Relatório Técnico: Sistema de Gerenciamento de Biblioteca Universitária

Autores:
Carlos Moreira;
Gustavo Alves;
Henrique Andrade;
Kendy Outi.

1. Modelagem do Banco de Dados

A modelagem do sistema foi dividida em duas etapas: o Diagrama Entidade-Relacionamento (DER) e o Diagrama do Modelo Relacional, que implementam as regras de herança de classes para entidades de Usuário e Material.

1.1 Diagrama Entidade-Relacionamento (DER)

O modelo conceitual estabelece as entidades principais e seus relacionamentos.

Entidades Principais:

- **Usuario:** Entidade genérica para Alunos, Professores e Funcionários.
- **Material:** Entidade genérica para Livros, Dissertações e Teses.
- **Autor:** Criador do material.
- **Exemplar:** Representa a cópia física de um Material.
- **Emprestimos** e **Reservas:** Registram as transações de uso.

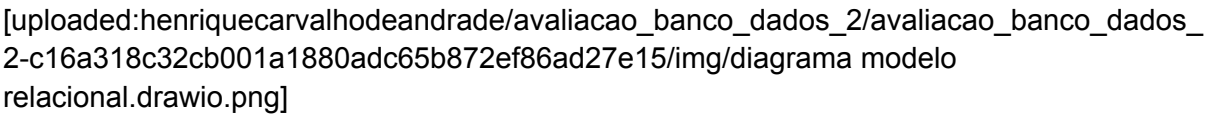
Diagrama Entidade-Relacionamento (DER):

[uploaded:henriquecarvalhodeandrade/avaliacao_banco_dados_2/avaliacao_banco_dados_2-c16a318c32cb001a1880adc65b872ef86ad27e15/img/diagrama entidade relacionamento.drawio.png]

1.2 Diagrama do Modelo Relacional

O modelo relacional utiliza a técnica de *Joined Table Inheritance* (Herança por Tabelas Unidas) para a superclasse `Usuario` (`usuarios`) e `Material` (`materiais`).

Diagrama do Modelo Relacional:



2. Justificativas das Escolhas de Cardinalidade e Integridade

As escolhas de cardinalidade e restrições de integridade visam garantir a consistência e a correta aplicação das regras de negócio do sistema.

2.1 Escolhas de Cardinalidade

- **1:N (Um para Muitos):**
  - `Material` (1) **->** `Exemplar` (N): Um material abstrato (título) pode ter muitas cópias físicas (exemplares).
  - `Usuario` (1) **->** `Emprestimo`/`Reserva` (N): Um utilizador pode realizar vários registros de empréstimos e reservas.
  - `Exemplar` (1) **->** `Emprestimo`/`Reserva` (N): Uma cópia específica (exemplar) pode estar historicamente associada a muitos registros de empréstimo e reservas.
- **N:M (Muitos para Muitos):**
  - `Material` (N) **<->** `Autor` (M): Um material pode ter múltiplos autores, e um autor pode escrever múltiplos materiais. Resolvido pela tabela associativa **`material_autor`**.
- **1:1 (Herança):**
  - `Usuario` (1) **<->** Subtipos (`Aluno`, `Professor`, `Funcionario`) (1): Relação 1:1 onde o `id` do subtipo é PK e FK para a superclasse, implementando a herança por tabelas unidas.

2.2 Escolhas de Integridade (Restrições)

- **Integridade de Entidade (PK):** Todas as tabelas possuem uma coluna `id` definida como `primary_key=True`, garantindo unicidade e não-nulidade.
- **Integridade Referencial (FK):** Chaves Estrangeiras, como `emprestimos.usuario_id`, garantem que os registros nos relacionamentos apontem apenas para entidades que existem nas tabelas referenciadas (ex: `usuarios.id`).
- **Integridade de Domínio (Lógicas):**
  - `matricula` Única em `usuarios`.
  - `data_devolucao` Anulável (`nullable=True`) em `emprestimos`, permitindo que um empréstimo exista enquanto o material está "em andamento" (não devolvido).
  - `multa` com valor padrão (`default=0`) em `emprestimos`.

3. Mapeamentos Objeto-Relacional (SQLAlchemy)

O sistema utiliza o SQLAlchemy ORM (Object-Relational Mapper) para mapear as classes Python para as tabelas do MySQL, com foco em herança (`Inheritance`) e relacionamento N:M (`Secondary` relationship).

3.1 Mapeamento de Herança (Joined Table Inheritance)

O mapeamento de herança é utilizado tanto para `Usuario` quanto para `Material`:

Classe Python	Tabela Mapeada	Base de Herança	Chave Estrangeira
Usuario	usuarios	Superclasse	N/A
Aluno	alunos	Subclasse	id (FK para usuarios.id)
Material	materiais	Superclasse	N/A
Livro	livros	Subclasse	id (FK para materiais.id)

Exemplo de Mapeamento da Classe `Usuario`:

```
Python
class Usuario(Base):
    __tablename__ = 'usuarios'
    id = Column(Integer, primary_key=True)
    tipo = Column(String(50))

    __mapper_args__ = {
        'polymorphic_identity': 'usuario',
        'polymorphic_on': tipo
    }
# ...
class Aluno(Usuario):
    __tablename__ = 'alunos'
    id = Column(Integer, ForeignKey('usuarios.id'), primary_key=True)
    curso = Column(String(100))

    __mapper_args__ = {
        'polymorphic_identity': 'aluno',
    }
```

Fonte: `classes_biblioteca.py`

3.2 Mapeamento de Relacionamento N:M

O relacionamento Muitos para Muitos entre `Material` e `Autor` é resolvido pela tabela associativa `material_autor`:

Python
```
class MaterialAutor(Base):
    __tablename__ = 'material_autor'
    autor_id = Column(Integer, ForeignKey('autores.id'), primary_key=True)
    material_id = Column(Integer, ForeignKey('materiais.id'), primary_key=True)

class Material(Base):
    # ...
    autores = relationship('Autor', secondary='material_autor', back_populates='obras')

class Autor(Base):
    # ...
    obras = relationship('Material', secondary='material_autor', back_populates='autores')
```

Fonte: `classes_biblioteca.py`

4. Exemplos de Queries (ORM e SQL)

O repositório fornece exemplos de como consultar o banco de dados usando tanto a linguagem SQL pura quanto a API do SQLAlchemy ORM.

4.1 Queries SQL

O arquivo `consultas_sql.sql` demonstra pesquisas de materiais, uso de `JOIN` para relacionamentos N:M e funções de agregação.

Objetivo	**Comando SQL**
Pesquisa por Filtros Múltiplos (título e ano)	SELECT T1.*, T2.* FROM materiais AS T1 JOIN livros AS T2 ON T1.id = T2.id WHERE T1.titulo = 'Machine Learning com Python' AND T1.ano_publicacao = 2023 LIMIT 1;
Pesquisa de Materiais por Autor (usando JOIN)	SELECT materiais.* FROM materiais JOIN material_autor ON materiais.id = material_autor.material_id JOIN autores ON autores.id = material_autor.autor_id WHERE autores.nome = 'Autor A';
Relatório Agregado (contagem total de empréstimos)	SELECT COUNT(id) FROM emprestimos;
Queries com LIKE	SELECT * FROM materiais WHERE titulo LIKE '%Dados%';

4.2 Queries ORM (SQLAlchemy)

O arquivo `consultas_orm.py` espelha os exemplos em SQL, mas utilizando a sintaxe orientada a objetos do SQLAlchemy.

Objetivo	**Comando SQLAlchemy ORM**
Pesquisa por Filtros Múltiplos (título e ano)	session.query(Livro).filter_by(titulo="Machine Learning com Python", ano_publicacao=2023).first()
Pesquisa de Materiais por Autor (acessando a lista obras)	autor_alvo.obras (onde autor_alvo é obtido por session.query(Autor).filter_by(nome="Autor A").first())
Relatório Agregado (contagem total)	session.query(func.count(Emprestimo.id)).scalar()
Queries com LIKE	session.query(Material).filter(Material.titulo.like('%Dados%')).all()