

# 1 - Modelagem de banco de dados relacional e algebra relacional – Dia 1

## #F3006 - Apresentação

A **modelagem de dados** é o processo de estruturar e documentar como os dados são armazenados, organizados e manipulados dentro de um sistema. Ela é essencial para garantir **coerência, eficiência e integridade** no banco de dados.

---

## Níveis da Modelagem de Dados

A modelagem ocorre em três níveis principais, cada um com um grau de detalhe diferente:

### ① Modelo Conceitual (Visão Abstrata)

- Representação de **alto nível**, focada na estrutura dos dados sem detalhes técnicos.
  - Define **entidades, atributos e relacionamentos** entre os dados.
  - Usado para comunicação com stakeholders não técnicos (analistas, gestores).
  - **Ferramenta comum:** Diagrama Entidade-Relacionamento (DER).
  - **Exemplo:**
    - Entidade **Cliente** (com atributos: nome, CPF, telefone).
    - Entidade **Pedido** (com atributos: número, data, valor).
    - Relacionamento: **Cliente faz Pedido** (1:N).
- 

### ② Modelo Lógico (Estrutura Formal)

- Tradução do modelo conceitual para um formato estruturado, adequado para bancos de dados.
- Define **tabelas, colunas, chaves primárias (PK), chaves estrangeiras (FK) e relacionamentos**.
- Ainda **independente de um SGBD** específico.
- **Exemplo:**

```
CREATE TABLE Cliente (  
    id_cliente INT PRIMARY KEY,  
    nome VARCHAR(100),  
    cpf CHAR(11) UNIQUE  
);  
  
CREATE TABLE Pedido (  
    id_pedido INT PRIMARY KEY,  
    id_cliente INT,  
    data_pedido DATE,  
    valor DECIMAL(10,2),  
    FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente)  
);
```

---

### ③ Modelo Físico (Implementação no Banco de Dados)

- **Dependente do SGBD** (MySQL, PostgreSQL, Oracle, etc.).
- Especifica **tipos de dados, índices, partições e otimizações** para performance.

- **Exemplo de otimizações:**
    - **Índices** para acelerar consultas (`CREATE INDEX idx_cliente ON Pedido(id_cliente);`).
    - **Normalização** para evitar redundância.
    - **Desnormalização** para otimizar leitura.
- 

## Princípios Fundamentais

### ✓ Normalização

- Processo que organiza os dados para **evitar redundância e inconsistências**.
- Aplicação de formas normais (1ª, 2ª, 3ª FN...).
- Exemplo: Separar dados repetitivos em tabelas diferentes (ex: cliente e endereço).

### ✓ Desnormalização

- Oposto da normalização: **duplica alguns dados para melhorar performance**.
  - Exemplo: Em um sistema de relatórios, pode ser melhor manter algumas informações repetidas para evitar JOINS complexos.
- 


## Resumo

Modelo	Objetivo	Características
<b>Conceitual</b>	Visão abstrata dos dados	Foca em entidades, atributos e relacionamentos
<b>Lógico</b>	Estrutura formal para o BD	Define tabelas, chaves e relacionamentos, mas sem detalhes do SGBD
<b>Físico</b>	Implementação no SGBD	Define tipos de dados, índices, otimizações

### #F3007 - Clube do livro

Empresa que será utilizada como case, é um ecommerce de livro

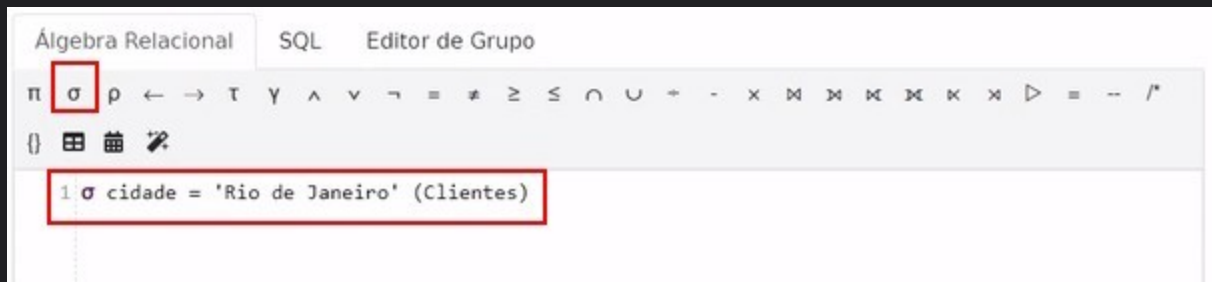
### #F3008 - Ferramenta e dados

**\_BASE DE DADOS** 

```
Livros (id_livro, nome_livro, autor, preco, qtd_estoque)
Vendedores = (id_vendedor, nome, anos_exp)
Clientes = (id_cliente, nome, cidade, email)
Vendas = (id_pedido, id_vendedor_vendas, id_livro, qtd_vendida)
LivroMaisVendidos = (id_livro, ano, mais_vendidos)
LivrosRecomendados = (id_livro, fonte_recomendacao, nota_media)
```

### #F3009 – Seleção $\sigma$ (sigma) – Trás linhas específicas conforme a seleção

A empresa Clube do livro gostaria de saber quais clientes irão se beneficiar do frete grátis para o Rio de Janeiro?

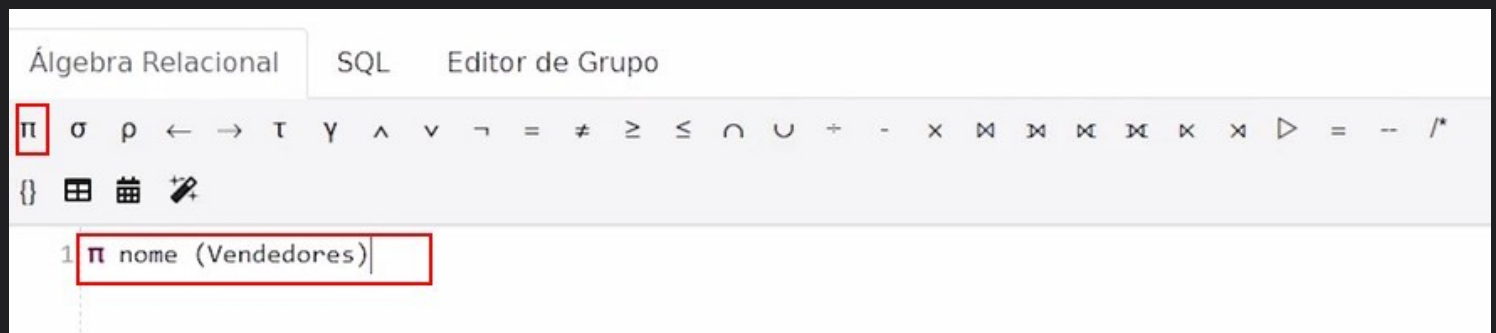


$\sigma \text{ cidade} = \text{'Rio de Janeiro'} (\text{Clientes})$   
Execution time: 0 ms

Clientes.id_cliente	Clientes.nome	Clientes.cidade	Clientes.email
2	'Maria Santos'	'Rio de Janeiro'	'maria.santos@example.com'
12	'Carolina Menezes'	'Rio de Janeiro'	'carolina.menezes@example.com'
18	'Laura Mendes'	'Rio de Janeiro'	'laura.mendes@example.com'

### #F3010 – Projeção $\pi$ (pi) – Trás colunas específicas conforme projeção

Se eu quiser colunas específicas da minha relação? Como fazer isso?



▶ Executar consulta

Download

Histórico



$\pi_{\text{nome}} ( \text{Vendedores} )$

Execution time: 2 ms

**Vendedores.nome**

'Lucas Silva'

'Camila Santos'

'Gabriel Almeida'

'Larissa Souza'

'Rafael Lima'

'Júlia Rocha'

Álgebra Relacional SQL Editor de Grupo

$\pi$   $\sigma$   $\rho$   $\leftarrow$   $\rightarrow$   $\tau$   $\gamma$   $\wedge$   $\vee$   $\neg$   $=$   $\neq$   $\geq$   $\leq$   $\cap$   $\cup$   $+$   $-$   $\times$   $\bowtie$   $\ltimes$   $\ltimes$   $\ltimes$   $\ltimes$   $\ltimes$   $\ltimes$   $\triangleright$   $=$   $--$   $/$

$\pi_{\text{nome, anos\_exp}} ( \text{Vendedores} )$

**Vendedores.nome** **Vendedores.anos\_exp**

'Lucas Silva'

5

'Camila Santos'

3

'Gabriel Almeida'

7

#F3011 - Linhas e colunas específicas – Como duas operações podem, de formas combinadas, filtrar linhas e colunas

Álgebra Relacional   SQL   Editor de Grupo

$\pi$   $\sigma$   $\rho$   $\leftarrow$   $\rightarrow$   $\tau$   $\gamma$   $\wedge$   $\vee$   $\neg$   $=$   $\neq$   $\geq$   $\leq$   $\cap$   $\cup$   $+$   $-$   $\times$   $\bowtie$   $\ltimes$   $\ltimes$   $\ltimes$   $\ltimes$   $\ltimes$   $\ltimes$   $\triangleright$   $=$   $-$   $/^*$

$\{\}$   $\text{table}$   $\text{grid}$   $\text{edit}$

1  $\pi$  nome,email (  $\sigma$  cidade = 'Rio de Janeiro' (Clientes))

Executar seleção   Download   Histórico

$\pi$  nome, email  
3 rows

$\sigma$  cidade = 'Rio de Janeiro'  
3 rows

Clientes  
20 rows

$\pi$  nome, email (  $\sigma$  cidade = 'Rio de Janeiro' ( Clientes ) )  
Execution time: 2 ms

Clientes.nome	Clientes.email
'Maria Santos'	'maria.santos@example.com'
'Carolina Menezes'	'carolina.menezes@example.com'
'Laura Mendes'	'laura.mendes@example.com'

#F3012 - Produto cartesiano (x)– Lista de possíveis livros para cada cliente.(é custoso tem que ter cuidado)

Álgebra Relacional   SQL   Editor de Grupo

$\pi$   $\sigma$   $\rho$   $\leftarrow$   $\rightarrow$   $\tau$   $\gamma$   $\wedge$   $\vee$   $\neg$   $=$   $\neq$   $\geq$   $\leq$   $\cap$   $\cup$   $+$   $-$   $\times$   $\bowtie$   $\ltimes$   $\ltimes$   $\ltimes$   $\ltimes$   $\ltimes$   $\ltimes$   $\triangleright$   $=$   $-$   $/^*$

$\{\}$   $\text{table}$   $\text{grid}$   $\text{edit}$

1  $\pi$  Livros.nome\_livro, Clientes.nome (Livros  $\times$  Clientes)

$\Pi$  Livros.nome\_livro, Clientes.nome ( Livros  $\times$  Clientes )

Execution time: 1 ms

Livros.nome_livro	Clientes.nome
'Percy Jackson e o Ladrão de Raios'	'João Silva'
'Percy Jackson e o Ladrão de Raios'	'Maria Santos'
'Percy Jackson e o Ladrão de Raios'	'Pedro Almeida'
'Percy Jackson e o Ladrão de Raios'	'Ana Souza'
'Percy Jackson e o Ladrão de Raios'	'Carlos Lima'

#F3013 – Junção – Como saber o nome da pessoa vendedora que realizou aquela venda?

Serve para relacionar e visualizar informações que estão em tabelas diferentes.

$\Pi$  id\_pedido, id\_vendedor\_vendas, id\_vendedor, nome (  $\sigma$  id\_vendedor\_vendas = id\_vendedor ( Vendas  $\times$  Vendedores ) )

Execution time: 2 ms

Vendas.id_pedido	Vendas.id_vendedor_vendas	Vendedores.id_vendedor	Vendedores.nome
1	3	3	'Gabriel Almeida'
2	1	1	'Lucas Silva'
3	2	2	'Camila Santos'
4	4	4	'Larissa Souza'
5	3	3	'Gabriel Almeida'
6	1	1	'Lucas Silva'
7	2	2	'Camila Santos'
8	4	4	'Larissa Souza'
9	3	3	'Gabriel Almeida'
10	1	1	'Lucas Silva'

```
1 -- π id_pedido, id_vendedor_vendas, id_vendedor, nome (σ id_vendedor_vendas = id_vendedor
2 (Vendas × Vendedores))
3 π id_pedido, id_vendedor_vendas, id_vendedor, nome (Vendas ⋈ id_vendedor_vendas = id_vendedor
4 Vendedores)
```

É um tipo de junção que vai fazer a relação entre duas tabelas onde os atributos são iguais em ambas tabelas

$\pi$  id\_pedido, id\_livro, nome\_livro ( Vendas  $\bowtie$  Livros )

Execution time: 0 ms

Vendas.id_pedido	Vendas.id_livro	Livros.nome_livro
1	7	'Gossip Girl Nao me Esqueca'
2	2	'Sangue de Lobo'
3	4	'O Simbolo Perdido'
4	8	'Cidade das Almas Perdidas'
5	6	'Pegasus e o Fogo do Olimpo'
6	6	'Pegasus e o Fogo do Olimpo'

Select DB (Clube do Livro) ▾

Livros

id_livro	number
nome_livro	string
autor	string
preco	number
qtd_estoque	number

Vendedores

id_vendedor	number
nome	string
anos_exp	number

Clientes

id_cliente	number
nome	string
cidade	string
email	string

Vendas

id_pedido	number
id_vendedor_vendas	number
id_livro	number
qtd_vendida	number

LivroMaisVendidos

id_livro	number
ano	number
mais_vendidos	number

LivrosRecomendados

id_livro	number
fonte_recomendacao	string
nota_media	number

#F3015 - Junção esquerda e direita – Será que existe alguma pessoa vendedora que não realizou nenhuma venda?

Junção a direita

$\Pi$  id\_vendedor\_vendas, id\_vendedor, nome ( Vendas  $\bowtie$  id\_vendedor\_vendas = id\_vendedor Vendedores )

Execution time: 2 ms

Vendas.id_vendedor_vendas	Vendedores.id_vendedor	Vendedores.nome
1	1	'Lucas Silva'
2	2	'Camila Santos'
3	3	'Gabriel Almeida'
4	4	'Larissa Souza'
null	5	'Rafael Lima'
null	6	'Júlia Rocha'

Trazer somente os que não venderam

$\sigma$  id\_vendedor\_vendas = null (  $\Pi$  id\_vendedor\_vendas, id\_vendedor, nome ( Vendas  $\bowtie$  id\_vendedor\_vendas = id\_vendedor Vendedores ) )

Execution time: 3 ms

Vendas.id_vendedor_vendas	Vendedores.id_vendedor	Vendedores.nome
null	5	'Rafael Lima'
null	6	'Júlia Rocha'



#F3016 – União – Quais os livros mais vendidos ou mais recomendados?

LivroMaisVendidos

id\_livro number

ano number

mais\_vendidos number

LivrosRecomendados

id\_livro number

fonte\_recomendacao string

nota\_media number

Mais vendidos:

$\pi_{id\_livro} ( \text{LivroMaisVendidos} ) \cup \pi_{id\_livro} ( \text{LivrosRecomendados} )$

Execution time: 2 ms

LivroMaisVendidos.id\_livro

3

7

9

12

14

16

18

19

## #F3017 – Intersecção – Se existe livros que são mais vendidos e também mais recomendados?



### 1. Junção (JOIN)

- A **junção** é uma operação que combina dados de duas ou mais tabelas com base em uma condição de relacionamento.
- Usa **chaves primárias e estrangeiras** para vincular registros relacionados.
- Exemplos de junções comuns:
  - **INNER JOIN**: Retorna apenas os registros que possuem correspondência em ambas as tabelas.
  - **LEFT JOIN**: Retorna todos os registros da tabela à esquerda e os correspondentes da tabela à direita (se existirem).
  - **RIGHT JOIN**: O contrário do LEFT JOIN.
  - **FULL JOIN**: Retorna todos os registros de ambas as tabelas, com correspondências onde existirem.

#### Exemplo de INNER JOIN:

Suponha duas tabelas:

Clientes

ID	Nome
1	Ana
2	João

Pedidos

ID	Cliente_ID	Produto
1	1	Celular
2	2	Notebook

sql

Copiar Editar

```
SELECT Clientes.Nome, Pedidos.Produto
FROM Clientes
INNER JOIN Pedidos ON Clientes.ID = Pedidos.Cliente_ID;
```

#### • Resultado:

Nome	Produto
Ana	Celular
João	Notebook

2. Interseção (INTERSECT)

- A **interseção** retorna os elementos que aparecem em **ambas as tabelas ou conjuntos de dados**, sem duplicatas.
- Em SQL, a operação **INTERSECT** é usada para encontrar registros comuns entre duas consultas.
- Diferente da junção, a interseção **não combina colunas diferentes**, apenas retorna os registros **exatamente iguais** entre duas tabelas.

Exemplo de INTERSECT:

Considere duas tabelas:

Tabela A

Nome
Ana
João
Carlos

Tabela B

Nome
João
Pedro

sql

Copiar

Editar

```
SELECT Nome FROM Tabela_A
INTERSECT
SELECT Nome FROM Tabela_B;
```

• Resultado:

Nome
João

 **Observação:** Nem todos os bancos de dados suportam `INTERSECT`. O MySQL, por exemplo, não tem esse operador nativamente.

Principais Diferenças

Característica	Junção (JOIN)	Interseção (INTERSECT)
Propósito	Combinar tabelas com base em uma relação	Retornar valores comuns entre duas tabelas
Combina colunas?	Sim	Não
Exemplo de uso	Unir clientes e pedidos por chave estrangeira	Encontrar usuários que estão em duas tabelas distintas
Disponibilidade	Suportado em SQL padrão	Nem todos os bancos suportam

Se precisar de mais detalhes, me avise! 🚀

#F3018 - Diferença - Quais são os livros que estão no cadastro do Clube do Livro, mas que não foram vendidos?

$\pi_{d\_livro}$  ( Livros ) -  $\pi_{id\_livro}$  ( Vendas )

Execution time: 2 ms

Livros.id\_livro

1

5

< 1 >

#F3019 - Renomear relações - (rô é ρ - renomea tabela especifica)

Pessoas candidatas para trabalhar com a 'Camila Santos'?

$\rho_v$  Vendedores x Vendedores

Execution time: 2 ms

V.id_vendedor	V.nome	V.anos_exp	Vendedores.id_vendedor	Vendedores.nome	Vendedor
1	'Lucas Silva'	5	1	'Lucas Silva'	
1	'Lucas Silva'	5	2	'Camila Santos'	
1	'Lucas Silva'	5	3	'Gabriel Almeida'	
1	'Lucas Silva'	5	4	'Larissa Souza'	
1	'Lucas Silva'	5	5	'Rafael Lima'	

$\sigma_{V.nome = 'Camila Santos'} ( \rho_V Vendedores \times Vendedores )$

Execution time: 1 ms

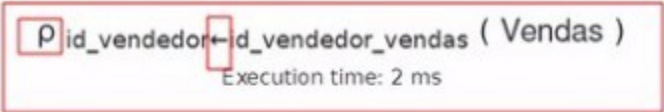
V.id_vendedor	V.nome	V.anos_exp	Vendedores.id_vendedor	Vendedores.nome	Vendedores.anos_exp
2	Camila Santos	3	1	'Lucas Silva'	5
2	'Camila Santos'	3	2	'Camila Santos'	5
2	'Camila Santos'	3	3	'Gabriel Almeida'	7
2	'Camila Santos'	3	4	'Larissa Souza'	5
2	'Camila Santos'	3	5	'Rafael Lima'	5
2	'Camila Santos'	3	6	'Júlia Rocha'	5

$\sigma_{V.anos\_exp < Vendedores.anos\_exp} ( \sigma_{V.nome = 'Camila Santos'} ( \rho_V Vendedores \times Vendedores ) )$

Execution time: 2 ms

V.nome	V.anos_exp	Vendedores.id_vendedor	Vendedores.nome	Vendedores.anos_exp
'Camila Santos'	3	1	'Lucas Silva'	5
'Camila Santos'	3	3	'Gabriel Almeida'	7

#F3020 - Renomear atributos (^ e) - Renomear relação e atributos

  
Execution time: 2 ms

Vendas.id_pedido	Vendas.id_vendedor	Vendas.id_livro	Vendas.qtd_vendida
1	3	7	2
2	1	2	5
3	2	4	3
4	4	8	1
5	3	6	2
6	1	6	4
7	2	3	1
8	4	10	3