Modelagem de dados é o processo de criar uma representação abstrata dos dados de um sistema, visando organizar, estruturar e definir como os dados serão armazenados, gerenciados e acessados. Ela permite que se entenda as necessidades de dados de uma aplicação, antecipando problemas de redundância e inconsistências.

Tipos de Modelagem de Dados

1. **Modelo Conceitual**:
    - **Objetivo:** Representar a estrutura de dados de maneira abstrata e sem detalhes de implementação.
    - **Componentes:**
        - **Entidades:** Representam objetos ou conceitos reais, como clientes, produtos ou pedidos.
        - **Atributos:** São propriedades ou características das entidades, como o nome do cliente ou o preço do produto.
        - **Relacionamentos:** Definem como as entidades se conectam, por exemplo, um cliente faz um pedido.
2. **Modelo Lógico**:
    - **Objetivo:** Representar os dados de forma mais detalhada, mas ainda sem preocupações com aspectos físicos de armazenamento.
    - **Componentes:**
        - **Tabelas:** Correspondem às entidades do modelo conceitual.
        - **Colunas/Atributos:** São os detalhes dos dados que cada tabela armazenará.
        - **Chaves Primárias (PK):** Identifica unicamente cada registro em uma tabela.
        - **Chaves Estrangeiras (FK):** Estabelecem vínculos entre tabelas (relacionamentos).
3. **Modelo Físico**:
    - **Objetivo:** Representar como os dados serão armazenados no banco de dados.
    - **Componentes:**
        - **Tabelas Físicas:** Implementações reais das tabelas no banco de dados.
        - **Índices:** Utilizados para melhorar a performance nas buscas.

        - **Tipos de Dados:** Definição do tipo (inteiro, texto, data, etc.) para cada coluna.
        - **Constraints (Restrições):** Definições de integridade, como "NOT NULL" ou "UNIQUE".

Conceito de Cardinalidade

A **cardinalidade** descreve a quantidade de instâncias de uma entidade que podem estar associadas a outra entidade em um relacionamento.

1. **1:1 (Um para Um):** Uma instância de uma entidade está associada a no máximo uma instância de outra entidade.
    - Exemplo: Cada pessoa tem um único número de CPF, e cada CPF pertence a uma única pessoa.
2. **1:N**
   (Um para Muitos): Uma instância de uma entidade está associada a várias instâncias de outra entidade, mas o contrário não é verdadeiro.
    - Exemplo: Um cliente pode fazer vários pedidos, mas cada pedido pertence a um único cliente.
3. **N:N**
   (Muitos para Muitos): Várias instâncias de uma entidade estão associadas a várias instâncias de outra entidade.
    - Exemplo: Um aluno pode se inscrever em vários cursos, e um curso pode ter vários alunos inscritos.

Exemplo Prático

Cenário: Vamos modelar um sistema de **biblioteca**.

1. **Modelo Conceitual:**
    - **Entidades:** Livro, Autor, Cliente, Empréstimo.
    - **Atributos:**
        - Livro: Título, ISBN, Editora, Ano.
        - Autor: Nome, Data de Nascimento.
        - Cliente: Nome, CPF, Endereço.
        - Empréstimo: Data de Empréstimo, Data de Devolução.
    - **Relacionamentos:**
        - Um livro pode ter vários autores (N).
        - Um cliente pode fazer vários empréstimos (1).

2. **Modelo Lógico:**
    - **Tabelas:** Livro, Autor, Cliente, Empréstimo, Livro_Autor.
    - **Chaves Primárias:**
        - Livro (ISBN).
        - Autor (AutorID).
        - Cliente (CPF).
        - Empréstimo (EmprestimoID).
    - **Chaves Estrangeiras:**
        - Livro_Autor (ISBN, AutorID).
        - Empréstimo (CPF, ISBN).

3. **Modelo Físico:**

    Tabelas Físicas com tipos de dados:

```
CREATE TABLE Livro (
  ISBN VARCHAR(13) PRIMARY KEY,
  Titulo VARCHAR(100),
  Editora VARCHAR(50),
  Ano INT
);

CREATE TABLE Autor (
  AutorID INT PRIMARY KEY AUTO_INCREMENT,
  Nome VARCHAR(100),
  DataNascimento DATE
);

CREATE TABLE Cliente (
  CPF CHAR(11) PRIMARY KEY,
  Nome VARCHAR(100),
  Endereco VARCHAR(200)
);

CREATE TABLE Emprestimo (
  EmprestimoID INT PRIMARY KEY AUTO_INCREMENT,
  CPF CHAR(11),
  ISBN VARCHAR(13),
  DataEmprestimo DATE,
  DataDevolucao DATE,
  FOREIGN KEY (CPF) REFERENCES Cliente(CPF),
  FOREIGN KEY (ISBN) REFERENCES Livro(ISBN)
);

CREATE TABLE Livro_Autor (
  ISBN VARCHAR(13),
```

```
    AutorID INT,
    PRIMARY KEY (ISBN, AutorID),
    FOREIGN KEY (ISBN) REFERENCES Livro(ISBN),
    FOREIGN KEY (AutorID) REFERENCES Autor(AutorID)
  );
```

Versão para postgresql

```
CREATE TABLE Cliente (
  CPF CHAR(11) PRIMARY KEY,
  Nome VARCHAR(100),
  Endereco VARCHAR(200)
);

CREATE TABLE Autor (
  AutorID serial PRIMARY KEY,
  Nome VARCHAR(100),
  DataNascimento DATE
);

CREATE TABLE Livro (
  ISBN VARCHAR(13) PRIMARY KEY,
  Titulo VARCHAR(100),
  Editora VARCHAR(50),
  Ano INT
);

CREATE TABLE Livro_Autor (
  ISBN VARCHAR(13),
  AutorID INT,
  PRIMARY KEY (ISBN, AutorID),
  FOREIGN KEY (ISBN) REFERENCES Livro(ISBN),
  FOREIGN KEY (AutorID) REFERENCES Autor(AutorID)
);

drop table emprestimo;
CREATE TABLE Emprestimo (
  EmprestimoID serial PRIMARY KEY,
  CPF CHAR(11),
  DataEmprestimo DATE,
  DataDevolucao DATE,
  FOREIGN KEY (CPF) REFERENCES Cliente(CPF)
);

CREATE TABLE Emprestimo (
  EmprestimoID serial PRIMARY KEY,
  CPF CHAR(11),
  ISBN VARCHAR(13),
  DataEmprestimo DATE,
  DataDevolucao DATE,
```

```
  FOREIGN KEY (CPF) REFERENCES Cliente(CPF),
  FOREIGN KEY (ISBN) REFERENCES Livro(ISBN)
);

alter table Emprestimo drop column ISBN;

CREATE TABLE LivroEmprestimo (
    Livro_ISBN varchar(13),
    EmprestimoID int
);

alter table LivroEmprestimo add constraint fk_emprestimo foreign key(EmprestimoID) references Emprestimo(EmprestimoID);
alter table LivroEmprestimo add constraint fk_livro foreign key(Livro_ISBN) references Livro(ISBN);
```