Documentação - FilmBox

Link no GitHub

- Ana Paula Natsumi Yuki
- Gabriel Henrique Pereira Carvalhaes
- Lucas Teixeira Reis
- Theo Diniz Viana

Documentação Técnica e Formulário:

1. **Como os índices são armazenados em disco?**

Existem dois tipos de indexação sendo usados: Hash Extensível e Árvore B Mais, os dois sendo baseados nos códigos fornecidos pelos professores Walisson e Marcos Kutova. Os índices são armazenados usando os diversos modelos de classe Par, que ditam como eles devem ser escritos e lidos em disco. A diferença é que na Árvore B Mais os pares são genéricos e no Hash são específicos.

2. **Como ocorre o acesso ao relacionamento 1 para N?**

Os relacionamentos 1 para N foram implementados usando uma classe chamada Review e um par chamado ParUsuárioLista. Devido a natureza das consultas que são baseadas em igualdade (as buscas por usuário são sempre feitas por ID e nunca em faixa) foi usado o Hash Extensível.

3. **Qual a estrutura usada para representar os registros?**

Em termos de orientação a objetos, os registros são representados por uma classe base que contém seus atributos e os métodos derivados da interface registro (getID, setID, toByteArray, fromByteArray). Os métodos toByteArray e fromByteArray são usados pela classe Arquivo (a classe Arquivo é responsável de fato pelo CRUD de bytes nos banco de dados). Os registros são armazenados em uma estrutura seguindo a lógica: LÁPIDE - TAM DO BLOCO DE DADOS - [ID - OUTROS CAMPOS].

4. **Como atributos multivalorados do tipo string foram tratados?**

Para os atributos multivalorados é escrito primeiro um int que indica o número de campos e em seguida para cada campo é escrito um int para indicar o tamanho do campo e logo em seguida os bytes daquele campo.

5. **Como foi implementada a exclusão lógica?**

A exclusão lógica foi implementada através de lápides. É marcado em um campo se o arquivo foi excluído ou não.

6. **Além das PKs, quais outras chaves foram utilizadas nesta etapa?**
   - userId: Para encontrar todas as Reviews e Listas de um usuário.
   - filmeId: Para encontrar todas as Reviews de um filme.
   - listaId: Para encontrar todos os Filmes contidos numa lista.
   - Filme.title: Para permitir a busca de filmes por nome.
   - Filme.score: Para permitir a busca de filmes por faixa de notas.
7. **Quais tipos de estruturas (hash, B + Tree, extensível, etc.) foram utilizadas para cada chave de pesquisa?**

HashExtensivel:

- Uso: É utilizado para buscas por igualdade exata (O(1)), sendo ideal para o armazenamento primário de entidades de acesso frequente por ID e para índices de relações simples.

- Chaves: User.id, Review.id, userId no índice idxUsuarioLista.

ArvoreBMais:

- Uso: É utilizada para o armazenamento primário de entidades (Filme, Lista) e, crucialmente, para todos os índices secundários que precisem de buscas por faixa ou que devam devolver resultados ordenados.
- Chaves: Filme.id, Lista.id, e para os índices secundários como Filme.title e Filme.score.

8. **Como foi implementado o relacionamento 1:N (explique a lógica da navegação entre registros e integridade referencial)?**

A navegação ocorre num processo de duas etapas: Consulta ao Índice Secundário: Primeiro, consulta-se o índice (ex: idxUsuarioReview) com a chave estrangeira (ex: userId) para obter uma lista de ponteiros (IDs da entidade alvo, ex: reviewIds).

- Busca dos Dados Principais: Em seguida, para cada ID obtido, faz-se uma busca rápida no armazenamento primário da entidade para recuperar o registo completo.
- Integridade Referencial: Não é imposta a nível da base de dados. É da responsabilidade da camada de Service garantir a consistência, por exemplo, ao deletar um User, o UserService deve orquestrar a remoção das suas Reviews e Listas.

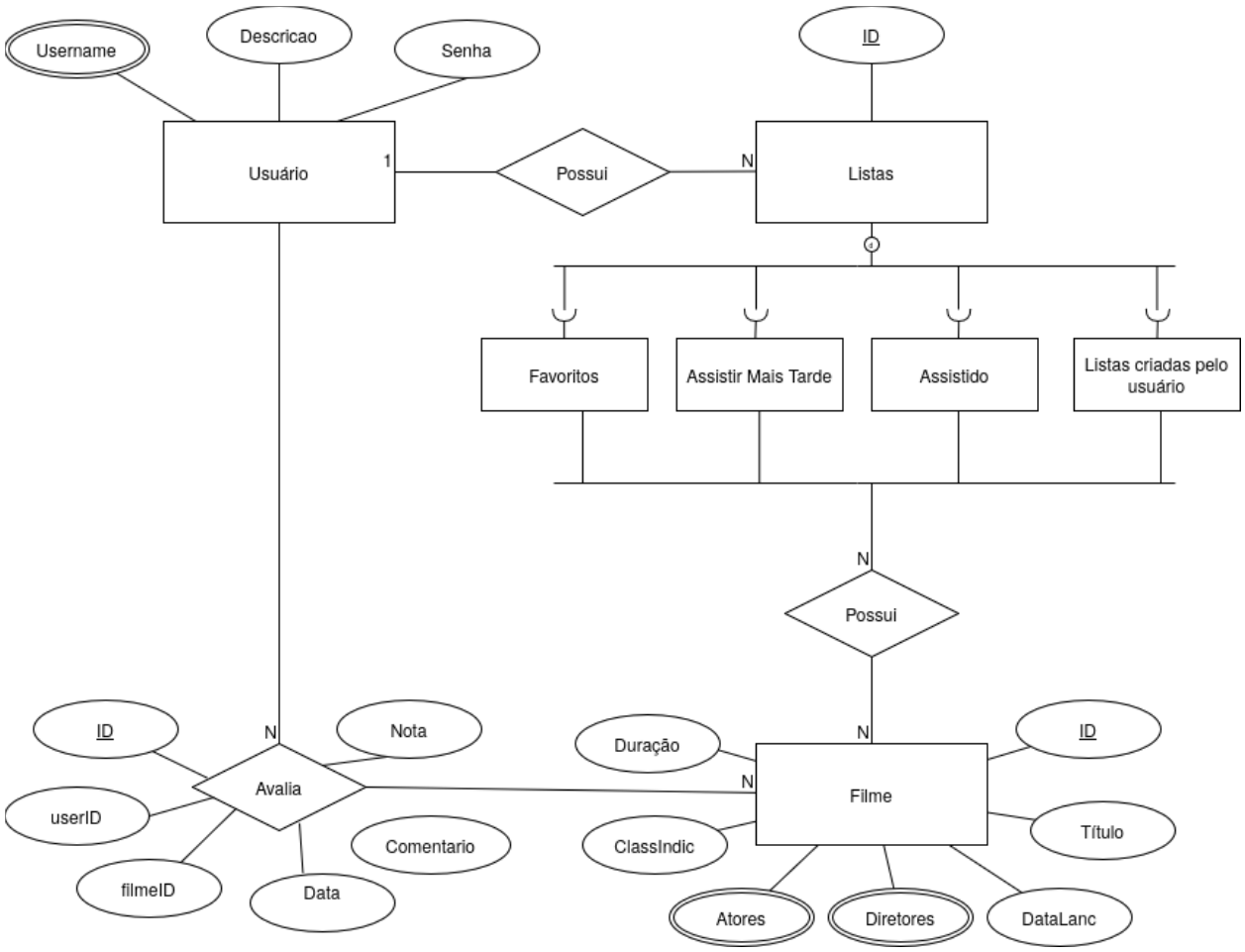
9. **Como os índices são persistidos em disco? (formato, atualização, sincronização com os dados).**

Cada índice é persistido em disco como um ou mais ficheiros binários (.db), geridos pelas classes HashExtensivel ou ArvoreBMais. A sincronização é imediata: quando um dado é criado ou alterado, a camada de Repository atualiza tanto o ficheiro de dados principal quanto todos os ficheiros de índice relevantes na mesma operação.

10. **Como está estruturado o projeto no GitHub (pastas, módulos, arquitetura)?**

O projeto no GitHub segue o padrão MVC + DAO (seguindo o padrão Spring Boot os DAO são chamados de repository) e usa o Spring Boot (a estrutura de pastas foi inicializada usando o SpringInitliazr).

Diagrama Entidade Relacionamento Atualizado


Username
Descricao
Senha
ID
Usuário
1
Possui
N
Listas
Favoritos
Assistir Mais Tarde
Assistido
Listas criadas pelo usuário
N
Possui
ID
N
Nota
Avalia
userID
filmeID
Data
Comentario
Duração
N
Filme
ID
Título
ClassIndic
Atores
Diretores
DataLanc