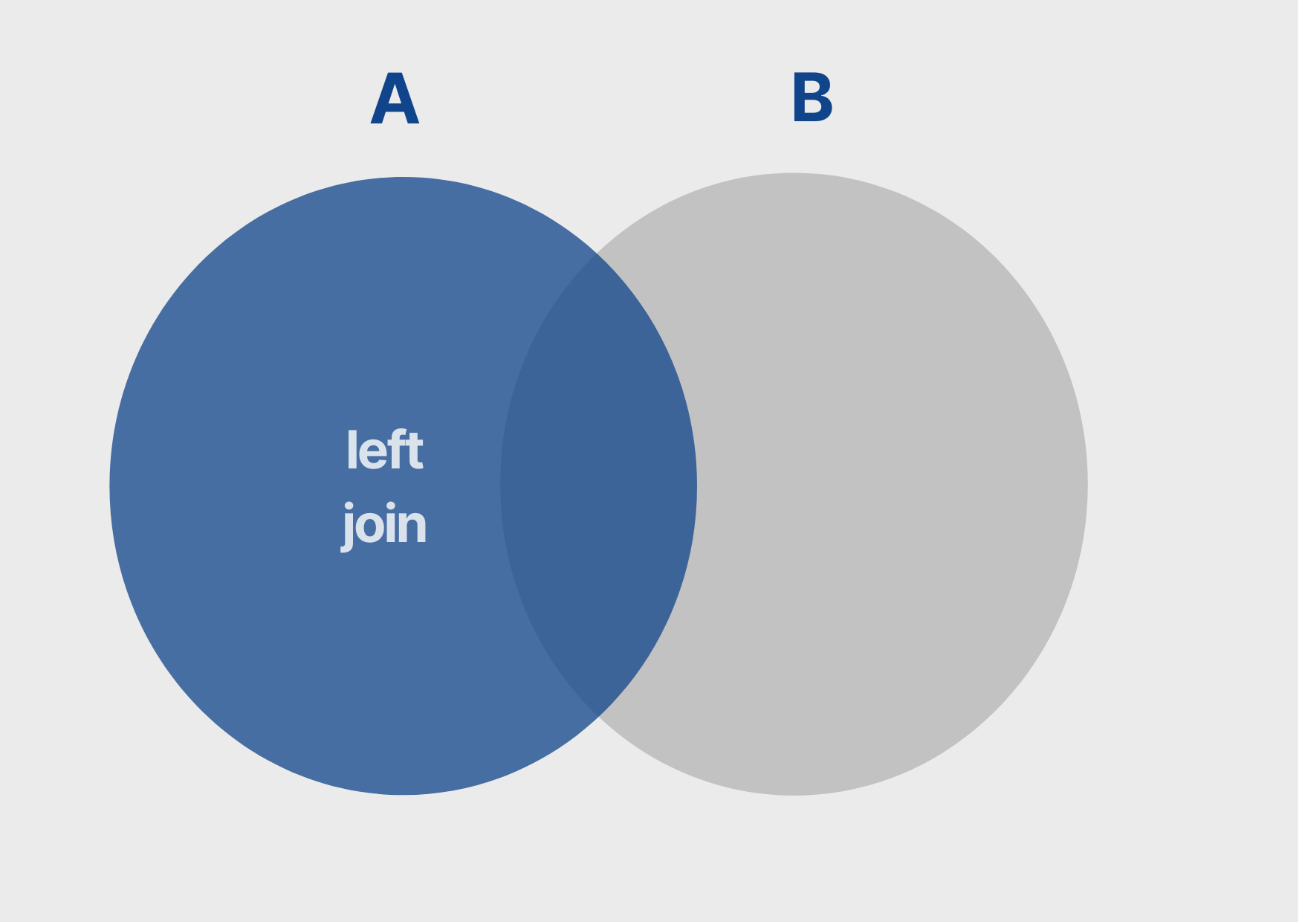
FROM TABELA\_A

INNER JOIN TABELA\_B

ON TABELA\_A.CAMPO = TABELA\_B.CAMPO

* **Left Join**

O LEFT JOIN baseia-se nas informações da tabela declarada à esquerda do comando ao se juntar com outra tabela.



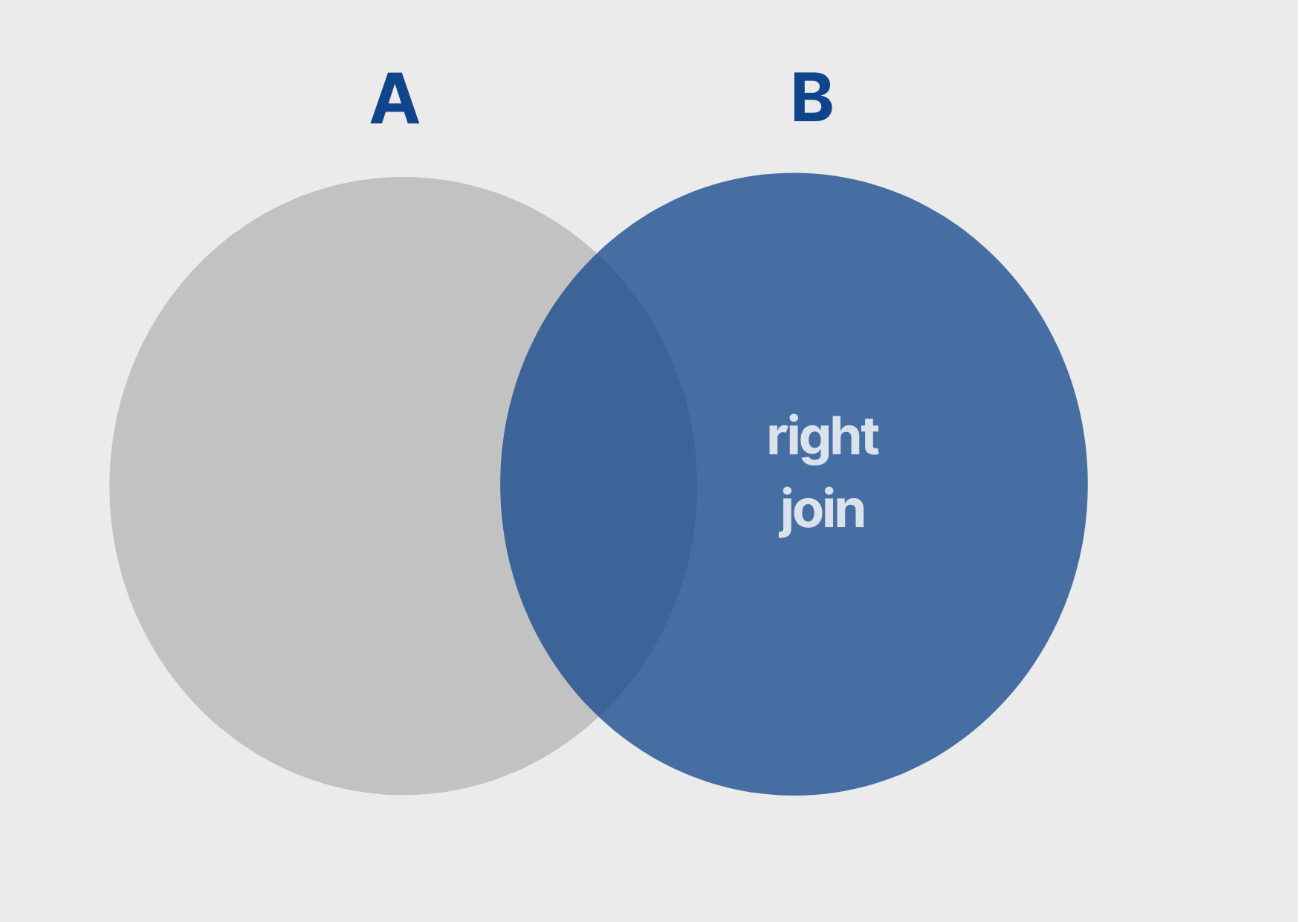
SELECT <CAMPOS>

FROM TABELA\_A

LEFT JOIN TABELA\_B

ON TABELA\_A.CAMPO = TABELA\_B.CAMPO

* **Right Join**



SELECT <CAMPOS>

FROM TABELA\_A

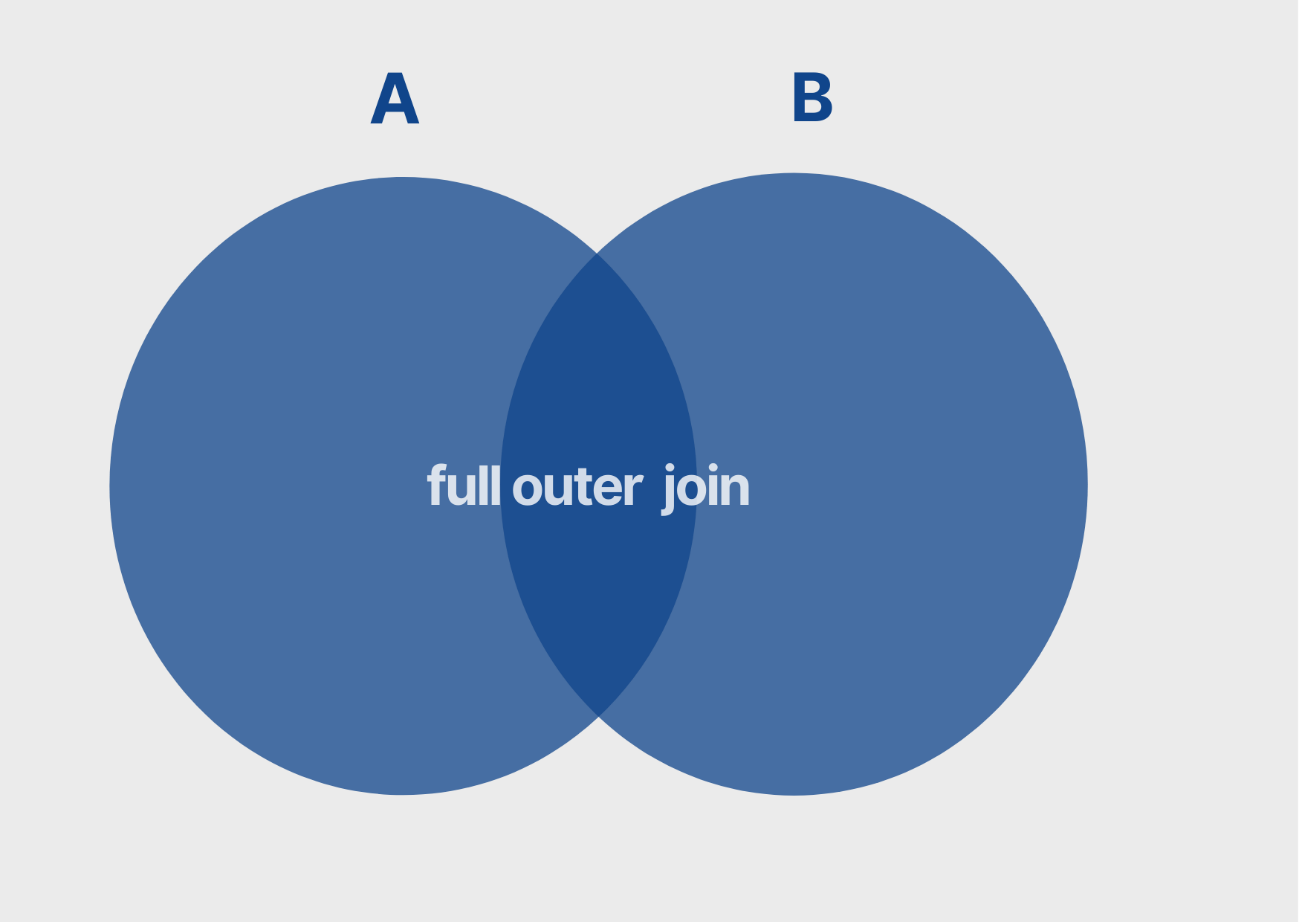
RIGHT JOIN TABELA\_B

ON TABELA\_A.CAMPO = TABELA\_B.CAMPO

* **Full Outer Join**

Esse comando apresenta a união entre duas tabelas.

1. Incluindo a interseção



SELECT <CAMPOS>

FROM TABELA\_A

FULL OUTER JOIN TABELA\_B

ON TABELA\_A.CAMPO = TABELA\_B.CAMPO

ou

SELECT <CAMPOS>

FROM TABELA\_A

LEFT JOIN TABELA\_B

ON TABELA\_A.CAMPO = TABELA\_B.CAMPO

UNION

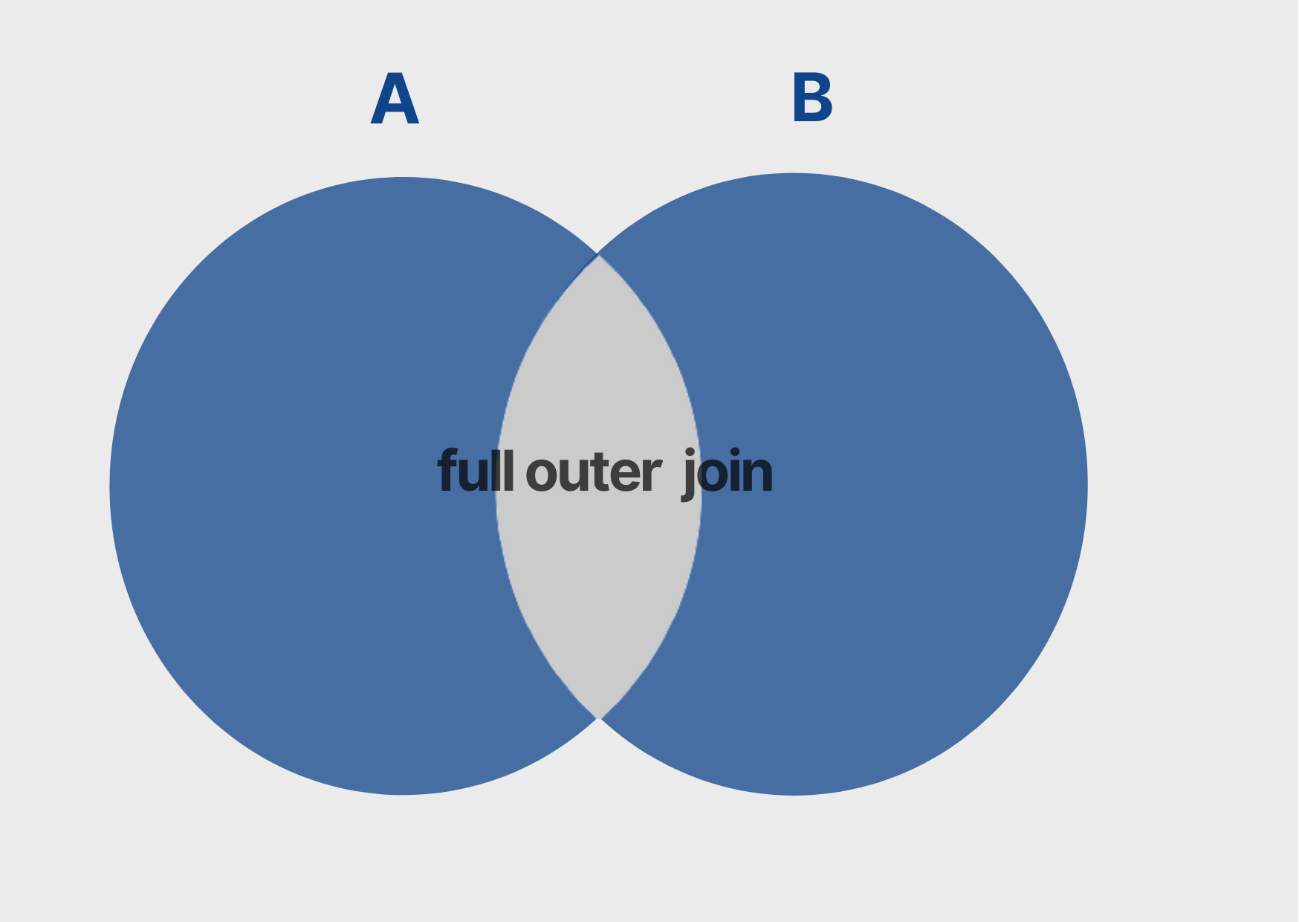
SELECT <CAMPOS>

FROM TABELA\_A

RIGHT JOIN TABELA\_B

ON TABELA\_A.CAMPO = TABELA\_B.CAMPO

1. Excluindo a interseção



SELECT <CAMPOS>

FROM TABELA\_A

INNER JOIN TABELA\_B

ON TABELA\_A.CAMPO = TABELA\_B.CAMPO

WHERE TABELA\_A.CAMPO IS NULL OR TABELA\_B.CAMPO IS NULL

Para saber mais, confira outros exemplos de joins em SQL [neste artigo.](https://www.alura.com.br/artigos/join-em-sql)