📄 Documentação Completa – Banco de Dados Escola de Música

Autores: Maria Vitória, Lucas Vinicius e Renan Souza

---

1. Introdução

Este documento apresenta a modelagem completa e os scripts SQL do banco de dados **Escola de Música**. O sistema foi desenvolvido para gerenciar de forma integrada informações sobre músicos, orquestras, instrumentos, sinfonias e suas respectivas atuações, contemplando:

- Scripts de criação da estrutura (DDL)
- Manipulação de dados (DML)
- Consultas analíticas e relatórios
- Views para facilitar análises gerenciais

O banco de dados suporta funcionalidades como controle de desempenho de músicos, gestão de repertório de orquestras, rastreamento de funções e instrumentos, além de análises estatísticas sobre apresentações e avaliações.

---

2. Estrutura do Banco de Dados (DDL)

2.1 Criação do Banco

CREATE DATABASE EscolaDeMusica;

USE EscolaDeMusica;

2.2 Tabelas Principais

Tabela: Orquestras

Armazena informações sobre as orquestras cadastradas no sistema.

CREATE TABLE Orquestras (

```
    idOrquestra INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(150) NOT NULL,
    cidade VARCHAR(100) NOT NULL,
    pais VARCHAR(50) NOT NULL,
    dataCriacao DATE NOT NULL,
    telefone VARCHAR(20),
    email VARCHAR(100),
    status ENUM('ativa', 'inativa', 'suspenso') DEFAULT 'ativa',
    INDEX idx_cidade (cidade)
);
```

Campos adicionais:

- `email`: Contato eletrônico da orquestra
- `status`: Situação operacional (ativa, inativa, suspenso)

Tabela: Musicos

Cadastro completo dos músicos com informações pessoais e profissionais.

```
CREATE TABLE Musicos (
    idMusicos INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    identidade VARCHAR(20) UNIQUE,
    nacionalidade VARCHAR(50) NOT NULL,
    dataNasc DATE NOT NULL,
    email VARCHAR(100),
    telefone VARCHAR(20),
```

```
    endereco VARCHAR(200),
    salario DECIMAL(10,2),
    status ENUM('ativo', 'inativo', 'licenca') DEFAULT 'ativo',
    idOrquestra INT,
    FOREIGN KEY(idOrquestra) REFERENCES Orquestras(idOrquestra) ON DELETE SET NULL
);
```

Campos adicionais:

- `telefone`: Contato telefônico
- `endereco`: Endereço residencial
- `salario`: Remuneração mensal
- `status`: Situação do músico (ativo, inativo, licença)

Tabela: Instrumentos

Catálogo de instrumentos musicais disponíveis.

```
CREATE TABLE Instrumentos (
    idInstrumento INT AUTO_INCREMENT PRIMARY KEY,
    nomeInstrumento ENUM(
        'flauta','oboe','clarinete','violino','viola',
        'violoncelo','contrabaixo','trompa','trompete',
        'trombone','tuba','harpa','piano','saxofone',
        'guitarra','bateria','teclado','fagote',
        'cavaquinho','bandolim'
    ) NOT NULL UNIQUE,
    categoria VARCHAR(50),
```

```
    preco DECIMAL(10,2)
);
```

Campos adicionais:

- `categoria`: Classificação do instrumento (ex: Sopros de Madeira, Cordas)
- `preco`: Valor estimado do instrumento

Tabela: FuncoesDosMusicos

Define as funções/cargos que os músicos podem desempenhar.

```
CREATE TABLE FuncoesDosMusicos (
    idFuncao INT AUTO_INCREMENT PRIMARY KEY,
    nomeFuncao ENUM(
        'maestro','flautista','oboísta','clarinetista',
        'violinista','violista','violoncelista','contrabaixista',
        'trompista','trompetista','trombonista','tubista',
        'harpaista','pianista','saxofonista','guitarrista',
        'baterista','tecladista','fagotista','cavaquinista',
        'bandolinista'
    ) NOT NULL UNIQUE,
    salarioBase DECIMAL(10,2)
);
```

Campo adicional:

- `salarioBase`: Salário base da função

Tabela: Sinfonias

Registro das obras musicais do repertório.

```
CREATE TABLE Sinfonias (
    idSinfonia INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    compositor VARCHAR(100) NOT NULL,
    dataComposicao DATE NOT NULL,
    generoMusical VARCHAR(50),
    duracao TIME,
    dificuldade ENUM('baixa', 'media', 'alta', 'muito_alta') DEFAULT 'media'
);
```

Campos adicionais:

- `duracao`: Tempo de execução da obra
- `dificuldade`: Nível de complexidade técnica

2.3 Tabelas de Relacionamento

Tocam (Musicos ↔ Instrumentos)

Relaciona quais instrumentos cada músico sabe tocar.

```
CREATE TABLE Tocam (
    idMusicos INT,
    idInstrumento INT,
    nivelProficiencia ENUM('iniciante', 'intermediario', 'avancado', 'expert')
        DEFAULT 'intermediario',
    PRIMARY KEY(idMusicos, idInstrumento),
```

FOREIGN KEY(idMusicos) REFERENCES Musicos(idMusicos) ON DELETE CASCADE,

FOREIGN KEY(idInstrumento) REFERENCES Instrumentos(idInstrumento) ON DELETE CASCADE

);

Campo adicional:

- `nivelProficiencia`: Domínio técnico do músico no instrumento

Executam (Orquestras ↔ Sinfonias)

Registra quais sinfonias são executadas por cada orquestra.

CREATE TABLE Executam (

idSinfonia INT,

idOrquestra INT,

dataEstreia DATE,

numeroApresentacoes INT DEFAULT 1,

PRIMARY KEY(idSinfonia, idOrquestra),

FOREIGN KEY(idSinfonia) REFERENCES Sinfonias(idSinfonia) ON DELETE CASCADE,

FOREIGN KEY(idOrquestra) REFERENCES Orquestras(idOrquestra) ON DELETE CASCADE

);

Campos adicionais:

- `dataEstreia`: Data da primeira apresentação
- `numeroApresentacoes`: Quantidade de vezes executada

Desempenham (Musicos ↔ Funções)

Indica quais funções os músicos assumem em suas orquestras.

```
CREATE TABLE Desempenham (
    idMusicos INT,
    idFuncao INT,
    dataAssumiu DATE,
    dataFim DATE,
    observacoes TEXT,
    PRIMARY KEY(idMusicos, idFuncao),
    FOREIGN KEY(idMusicos) REFERENCES Musicos(idMusicos) ON DELETE CASCADE,
    FOREIGN KEY(idFuncao) REFERENCES FuncoesDosMusicos(idFuncao) ON DELETE CASCADE
);
```

Campo adicional:

- `observacoes`: Notas sobre o desempenho da função

Atuam (Musicos ↔ Sinfonias ↔ Funções ↔ Instrumentos)

Tabela central que relaciona todas as atuações dos músicos em sinfonias específicas.

```
CREATE TABLE Atuam (
    idMusicos INT,
    idSinfonia INT,
    idFuncao INT,
    idInstrumento INT,
    dataInicio DATE NOT NULL,
    dataFim DATE,
```

avaliacaoDesempenho ENUM('ruim', 'regular', 'bom', 'otimo', 'excelente'),

PRIMARY KEY(idMusicos, idSinfonia, idFuncao),

FOREIGN KEY(idMusicos) REFERENCES Musicos(idMusicos) ON DELETE CASCADE,

FOREIGN KEY(idSinfonia) REFERENCES Sinfonias(idSinfonia) ON DELETE CASCADE,

FOREIGN KEY(idFuncao) REFERENCES FuncoesDosMusicos(idFuncao) ON DELETE CASCADE,

FOREIGN KEY(idInstrumento) REFERENCES Instrumentos(idInstrumento) ON DELETE CASCADE

);

Campos adicionais:

- `dataFim`: Data de término da atuação
- `avaliacaoDesempenho`: Avaliação qualitativa do desempenho

---

3. Scripts de Manipulação de Dados (DML)

3.1 Exemplos de INSERT

-- Inserir Orquestras

INSERT INTO Orquestras (nome, cidade, pais, dataCriacao, telefone, status)

VALUES

('Orquestra Sinfônica de Recife', 'Recife', 'Brasil', '1990-03-15', '(81) 3421-8800', 'ativa'),

('Orquestra Filarmônica de São Paulo', 'São Paulo', 'Brasil', '1985-05-10', '(11) 3123-4567', 'ativa');

-- Inserir Músicos

INSERT INTO Musicos (nome, identidade, nacionalidade, dataNasc, email, salario, status, idOrquestra)

```
VALUES

('Roberto Silva', '147258369', 'Brasil', '1984-02-28', 'roberto.silva@email.com', 5000.00, 'ativo', 1),

('Patricia Oliveira', '258369147', 'Brasil', '1989-09-14', 'patricia.oliveira@email.com', 4500.00, 'ativo', 1);


-- Inserir Instrumentos

INSERT INTO Instrumentos (nomeInstrumento, categoria, preco)

VALUES

('flauta', 'Sopros de Madeira', 2800.00),

('violino', 'Cordas', 5500.00);
```

3.2 Exemplos de UPDATE

```
-- Atualizar salário de músicos

UPDATE Musicos SET salario = 6000.00 WHERE idMusicos = 1;


-- Alterar status para licença

UPDATE Musicos SET status = 'licenca' WHERE idMusicos = 4;


-- Atualizar informações de contato

UPDATE Musicos SET telefone = '(81) 98888-1234', endereco = 'Rua Nova, 999' WHERE idMusicos = 1;


-- Atualizar avaliações de desempenho

UPDATE Atuam SET avaliacaoDesempenho = 'excelente'

WHERE idMusicos = 1 AND idSinfonia = 1 AND idFuncao = 5;
```

```
-- Atualizar número de apresentações
UPDATE Executam SET numeroApresentacoes = 6
WHERE idSinfonia = 1 AND idOrquestra = 1;

-- Atualizar nível de proficiência
UPDATE Tocam SET nivelProficiencia = 'expert'
WHERE idMusicos = 1 AND idInstrumento = 4;
```

3.3 Exemplos de DELETE

```
-- Remover relacionamento músico-instrumento
DELETE FROM Tocam WHERE idMusicos = 10 AND idInstrumento = 3;

-- Remover atuação específica
DELETE FROM Atuam WHERE idMusicos = 10 AND idSinfonia = 10 AND idFuncao = 1;

-- Remover execução de sinfonia por orquestra
DELETE FROM Executam WHERE idSinfonia = 10 AND idOrquestra = 1;
```

---

4. Consultas SQL Úteis

4.1 Músicos que tocam mais de um instrumento

```
SELECT m.nome AS Nome,
    COUNT(t.idInstrumento) AS "Total Instrumentos",
```

```
    GROUP_CONCAT(i.nomeInstrumento) AS Instrumentos
FROM Musicos m
JOIN Tocam t ON m.idMusicos = t.idMusicos
JOIN Instrumentos i ON t.idInstrumento = i.idInstrumento
GROUP BY m.idMusicos, m.nome;
```

4.2 Sinfonias por compositor com orquestras executoras

```
SELECT
   s.compositor,
   COUNT(DISTINCT s.idSinfonia) as total_sinfonias,
   COUNT(DISTINCT e.idOrquestra) as total_orquestras_executoras,
   GROUP_CONCAT(DISTINCT s.nome ORDER BY s.nome SEPARATOR ', ') as sinfonias,
   GROUP_CONCAT(DISTINCT o.nome ORDER BY o.nome SEPARATOR ', ') as
orquestras_executoras,
   GROUP_CONCAT(DISTINCT o.cidade ORDER BY o.cidade SEPARATOR ', ') as
cidades_apresentacao
FROM Sinfonias s
LEFT JOIN Executam e ON s.idSinfonia = e.idSinfonia
LEFT JOIN Orquestras o ON e.idOrquestra = o.idOrquestra
GROUP BY s.compositor
ORDER BY total_sinfonias DESC, s.compositor;
```

4.3 Instrumentos mais utilizados

```
SELECT i.nomeInstrumento,
    COUNT(at.idMusicos) AS total_uso
```

```
FROM Instrumentos i
JOIN Atuam at ON i.idInstrumento = at.idInstrumento
GROUP BY i.idInstrumento, i.nomeInstrumento
ORDER BY total_uso DESC;
```

4.4 Maestros e suas sinfonias

```
SELECT m.nome AS Maestro,
    s.nome AS Sinfonia,
    s.compositor AS Compositor,
    DATE_FORMAT(at.DataInicio, '%d/%m/%Y') AS DataInicio
FROM Atuam at
JOIN Musicos m ON at.idMusicos = m.idMusicos
JOIN Sinfonias s ON at.idSinfonia = s.idSinfonia
JOIN FuncoesDosMusicos f ON at.idFuncao = f.idFuncao
WHERE f.nomeFuncao = 'maestro'
ORDER BY DataInicio;
```

4.5 Músicos com avaliação máxima

```
SELECT
   m.nome AS Músico,
   s.nome AS Sinfonia,
   f.nomeFuncao AS Função,
   atu.avaliacaoDesempenho AS Avaliação
FROM Atuam atu
```

```
JOIN Musicos m ON atu.idMusicos = m.idMusicos
JOIN Sinfonias s ON atu.idSinfonia = s.idSinfonia
JOIN FuncoesDosMusicos f ON atu.idFuncao = f.idFuncao
WHERE atu.avaliacaoDesempenho = 'excelente'
ORDER BY s.nome, m.nome;
```

4.6 Orquestras com média salarial

```
SELECT o.nome AS Orquestra,
    COUNT(m.idMusicos) AS "Total Músicos",
    CONCAT("R$ ", FORMAT(AVG(m.salario), 2, 'de_DE')) AS MediaSalarial
FROM Orquestras o
JOIN Musicos m ON o.idOrquestra = m.idOrquestra
GROUP BY o.idOrquestra, o.nome
ORDER BY AVG(m.salario) DESC;
```

4.7 Sinfonias mais complexas com média de avaliação

```
SELECT
  s.nome AS Sinfonia,
  s.dificuldade AS Dificuldade,
  FORMAT(AVG(
    CASE atu.avaliacaoDesempenho
      WHEN 'ruim' THEN 1
      WHEN 'regular' THEN 2
      WHEN 'bom' THEN 3
```

```
            WHEN 'otimo' THEN 4
            WHEN 'excelente' THEN 5
        END
    ), 2) AS MediaAvaliacao
FROM Sinfonias s
JOIN Atuam atu ON s.idSinfonia = atu.idSinfonia
GROUP BY s.idSinfonia
ORDER BY MediaAvaliacao DESC;
```

4.8 Sinfonias por dificuldade e apresentações

```
SELECT s.nome AS Sinfonia,
       s.compositor AS Compositor,
       s.dificuldade AS Dificuldade,
       COUNT(e.idOrquestra) AS "Total Orquestras",
       SUM(e.numeroApresentacoes) AS TotalApresentacoes
FROM Sinfonias s
LEFT JOIN Executam e ON s.idSinfonia = e.idSinfonia
GROUP BY s.idSinfonia, s.nome, s.dificuldade
ORDER BY TotalApresentacoes DESC,
         FIELD(s.dificuldade, 'muito_alta', 'alta', 'media', 'baixa');
```

---

5. Views SQL

5.1 vw_musicos_orquestras

Visão geral dos músicos e suas orquestras.

```
CREATE VIEW vw_musicos_orquestras AS
SELECT m.idMusicos,
    m.nome AS "Nome Musicos",
    m.email AS Email,
    o.nome AS nomeOrquestra,
    o.cidade,
    o.pais
FROM Musicos m
LEFT JOIN Orquestras o ON m.idOrquestra = o.idOrquestra;
```

5.2 vw_musicos_instrumentos

Relacionamento músicos e instrumentos que dominam.

```
CREATE VIEW vw_musicos_instrumentos AS
SELECT m.nome AS Musicos,
    i.nomeInstrumento AS "Nome Instrumento"
FROM Musicos m
JOIN Tocam t ON m.idMusicos = t.idMusicos
JOIN Instrumentos i ON t.idInstrumento = i.idInstrumento;
```

5.3 vw_sinfonias_orquestras

Sinfonias com informações das orquestras executoras.

```
CREATE VIEW vw_sinfonias_orquestras AS
SELECT s.nome AS sinfonia,
```

```
    s.compositor,
    s.generoMusical,
    o.nome AS orquestra,
    o.cidade,
    o.pais
FROM Sinfonias s
JOIN Executam e ON s.idSinfonia = e.idSinfonia
JOIN Orquestras o ON e.idOrquestra = o.idOrquestra;
```

5.4 vw_atuacoes_completas

Visão completa das atuações dos músicos.

```
CREATE VIEW vw_atuacoes_completas AS
SELECT m.nome AS Musicos,
    s.nome AS Sinfonia,
    s.Compositor,
    f.nomeFuncao,
    i.nomeInstrumento,
    DATE_FORMAT(at.dataInicio, '%d/%m/%Y') AS DataInicio,
    o.nome AS Orquestra
FROM Atuam at
JOIN Musicos m ON at.idMusicos = m.idMusicos
JOIN Sinfonias s ON at.idSinfonia = s.idSinfonia
JOIN FuncoesDosMusicos f ON at.idFuncao = f.idFuncao
JOIN Instrumentos i ON at.idInstrumento = i.idInstrumento
```

```
JOIN Orquestras o ON m.idOrquestra = o.idOrquestra;
```

5.5 vw_estatisticas_orquestra

Estatísticas agregadas por orquestra.

```
CREATE VIEW vw_estatisticas_orquestra AS
SELECT o.nome AS Orquestra,
    COUNT(DISTINCT m.idMusicos) AS TotalMusicos,
    COUNT(DISTINCT e.idSinfonia) AS "Total Sinfonias",
    o.cidade AS Cidade,
    o.pais AS País
FROM Orquestras o
LEFT JOIN Musicos m ON o.idOrquestra = m.idOrquestra
LEFT JOIN Executam e ON o.idOrquestra = e.idOrquestra
GROUP BY o.idOrquestra
ORDER BY TotalMusicos;
```

5.6 vw_musicos_faixa_salarial

Distribuição de músicos por faixa salarial.

```
CREATE VIEW vw_musicos_faixa_salarial AS
SELECT
  CASE
    WHEN salario < 4000 THEN 'Baixo'
    WHEN salario BETWEEN 4000 AND 6000 THEN 'Médio'
    WHEN salario > 6000 THEN 'Alto'
```

```
        ELSE 'Não informado'
    END AS faixa_salarial,
    COUNT(*) as "Quantidade Músicos",
    CONCAT("R$ ", FORMAT(AVG(salario), 2, 'de_DE')) as "Salário Médio"
FROM Musicos
WHERE salario IS NOT NULL
GROUP BY faixa_salarial;
```

5.7 vw_instrumentos_categoria_preco

Análise de instrumentos por categoria e preço.

```
CREATE VIEW vw_instrumentos_categoria_preco AS
SELECT categoria,
    COUNT(*) as total_instrumentos,
    CONCAT("R$ ", FORMAT(AVG(preco), 2, 'de_DE')) as preco_medio,
    CONCAT("R$ ", FORMAT(MIN(preco), 2, 'de_DE')) as preco_minimo,
    CONCAT("R$ ", FORMAT(MAX(preco), 2, 'de_DE')) as preco_maximo
FROM Instrumentos
WHERE categoria IS NOT NULL
GROUP BY categoria;
```

5.8 vw_sinfonias_dificuldade

Sinfonias agrupadas por nível de dificuldade.

```
CREATE VIEW vw_sinfonias_dificuldade AS
SELECT dificuldade,
```

```
    COUNT(*) as total_sinfonias,
    FORMAT(AVG(TIME_TO_SEC(duracao)/60), 2) as duracao_media_minutos
FROM Sinfonias
WHERE dificuldade IS NOT NULL
GROUP BY dificuldade
ORDER BY
  CASE dificuldade
    WHEN 'baixa' THEN 1
    WHEN 'media' THEN 2
    WHEN 'alta' THEN 3
    WHEN 'muito_alta' THEN 4
  END;
```

5.9 vw_orquestras_status_regiao

Distribuição geográfica das orquestras por status.

```
CREATE VIEW vw_orquestras_status_regiao AS
SELECT status,
    CASE
      WHEN cidade IN ('São Paulo', 'Campinas') THEN 'Sudeste-SP'
      WHEN cidade IN ('Rio de Janeiro') THEN 'Sudeste-RJ'
      WHEN cidade IN ('Belo Horizonte') THEN 'Sudeste-MG'
      WHEN cidade IN ('Porto Alegre', 'Florianópolis', 'Curitiba') THEN 'Sul'
      WHEN cidade IN ('Salvador', 'Recife', 'Fortaleza', 'Natal', 'Maceió') THEN 'Nordeste'
      WHEN cidade IN ('Brasília', 'Goiânia') THEN 'Centro-Oeste'
```

```
        WHEN cidade IN ('Manaus') THEN 'Norte'
        ELSE 'Outras'
    END AS regiao,
    COUNT(*) as total_orquestras
FROM Orquestras
GROUP BY status, regiao;
```

5.10 vw_performance_artistas

Análise de performance individual dos músicos.

```
CREATE VIEW vw_performance_artistas AS
SELECT m.nome as Músico,
    COUNT(DISTINCT at.idSinfonia) as "Total Sinfonia",
    COUNT(DISTINCT t.idInstrumento) as "Total Instrumentos",
    COUNT(DISTINCT d.idFuncao) as "Total Funções",
    CONCAT("R$ ", FORMAT(m.salario, 2, 'de_DE')) AS Salário,
    o.nome as Orquestra
FROM Musicos m
LEFT JOIN Atuam at ON m.idMusicos = at.idMusicos
LEFT JOIN Tocam t ON m.idMusicos = t.idMusicos
LEFT JOIN Desempenham d ON m.idMusicos = d.idMusicos
LEFT JOIN Orquestras o ON m.idOrquestra = o.idOrquestra
GROUP BY m.idMusicos;
```

---

6. Principais Alterações e Melhorias

6.1 Campos Adicionados via ALTER TABLE

-- Orquestras

ALTER TABLE Orquestras ADD COLUMN email VARCHAR(100);

ALTER TABLE Orquestras ADD COLUMN status ENUM('ativa', 'inativa', 'suspenso') DEFAULT 'ativa';

-- Músicos

ALTER TABLE Musicos ADD COLUMN telefone VARCHAR(20);

ALTER TABLE Musicos ADD COLUMN endereco VARCHAR(200);

ALTER TABLE Musicos ADD COLUMN salario DECIMAL(10,2);

ALTER TABLE Musicos ADD COLUMN status ENUM('ativo', 'inativo', 'licenca') DEFAULT 'ativo';

-- Instrumentos

ALTER TABLE Instrumentos ADD COLUMN categoria VARCHAR(50);

ALTER TABLE Instrumentos ADD COLUMN preco DECIMAL(10,2);

-- Sinfonias

ALTER TABLE Sinfonias ADD COLUMN duracao TIME;

ALTER TABLE Sinfonias ADD COLUMN dificuldade ENUM('baixa', 'media', 'alta', 'muito_alta') DEFAULT 'media';

-- Tabelas de Relacionamento

ALTER TABLE Tocam ADD COLUMN nivelProficiencia ENUM('iniciante', 'intermediario', 'avancado', 'expert') DEFAULT 'intermediario';

```
ALTER TABLE Executam ADD COLUMN dataEstreia DATE;

ALTER TABLE Executam ADD COLUMN numeroApresentacoes INT DEFAULT 1;

ALTER TABLE Desempenham ADD COLUMN observacoes TEXT;

ALTER TABLE Atuam ADD COLUMN dataFim DATE;

ALTER TABLE Atuam ADD COLUMN avaliacaoDesempenho ENUM('ruim', 'regular', 'bom',
'otimo', 'excelente');
```

6.2 Funcionalidades Implementadas

- **Controle de Status**: Monitoramento do estado operacional de orquestras e músicos
- **Gestão Salarial**: Rastreamento de remunerações e faixas salariais
- **Avaliação de Desempenho**: Sistema de avaliação qualitativa das atuações
- **Controle de Proficiência**: Níveis de domínio técnico dos instrumentos
- **Estatísticas de Apresentações**: Contagem e histórico de execuções
- **Análise Regional**: Distribuição geográfica das orquestras brasileiras

---

7. Diagrama Entidade-Relacionamento (Conceitual)

Entidades Principais:

- **Orquestras**: Organizações musicais
- **Musicos**: Profissionais da música
- **Instrumentos**: Ferramentas musicais
- **Sinfonias**: Obras musicais
- **FuncoesDosMusicos**: Cargos/papéis profissionais

Relacionamentos:

- Músicos **pertencem a** Orquestras (N:1)
- Músicos **tocam** Instrumentos (N:N)
- Músicos **desempenham** Funções (N:N)
- Orquestras **executam** Sinfonias (N:N)
- Músicos **atuam em** Sinfonias com Funções e Instrumentos (N:N:N:N)

---

8. Considerações sobre Integridade

8.1 Chaves Estrangeiras

Todas as relações utilizam `ON DELETE CASCADE` para manter integridade referencial, exceto:

- `Musicos.idOrquestra`: usa `ON DELETE SET NULL` para preservar histórico

8.2 Validações

- Emails únicos para músicos
- Identidades únicas
- Instrumentos e funções com valores ENUM predefinidos
- Datas obrigatórias para atuações
- Índice em `cidade` para otimizar buscas geográficas

---

9. Recomendações de Uso

9.1 Backup e Manutenção

- Realizar backups diários do banco
- Documentar todas as alterações estruturais
- Testar scripts de DELETE em ambiente de desenvolvimento

9.2 Otimização

- Criar índices adicionais em campos frequentemente consultados
- Monitorar performance das views complexas
- Implementar procedures para operações recorrentes

9.3 Expansões Futuras

- Sistema de reserva de ensaios
- Controle de repertório por temporada
- Gestão de cachês e contratos
- Histórico de mudanças de orquestra
- Sistema de avaliação 360° entre músicos

---

10. Conclusão

Este banco de dados oferece uma solução completa e robusta para gestão de uma escola ou organização musical. A estrutura normalizada permite escalabilidade, enquanto as views facilitam análises gerenciais complexas. Os campos adicionais implementados via ALTER TABLE demonstram a evolução do sistema para atender necessidades administrativas e analíticas avançadas.

Próximos Passos:

1. Implementar triggers para auditoria
2. Criar stored procedures para relatórios complexos
3. Desenvolver API REST para integração com sistemas externos
4. Implementar sistema de autenticação e permissões

Data de Última Atualização: Outubro/2024
Versão do Banco: 2.0
Compatibilidade: MySQL 5.7+