Sistema de Gerenciamento de Biblioteca: Modelagem de Dados Relacional para Lumen Archive

Sumário

1. Introdução e Contexto de Negócio

O projeto de modelagem de dados apresentado neste documento foi elaborado objetivando atender às necessidades da biblioteca "Lumen Archive", uma biblioteca digital com foco em acervo acadêmico. A Lumen Archive opera exclusivamente online, oferecendo serviços de consulta e empréstimo de livros para estudantes e pesquisadores em universidades federais.

A rápida expansão do acervo e da base de usuários tornou inviável a gestão por meio de métodos manuais ou planilhas simples. A falta de um sistema estruturado gera diversos problemas, como dificuldade em rastrear o status dos empréstimos, inconsistência nos registros dos livros e falta de capacidade para gerar relatórios estratégicos sobre a popularidade de autores, gêneros e o uso geral da plataforma.

1.1. Necessidade de Negócio e o Problema a Ser Resolvido

O principal desafio de negócio da Lumen Archive é a escalabilidade e a integridade dos dados. Atualmente, o gerenciamento de dados enfrenta as seguintes dificuldades:

- **Rastreamento de Empréstimos (Loans):** Não há rastreamento em tempo real do status dos empréstimos, o que leva a erros no cálculo das datas de devolução (loan_due_date) e dificulta a identificação de livros em atraso (Overdue), prejudicando a disponibilidade do acervo.
- **Performance:** A ausência de um sistema estruturado dificulta a realização de buscas eficientes por título, autor ou editora, tornando o processo lento, manual e sujeito a erros, especialmente com o crescimento acelerado do acervo.
- **Auditoria:** Sem um sistema formal, não há controle padronizado sobre quando cada registro foi criado ou atualizado, prejudicando a rastreabilidade, governança de dados e conformidade institucional.
- **Consistência dos Metadados:** A falta de um modelo de dados definido gera inconsistências nos metadados (como variações de nome de autores, nacionalidades ou editoras), criando duplicidades, retrabalho e consultas imprecisas.

1.2. Justificativa e Objetivo do Projeto de Modelagem

Este projeto visa resolver as dores de negócio da Lumen Archive por meio da criação de um Modelo de Dados Lógico e Normalizado. O objetivo principal é desenvolver um schema relacional robusto que garanta a atomicidade, consistência, isolamento e durabilidade (ACID) das transações e suporte futuras expansões da plataforma.

Objetivo	Descrição	Tabela(s) Envolvida(s)
Integridade Referencial	Garantir que nenhum empréstimo seja registrado para um usuário ou livro inexistente, utilizando Chaves Estrangeiras (FKs) que referenciam users(usr_id) e book(book_isbn)	book, users, loan
Eliminação de Redundância	Aplicação da Terceira Forma Normal (3FN) ao isolar informações repetitivas (como Editoras e Autores) em tabelas próprias, evitando duplicação de nomes, inconsistências de metadados e anomalias de atualização.	publisher, author
Resolução N:M	Modelar corretamente a relação de múltiplos autores para múltiplos livros, essencial para a precisão do acervo acadêmico.	book_author
Rastreabilidade e Auditoria	Incorporar campos de auditoria (created_at, updated_at) em todas as entidades para rastrear o ciclo de vida de cada registro.	Todas as tabelas

Ao final deste projeto, a Lumen Archive terá um blueprint de banco de dados estruturado, escalável e de alto desempenho, pronto para ser implementado em um Sistema Gerenciador de Banco de Dados (SGBD) relacional de nível empresarial. Este modelo não só resolve os problemas operacionais atuais, mas também serve como base sólida para análises de dados futuras e tomada de decisão estratégica.

2. Descrição Detalhada do Modelo Lógico

O modelo lógico apresentado foi projetado em conformidade com a Terceira Forma Normal (3FN), garantindo ausência de redundâncias, dependências parciais ou transitivas. Este modelo é composto por 5 tabelas principais e 1 tabela de junção, interligadas por Chaves Primárias (PK) e Chaves Estrangeiras (FK), conforme detalhado a seguir.

2.1 Relacionamentos de Cardinalidade

2.1.1 Relacionamento BOOK e PUBLISHER (1:N)

- **Regra:** Uma Editora (publisher) pode publicar muitos Livros (book), mas um Livro é publicado por apenas uma Editora.
- **Implementação:** A tabela book contém a Chave Estrangeira book_publisher_id, que referencia a Chave Primária pub_id da tabela publisher.

2.1.2 Relacionamento USER e LOAN (1:N)

- **Regra:** Um Usuário (users) pode ter muitos Empréstimos (loan), mas cada Empréstimo pertence a um único Usuário.
- **Implementação:** A tabela loan contém a Chave Estrangeira usr_id, que referencia a Chave Primária usr_id da tabela users.

2.1.3 Relacionamento BOOK e LOAN (1:N)

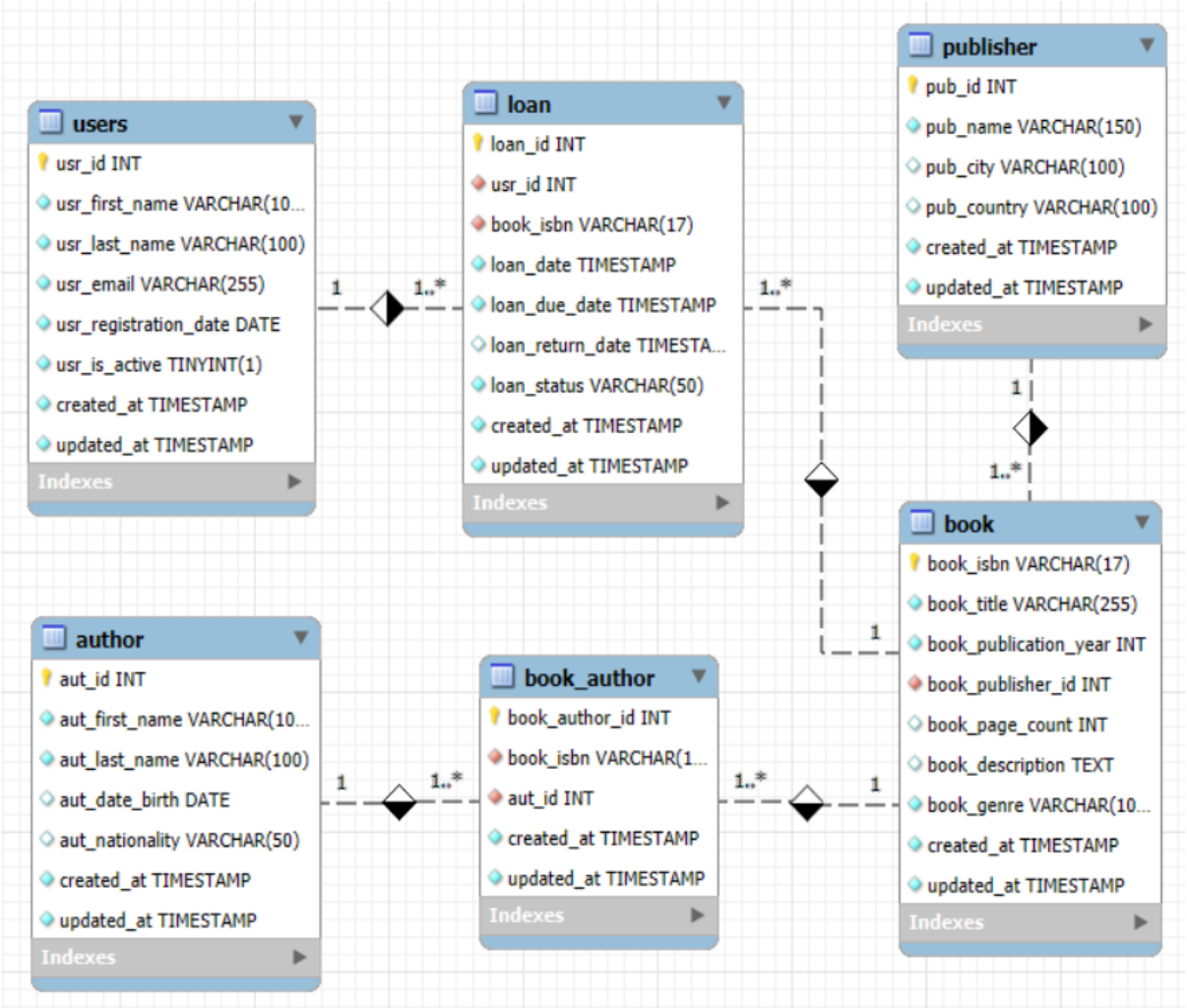
- **Regra:** Um Livro (book) pode ser emprestado muitas vezes (gerando vários registros em loan), mas cada registro de Empréstimo está ligado a um único Livro.
- **Implementação:** A tabela loan contém a Chave Estrangeira book_isbn, que referencia a Chave Primária book_isbn da tabela book.

2.1.4 Resolução do Relacionamento AUTHOR e BOOK (N:M)

- **Regra:** Um Livro pode ter múltiplos Autores, e um Autor pode escrever múltiplos Livros.
- **Resolução** (Tabela de Junção): Este relacionamento Muitos-para-Muitos é resolvido pela tabela book_author.
    - Implementação: A tabela book_author contém duas Chaves Estrangeiras: aut_id (referenciando author) e book_isbn (referenciando book).
    - Unicidade: A combinação destas duas FKs é única (UNIQUE (book_isbn, aut_id)), garantindo que um mesmo autor não seja listado duas vezes para o mesmo livro.
- **Justificativa**:

  Para respeitar a 1FN e evitar múltiplos autores numa mesma coluna, o que causaria redundância, inconsistência, dificuldade de consulta e comprometimento da integridade referencial, foi criada a tabela de junção book_author. Esta tabela mantém os dados atômicos e garante a integridade.

2.2 Diagrama de Entidade e Relacionamento (DER)



3. Definição das Estruturas de Banco de Dados (DDL)

Nesta parte, estão descritos os scripts de criação das tabelas que compõem o modelo de dados do sistema. Os scripts DDL estabelecem a criação das entidades, suas respectivas chaves primárias e estrangeiras, bem como os atributos essenciais que compõem cada tabela.

3.1 Criação das Tabelas do Projeto

```sql
CREATE TABLE author (
    aut_id INT PRIMARY KEY AUTO_INCREMENT,
    aut_first_name VARCHAR(100) NOT NULL,
    aut_last_name VARCHAR(100) NOT NULL,
    aut_date_birth DATE,
    aut_nationality VARCHAR(50),

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP NOT NULL
);

CREATE TABLE publisher (
    pub_id INT PRIMARY KEY AUTO_INCREMENT,
    pub_name VARCHAR(150) NOT NULL UNIQUE,
    pub_city VARCHAR(100),
    pub_country VARCHAR(100),

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP NOT NULL
);

CREATE TABLE users (
    usr_id INT PRIMARY KEY AUTO_INCREMENT,
    usr_first_name VARCHAR(100) NOT NULL,
    usr_last_name VARCHAR(100) NOT NULL,
    usr_email VARCHAR(255) NOT NULL UNIQUE,
    usr_registration_date DATE NOT NULL,
    usr_is_active BOOLEAN NOT NULL DEFAULT TRUE,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP NOT NULL
);
```

```sql
CREATE TABLE book (
    book_isbn VARCHAR(17) PRIMARY KEY,
    book_title VARCHAR(255) NOT NULL,
    book_publication_year INT NOT NULL,
    book_publisher_id INT NOT NULL,
    book_page_count INT,
    book_description TEXT,
    book_genre VARCHAR(100) NOT NULL,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP NOT NULL,

    FOREIGN KEY (book_publisher_id) REFERENCES publisher(pub_id)
);

CREATE TABLE book_author (
    book_author_id INT PRIMARY KEY AUTO_INCREMENT,
    book_isbn VARCHAR(17) NOT NULL,
    aut_id INT NOT NULL,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP NOT NULL,

    FOREIGN KEY (book_isbn) REFERENCES book(book_isbn),
    FOREIGN KEY (aut_id) REFERENCES author(aut_id),
    UNIQUE (book_isbn, aut_id)
);

CREATE TABLE loan (
    loan_id INT PRIMARY KEY AUTO_INCREMENT,
    usr_id INT NOT NULL,
    book_isbn VARCHAR(17) NOT NULL,
    loan_date TIMESTAMP NOT NULL,
    loan_due_date TIMESTAMP NOT NULL,
    loan_return_date TIMESTAMP,
    loan_status VARCHAR(50) NOT NULL DEFAULT 'Ongoing',

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP NOT NULL,

    FOREIGN KEY (usr_id) REFERENCES users(usr_id),
    FOREIGN KEY (book_isbn) REFERENCES book(book_isbn)
);
```

4. Consultas para análise de dados (Bônus)

A seção a seguir apresenta um conjunto de queries SQL criadas com o objetivo de levantar informações relevantes sobre a Lumen Archive. Essas consultas permitem analisar dados relacionados a empréstimos, usuários, autores e editoras, demonstrando como o modelo relacional responde de forma eficaz a cenários reais de gestão do acervo digital

4.1 Consultar Livros Atualmente Atrasados (Overdue)

Objetivo: Identificar rapidamente os livros que deveriam ter sido devolvidos, mas que ainda não tiveram o campo loan_return_date preenchido, e rastrear o usuário responsável. Esta consulta é crucial para a gestão da disponibilidade do acervo.

```sql
SELECT
    l.loan_id,
    b.book_title AS book_title,
    CONCAT(u.usr_first_name, ' ', u.usr_last_name) AS user_name,
    u.usr_email AS user_email,
    l.loan_due_date AS due_date,
    DATEDIFF(CURRENT_TIMESTAMP(), l.loan_due_date) AS days_overdue
FROM loan l
JOIN book b ON l.book_isbn = b.book_isbn
JOIN users u ON l.usr_id = u.usr_id
WHERE
    l.loan_status = 'Ongoing' AND
    l.loan_due_date < CURRENT_TIMESTAMP();
```

4.2 Determinar Autores Mais Populares e Suas Editoras

Objetivo: Gerar um ranking dos autores mais emprestados para guiar decisões de aquisição de novo acervo e renovação de licenças.

```sql
SELECT
    CONCAT(a.aut_first_name, ' ', a.aut_last_name) AS author_name,
    p.pub_name AS main_publisher,
    COUNT(l.loan_id) AS total_loans
FROM author a
JOIN book_author ba ON a.aut_id = ba.aut_id
JOIN book b ON ba.book_isbn = b.book_isbn
JOIN loan l ON b.book_isbn = l.book_isbn
JOIN publisher p ON b.book_publisher_id = p.pub_id
GROUP BY
    author_name, p.pub_name
ORDER BY
    total_loans DESC
LIMIT 10;
```

4.3 Listar Livros com Múltiplos Autores (Essencial para Acervo Acadêmico)

Objetivo: Isolar e listar todos os livros que possuem colaboração, validando o uso da tabela de junção book_author e auxiliando na categorização de pesquisa.

```sql
SELECT
    B.book_title AS book_title,
    COUNT(BA.aut_id) AS author_count,
    GROUP_CONCAT(A.aut_last_name ORDER BY A.aut_last_name SEPARATOR ', ') AS last_names_list

FROM book_author BA
JOIN book B ON BA.book_isbn = B.book_isbn
JOIN author A ON BA.aut_id = A.aut_id
GROUP BY
    B.book_title
HAVING
    author_count > 1
ORDER BY
    author_count DESC;
```

4.4 Consultar o Histórico de Usuários Inativos

Objetivo: Identificar usuários que estão inativos (usr_is_active = FALSE) para fins de governança de dados ou reengajamento, juntamente com o último livro que eles emprestaram.

```sql
SELECT
    CONCAT(u.usr_first_name, ' ', u.usr_last_name) AS user_name,
    u.usr_email AS email,
    l.last_loan_date,
    b.book_title AS last_book_title
FROM users u
LEFT JOIN (
    SELECT
        usr_id,
        MAX(loan_date) AS last_loan_date
    FROM loan
    GROUP BY usr_id
) l ON u.usr_id = l.usr_id
LEFT JOIN loan lo ON lo.usr_id = u.usr_id AND lo.loan_date = l.last_loan_date
LEFT JOIN book b ON b.book_isbn = lo.book_isbn
WHERE u.usr_is_active = FALSE
ORDER BY l.last_loan_date DESC;
```

4.5 Identificar Editoras com Maior Tempo Médio de Retenção de Livros

Objetivo: Calcular o tempo médio (em dias) que os livros de cada editora permanecem emprestados antes de serem devolvidos. Isso ajuda a identificar editoras cujos livros são mais populares, têm maior retenção ou demandam mais atenção na gestão do acervo.

```sql
SELECT
    P.pub_name AS publisher_name,
    COUNT(L.loan_id) AS total_loans,
    AVG(DATEDIFF(L.loan_return_date, L.loan_date)) AS avg_loan_days,
    MAX(DATEDIFF(L.loan_return_date, L.loan_date)) AS max_loan_days
FROM loan L
JOIN book B ON L.book_isbn = B.book_isbn
JOIN publisher P ON B.book_publisher_id = P.pub_id
WHERE
    L.loan_status = 'Returned'
    AND L.loan_return_date IS NOT NULL
GROUP BY P.pub_name
HAVING
    total_loans > 3  -- Only considers publishers with at least 3 loans
ORDER BY
    avg_loan_days DESC;
```

5 Preparação do Ambiente Virtual Python para Execução do Seed (Bônus)

Nesta etapa, é descrito o processo de configuração do ambiente virtual necessário para a execução do script de inserção de dados no SGBD. As instruções partem do pressuposto de que o Python já está instalado e funcionando corretamente em seu computador.

5.1 Criando o repositório do projeto para execução de scripts python

No PowerShell ou CMD:

```
mkdir PYTHON_ENVIRONMENT
```

Entrar no diretório:

```
cd .\PYTHON_ENVIRONMENT\
```

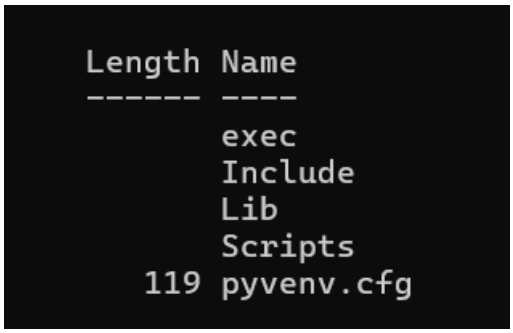
Essa diretório vai armazenar o ambiente e seus scripts.

5.2 Criar o ambiente virtual

Dentro do diretório PYTHON_ENVIRONMENT, execute:

```
python -m venv modelagem_dados_env
```

Isso cria a estrutura:



Obs: o diretório exec, foi criado manualmente para a inserção do script python para inserção de dados fictícios no banco de dados criado nas etapas anteriores do projeto.

5.3 Ativar o ambiente virtual

```
.\modelagem_dados_env\Scripts\activate
```

Se ativou corretamente, seu terminal vai mudar para:

```
(modelagem_dados_env) PS C:\Users\lucas\Downloads\PYTHON_ENVIRONMENT\modelagem_dados_env>
```

A partir daqui tudo **que você executar no prompt (pip install, python, etc.) será dentro desse ambiente**.

5.4 Instalar bibliotecas usando pip

Com o ambiente ativado:

pip install <nome_da_biblioteca>

Exemplo:

pip install sqlalchemy

5.5 Verificar bibliotecas instaladas

Listar todas:

pip list

```
(modelagem_dados_env) PS
>> pip list
Package            Version
------------------ -------
cffi               2.0.0
cryptography       46.0.3
Faker              38.2.0
greenlet           3.3.0
pip                25.3
pycparser          2.23
PyMySQL            1.1.2
setuptools         63.2.0
SQLAlchemy         2.0.45
typing_extensions 4.15.0
tzdata             2025.2
```

Ver detalhes de uma lib:

Pip show <nome_lib>

```
>> pip show sqlalchemy
Name: SQLAlchemy
Version: 2.0.45
Summary: Database Abstraction Library
Home-page: https://www.sqlalchemy.org
Author: Mike Bayer
Author-email: mike_mp@zzzcomputing.com
License: MIT
Location: c:\users\lucas\downloads\python_environment\modelagem_dados_env\lib\site-packages
Requires: greenlet, typing-extensions
Required-by:
```

5.6 Rodar qualquer script Python dentro do ambiente:

.\<nome_do_script.py>

Exemplo:

```
python .\seed_library.py
```

5.7 Desativar o ambiente virtual

Quando terminar:

Deactivate

Após esse procedimento, o nome do ambiente virtual (exibido anteriormente em verde) deixa de aparecer no terminal, restando apenas o caminho do diretório atual. Isso confirma que o ambiente foi desativado com sucesso.

```
(modelagem_dados_env) PS C:\Users\lucas\Downloads\PYTHON_ENVIRONMENT\modelagem_dados_env> deactivate
PS C:\Users\lucas\Downloads\PYTHON_ENVIRONMENT\modelagem_dados_env>
```

6 Bibliotecas Utilizadas na Execução do Seed do Projeto

cryptography 46.0.3

SQLAlchemy 2.0.45

PyMySQL 1.1.2