MYSQL: ARQUITETURA E PLANEJAMENTO DO BANCO DE DADOS - RESUMO

1. **Introdução ao MySQL:**
    - Breve apresentação do MySQL como sistema de gerenciamento de banco de dados (SGBD) relacional.
    - Importância do planejamento adequado do banco de dados para o sucesso de um projeto.

Introdução ao MySQL:

Bem-vindos ao tópico de Introdução ao MySQL! Nesta parte do nosso curso, vamos explorar o MySQL como um sistema de gerenciamento de banco de dados (SGBD) relacional e entender a importância do planejamento adequado do banco de dados para o sucesso de um projeto.

MySQL como SGBD Relacional:

O MySQL é um SGBD de código aberto muito popular e amplamente utilizado em todo o mundo.

Ele oferece uma solução robusta e eficiente para armazenar, gerenciar e manipular grandes volumes de dados de forma organizada.

Sendo um banco de dados relacional, o MySQL baseia-se no modelo relacional, que utiliza tabelas para armazenar os dados e estabelece relações entre essas tabelas por meio de chaves.

Importância do Planejamento do Banco de Dados:

Um bom planejamento do banco de dados é fundamental para o sucesso de qualquer projeto de software que envolva a utilização de dados.

Durante o desenvolvimento de um sistema, o banco de dados desempenha um papel essencial, uma vez que é responsável por armazenar e recuperar informações de maneira confiável e eficiente.

Ao planejar adequadamente o banco de dados, podemos garantir:

1. **Organização e Estrutura:** Definir a estrutura correta para as tabelas, estabelecendo relacionamentos entre elas de forma coerente. Isso permite que os dados sejam armazenados de maneira organizada, facilitando sua recuperação e garantindo a integridade dos dados.

2. **Desempenho e Otimização:** Projetar o banco de dados de forma otimizada para atender às necessidades do sistema, garantindo que as consultas e operações sejam executadas de maneira rápida e eficiente.

Isso é essencial para sistemas que precisam lidar com um grande número de acessos simultâneos.

3. **Escalabilidade:** Planejar o banco de dados de forma que ele possa crescer conforme a demanda, permitindo a inclusão de novos dados e funcionalidades sem comprometer o desempenho do sistema.

4. **Segurança dos Dados:** Estabelecer as permissões de acesso adequadas para garantir que apenas usuários autorizados possam acessar e modificar os dados, protegendo assim a integridade e a confidencialidade das informações.

5. **Manutenção e Evolução:** Facilitar a manutenção e evolução do sistema ao longo do tempo, permitindo que novas funcionalidades sejam adicionadas e que mudanças nos requisitos do sistema sejam realizadas de forma mais simples.

Portanto, o planejamento adequado do banco de dados é um dos pilares fundamentais para o sucesso de qualquer projeto de software.

2. **Arquitetura do MySQL:**
    - Visão geral dos componentes essenciais da arquitetura do MySQL.
    - Estrutura cliente-servidor e como as conexões são estabelecidas.
    - Funcionamento do servidor MySQL e sua interação com o armazenamento de dados.

Arquitetura do MySQL:

Neste tópico, exploraremos a arquitetura do MySQL, compreendendo os componentes essenciais, a estrutura cliente-servidor e o funcionamento do servidor MySQL em sua interação com o armazenamento de dados.

Componentes Essenciais da Arquitetura do MySQL:

A arquitetura do MySQL é composta por três principais componentes:

1. **Servidor MySQL:** É o núcleo do sistema de gerenciamento de banco de dados. Ele é responsável por receber e processar as solicitações dos clientes, gerenciar as conexões com o banco de dados, executar consultas e comandos, bem como gerenciar os dados armazenados.

2. **Clientes MySQL:** São os aplicativos ou programas que se conectam ao servidor MySQL para acessar, manipular e consultar os dados do banco de dados. Os clientes podem variar desde aplicativos de linha de comando até interfaces gráficas e aplicações web.

3. **Armazenamento de Dados:** É onde os dados do banco de dados são fisicamente armazenados. O MySQL suporta diferentes mecanismos de armazenamento, como MyISAM e InnoDB, cada um com suas próprias características e vantagens.

Estrutura Cliente-Servidor e Estabelecimento de Conexões:

A arquitetura do MySQL segue o modelo cliente-servidor, o que significa que o servidor MySQL é responsável por processar as solicitações dos clientes e retornar os resultados correspondentes.

O processo de estabelecimento de conexões ocorre da seguinte forma:

1. **Cliente envia solicitação:** O cliente MySQL envia uma solicitação ao servidor MySQL para estabelecer uma conexão.

Essa solicitação contém informações como o nome de usuário e a senha do cliente.

2. **Servidor responde:** O servidor MySQL recebe a solicitação do cliente e, caso as credenciais sejam válidas, estabelece a conexão.

3. **Comunicação bidirecional:** Uma vez estabelecida a conexão, o cliente e o servidor podem se comunicar bidirecionalmente.

O cliente envia comandos SQL ao servidor para realizar operações no banco de dados, e o servidor retorna os resultados das consultas e operações.

Funcionamento do Servidor MySQL e sua Interação com o Armazenamento de Dados:

O servidor MySQL recebe as solicitações dos clientes e as processa por meio de um motor de armazenamento específico.

Cada motor de armazenamento possui características e otimizações próprias.

Por exemplo, o mecanismo InnoDB é conhecido por suportar transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade), garantindo a integridade dos dados.

Quando um cliente executa comandos SQL, o servidor MySQL analisa esses comandos e decide como executá-los.

O servidor interage com o mecanismo de armazenamento adequado para realizar as operações necessárias no banco de dados.

Em seguida, o servidor retorna os resultados ao cliente.

Entender a arquitetura do MySQL é essencial para desenvolver aplicações eficientes e escaláveis.

3. **Modelagem de Dados:**
    - Explicação sobre a modelagem de dados e a importância de uma boa representação do sistema.
    - Discussão sobre os principais modelos conceituais, como o modelo entidade-relacionamento (ER).
    - Exemplos práticos de como mapear os requisitos de negócio em uma estrutura de banco de dados.

Neste tópico, abordaremos a modelagem de dados, que consiste em representar a estrutura e as relações dos dados em um sistema, bem como a importância de uma boa representação para o sucesso do projeto.

O que é Modelagem de Dados e sua Importância:

A modelagem de dados é um processo fundamental no desenvolvimento de sistemas, pois permite representar de forma organizada as informações que serão armazenadas no banco de dados.

Essa representação é essencial para garantir a integridade, a consistência e a compreensão dos dados ao longo do ciclo de vida do projeto.

Uma modelagem de dados bem-feita possibilita:

1. **Visão Estruturada:** Proporcionar uma visão clara e estruturada dos dados que serão manipulados pelo sistema, facilitando o entendimento e a comunicação entre os membros da equipe.

2. **Identificação de Entidades e Relacionamentos:** Permitir a identificação das entidades (objetos, pessoas, conceitos) e seus relacionamentos no contexto do sistema.

3. **Identificação de Atributos:** Identificar os atributos (características) das entidades, que representam os dados específicos de cada entidade.

4. **Evitar Redundância e Inconsistência:** Evitar a redundância de dados e a inconsistência de informações, garantindo a integridade e a confiabilidade dos dados.

5. **Base para o Planejamento do Banco de Dados:** Servir como base para o projeto do banco de dados, incluindo a criação de tabelas, chaves e relacionamentos.

Modelo Entidade-Relacionamento (ER):

O modelo Entidade-Relacionamento (ER) é uma das técnicas mais utilizadas para representar a modelagem de dados.

Ele consiste em utilizar diagramas para representar as entidades (tabelas) e os relacionamentos entre essas entidades.

- Entidades: São objetos ou conceitos que serão representados no banco de dados, podendo ser coisas físicas ou conceituais.
- Atributos: São as características das entidades, ou seja, as informações que precisam ser armazenadas.
- Relacionamentos: Representam as associações entre as entidades, podendo ser do tipo um para um, um para muitos ou muitos para muitos.

Exemplos Práticos de Mapeamento de Requisitos em uma Estrutura de Banco de Dados:

Para exemplificar o processo de mapeamento de requisitos em uma estrutura de banco de dados, vamos considerar um cenário de um sistema de gerenciamento de biblioteca.

Requisitos:

- O sistema deve armazenar informações sobre livros, autores e usuários.
- Cada livro possui um título, um autor e uma data de publicação.
- Cada autor possui um nome e um país de origem.
- Cada usuário possui um nome, um e-mail e uma lista de livros emprestados.

Nesse exemplo, podemos mapear os requisitos da seguinte forma:

- Entidades:
  - Livro (atributos: título, data de publicação)
  - Autor (atributos: nome, país de origem)
  - Usuário (atributos: nome, e-mail)

- Relacionamentos:
  - Livro está relacionado com Autor (relacionamento "escrito por")
  - Usuário está relacionado com Livro (relacionamento "empresta")

Com essa representação, podemos visualizar claramente as entidades envolvidas, seus atributos e os relacionamentos entre elas.

A modelagem de dados é um passo crucial para o desenvolvimento de um sistema eficiente e organizado.

Nos próximos tópicos, continuaremos explorando o planejamento do banco de dados no MySQL, incluindo a normalização e a definição das tabelas, chaves e índices.

Vamos avançar no conhecimento para criar sistemas sólidos e bem estruturados!

4. **Normalização de Dados:**
    - Definição dos conceitos de normalização e a importância de evitar redundância e inconsistência nos dados.
    - Explanação das principais formas normais (1NF, 2NF, 3NF) e suas vantagens.
    - Como aplicar a normalização na modelagem do banco de dados.

A normalização de dados é um processo importante na modelagem de banco de dados, visando evitar redundância e inconsistência nos dados, garantindo a integridade e eficiência do sistema.

Definição de Normalização e a Importância:

A normalização é um conjunto de regras aplicadas às tabelas de um banco de dados para eliminar redundâncias e inconsistências, minimizando o espaço de armazenamento e melhorando o desempenho das consultas.

Através da normalização, buscamos garantir a integridade dos dados e facilitar a manutenção do banco de dados à medida que o sistema evolui.

Principais Formas Normais (1NF, 2NF, 3NF) e Suas Vantagens:

1. **Primeira Forma Normal (1NF):**
    - A primeira forma normal exige que cada atributo de uma tabela contenha apenas valores atômicos, ou seja, valores indivisíveis.
    - Isso significa que não deve haver atributos multivalorados ou campos repetitivos em uma mesma coluna.
    - A 1NF evita redundância de dados, facilita a busca e manipulação dos registros e é o primeiro passo para a normalização.

2. **Segunda Forma Normal (2NF):**
    - A segunda forma normal é alcançada após atender aos critérios da 1NF e identificar chaves parciais em uma tabela.
    - Nessa forma normal, criam-se novas tabelas para evitar dependências parciais em relação à chave primária.
    - Isso reduz a redundância de dados e melhora a integridade do banco de dados.

3. **Terceira Forma Normal (3NF):**
    - A terceira forma normal é atingida quando uma tabela está na 2NF e não possui dependências transitivas não baseadas na chave primária.
    - Em outras palavras, todos os atributos não chave devem depender diretamente da chave primária.
    - A 3NF elimina a dependência indireta entre atributos, reduzindo ainda mais a redundância e melhorando o desempenho das consultas.

Como Aplicar a Normalização na Modelagem do Banco de Dados:

Para aplicar a normalização na modelagem do banco de dados, siga os seguintes passos:

1. Identifique as tabelas e seus atributos.
2. Verifique se cada atributo contém apenas valores atômicos (1NF).
3. Identifique as chaves parciais e as dependências parciais (2NF).
4. Separe as tabelas para eliminar as dependências parciais e as redundâncias.
5. Verifique as dependências transitivas e elimine-as (3NF).

É importante ressaltar que a normalização não precisa ser aplicada de forma rigorosa em todos os casos, e a decisão de até que ponto normalizar dependerá das necessidades específicas do projeto.

A normalização é uma etapa crítica na modelagem de banco de dados, garantindo que o sistema seja organizado, eficiente e de fácil manutenção.

5. **Planejamento do Banco de Dados:**
    - Identificação dos requisitos e regras de negócio que irão influenciar a estrutura do banco de dados.
    - Definição das entidades, atributos, relacionamentos e chaves.
    - Organização dos dados em tabelas e definição de índices para otimização de consultas.

O planejamento do banco de dados é uma etapa essencial na criação de sistemas eficientes e bem-estruturados.

Neste tópico, abordaremos a identificação dos requisitos e regras de negócio que influenciam a estrutura do banco de dados, a definição das entidades, atributos, relacionamentos e chaves, bem como a organização dos dados em tabelas e a definição de índices para otimização de consultas.

Identificação dos Requisitos e Regras de Negócio:

Antes de iniciar o planejamento do banco de dados, é fundamental compreender os requisitos do sistema e as regras de negócio que guiarão o desenvolvimento.

Essa etapa envolve a análise cuidadosa dos documentos de especificação do projeto, a interação com os stakeholders e o entendimento profundo das necessidades dos usuários finais.

Ao identificar os requisitos e regras de negócio, podemos definir os principais objetos que comporão o banco de dados e os relacionamentos entre eles.

Definição de Entidades, Atributos, Relacionamentos e Chaves:

Com base na identificação dos requisitos e regras de negócio, procedemos com a definição das entidades, que são os objetos ou conceitos que serão representados no banco de dados.

Cada entidade possui atributos, que são as características específicas relacionadas a ela.

Além disso, devemos estabelecer os relacionamentos entre as entidades.

Esses relacionamentos representam as associações e dependências entre os objetos do sistema.

Podem ser do tipo um para um, um para muitos ou muitos para muitos, dependendo das necessidades do projeto.

Também é essencial identificar as chaves primárias e estrangeiras que garantirão a integridade e a consistência dos dados.

A chave primária é um atributo que identifica unicamente cada registro em uma tabela, enquanto a chave estrangeira estabelece uma relação entre duas tabelas, referenciando a chave primária de outra tabela.

Organização dos Dados em Tabelas e Definição de Índices:

Após a definição das entidades, atributos, relacionamentos e chaves, organizamos os dados em tabelas.

Cada entidade corresponderá a uma tabela do banco de dados.

As tabelas devem ser projetadas de forma a evitar a redundância e a inconsistência de dados, de acordo com os conceitos de normalização que abordamos anteriormente.

Para otimizar o desempenho das consultas, é recomendável definir índices nas colunas que serão frequentemente utilizadas em cláusulas de busca, como as colunas das chaves primárias e estrangeiras.

Os índices aceleram a busca por registros no banco de dados, melhorando a eficiência das operações.

Ao final do planejamento do banco de dados, teremos uma estrutura organizada, coerente e eficiente, pronta para receber os dados do sistema e atender às necessidades dos usuários finais.

6. **Criação do Banco de Dados no MySQL:**
    - Exemplificação dos comandos SQL para criar tabelas, definir chaves primárias e estrangeiras.
    - Demonstração de como criar índices para melhorar o desempenho das consultas.
    - Considerações sobre tipos de dados e limitações do MySQL.

Criação do Banco de Dados no MySQL:

Neste tópico, vamos aprender como criar o banco de dados no MySQL, utilizando comandos SQL para criar tabelas, definir chaves primárias e estrangeiras, além de criar índices para melhorar o desempenho das consultas.

Também faremos considerações importantes sobre tipos de dados e limitações do MySQL.

Exemplificação dos Comandos SQL para Criação de Tabelas e Definição de Chaves:

Para criar uma tabela no MySQL, utilizamos o comando CREATE TABLE.

Vamos exemplificar a criação de algumas tabelas considerando o cenário do nosso sistema de gerenciamento de biblioteca.

```sql
-- Tabela para armazenar informações dos livros
CREATE TABLE Livro (
    id INT AUTO_INCREMENT PRIMARY KEY,
    titulo VARCHAR(100) NOT NULL,
    data_publicacao DATE,
    autor_id INT,
    FOREIGN KEY (autor_id) REFERENCES Autor(id)
);

-- Tabela para armazenar informações dos autores
CREATE TABLE Autor (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    pais_origem VARCHAR(50)
);
```

Nesse exemplo, criamos três tabelas: Livro, Autor e Usuario.

Na tabela Livro, definimos uma chave primária `id` como chave única para cada registro, e um campo `autor_id` como chave estrangeira que faz referência à tabela Autor.

Demonstração de Como Criar Índices:

Para criar índices no MySQL, utilizamos o comando `CREATE INDEX`. Índices são úteis para acelerar a busca e recuperação de dados em colunas específicas. Vamos criar um índice na coluna `autor_id` da tabela Livro:

```
CREATE INDEX idx_autor_id ON Livro (autor_id);
```

Com esse índice, as consultas que utilizarem a coluna `autor_id` serão otimizadas, pois o MySQL poderá localizar os registros mais rapidamente.

Considerações sobre Tipos de Dados e Limitações do MySQL:

É importante escolher os tipos de dados adequados para cada coluna, de acordo com o tipo de informação que será armazenada.

O MySQL suporta diversos tipos de dados, como INT (inteiro), VARCHAR (texto), DATE (data), entre outros.

É recomendado utilizar o tipo de dado mais apropriado para evitar desperdício de espaço e garantir a integridade dos dados.

Além disso, é essencial conhecer as limitações do MySQL, como o tamanho máximo de uma tabela, a quantidade máxima de índices por tabela, a quantidade máxima de colunas em uma tabela, entre outros.

Essas limitações devem ser consideradas durante o planejamento do banco de dados para garantir a escalabilidade e o bom funcionamento do sistema.

Com a criação das tabelas, definição de chaves e índices, e a utilização adequada dos tipos de dados, teremos um banco de dados bem estruturado e otimizado para suportar as operações do nosso sistema de gerenciamento de biblioteca.

7. **Garantia da Integridade dos Dados:**
    - Discussão sobre a importância de garantir a integridade dos dados no banco de dados.
    - Uso de restrições (constraints) para evitar inserção de dados inválidos.
    - Como trabalhar com transações para manter a consistência dos dados.

Neste tópico, abordaremos a importância de garantir a integridade dos dados no banco de dados, o uso de restrições (constraints) para evitar inserção de dados inválidos e como trabalhar com transações para manter a consistência dos dados.

Importância da Integridade dos Dados:

A integridade dos dados é fundamental para a confiabilidade e precisão das informações armazenadas no banco de dados.

Garantir que os dados sejam válidos, consistentes e sem duplicações é essencial para que o sistema funcione corretamente e forneça resultados confiáveis aos usuários.

Uso de Restrições (Constraints) para Evitar Inserção de Dados Inválidos:

As restrições, também conhecidas como constraints, são regras definidas no banco de dados que impõem limitações sobre os dados que podem ser inseridos, atualizados ou excluídos.

Existem diferentes tipos de restrições:

1. **Chave Primária (Primary Key):** Garante que cada registro em uma tabela seja único e tenha um valor de identificação exclusivo.

```
CREATE TABLE alunos (
    id INT PRIMARY KEY,
    nome VARCHAR(50),
    idade INT
);
```

2. **Chave Estrangeira (Foreign Key):** Define uma relação entre duas tabelas, garantindo que os valores em uma coluna correspondam aos valores em outra coluna de outra tabela.

```
CREATE TABLE cursos (
    id INT PRIMARY KEY,
    nome VARCHAR(50),
    professor_id INT,
    FOREIGN KEY (professor_id) REFERENCES professores(id)
);
```

3. **Restrição de Not Null:** Impede que um campo receba valores nulos, ou seja, que não tenham valor.

```
CREATE TABLE funcionarios (
    id INT PRIMARY KEY,
    nome VARCHAR(50) NOT NULL,
    cargo VARCHAR(50) NOT NULL
);
```

4. **Restrição de Unique:** Garante que os valores em uma coluna sejam únicos, mas permite que haja registros com valores nulos.

```
CREATE TABLE clientes (
    id INT PRIMARY KEY,
    cpf VARCHAR(11) UNIQUE,
    email VARCHAR(100) UNIQUE
);
```

5. **Restrição de Check:** Permite que você especifique uma condição para os valores que podem ser inseridos em uma coluna.

```
CREATE TABLE pedidos (
    id INT PRIMARY KEY,
    valor_total DECIMAL(10, 2),
    status VARCHAR(20) CHECK (status IN ('aberto', 'fechado',
);
```

Como Trabalhar com Transações para Manter a Consistência dos Dados:

Transações são conjuntos de operações que são tratadas como uma unidade única de trabalho. O uso de transações é essencial para garantir a consistência dos dados em operações que envolvem múltiplas atualizações no banco de dados.

Uma transação deve ser concluída com sucesso (commit) ou desfeita (rollback) caso ocorra algum erro, mantendo assim a integridade dos dados.

Exemplo de uso de transações:

```
START TRANSACTION;

-- Realize as operações de inserção, atualização ou exclusão n

-- Se todas as operações ocorrerem corretamente, efetue o comm
COMMIT;

-- Se ocorrer algum erro ou problema, desfaça as operações rea
ROLLBACK;
```

Trabalhar com transações é especialmente importante em situações onde é necessário manter a consistência dos dados, como em sistemas financeiros ou sistemas de reserva de passagens, por exemplo.

8. **Exemplos Práticos:**
    - Criar um cenário fictício e desenvolver o planejamento e a criação do banco de dados correspondente.
    - Apresentar desafios comuns encontrados durante o planejamento e modelagem do banco de dados.

Cenário Fictício: Sistema de Gerenciamento de Escola

Vamos criar um cenário fictício de um sistema de gerenciamento de escola, que inclui informações sobre alunos, cursos e notas.

Planejamento e Criação do Banco de Dados:

1. Identificação dos Requisitos e Regras de Negócio:
    - O sistema deve armazenar informações dos alunos, como nome, data de nascimento e endereço.
    - Deve ser possível cadastrar cursos, com nome, duração e descrição.
    - Os alunos podem se matricular em cursos, e cada matrícula está associada a um aluno e a um curso específico.
    - Deve ser possível registrar as notas dos alunos em cada curso que estão matriculados.

2. Definição de Entidades, Atributos e Relacionamentos:
    - Entidades: Aluno, Curso, Matrícula

Atributos:

- Aluno: id, nome, data_nascimento, endereco
- Curso: id, nome, duracao, descricao
- Matrícula: id, aluno_id (chave estrangeira para Aluno), curso_id (chave estrangeira para Curso), data_matricula

Relacionamentos:

- Aluno tem uma ou várias Matrículas (relacionamento 1 para muitos)
- Curso tem várias Matrículas (relacionamento 1 para muitos)

3. Definição de Chaves:
    - Chave Primária: id (para Aluno, Curso, Matrícula)
    - Chave Estrangeira: aluno_id (para Matrícula, referenciando Aluno), curso_id (para Matrícula, referenciando Curso)

4. Criação das Tabelas e Índices:

```
CREATE TABLE Aluno (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    data_nascimento DATE,
    endereco VARCHAR(200)
);
```

```sql
CREATE TABLE Curso (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    duracao INT,
    descricao TEXT
);
```

```sql
CREATE TABLE Matricula (
    id INT AUTO_INCREMENT PRIMARY KEY,
    aluno_id INT,
    curso_id INT,
    data_matricula DATE,
    FOREIGN KEY (aluno_id) REFERENCES Aluno(id),
    FOREIGN KEY (curso_id) REFERENCES Curso(id)
);
```

```sql
CREATE INDEX idx_aluno_id ON Matricula (aluno_id);
CREATE INDEX idx_curso_id ON Matricula (curso_id);
```

Desafios Comuns no Planejamento e Modelagem do Banco de Dados:

- **Identificação e Definição dos Requisitos:** Garantir que todos os requisitos do sistema sejam compreendidos e traduzidos corretamente para a modelagem do banco de dados.

- **Normalização:** A normalização pode ser desafiadora, especialmente ao lidar com projetos complexos, e requer um bom entendimento das formas normais.

- **Tratamento de Relacionamentos:** Definir corretamente os relacionamentos entre as entidades é fundamental para a integridade dos dados. Lidar com relacionamentos muitos para muitos pode ser um desafio específico.

- **Otimização de Consultas:** À medida que o banco de dados cresce, a otimização de consultas para garantir o desempenho adequado pode ser um desafio.

9. **Trabalho Prático:**
    - Propor um exercício em que os alunos devem planejar o banco de dados para um sistema específico.
    - Incentivar a aplicação dos conceitos aprendidos na criação do esquema do banco de dados.

Exercício: Sistema de Gerenciamento de Biblioteca

Desenvolva o planejamento e a criação do banco de dados para um sistema de gerenciamento de biblioteca.

O sistema deve armazenar informações sobre livros, autores e usuários. Cada livro possui um título, um autor e uma data de publicação.

Cada autor possui um nome e um país de origem. Cada usuário possui um nome, um e-mail e uma lista de livros emprestados.

Apliquem os conceitos aprendidos anteriormente, como a definição de entidades, atributos, relacionamentos, chaves e índices e restrições (constraints) para garantir a integridade dos dados.

Ao final do exercício, os alunos terão criado um esquema de banco de dados sólido e bem-estruturado para o sistema de gerenciamento de biblioteca.

Esse tipo de atividade prática ajuda os alunos a consolidarem os conhecimentos adquiridos e a aplicarem as habilidades na criação de sistemas reais.