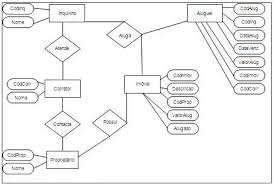
**O que é a modelagem DER e MER**

Basicamente os dois são modelos utilizados na linguagem de dados, com o MER(Modelo entidade-Relacionamento) sendo um modelo conceitual e o DER(Diagrama Entidade-Relacionamento) sendo sua representação através de um diagrama. Enquanto e MER descreve entidades(objetos do mundo real) e seus relacionamentos como Livro-Escrever, ou Carro-Andar, enquanto o DER representa isso de uma maneira mais gráfica através de diagramas, facilitando a compreensão e a comunicação

Exemplo

de DER:

**O que são ferramentas de modelagem?:**

As ferramentas de modelagem são softwares (onlines e instaláveis( no meu caso eu uso o mySQL, o MongoDB e o PostgreSQL)) usados para planejar, desenhar e documentar sistemas, especialmente no que diz a respeito á estrutura de dados.

Ele permite representar conceitos abstratos de um banco de dados de forma visual, para que analistas, desenvolvedores e gestores possam:

* Entender a estrutura dos dados.
* Comunicar ideias com clareza.
* Gerar automaticamente o código SQL (em alguns casos).
* Validar regras de negócios e integridade.

| **Nível** | **Descrição** | **Exemplos** |
| --- | --- | --- |
| **Conceitual** | Focado no negócio. Usa entidades e relacionamentos sem se preocupar com SGBD. | Modelo ER simples, sem tipos de dados. |
| **Lógico** | Já pensa em chaves primárias, tipos de dados, integridade. Ainda independente do SGBD. | MER com atributos e tipos. |
| **Físico** | Inclui detalhes específicos do banco (índices, tabelas, constraints, engine etc.). | Geração de script SQL para MySQL, PostgreSQL etc. |

**Funcionalidades comum das ferramentas de modelagem**

| **Funcionalidade** | **Explicação** |
| --- | --- |
| 🎨 Editor Visual | Interface para desenhar entidades, atributos e relacionamentos. |
| 🔑 Definição de Chaves | Suporte a chave primária, estrangeira e únicas. |
| 🛠️ Geração de SQL | Converte o modelo para código SQL para criação do banco. |
| 🔄 Engenharia reversa | Conecta a um banco real e gera o diagrama a partir dele. |
| 📄 Exportação | Exporta para imagem, PDF, SQL, XML etc. |
| 🤝 Colaboração | Permite que várias pessoas editem o mesmo modelo (em algumas ferramentas online). |

**Explicando de uma forma mais detalhada sobre Modelagem ER**

A modelagem de Entidade-Relacionamento (ER) é uma técnica criada por Peter Chen em 1976 para descrever a estrutura de dados de uma forma visual e conceitual.

Essa modelagem permite representar:

* As entidades (coisas ou objetos importantes para o sistema).
* Seus atributos (características).
* E os relacionamentos entre essas entidades.

Ela não depende de um banco de dados especifico, o que a torna útil para a fase de analise e projeto do sistema.

**Componentes da Modelagem ER**

**1.ENTIDADES:**

* Representam objetos ou conceitos do mundo real.
* **Ex: Aluno, Professor, Curso.**
* Representadas graficamente como retângulos.

**Tipos de entidades:**

* **Forte:** Tem existência própria ( Ex: **Aluno** ).
* **Fraca:** Depende de outra entidade para existir ( Ex: **Dependente** de **Funcionário).**

**2.Atributos:**

* São as propriedades que descrevem as entidades.
* Ex: nome, data\_nascimento, matrícula.

Tipos

| **Tipo** | **Exemplo** | **Descrição** |
| --- | --- | --- |
| Simples | CPF | Indivisível |
| Composto | Endereço → Rua, Cidade | Pode ser dividido |
| Multivalorado | Telefones | Pode ter vários valores |
| Derivado | Idade | Calculado a partir de outro atributo (ex: data\_nascimento) |

**3.Relacionamentos**

* Indicam como entidades se conectam.
* Ex: Um **Aluno** matriculou-se em um **Curso**.

**Tipos de Relacionamento:**

* **1:1**( um para um )
* **1:N**(um para muitos)
* **N:M**(muitos para muitos)

Relacionamentos podem ter atributos próprios, como a data da matricula.

**4.Cardinalidade**

Define quantas ocorrências de uma entidade estão relacionadas a outra. Ex:

* Um **Professor** pode dar varias **Disciplinas**, mas cada **Disciplina** tem apenas um **Professor** -> relacionamento 1:N.

**O que é MER mais aprofundadamente?**

O Modelo Entidade-Relacionamento (**MER**) é a representação gráfica do modelo ER. Ele é usado para documentar visualmente o que foi definido na modelagem.

O MER é composto por:

* Entidades como retângulos
* Relacionamento como losangos
* Atributos como elipses
* Conectores que indicam cardinalidade (ex: 1, N )

Exemplo:

[Aluno] ----- (Matricula) ----- [Curso]

| |

[nome, idade] [nome, carga\_horária]

Ferramentas de Modelagem de Dados, mais aprofundamento nas pesquisas anteriores

**Introdução**

A modelagem de dados é uma etapa critica no desenvolvimento de sistemas de informação. Através dela, conseguimos representar logicamente a estrutura e o comportamento dos dados antes de qualquer implementação pratica. Essa representação precisa ser clara, precisa e comunicável – por isso, utilizamos ferramentas de modelagem.

Essas ferramentas são softwares especializados que nos permitem criar diagramas visuais e lógicos( como diagramas ER,UML ou diagramas físicos de banco de dados ) com a maior rapidez, segurança e organização.

Finalidades das Ferramentas de Modelagem

As ferramentas de modelagem são empregadas para:

* Representar entidades, atributos e relacionamentos de um banco de dados.
* Planejar estrutura de tabelas, chaves primarias e estrangeiras.
* Validar regras de integridade e cardinalidade
* Criar documentação visual de sistemas.
* Automatizar a geração de scripts SQL e engenharia reversa

Tipos de Modelagem Auxiliadas por Ferramentas

| **Nível de Modelagem** | **Descrição** | **Exemplo** |
| --- | --- | --- |
| **Conceitual** | Alta abstração. Sem foco em implementação. | Diagrama ER com entidades e relacionamentos. |
| **Lógico** | Estrutura com tipos de dados, chaves e normalização. | Tabelas e colunas definidas, mas sem engine. |
| **Físico** | Detalhes específicos do SGBD. | Scripts SQL para criação no MySQL/PostgreSQL. |

**Principais Funcionalidades das Ferramentas**

As funcionalidades variam, mas as melhores ferramentas oferecem:

1. **Editor Visual Drag-and-Drop**
   * Arraste e conecte elementos facilmente.
2. **Editor de Código ou DSL (Domain Specific Language)**
   * Permite escrever tabelas com sintaxe própria (ex: dbdiagram.io).
3. **Geração de SQL**
   * Criação automática de comandos CREATE TABLE, ALTER TABLE etc.
4. **Engenharia Reversa**
   * Conecta-se a um banco de dados real e gera o modelo a partir dele.
5. **Exportação e Compartilhamento**
   * Gera arquivos PDF, PNG, SQL ou link online para visualização em equipe.
6. **Validação de Modelo**
   * Verifica conflitos como ausência de chave primária, relacionamentos mal definidos etc.

| **Ferramenta** | **SQL Automático** | **Engenharia Reversa** | **Online** | **Multibanco** | **Ideal para** |
| --- | --- | --- | --- | --- | --- |
| MySQL Workbench | ✅ | ✅ | ❌ | ❌ | MySQL puro |
| dbdiagram.io | ✅ | ❌ | ✅ | ✅ (geração SQL) | Prototipagem e ensino |
| DBeaver | ✅ | ✅ | ❌ | ✅ | Devs e DBAs |
| pgModeler | ✅ | ✅ | ❌ | ❌ (PostgreSQL) | Sistemas PostgreSQL |
| draw.io | ❌ | ❌ | ✅ | ✅ (visuais) | Apresentações/documentação |

**Um texto aprofundado em Cassos de Uso e em Classes**

**Primeiramente o que é UML?**

A UML(Unified Modeling Language ) é uma linguagem visual padronizada para a modelagem de sistemas orientadas a objetos.Ela Permite que analistas, desenvolvedores e stakeholders comuniquem e documentem a estrutura e o comportamento de um sistema de forma clara em concisa, independente da linguagem que é usada na programação.

UML não é um processo de desenvolvimento em si, mas uma linguagem de modelagem que pode ser usada com qualquer metodologia ( ágil, cascata, RUP, etc ).

**Diagrama de casos de Uso**

O Diagrama de Casos De Uso (Use Case Diagram) tem o objetivo de representar as funcionalidades do sistema sob o ponto de vista dos usuários(atores). Ele ajuda a identificar o que o sistema deve fazer, sem detalhar como isso será implementado.

É especialmente útil nas fases iniciais de analise de requisitos, para alinhar o entendimento entre a equipe técnica e cliente.

**Elementos Principais**

| **Elemento** | **Símbolo** | **Descrição** |
| --- | --- | --- |
| **Ator** | Boneco | Quem interage com o sistema (usuário, sistema externo etc.) |
| **Caso de uso** | Elipse | Função ou serviço oferecido pelo sistema |
| **Sistema** | Retângulo | Delimita os casos de uso que fazem parte do sistema |
| **Relacionamentos** | Linhas (simples, com setas) | Mostram como atores e casos de uso se conectam |

**Relacionamento entre casos de uso:**

* **Include(<<include>>):** Um caso de uso **obrigatoriamente chama outro** ( reutilização ).
* **Ex:** “Pagar como Cartão” <<include>> “Validar Pagamento”.
* **Extend(<<extend>>):**Um caso de uso **pode opcionalmente executar outro**, se houver uma condição.
* **Ex:”**Realizar Compra” <<extend>> “Aplicar Cupom de Desconto”.

### Exemplo

**Sistema de Biblioteca**:

* Atores: Aluno, Bibliotecário
* Casos de uso: Buscar livro, Reservar livro, Devolver livro

Aluno ———> (Buscar livro)

———> (Reservar livro)

Bibliotecário ———> (Cadastrar livro)

———> (Devolver livro)

Esse diagrama mostra o que o sistema deve permitir e **quem usa cada funcionalidade**.

**Diagrama de Classes**

O Diagrama de Classes( Class Diagram ) descreve a estrutura estática do sistema, mostrando as classes, seus atributos, métodos e os relacionamentos entre elas

### Elementos principais

1. **Classe**  
   Representada por um retângulo com três divisões:

---------------------

| NomeDaClasse |

---------------------

| atributos |

---------------------

| métodos |

---------------------

1. **Atributos**
   * Definem o **estado** de um objeto (variáveis).
   * Ex: nome: String, idade: int
2. **Métodos**
   * Definem o **comportamento** (funções da classe).
   * Ex: emprestarLivro(), calcularMulta()
3. **Visibilidade**
   * + Público
   * - Privado
   * # Protegido

| **Relacionamento** | **Símbolo** | **Significado** |
| --- | --- | --- |
| **Associação** | Linha simples | Uma classe usa ou se relaciona com outra |
| **Agregação** | Linha com losango branco | Relação "tem/contém", mas independente |
| **Composição** | Linha com losango preto | Relação "faz parte de", dependente |
| **Generalização (Herança)** | Linha com triângulo | Uma classe herda atributos e métodos de outra |

### **Exemplo de Diagrama de Classes**

**Sistema de Biblioteca**:

* Classe: Livro

Atributos: titulo, autor, codigo

Métodos: emprestar(), devolver()

* Classe: Usuario

Atributos: nome, matricula

Métodos: reservarLivro()

* Relacionamento:

Usuario **possui** vários Livro → associação 1:N

----------------

| Livro |

----------------

| titulo |

| autor |

| codigo |

----------------

| emprestar() |

| devolver() |

----------------

----------------

| Usuario |

----------------

| nome |

| matricula |

----------------

| reservarLivro() |

----------------

Usuario 1 -------- \* Livro

**Casos de Uso vs Diagrama de Classes**

| **Aspecto** | **Casos de Uso** | **Diagrama de Classes** |
| --- | --- | --- |
| Foco | Funcionalidades (comportamento) | Estrutura do sistema (dados e lógica) |
| Visão | Do usuário | Do desenvolvedor / arquiteto |
| Nível | Alto nível (abstrato) | Técnico (detalhado) |
| Quando usar | Levantamento de requisitos | Projeto técnico e implementação |
| Complementaridade | Identifica **o que fazer** | Mostra **como será feito** |

**Projeto de Sistema**

O Projeto de sistema é basicamente um processo estruturado de etapas que incluem, Planejamento, analise, desenvolvimento e manutenção, que juntas garantem a funcionalidade, escalabilidade do sistema, em uma explicação mais detalhada:

1. Planejamento

A etapa de planejamento é fundamental para definir os objetivos do sistema e traçar um caminho claro para o seu desenvolvimento, um passo a passo simples seria

* **Identificar os problemas:** Converse com os stakeholders (usuários, gestores) para entender as dificuldades atuais em como o sistema pode solucioná-las
* **Defina o Escopo:** Determine o que o sistema incluirá e o que estará fora do seu alcance
* **Monte um cronograma:** Use ferramentas como o GitHup Projects( recomendação pessoal ) para organizar as tarefas e prazos
* **Estimo de custos:** Este aqui é meio fora da minha modalidade mas basicamente, inclui os custos de equipamentos, programas, equipes, licenciamentos entre outros.

1. Análise de Requisitos

Essa fase envolve a coleta detalhada das funcionalidades esperadas e das características do sistema. Para realizar esta tarefa

* **Liste Requisitos Funcionais:** Coloque aqui as principais funcionalidades que seu sistema permitir realizar ( ex: permitir cadastro de clientes, gerar relatórios financeiros).
* **Determine Requisitos não Funcionais:** Aqui se é especificado as características do sistema, como o desempenho esperado, segurança, usabilidade, basicamente características que o usuário não irá necessariamente ver.
* **Crie Diagramas:** Basicamente nesta parte você tem que colocar os diagramas de seu projeto para mapear fluxos de dados e casos de uso, aqui se é usado ferramentas como o draw.io( que pode ser instalado como uma extensão ) ou qualquer outro tipo de software ou site que você considerar melhor.

1. Projeto (design)

Nesta parte se é colocada o protótipo de alta(muitos detalhes, cores especificas, posições especificas, basicamente como programador vai programara para ser o site, detalhe por detalhe), media(pode colocar as posições especificas, basicamente é como o seu site vai ficar no geral, mas o programador tem alguma margem de manobra para fazer alterações, diferente do de alta fidelidade) ou baixa(Pouco detalhe, nenhuma cor em especifico apenas onde via ficar cada coisa, neste tipo de protótipo o programador tem mais liberdade para fazer mudanças) fidelidade.

Aqui também será como será a estrutura do arquivo( como o MVC – Model View Controller por exemplo) e onde o programador verá qual tipo de banco de dados será usado neste projeto.

1. Desenvolvimento

Esta etapa é onde os programadores colocam a mão na massa, por assim dizer, aqui é onde será escolhida a linguagem de programação(java, PHP, JavaScript, Python) e onde será escolhidos os frameworks que serão usados dando no frontend ( Angular/React) e para backend (express,Django) .

Também serão implementados recursos Modulares, no qual iremos testar individualmente cada um deles antes de integrarmos eles.

E onde também será feita o versionamento do nosso programa através de it/github, para gerenciar as alterações e evitar a perda dos trabalhos.

1. Teste

Bem esta parte não tem muito o que dizer é basicamente realizar três tipos de teste em nosso programa antes de entregarmos que seria os teste unidarios( através de jUnit ou Pytest ), Fazer o teste de integração, que é para basicamente para ver se cada um dos módulos que fizemos trabalha de uma maneira boa em conjunto e a fase de aceitação, no qual entregamos para os nossos stakeholders validar se o sistema cumpre suas expectativas.

1. Implantação e Manutenção( não será usada na competição)

Esta parte é realmente a parte final de todo este processo, basicamente nos iremos lançar para o publico, ou melhor implementar definitivamente o nosso programa(lembre-se de usar servers em nuvem como o aws e azure { em nossa competição, ignore esta parte }), oferecer treinamento e monitorar e atualizar o site sempre que possível.

### Consultas em MySQL: Como Realizar e Aplicar com Eficiência

As consultas em MySQL são a base para a manipulação e recuperação de dados armazenados em bancos de dados relacionais. Com elas, é possível acessar informações, realizar cálculos, organizar registros e até modificar os dados existentes. Este processo é essencial para desenvolvedores, analistas de dados e administradores de banco de dados que lidam com grandes volumes de informações. A seguir, apresentamos como realizar consultas em MySQL, explicando os principais comandos e etapas.

#### 1. Preparação do Ambiente

Antes de realizar consultas, é necessário configurar e acessar o banco de dados MySQL. Para isso:

* **Instale o MySQL**: Use um gerenciador de pacotes como apt (Linux) ou faça o download no site oficial do MySQL.
* **Acesse o servidor**: Utilize o terminal ou ferramentas como MySQL Workbench para conectar-se ao banco.
* **Crie ou importe o banco de dados**: Use o comando CREATE DATABASE nome\_do\_banco; ou importe um arquivo .sql com mysql -u usuario -p nome\_do\_banco < arquivo.sql

#### 2. Consultas Básicas: SELECT

O comando SELECT é o mais utilizado em consultas e permite buscar dados específicos. Para usá-lo:

* **Selecionar todas as colunas**:

SELECT \* FROM nome\_tabela;

* **Selecionar colunas específicas**:

SELECT coluna1, coluna2 FROM nome\_tabela;

* **Filtrar resultados com WHERE**:

SELECT \* FROM nome\_tabela WHERE coluna = 'valor';

* **Ordenar resultados**:

SELECT \* FROM nome\_tabela ORDER BY coluna ASC;

#### 3. Consultas com Agregações

As funções de agregação são úteis para análises estatísticas e sumarização de dados. Exemplos:

* **Contar registros**:

SELECT COUNT(\*) FROM nome\_tabela;

* **Obter valores máximos e mínimos**:

SELECT MAX(coluna), MIN(coluna) FROM nome\_tabela;

* **Calcular médias ou somas**:

SELECT AVG(coluna), SUM(coluna) FROM nome\_tabela;

* **Agrupar resultados com GROUP BY**:

SELECT coluna, COUNT(\*) FROM nome\_tabela GROUP BY coluna;

#### 4. Consultas Avançadas

Consultas mais complexas permitem trabalhar com múltiplas tabelas e condições. Exemplos incluem:

* **Joins (junção de tabelas)**:

SELECT tabela1.coluna1, tabela2.coluna2

FROM tabela1

JOIN tabela2 ON tabela1.chave\_primaria = tabela2.chave\_estrangeira;

* **Subconsultas**:

SELECT coluna FROM tabela WHERE coluna IN (SELECT coluna FROM outra\_tabela);

* **Consultas com CASE**:

SELECT coluna,

CASE

WHEN coluna > 100 THEN 'Alto'

ELSE 'Baixo'

END AS classificacao

FROM tabela;

#### 5. Inserção, Atualização e Exclusão de Dados

Além de consultas, MySQL permite manipular dados diretamente:

* **Inserir novos registros**:

INSERT INTO tabela (coluna1, coluna2) VALUES ('valor1', 'valor2');

* **Atualizar dados existentes**:

UPDATE tabela SET coluna = 'novo\_valor' WHERE coluna = 'valor\_antigo';

* **Excluir registros**:

DELETE FROM tabela WHERE coluna = 'valor';

#### 6. Boas Práticas e Otimização

Para garantir eficiência em consultas MySQL:

* **Crie índices**: Use o comando CREATE INDEX em colunas frequentemente utilizadas em filtros ou joins.
* **Evite SELECT \* em grandes tabelas**: Liste apenas as colunas necessárias.
* **Use LIMIT**: Restrinja o número de registros retornados para melhorar o desempenho.

SELECT \* FROM tabela LIMIT 10;

* **Monitore desempenho**: Use a função EXPLAIN para verificar como o MySQL executa uma consulta e identificar gargalos