Implementing a database for a bookstore using MySQL Workbench, Python and Docker

Johnatas P. R. da Silva, Pedro H. S. dos Santos

Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Postal Code 69067-005 – Manaus – AM – Brazil

{pedro.henrique,johnatas.silva}@icomp.ufam.edu.br

Abstract. *This paper covers the implementation of a database aimed at a bookstore using DBMS MySQL Workbench, the Python language to populate. Tables and perform queries, and Docker to create containers. A VIEW and the use of indexes were implemented. The ERR diagram was generated using the DBMS reverse engineering tool.*

Resumo. *Este artigo aborda a implementação de um banco de dados voltado para uma livraria usando SGBD MySQL Workbench, a linguagem Python para popular as tabelas e realizar consultas e o Docker para a criação de contêiner. Foi implementado uma VIEW e o uso de índices. Foi gerado o diagrama EER por meio da ferramenta de engenharia reversa do SGBD.*

1. Introdução

Este artigo visa apresentar uma solução de banco de dados para uma livraria utilizando um banco de dados relacional. Um esquema conceitual foi montado e posteriormente o banco e as tabelas foram criados, bem como os relacionamentos foram definidos por meio de chaves estrangeiras a partir de um script em SQL. Então, o diagrama foi gerado por meio da ferramenta de engenharia reversa presente no MySQL Workbench. Após isso, criou-se um programa em Python para popular as tabelas criadas por meio da biblioteca Faker. Utilizou-se também a bibliotecas MySQL Conector/Python para conectar o código em Python com o banco de dados. O Docker também foi utilizado para criação de contêiner para encapsular o ambiente de trabalho.

2. Objetivos

O objetivo principal desse trabalho é desenvolver uma solução que envolve banco de dados relacional para uma livraria.

2.1 Objetivos Específicos

Para cumprir o objetivo geral, temos os seguintes objetivos específicos:

1. Implementar o esquema do bando de dados no SGBD MySQL Workbench, criando as tabelas, relacionamentos e atributos necessários para a solução do problema.

2. Criar um script em Python para gerar dados e popular as tabelas do banco de dados por meio da biblioteca Faker.

3. Criar consultas em SQL que abordem as operações de seleção, inserção e atualização de dados.

4. Identificar e definir índices em atributos que serão relevantes para a otimização das consultas.

5. Criar uma view com o objetivo de consultar os maiores estoques de livros disponíveis na livraria.

3. Ferramentas, Linguagens e SGBD

Primeiramente optou-se por seguir a sugestão dada no comando do trabalho para a utilização do Docker, que é uma plataforma de software utilizada, dentre outras coisas, para a implantação de contêineres para a criação de um ambiente leve, seguro e portátil para softwares.

O SGBD escolhido foi o MySQL Workbench por conta da familiaridade com o uso da ferramenta em detrimento do PostgreSQL que demandaria mais tempo de aprendizagem. Além disso, o MySQL é um sistema mais simples para o propósito do trabalho.

A linguagem escolhida foi o Python devido a familiaridade com o uso da linguagem. Além disso, utilizou-se as bibliotecas MySQL Connect/Python para realizar a conexão entre o script com o banco de dados.

A biblioteca Faker foi utilizada para gerar dados fictícios com o objetivo de povoar as tabelas criadas e para posterior consulta.

4. Esquemas e Consultas

Para a criação do esquema, foi elaborado um script em SQL no MySQL Workbench, sendo *bd_livraria* o nome definido para o banco de dados. Segue o script de criação do esquema:

```
CREATE DATABASE db_livraria;
```

```
USE db_livraria;

CREATE TABLE IF NOT EXISTS Autor (
        cod_autor SMALLINT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    data_nasc DATE NOT NULL,
    pais_nasc VARCHAR(100) NOT NULL,
    biografia VARCHAR(300) NOT NULL
);

CREATE TABLE IF NOT EXISTS Livro (
        cod_livro SMALLINT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    lingua VARCHAR(45) NOT NULL,
    ano YEAR NOT NULL
);

CREATE TABLE IF NOT EXISTS Autor_has_Livro (
        cod_autor SMALLINT NOT NULL,
    cod_livro SMALLINT NOT NULL,
    FOREIGN KEY (cod_autor) REFERENCES Autor(cod_autor),
    FOREIGN KEY (cod_livro) REFERENCES Livro(cod_livro)
);

CREATE TABLE IF NOT EXISTS Editora (
        cod_editora SMALLINT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(45) NOT NULL,
    endereco VARCHAR(255) NOT NULL,
    telefone VARCHAR(55) NOT NULL
```

```
);

CREATE TABLE IF NOT EXISTS Edicao (
        cod_isbn VARCHAR(55) PRIMARY KEY,
    preco DOUBLE NOT NULL,
    ano YEAR NOT NULL,
    num_pag INT NOT NULL,
    qtde_estoque INT NOT NULL,
    cod_livro SMALLINT NOT NULL,
    cod_editora SMALLINT NOT NULL,
    FOREIGN KEY (cod_livro) REFERENCES Livro(cod_livro),
    FOREIGN KEY (cod_editora) REFERENCES Editora(cod_editora)
);
```

Após isso, definiu-se os atributos que seriam indexados. As tabelas que tiveram seus atributos indexados foram Livro, Autor e Editora. Em todas as tabelas escolhidas, optou-se por indexar o atributo *nome* após análise de que é comum a pesquisa pelo nome de um objeto dentro do banco. Sendo assim, decidiu-se otimizar a busca por nome nessas tabelas.

```
CREATE INDEX index_nome_autor ON Autor(nome);
CREATE INDEX index_nome_livro ON Livro(nome);
CREATE INDEX index_nome_editora ON Editora(nome);
```

Foi criada uma view para consultar todas as informações relacionadas aos livros com maior quantidade de edições em estoque no atual momento da pesquisa.

```
CREATE VIEW view_livro_info AS
SELECT Editora.nome AS Nome_Editora, Edicao.cod_isbn AS ID_Edicao, Livro.nome AS Titulo_Livro, Edicao.qtde_estoque AS Quantidade_Estoque
FROM Livro
```

```
JOIN Edicao ON Livro.cod_livro = Edicao.cod_livro
JOIN Editora ON Edicao.cod_editora = Editora.cod_editora
WHERE Edicao.qtde_estoque = (
        SELECT MAX(qtde_estoque)
    FROM Edicao
);
```

Para realizar a consulta da view, utilizou-se a seguinte consulta:

```
SELECT * FROM view_livro_info;
```

O esquema criado gerou o seguinte diagrama por meio de engenharia reversa:

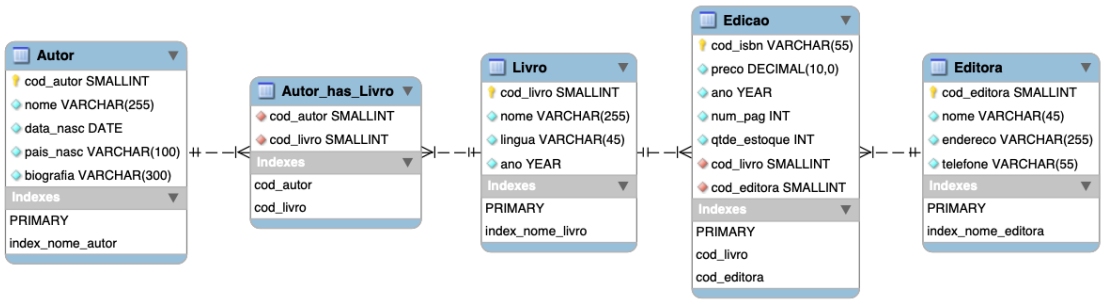

Figura 1. Diagrama EER do banco de dados gerado por meio do código em SQL.

Observe que há três tabelas principais *Autor*, *Livro* e *Editora*, e duas tabelas relacionamentos *Autor_has_Livro* e *Edicao*.

As seguintes consultas foram escritas em SQL e utilizadas no script Python:

1. (SELECT) Listar os nomes de todos os autores que têm edições de seus livros publicados com uma determinada editora (id da editora dado como entrada).

```
SELECT DISTINCT Autor.nome
FROM Autor
JOIN Autor_has_Livro ON Autor.cod_autor = Autor_has_Livro.cod_autor
JOIN Livro ON Autor_has_Livro.cod_livro = Livro.cod_livro
JOIN Edicao ON Livro.cod_livro = Edicao.cod_livro
JOIN Editora ON Edicao.cod_editora = Editora.cod_editora
WHERE Editora.cod_editora = '<cod_editora>';
```

Resultado da consulta:

Como resultado, obteve-se uma lista com o nome dos autores com livros publicados por uma determinada editora. O vídeo com a consulta está disponível no fim dessa seção.

2. (SELECT) Dada uma palavra "XXX" dada como entrada, listar as informações das edições (número da edição, editora, título do livro e seu primeiro autor) que tenha a palavra dada no título do livro da edição.

```
SELECT Edicao.cod_isbn AS Numero_Edicao, Editora.nome AS Nome_Editora, Livro.nome AS Titutlo_livro, Autor.nome AS Primeiro_Autor
FROM Edicao
JOIN Livro ON Edicao.cod_livro = Livro.cod_livro
JOIN Autor_has_Livro ON Livro.cod_livro = Autor_has_Livro.cod_livro
JOIN Autor ON Autor_has_Livro.cod_autor = Autor.cod_autor
JOIN Editora ON Edicao.cod_editora = Editora.cod_editora
WHERE Livro.nome LIKE '%<entrada>%'
ORDER BY Edicao.cod_isbn;
```

Resultado da consulta:

Como resultado, obteve-se uma lista com o nome dos livros baseado em uma string dada como entrada, como a string teve correspondência no

banco, as informações foram obtidas. O vídeo com a consulta está disponível no fim dessa seção.

3. (SELECT) Dada uma string "XXX" dada como entrada, listar as informações das edições (id das edições, editoras, títulos dos livros) onde a string fornecida esteja presente no nome de pelo menos um dos autores dos livros.

   ```
   SELECT DISTINCT Edicao.cod_isbn AS Id_Edicao, Editora.nome AS Nome_Editora, Livro.nome AS Titulo_Livro
   FROM Edicao
   JOIN Livro ON Edicao.cod_livro = Livro.cod_livro
   JOIN Autor_has_Livro ON Livro.cod_livro = Autor_has_Livro.cod_livro
   JOIN Autor ON Autor_has_Livro.cod_autor = Autor.cod_autor
   JOIN Editora ON Edicao.cod_editora = Editora.cod_editora
   WHERE Autor.nome LIKE '%<entrada>%';
   ```

   Resultado da pesquisa:

   Como resultado, obteve-se uma lista com as informações das edições baseado em uma string dada como entrada, como a string teve correspondência no nome de um dos autores, as informações foram obtidas. O vídeo com a consulta está disponível no fim dessa seção.

4. (UPDATE) Atualizar a quantidade de estoque de todas as edições de livros de uma editora dada como entrada - aumentando em 20%.

   ```
   UPDATE Edicao
   JOIN Livro ON Edicao.cod_livro = Livro.cod_livro
   JOIN Editora ON Edicao.cod_editora = Editora.cod_editora
   SET Edicao.qtde_estoque = Edicao.qtde_estoque * 1.2
   WHERE Editora.nome = '<Nome_da_Editora>';
   ```

Resultado da operação:

Após dar como entrada o nome de uma editora, o banco realizou uma operação para aumentar em 20% a quantidade de livros no estoque da livraria. Esse resultado pode ser conferido ao consultar o banco para verificar a efetividade da operação. O vídeo com a operação está disponível no fim dessa seção.

5. (INSERT) Inserir uma nova edição de um livro que já existe, considerando que essa edição continua associada à editora anterior.

INSERT INTO Edicao (cod_isbn, preco, ano, num_pag, qtde_estoque, cod_livro, cod_editora)

VALUES ('<Novo_ISBN>', Novo_Preco, Novo_Ano, Novo_Num_Pag, Nova_Qtde_Estoque, Codigo_Livro_Existente, Codigo_Editora_Anterior);

Resultado da operação:

Após dar como entrada um novo código ISBN, que é uma chave primária, foram inseridas as outras informações, as chaves primárias de um livro e de uma editora existente foram dadas como chaves estrangeiras. Assim, um novo registro foi adicionado ao banco. O vídeo com a operação está disponível no fim dessa seção.

Neste link há a pasta com o vídeo para a demonstração da criação do esquema, do povoamento das tabelas, das consultas e demonstração da view, e os scripts em SQL e Python.

5. Conclusão

Com base na execução do trabalho, notou-se a importância da clareza no texto base para a modelação do banco de dados, pois, como base no conceito do modelo, foi possível determinar os relacionamentos e a cardinalidade, influenciando diretamente na escrita do script em SQL para a geração do banco de dados. Sobre o SGBD, o MySQL Workbench é fácil de usar, possui uma gama útil de ferramentas e recursos que essenciais para a elaboração do diagrama EER. A escolha do Python também foi uma decisão acertada devido aos recursos da linguagem, como as bibliotecas de conexão com o banco de dados e a biblioteca de geração de dados fictícios, além disso a linguagem é fácil de entender e implementar. Em relação ao uso da View, ela facilita a visualização de tabelas sem ocupar um espaço a mais no armazenamento do computador. Ademais, entender como funciona a estrutura do banco e seus relacionamentos é importante para escrever consultas em SQL eficientes e legíveis.

6. Referências

Faraglia, Daniele (2024) "Faker 24.3.0 documentation", https://faker.readthedocs.io/en/master/#, março de 2024.

Docker Inc. (2024) "What is a Docker? Accelerate how you build, share, and run applications", https://www.docker.com, Março de 2024.

Oracle (2024) "MySQL Connector/Python Developer Guide", https://dev.mysql.com/doc/connector-python/en/, Março de 2024.

Oracle (2024) "MySQL Community Downloads", https://dev.mysql.com/downloads/, Março de 2024.

Elmasri R., and Navathe Shamkant B., (2011). "Sistemas de Banco de Dados", Pearson Education do Brasil.