Banco de Dados I

Modelo físico - comentado

Gestão da Tecnologia da Informação - 2025.01

Sumário

1	Configurações Iniciais	2
2	Criação do Schema	3
3	Tabela alunos	4
4	Tabela enderecos	5
5	Tabela disciplinas	7
6	Tabela professores	8
7	Tabela disciplinas_professores	9
8	Tabela alunos_disciplinas	11
9	Tabela faltas	13
10	Restauração das Configurações	15

1 Configurações Iniciais

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,
    STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,
    ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
```

Essas instruções salvam os valores atuais de algumas configurações do MySQL e os desabilitam temporariamente. Isso evita erros durante a criação das tabelas e relacionamentos.

2 Criação do Schema

```
CREATE SCHEMA IF NOT EXISTS 'mydb' DEFAULT CHARACTER SET utf8;  
USE 'mydb';
```

Cria um novo banco de dados chamado 'mydb' com codificação UTF-8 e define-o como o banco atual para execução dos comandos subsequentes.

3 Tabela alunos

```
CREATE TABLE IF NOT EXISTS 'alunos' (  
  'id' INT NOT NULL AUTO_INCREMENT,  
  'nome' VARCHAR(100) NOT NULL,  
  'email_academico' VARCHAR(150) NULL,  
  PRIMARY KEY ('id'),  
  UNIQUE INDEX 'email_academico_UNIQUE' (('email_academico' ASC))  
ENGINE = InnoDB;
```

Explicação Detalhada

CREATE TABLE IF NOT EXISTS

Este comando cria uma nova tabela no banco de dados. O modificador **IF NOT EXISTS** evita erro se a tabela `alunos` já existir, impedindo que ela seja recriada.

Colunas da Tabela

- **id INT NOT NULL AUTO_INCREMENT**: cria uma coluna chamada `id`, do tipo inteiro, que não aceita valores nulos. O atributo `AUTO_INCREMENT` faz com que novos valores sejam gerados automaticamente e sequencialmente.
- **nome VARCHAR(100) NOT NULL**: define uma coluna do tipo texto (até 100 caracteres), obrigatória, que armazena o nome do aluno.
- **email_academico VARCHAR(150) NULL**: cria uma coluna do tipo texto (até 150 caracteres), que pode ser deixada em branco (valor nulo).

Chave Primária

`PRIMARY KEY (id)` define a coluna `id` como chave primária da tabela, garantindo que seu valor seja único e não nulo.

Índice Único

`UNIQUE INDEX email_academico_UNIQUE (email_academico ASC)` cria um índice que impede a repetição de valores no campo `email_academico`, assegurando a unicidade.

Engine de Armazenamento

`ENGINE = InnoDB` define que a tabela usará o mecanismo de armazenamento `InnoDB`, que permite transações, integridade referencial e desempenho robusto para operações relacionais.

4 Tabela enderecos

```
CREATE TABLE IF NOT EXISTS 'enderecos' (  
  'id' INT NOT NULL AUTO_INCREMENT,  
  'CEP' CHAR(9) NOT NULL,  
  'rua' VARCHAR(45) NULL,  
  'complemento' VARCHAR(45) NULL,  
  'alunos_id' INT NOT NULL,  
  PRIMARY KEY ('id'),  
  INDEX 'fk_enderecos_alunos_idx' ('alunos_id' ASC),  
  CONSTRAINT 'fk_enderecos_alunos'  
    FOREIGN KEY ('alunos_id')  
    REFERENCES 'alunos' ('id'))  
ENGINE = InnoDB;
```

Relaciona os endereços com os alunos por meio de uma chave estrangeira.

Explicação Detalhada

CREATE TABLE IF NOT EXISTS

Cria a tabela `enderecos`, caso ela ainda não exista no banco de dados. Isso evita erro de duplicação.

Colunas da Tabela

- `id INT NOT NULL AUTO_INCREMENT`: identificador único para cada endereço. A geração é automática e incremental.
- `CEP CHAR(9) NOT NULL`: campo fixo de 9 caracteres (ex: 00000-000), obrigatório.
- `rua VARCHAR(45) NULL`: nome da rua, campo de texto com até 45 caracteres, opcional.
- `complemento VARCHAR(45) NULL`: complemento do endereço, também opcional.
- `alunos_id INT NOT NULL`: chave estrangeira que se relaciona com a tabela `alunos`, obrigatória.

Chave Primária

`PRIMARY KEY (id)` define o campo `id` como identificador único da tabela.

Índice

`INDEX fk_enderecos_alunos_idx (alunos_id ASC)` cria um índice para otimizar buscas baseadas no campo `alunos_id`.

Chave Estrangeira

- `CONSTRAINT fk_enderecos_alunos FOREIGN KEY (alunos_id)` cria uma restrição que obriga o valor de `alunos_id` a existir previamente na tabela `alunos`.
- Isso estabelece a integridade referencial entre os dados, garantindo que não haja endereços sem aluno associado.

Engine de Armazenamento

ENGINE = InnoDB define o uso do mecanismo de armazenamento InnoDB, que suporta integridade referencial e transações.

5 Tabela disciplinas

```
CREATE TABLE IF NOT EXISTS 'disciplinas' (  
  'id' INT NOT NULL AUTO_INCREMENT,  
  'nome' VARCHAR(100) NOT NULL,  
  'descricao' VARCHAR(300) NOT NULL,  
  'carga_horaria' INT NOT NULL,  
  'obrigatoria' INT NOT NULL,  
  PRIMARY KEY ('id'))  
ENGINE = InnoDB;
```

Define a tabela de disciplinas com campos básicos, incluindo carga horária e se a disciplina é obrigatória.

Explicação Detalhada

CREATE TABLE IF NOT EXISTS

Cria a tabela `disciplinas` somente se ela ainda não existir no banco de dados, evitando duplicação.

Colunas da Tabela

- `id INT NOT NULL AUTO_INCREMENT`: identificador único da disciplina, gerado automaticamente.
- `nome VARCHAR(100) NOT NULL`: nome da disciplina, obrigatório, com limite de 100 caracteres.
- `descricao VARCHAR(300) NOT NULL`: breve descrição do conteúdo da disciplina, também obrigatória.
- `carga_horaria INT NOT NULL`: número de horas da disciplina. Espera-se um valor inteiro, como 60, 80 etc.
- `obrigatoria INT NOT NULL`: valor que indica se a disciplina é obrigatória (poderia ser 1 para sim e 0 para não, por exemplo).

Chave Primária

`PRIMARY KEY (id)` indica que o campo `id` é o identificador exclusivo para cada registro da tabela.

Engine de Armazenamento

`ENGINE = InnoDB` define o uso do mecanismo de armazenamento InnoDB, necessário para suporte a transações e integridade referencial.

6 Tabela professores

```
CREATE TABLE IF NOT EXISTS 'professores' (  
  'id' INT NOT NULL AUTO_INCREMENT,  
  'nome' VARCHAR(100) NOT NULL,  
  'email_academico' VARCHAR(150) NOT NULL,  
  PRIMARY KEY ('id'),  
  UNIQUE INDEX 'email_academico_UNIQUE' (('email_academico' ASC))  
ENGINE = InnoDB;
```

Tabela com informações de professores, também com índice único no e-mail.

Explicação Detalhada

CREATE TABLE IF NOT EXISTS

Cria a tabela `professores` apenas se ela ainda não existir, evitando erros caso já esteja presente no banco de dados.

Colunas da Tabela

- `id INT NOT NULL AUTO_INCREMENT`: identificador único de cada professor. O valor é gerado automaticamente e não pode ser nulo.
- `nome VARCHAR(100) NOT NULL`: nome completo do professor. Deve ser preenchido obrigatoriamente e pode ter até 100 caracteres.
- `email_academico VARCHAR(150) NOT NULL`: endereço de e-mail institucional ou acadêmico do professor, com até 150 caracteres. Também é obrigatório.

Chave Primária

`PRIMARY KEY (id)` define o campo `id` como a chave primária, garantindo a unicidade de cada registro na tabela.

Índice Único

`UNIQUE INDEX email_academico_UNIQUE (email_academico ASC)` garante que não existam dois professores com o mesmo e-mail acadêmico, evitando duplicidade de registros com esse dado sensível.

Engine de Armazenamento

`ENGINE = InnoDB` especifica o uso do mecanismo InnoDB, que oferece suporte a integridade referencial e transações ACID.

7 Tabela disciplinas_professores

```
CREATE TABLE IF NOT EXISTS 'disciplinas_professores' (  
  'disciplinas_id' INT NULL,  
  'professores_id' INT NULL,  
  'id' INT NOT NULL,  
  PRIMARY KEY ('id'),  
  INDEX 'fk_disciplinas_has_professores_professores1_idx' (  
    professores_id' ASC),  
  INDEX 'fk_disciplinas_has_professores_disciplinas1_idx' (  
    disciplinas_id' ASC),  
  CONSTRAINT 'fk_disciplinas_has_professores_disciplinas1'  
    FOREIGN KEY ('disciplinas_id')  
    REFERENCES 'disciplinas' ('id'),  
  CONSTRAINT 'fk_disciplinas_has_professores_professores1'  
    FOREIGN KEY ('professores_id')  
    REFERENCES 'professores' ('id'))  
ENGINE = InnoDB;
```

Relaciona professores às disciplinas lecionadas, com uma tabela de junção intermediária.

Explicação Detalhada

Objetivo da Tabela

A tabela `disciplinas_professores` representa uma tabela de associação (ou tabela intermediária) que modela um relacionamento muitos-para-muitos entre as tabelas `disciplinas` e `professores`.

Colunas

- `disciplinas_id` INT NULL: campo que armazena o ID de uma disciplina. Pode ser nulo.
- `professores_id` INT NULL: campo que armazena o ID de um professor. Pode ser nulo.
- `id` INT NOT NULL: identificador único da relação. Será a chave primária da tabela.

Chave Primária

`PRIMARY KEY (id)`: define que o campo `id` é o identificador único de cada linha da tabela.

Índices

- `fk_disciplinas_has_professores_disciplinas1_idx`: índice para melhorar o desempenho nas buscas baseadas no campo `disciplinas_id`.
- `fk_disciplinas_has_professores_professores1_idx`: índice para o campo `professores_id`.

Chaves Estrangeiras

- `fk_disciplinas_has_professores_disciplinas1`: define o campo `disciplinas_id` como chave estrangeira que referencia o campo `id` da tabela `disciplinas`.
- `fk_disciplinas_has_professores_professores1`: define o campo `professores_id` como chave estrangeira que referencia o campo `id` da tabela `professores`.

Engine de Armazenamento

`ENGINE = InnoDB`: utiliza o mecanismo de armazenamento InnoDB, que dá suporte a chaves estrangeiras e integridade referencial.

8 Tabela alunos_disciplinas

```
CREATE TABLE IF NOT EXISTS 'alunos_disciplinas' (  
  'alunos_id' INT NULL,  
  'disciplinas_id' INT NULL,  
  'id' INT NOT NULL AUTO_INCREMENT,  
  PRIMARY KEY ('id'),  
  INDEX 'fk_alunos_has_disciplinas_disciplinas1_idx' (('disciplinas_id'  
    ASC),  
  INDEX 'fk_alunos_has_disciplinas_alunos1_idx' ('alunos_id' ASC),  
  CONSTRAINT 'fk_alunos_has_disciplinas_alunos1'  
    FOREIGN KEY ('alunos_id')  
    REFERENCES 'alunos' ('id'),  
  CONSTRAINT 'fk_alunos_has_disciplinas_disciplinas1'  
    FOREIGN KEY ('disciplinas_id')  
    REFERENCES 'disciplinas' ('id'))  
ENGINE = InnoDB;
```

Tabela que associa alunos às disciplinas que estão cursando.

Objetivo da Tabela

A tabela `alunos_disciplinas` registra o relacionamento entre alunos e as disciplinas que estão cursando, modelando uma relação muitos-para-muitos.

Colunas

- `alunos_id` INT NULL: identifica o aluno associado. Permite valor nulo.
- `disciplinas_id` INT NULL: identifica a disciplina associada. Permite valor nulo.
- `id` INT NOT NULL AUTO_INCREMENT: identificador único da linha, gerado automaticamente.

Chave Primária

PRIMARY KEY (id): define o campo `id` como chave primária da tabela.

Índices

- `fk_alunos_has_disciplinas_disciplinas1_idx`: índice para o campo `disciplinas_id`.
- `fk_alunos_has_disciplinas_alunos1_idx`: índice para o campo `alunos_id`.

Chaves Estrangeiras

- `fk_alunos_has_disciplinas_alunos1`: relaciona `alunos_id` ao campo `id` da tabela `alunos`.
- `fk_alunos_has_disciplinas_disciplinas1`: relaciona `disciplinas_id` ao campo `id` da tabela `disciplinas`.

Engine de Armazenamento

ENGINE = InnoDB: define o mecanismo de armazenamento como InnoDB, que permite o uso de chaves estrangeiras e transações com integridade referencial.

9 Tabela faltas

```
CREATE TABLE IF NOT EXISTS 'faltas' (  
  'id' INT NOT NULL AUTO_INCREMENT,  
  'alunos_id' INT NOT NULL,  
  'professores_id' INT NOT NULL,  
  'disciplinas_id' INT NOT NULL,  
  'total_faltas' INT NULL,  
  PRIMARY KEY ('id'),  
  INDEX 'fk_faltas_alunos1_idx' ('alunos_id' ASC),  
  INDEX 'fk_faltas_professores1_idx' ('professores_id' ASC),  
  INDEX 'fk_faltas_disciplinas1_idx' ('disciplinas_id' ASC),  
  CONSTRAINT 'fk_faltas_alunos1'  
    FOREIGN KEY ('alunos_id')  
    REFERENCES 'alunos' ('id'),  
  CONSTRAINT 'fk_faltas_professores1'  
    FOREIGN KEY ('professores_id')  
    REFERENCES 'professores' ('id'),  
  CONSTRAINT 'fk_faltas_disciplinas1'  
    FOREIGN KEY ('disciplinas_id')  
    REFERENCES 'disciplinas' ('id'))  
ENGINE = InnoDB;
```

Tabela que registra o número de faltas dos alunos, vinculando ao professor e à disciplina.

Objetivo da Tabela

A tabela `faltas` registra o número de faltas que cada aluno teve em uma disciplina ministrada por um determinado professor. Representa a associação entre aluno, disciplina e professor, com uma informação adicional: o total de faltas.

Colunas

- `id` INT NOT NULL AUTO_INCREMENT: identificador único da falta, gerado automaticamente.
- `alunos_id` INT NOT NULL: ID do aluno relacionado.
- `professores_id` INT NOT NULL: ID do professor responsável pela disciplina.
- `disciplinas_id` INT NOT NULL: ID da disciplina onde a falta ocorreu.
- `total_faltas` INT NULL: quantidade de faltas registradas.

Chave Primária

PRIMARY KEY (`id`): define o campo `id` como identificador único da tabela.

Índices

- `fk_faltas_alunos1_idx`: índice sobre o campo `alunos_id`.
- `fk_faltas_professores1_idx`: índice sobre o campo `professores_id`.
- `fk_faltas_disciplinas1_idx`: índice sobre o campo `disciplinas_id`.

Chaves Estrangeiras

- `fk_faltas_alunos1`: relaciona `alunos_id` ao campo `id` da tabela `alunos`.
- `fk_faltas_professores1`: relaciona `professores_id` ao campo `id` da tabela `professores`.
- `fk_faltas_disciplinas1`: relaciona `disciplinas_id` ao campo `id` da tabela `disciplinas`.

Engine de Armazenamento

`ENGINE = InnoDB`: usa o mecanismo InnoDB, que suporta chaves estrangeiras e integridade referencial.

10 Restauração das Configurações

```
SET SQL_MODE=@OLD_SQL_MODE;  
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;  
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Restaura as configurações iniciais do MySQL, garantindo que o ambiente volte ao estado anterior.