Pontifícia Universidade Católica de Minas Gerais

Instituto de Ciências Exatas e Informática

AED III - Trabalho Prático
Fase III do TP

Alexandre Versiani Raposo

https://github.com/LexVRaps/TP-AEDS3.git

Formulário

1. Qual foi o relacionamento N:N escolhido e quais tabelas ele conecta?

O relacionamento N:N conecta as entidades Rota e Horario. Uma rota pode ter múltiplos horários(exemplo: BH-Rio às 14:00, 16:00 e 20:00) e um mesmo horário pode ser usado em múltiplas rotas(exemplo: BH-Rio às 14:00 e São Paulo-Vitória às 14:00)

2. Qual estrutura de índice foi utilizada (B+ ou Hash Extensível)? Justifique a escolha.

Foi utilizada B+ porque ela é mais eficiente do que a hash quando se trata de intervalos, “range”. Ela permite buscar todos os horários de uma rota ou todas as rotas de um horário.

3. Como foi implementada a chave composta da tabela intermediária?

A chave composta que representa a associação entre Rota e Horario é implementada pela classe ParIntInt. Essa classe armazena dois inteiros, num1 e num2, que correspondem ao id da Rota e ao id do Horario.

```
public int compareTo(ParIntInt a) {
    if (this.num1 != a.num1)
      return this.num1 - a.num1;
    else
      return this.num2 == -1 ? 0 : this.num2 - a.num2;
  }
```

Esse método estabelece a ordem dos pares na Árvore B+, primeiro comparando o num1 e em caso de empate, comparando o num2. Isso efetivamente, cria uma chave primária composta com o id de ambos.

4. Como é feita a busca eficiente de registros por meio do índice?

Ela é feita através de hash extensível e Árvore B+, dependendo do índice.
Hash extensível: O(1) para busca de id
Árvore B+: O(log n) para relacionamentos
Ex:

```
public T read(int id) throws Exception {
    T obj = construtor.newInstance();
    int tam;
    //Busca no índice hash pelo ID
    ParIDEndereco pie = indiceDireto.read(id);
    long endereco = pie != null ? pie.getEndereco() : -1;
    if (endereco != -1) {
      // Acesso direto ao registro do arquivo
      arquivo.seek(endereco + 1);
      tam = arquivo.readShort();
      byte[] ba = new byte[tam];
      arquivo.read(ba);
      obj.fromByteArray(ba);
      return obj;
    } else
      return null;
  }
```

```
public ArrayList<Integer> buscarHorariosDaRota(int rotaId) throws Exception {
        ArrayList<Integer> horarioIds = new ArrayList<>();
        ArrayList<ParIntInt> pares = relRotaHorario.read(new ParIntInt(rotaId,
-1));

        for (ParIntInt par : pares) {
            horarioIds.add(par.get2());
        }

        return horarioIds;
    }
```

```
    public ArrayList<Integer> buscarRotasDoHorario(int horarioId) throws
Exception {
        ArrayList<Integer> rotaIds = new ArrayList<>();
        ArrayList<ParIntInt> pares = relHorarioRota.read(new
ParIntInt(horarioId, -1));

        for (ParIntInt par : pares) {
            rotaIds.add(par.get2());
        }

        return rotaIds;
    }
```

5. Como o sistema trata a integridade referencial (remoção/atualização) entre as tabelas?

Ex:

```
public boolean delete(int id) throws Exception {
        Rota obj = super.read(id);
        if (obj != null) {
            // Exclusão em cascata do relacionamento N:N
            ArrayList<ParIntInt> pares = relRotaHorario.read(new ParIntInt(id,
-1));
            for (ParIntInt par : pares) {
                relRotaHorario.delete(par);
                relHorarioRota.delete(new ParIntInt(par.get2(), par.get1()));
            }
            return super.delete(id);
        }
        return false;
    }
```

6. Como foi organizada a persistência dos dados dessa nova tabela (mesmo padrão de cabeçalho e lápide)?

As entidades principais, como rotas.db e horarios.db, usam cabeçalho com último ID + lápide. Porém, os relacionamentos N:N usam o formato próprio da Árvore B+, sem lápide e cabeçalho.

7. Descreva como o código da tabela intermediária se integra com o CRUD das

tabelas principais.

Ex: Create

```
public void incluir() {

//...

// Cria a rota primeiro

Rota rota = new Rota(origem, destino, distancia);

    try {

        int id = arqRotas.create(rota);

        System.out.println("Rota criada com ID: " + id);

       // Oferece associar horários depois

        System.out.print("Deseja associar horários a esta rota agora? (S/N):
");

        String resposta = console.nextLine();

        if (resposta.equalsIgnoreCase("S")) {

            associarHorarioExistente(id);

        }

}
```

```
private void associarHorarioExistente(int rotaId) {

       //...

       Lista Horários disponíveis

       ArrayList<Horario> horarios = arqHorarios.readAll();

       ...

       Usuário escolhe horário para associar

       arqRotas.associarHorario(rotaId, horarioId);

}
```

Update não influencia nos relacionamentos.

Ex: arquivo MenuRotas
1- Incluir (Influencia apenas no CRUD básico)

2- Buscar (Influencia apenas no CRUD básico)

3- Alterar (Influencia apenas no CRUD básico)

4- Excluir (Influencia apenas no CRUD básico)

5- Associar Horário (Influencia no Relacionamento)

6- Desassociar Horário (Influencia no Relacionamento)

7- Buscar Horários da Rota (Influencia no Relacionamento)

8. Descreva como está organizada a estrutura de diretórios e módulos no repositório após esta fase.

```
projeto/
│   ├── aeds3/
│   │   ├── Arquivo.java
│   │   ├── ArvoreBMais.java
│   │   ├── HashExtensivel.java
│   │   ├── ParIntInt.java
│   │   ├── ParIntIntHash.java
│   │   ├── ParIDEndereco.java
│   │   ├── Registro.java
│   │   ├── RegistroArvoreBMais.java
│   │   └── RegistroHashExtensivel.java
│   │
│   ├── arquivos/
│   │   ├── ArquivoRotas.java          # métodos N:N (Rota↔Horário)
│   │   ├── ArquivoHorarios.java
│   │   ├── ArquivoUsuarios.java
│   │   ├── ArquivoAssentos.java
│   │   └── ArquivoPassagens.java      # relacionamentos 1:N (Usuário→Passagem)
│   │
```

```
│   ├── entidades/
│   │   ├── Rota.java
│   │   ├── Horario.java
│   │   ├── Usuario.java
│   │   ├── Assento.java
│   │   ├── Passagem.java
│   │   └── UsuarioPassagem.java
│   │
│   └── menus/
│       ├── Principal.java
│       ├── MenuRotas.java
│       ├── MenuHorarios.java
│       ├── MenuUsuarios.java
│       ├── MenuAssentos.java
│       └── MenuPassagens.java
│
└── dados/
    ├── rotas.db
    ├── rotas.hash_d.db
    ├── rotas.hash_c.db
    ├── horarios.db
    ├── horarios.hash_d.db
    ├── horarios.hash_c.db
    ├── usuarios.db
    ├── assentos.db
    ├── passagens.db
    ├── rota_horario.btree.db          #  Árvore B+ Rota→Horário (N:N)
    ├── horario_rota.btree.db          #  Árvore B+ Horário→Rota (N:N)
    └── [outros .hash e .btree]        # Demais índices do sistema
```