Relatório de Projeto: Banco de Dados – Escola de Música

Nome do Projeto: Banco de Dados – Escola de Música

Alunos: Hugo, Wesley e Pedro

Disciplina: Banco de Dados

Professor: Danilo Farias

Data: Outubro de 2025

Introdução / Minimundo

A Escola de Música é uma instituição dedicada ao ensino e à prática musical, com foco em orquestras profissionais e amadoras. O cenário envolve diversos elementos interconectados: músicos talentosos que compõem as orquestras, instrumentos variados (como cordas, sopros e percussão) alocados para uso, funções específicas (ex: violinista, regente) e sinfonias clássicas ou contemporâneas que são ensaiadas e apresentadas. As orquestras podem ser de diferentes gêneros musicais, localizadas em várias cidades e países, e os músicos possuem dados pessoais como idade, nacionalidade e contatos.
O objetivo principal deste banco de dados é organizar e gerenciar esses dados de forma eficiente, permitindo consultas rápidas sobre alocação de músicos em orquestras, disponibilidade de instrumentos, histórico de sinfonias e relatórios analíticos (ex: idade média dos músicos ou contagem por função). Um banco de dados bem estruturado é essencial para a escola, pois facilita a coordenação de ensaios, evita duplicações de dados, garante integridade referencial entre entidades e suporta decisões administrativas, como contratações ou manutenções de instrumentos. Isso otimiza a operação diária e melhora a experiência dos envolvidos na comunidade musical.

Modelo Lógico (MER)

O Modelo Entidade-Relacionamento (MER) representa as entidades principais e seus relacionamentos de forma conceitual. As entidades incluem:

- Musicos: Representa os indivíduos que participam das orquestras, com atributos como nome, data de nascimento, nacionalidade, email e telefone.
- Orquestras: Entidades que agrupam músicos, com atributos como nome, data de criação, cidade e país.
- Instrumentos: Objetos alocados aos músicos, com atributos como nome, tipo (ex: Corda, Sopro) e estado (ex: Disponível, Usado).
- Funcoes: Papéis exercidos pelos músicos (ex: Violinista, Regente), com atributos como nome da função e descrição.
- Sinfonias: Obras musicais associadas às orquestras, com atributos como título, compositor e duração.

Relacionamentos principais:

- Musicos pertence a Orquestras (1:N – um músico em uma orquestra, uma orquestra com múltiplos músicos).
- Musicos exerce Funcoes (N:1 – múltiplos músicos em uma função, uma função para múltiplos músicos).
- Musicos usa Instrumentos (N:1 – múltiplos músicos usam um instrumento, mas tipicamente 1:1 para alocação).
- Orquestras apresenta Sinfonias (1:N – uma orquestra apresenta múltiplas sinfonias).

[Inserir imagem do MER aqui: Diagrama com entidades como retângulos, relacionamentos como losangos e atributos listados. Exemplo: Musicos --(pertence a)-- Orquestras.]
Este modelo garante a normalização para evitar redundâncias e suporta consultas complexas via joins.

Modelo Físico (MR)

O Modelo Relacional (MR) implementa o MER em tabelas relacionais, com chaves primárias (PK), chaves estrangeiras (FK) e tipos de dados apropriados. Decisões principais:

- Tipos de dados: VARCHAR para nomes e textos (escaláveis), DATE para datas, DECIMAL para durações (precisão), BOOLEAN para flags como 'ativo'.
- Relacionamentos: Implementados via FK (ex: id_orquestra em Musicos referencia Orquestras).

- Restrições: NOT NULL em campos essenciais (ex: nome), UNIQUE em emails, CHECK para idades positivas, DEFAULT para valores como ativo=TRUE. Índices em FK para performance em joins.
- Normalização: 3NF para eliminar dependências transitivas (ex: funções separadas de músicos).

Tabelas principais:

- musicos: id_musico (PK, INT AUTO_INCREMENT), nome (VARCHAR(60)), data_nascimento (DATE), nacionalidade (VARCHAR(50)), email (VARCHAR(100) UNIQUE), telefone (VARCHAR(30) NULL), id_orquestra (FK), id_funcao (FK), pais_origem (VARCHAR(50)), data_falecimento (DATE NULL), ativo (BOOLEAN DEFAULT TRUE).
- orquestras: id_orquestra (PK), nome (VARCHAR(100)), data_criacao (DATE), cidade (VARCHAR(100)), pais (VARCHAR(50)), genero_musical (VARCHAR(50)).
- instrumentos: id_instrumento (PK), nome_instrumento (VARCHAR(100)), tipo (VARCHAR(50)), estado (VARCHAR(50)), marca (VARCHAR(50)).
- funcoes: id_funcao (PK), funcao (VARCHAR(50) NOT NULL). *(Nota: Renomeado de nome_funcao para funcao)*
- sinfonias: id_sinfonia (PK), titulo (VARCHAR(200)), compositor (VARCHAR(100)), duracao_min (DECIMAL(5,2)).

[Inserir imagem do MR aqui: Diagrama de tabelas com colunas, PK/FK destacadas e setas para relacionamentos.]

Essas decisões priorizam integridade, escalabilidade e consultas eficientes.

Scripts SQL

Criação de Tabelas e Views (DDL)

Os scripts de criação de tabelas não foram fornecidos explicitamente, mas assumem-se baseados no MR. Aqui, foco em views criadas para relatórios simplificados. Cada view representa uma visão agregada ou joinada para facilitar consultas.

```
1USE escolamusica;
2
3-- View 1: Músicos e suas orquestras (facilita relatórios de alocação)
4CREATE VIEW view_musicos_orquestra AS
```

```sql
SELECT m.nome AS nome_musico, o.nome AS nome_orquestra, o.cidade
FROM musicos m
JOIN orquestras o ON m.id_orquestra = o.id_orquestra;

-- View 2: Músicos e suas funções (mostra papéis exercidos)
CREATE VIEW view_musicos_funcoes AS
SELECT m.nome AS nome_musico, f.funcao AS funcao  -- Nota: Usando 'funcao' após renomeação
FROM musicos m
JOIN funcoes f ON m.id_funcao = f.id_funcao;

-- View 3: Sinfonias e compositores (relatório de repertório)
CREATE VIEW view_sinfonias_compositores AS
SELECT s.titulo AS sinfonia, s.compositor  -- Nota: Assumindo 'titulo' como nome_sinfonia
FROM sinfonias s;

-- View 4: Instrumentos e estado (gerencia disponibilidade)
CREATE VIEW view_instrumentos_estado AS
SELECT nome_instrumento AS instrumento, estado
FROM instrumentos;

-- View 5: Orquestras por cidade (análise geográfica)
CREATE VIEW view_orquestras_cidade AS
SELECT nome AS orquestra, cidade
FROM orquestras;

-- View 6: Orquestras por país (visão internacional)
CREATE VIEW view_orquestras_pais AS
SELECT nome AS orquestra, pais
FROM orquestras;

-- View 7: Músicos e sinfonias (associação via orquestra)
CREATE VIEW view_musicos_sinfonias AS
SELECT m.nome AS musico, s.titulo AS sinfonia
FROM musicos m
JOIN sinfonias s ON m.id_orquestra = s.id_orquestra;  -- Assumindo FK em sinfonias

-- View 8: Instrumentos por tipo (classificação)
CREATE VIEW view_instrumentos_tipo AS
SELECT nome_instrumento AS instrumento, tipo
FROM instrumentos;

-- View 9: Contagem de músicos por função (estatística)
CREATE VIEW view_funcoes_contagem AS
SELECT f.funcao, COUNT(m.id_musico) AS total_musicos
FROM funcoes f
```

```
LEFT JOIN musicos m ON f.id_funcao = m.id_funcao
GROUP BY f.funcao;

-- View 10: Orquestras e média de idade dos músicos (análise demográfica)
CREATE VIEW view_orquestras_idade AS
SELECT o.nome AS orquestra, AVG(DATEDIFF(CURDATE(),
m.data_nascimento)/365.25) AS media_idade
FROM orquestras o
JOIN musicos m ON o.id_orquestra = m.id_orquestra

GROUP BY o.nome;
```

Essas views simplificam consultas complexas, melhorando a performance e a usabilidade para usuários não técnicos. Cada view é explicada acima: por exemplo, `view_musicos_orquestra` retorna nomes de músicos, orquestras e cidades para relatórios de alocação; `view_orquestras_idade` calcula médias de idade para análises demográficas.

Alterações (ALTER TABLE)

Essas alterações refinam o esquema inicial, adicionando campos, alterando tipos e permitindo flexibilidade.

```
USE escolamusica;

-- 1. Alterar tamanho da coluna 'nome' em musicos (aumenta capacidade para
nomes longos)
ALTER TABLE musicos
 MODIFY nome VARCHAR(60);

-- 2. Adicionar coluna 'genero_musical' na tabela orquestras (classifica
estilos)
ALTER TABLE orquestras
 ADD genero_musical VARCHAR(50);

-- 3. Alterar tipo da coluna 'telefone' em musicos para permitir null (nem
todos têm telefone)
ALTER TABLE musicos
 MODIFY telefone VARCHAR(30) NULL;

-- 4. Adicionar coluna 'pais_origem' na tabela musicos (rastreia diversidade)
ALTER TABLE musicos
 ADD pais_origem VARCHAR(50);

```

```sql
-- 5. Alterar tipo da coluna 'duracao_min' em sinfonias para DECIMAL (precisão em minutos)
ALTER TABLE sinfonias
MODIFY duracao_min DECIMAL(5,2);

-- 6. Adicionar coluna 'marca' na tabela instrumentos (detalhes de fabricação)
ALTER TABLE instrumentos
ADD marca VARCHAR(50);

-- 7. Renomear coluna 'nome_funcao' para 'funcao' na tabela funcoes (padronização)
ALTER TABLE funcoes
CHANGE nome_funcao funcao VARCHAR(50) NOT NULL;

-- 8. Adicionar coluna 'data_falecimento' em musicos (histórico para músicos falecidos)
ALTER TABLE musicos
ADD data_falecimento DATE;

-- 9. Adicionar coluna 'ativo' em musicos (indica status na orquestra)
ALTER TABLE musicos
ADD ativo BOOLEAN DEFAULT TRUE;

-- 10. Alterar coluna 'cidade' em orquestras para VARCHAR(100) (cidades longas)
ALTER TABLE orquestras

MODIFY cidade VARCHAR(100);
```

Inserts (DML)

[Nota: Scripts de INSERT não foram fornecidos no input. Em um relatório real, incluiria exemplos como: INSERT INTO musicos (nome, data_nascimento, ...) VALUES ('João Silva', '1980-05-15', ...); Comentário: Inserindo dados iniciais de músicos para popular o banco.]

Updates e Deletes (DML)

Esses comandos exemplificam manutenção de dados, corrigindo ou removendo registros obsoletos.

Updates:

```sql
-- 1. Atualizar o telefone do músico com id 1 (correção de contato)
```

```sql
UPDATE musicos
SET telefone = '81911122233'
WHERE id_musico = 1;

-- 2. Atualizar o email do músico com id 2 (atualização de email)
UPDATE musicos
SET email = 'fernanda.nova@email.com'
WHERE id_musico = 2;

-- 3. Atualizar o nome do músico com id 3 (correção ortográfica)
UPDATE musicos
SET nome = 'Thiago R. Rocha'
WHERE id_musico = 3;

-- 4. Atualizar a nacionalidade do músico com id 4 (mudança de dados)
UPDATE musicos
SET nacionalidade = 'Argentina'
WHERE id_musico = 4;

-- 5. Atualizar o telefone do músico com id 5 (novo número)
UPDATE musicos
SET telefone = '81999990000'
WHERE id_musico = 5;

-- 6. Atualizar o nome da orquestra com id 6 (rebranding)
UPDATE orquestras
SET nome = 'Orquestra Clássica do Recife'
WHERE id_orquestra = 6;

-- 7. Atualizar o país da orquestra com id 7 (mudança de sede)
UPDATE orquestras
SET pais = 'Portugal'
WHERE id_orquestra = 7;

-- 8. Atualizar o estado do instrumento com id 8 (após uso)
UPDATE instrumentos
SET estado = 'Usado'
WHERE id_instrumento = 8;

-- 9. Atualizar o tipo do instrumento com id 9 (reclassificação)
UPDATE instrumentos
SET tipo = 'Sopro'
WHERE id_instrumento = 9;

-- 10. Atualizar o nome da função com id 10 (ajuste descritivo)
UPDATE funcoes
SET funcao = 'Regente Assistente'  -- Nota: Usando 'funcao' após renomeação
```

```
49WHERE id_funcao = 10;
```

Deletes:

```
1-- 1. Deletar o músico com id 11 (demissão ou erro de cadastro)
2DELETE FROM musicos
3WHERE id_musico = 11;
4
5-- 2. Deletar o músico com id 12 (similar ao anterior)
6DELETE FROM musicos
7WHERE id_musico = 12;
8
9-- 3. Deletar o instrumento com id 13 (descarte)
10DELETE FROM instrumentos
11WHERE id_instrumento = 13;
12
13-- 4. Deletar o instrumento com id 14 (similar)
14DELETE FROM instrumentos
15WHERE id_instrumento = 14;
16
17-- 5. Deletar a função com id 15 (função obsoleta)
18DELETE FROM funcoes
19WHERE id_funcao = 15;
20
21-- 6. Deletar a orquestra com id 16 (dissolução)
22DELETE FROM orquestras
23WHERE id_orquestra = 16;
24
25-- 7. Deletar a orquestra com id 17 (similar)
26DELETE FROM orquestras
27WHERE id_orquestra = 17;
28
29-- 8. Deletar a sinfonia com id 18 (removida do repertório)
30DELETE FROM sinfonias
31WHERE id_sinfonia = 18;
32
33-- 9. Deletar a sinfonia com id 19 (similar)
34DELETE FROM sinfonias
35WHERE id_sinfonia = 19;
36
37-- 10. Deletar a função com id 20 (obsoleta)
38DELETE FROM funcoes

39WHERE id_funcao = 20;
```

Consultas / Relatórios (DQL)

Aqui, listo 20 consultas SELECT representativas, comentando o que cada uma retorna. Elas cobrem listagens, contagens, filtros e análises. Resultados simulados assumem dados de exemplo (ex: 50 músicos, 10 orquestras). Essas consultas demonstram a versatilidade do banco para responder perguntas como “Quais músicos estão em cada orquestra?”, “Qual a idade média dos músicos?” ou “Instrumentos disponíveis por tipo”.

1. Listar todos os músicos com suas orquestras

```
SELECT m.nome AS Musico, o.nome AS Orquestra
FROM musicos m
```

2. `JOIN orquestras o ON m.id_orquestra = o.id_orquestra;`
3. *Retorna:* Lista de pares (ex: "João Silva" - "Orquestra Sinfônica do Recife"). Útil para visão geral de alocações. *Simulado:* 50 linhas.
4. Listar músicos e suas funções

```
SELECT m.nome AS Musico, f.funcao AS Funcao
FROM musicos m
```

5. `JOIN funcoes f ON m.id_funcao = f.id_funcao;`
6. *Retorna:* Pares como "Maria Oliveira" - "Violinista". Mostra distribuição de papéis. *Simulado:* 50 linhas.
7. Mostrar o nome dos músicos, orquestra e função

```
SELECT m.nome, o.nome AS Orquestra, f.funcao AS Funcao
FROM musicos m
JOIN orquestras o ON m.id_orquestra = o.id_orquestra
```

8. `JOIN funcoes f ON m.id_funcao = f.id_funcao;`
9. *Retorna:* Tabela completa (ex: Nome | Orquestra | Função). Relatório integrado. *Simulado:* 50 linhas.
10. Contar quantos músicos existem por orquestra

```
SELECT o.nome AS Orquestra, COUNT(m.id_musico) AS Quantidade_Musicos
```

```
2FROM musicos m
3JOIN orquestras o ON m.id_orquestra = o.id_orquestra
```

11. ```
    4GROUP BY o.nome;
    ```
12. *Retorna:* Ex: "Orquestra Sinfônica" - 25 músicos. Análise de tamanho. *Simulado:* 10 linhas.
13. Mostrar os músicos nascidos após 1990

```
1SELECT nome, data_nascimento
2FROM musicos
```

14. ```
    3WHERE data_nascimento > '1990-01-01';
    ```
15. *Retorna:* Lista de jovens músicos (ex: 15 registros). Foco em novos talentos. *Simulado:* 15 linhas.
16. Listar orquestras e os países em que estão

```
1SELECT nome, pais
```

17. ```
    2FROM orquestras;
    ```
18. *Retorna:* Todas as orquestras com localização (ex: 10 linhas). *Simulado:* 10 linhas.
19. Mostrar todos os instrumentos cadastrados

```
1SELECT nome_instrumento, tipo, estado
```

20. ```
    2FROM instrumentos;
    ```
21. *Retorna:* Inventário completo (ex: "Violino Stradivarius" - "Corda" - "Disponível"). *Simulado:* 30 linhas.
22. Mostrar sinfonias e seus respectivos compositores

```
1SELECT titulo, compositor
```

23. ```
    2FROM sinfonias;
    ```
    ```

24. *Retorna:* Repertório (ex: "Sinfonia nº5" - "Beethoven"). *Simulado:* 20 linhas.
25. Contar quantos instrumentos existem por tipo
26. `1undefined`