Módulo 3: Banco de Dados – Bloco 2: Modelagem de Banco de Dados – 20/03/2023

P2 – Projeto Prático de Modelagem de Banco de Dados.

Objetivos:

Nesse projeto prático, iremos modelar um banco de dados para uma escola de idiomas que quer informatizar seu sistema de matrícula.

Etapas:

1. Levantamento dos requisitos.
2. Identificação das entidades e dos relacionamentos.
3. Modelagem conceitual usando DER.
4. Dicionário.
5. Modelagem Lógica.
6. Normalização.
7. Modelo Físico.
8. Testes.

Regras de Negócio:

- A escola de idiomas oferece cursos em várias línguas (ex. Inglês, Espanhol e Francês).
- Cada curso é dividido nos níveis Básico, Intermediário e Avançado.
- Cada nível possui turmas que variam com o período e a frequência de aulas (ex. Segunda e Quarta de manhã, Terça e Quinta de Noite, Sábado de Tarde).
- Cada turma aceita de 1 a 10 alunos.
- Cada professor pertence a apenas um curso (Inglês ou Espanhol ou Francês).
- Cada professor pode dar aula para até 3 turmas.
- Cada aluno pode fazer parte de apenas uma turma de um curso.

Entidades e Atributos:

- Idioma / idioma
  - Código do Idioma / id_idioma / int / PK / AI
  - Nome do Idioma /nome_idioma / varchar(15) / NN

- Nível / nivel
  - Código do Nível / id_nivel / char(1) / PK / AI
  - Nome do Nível /nome_nivel / varchar (15) / NN

- Idioma_Nível / idioma_nivel
  - Código Idioma-Nível / id_idioma_nivel / int / PK / AI
  - Código do Idioma / id_idioma / int / FK
  - Código do Nível / id nivel / int / FK
- Turma / turma
  - Código da Turma / id_turma / int / PK / AI
  - Código do Idioma / id_idioma / int / FK
  - Dias de aula / dias_aula / varchar (40) / NN
  - Data início / turma_inicio / date / NN
  - Data fim / turma_fim / date / NN
  - Numero de alunos / numero_alunos / int / NN
  - Carga Horária / carga_horaria / int / NN

- Boletim / boletim
  - Código do boletim / id_boletim / int / PK / AI
  - Código da Turma/ id_turma / int / FK
  - Cadastro do aluno / id_aluno / int / FK
  - Nota / nota_aluno / decimal / NULL
  - Frequência (%) / freq_aluno (int) / NULL

- Aluno / aluno
  - Cadastro aluno/ id_aluno / int / PK
  - Nome / nome_aluno / varchar (20) / NN
  - Sobrenome / sobrenome_aluno / varchar (20) / NN
  - Nascimento / nasc_aluno / date / NN
  - Telefone / telefone_aluno
    - tel_fixo: char (10) / NULL
    - celular: char(11) / NULL
  - Endereço /endereco_aluno
    - rua / varchar(50) / NN
    - numero / int / NN
    - complemento / varchar (20) / NULL
    - bairro / varchar (20) / NULL
    - CEP / char(8) / NN
    - cidade / varchar (30) / NN
  - Email / email_aluno / varchar (40) / NULL
  - Idioma / id_idioma / int / FK / NN

- Professor / professor
  - Cadastro professor / id_professor / int / PK
  - Nome / nome_professor / varchar (20) / NN
  - Sobrenome / sobrenome_professor / varchar (20) / NN
  - Idioma / id_idioma / int / FK / NN

Relacionamentos:

- idioma (1,n) possui (1,n) nível – Cardinalidade Muitos para Muitos (N,M)
  - idioma (1,1) compõe (1,n) idioma_nível (1,n) possui (1,1) nível.
- nível (1,1) gera (1,n) turmas
- turmas (1,1) possuem (1,n) alunos
- professores (1,n) pertencem a (1,1)idioma
- professores (1,1) lecionam (1,n) turmas
- turmas (1,1) geram (1,n) boletim
- boletim (1,1) pertence a (1,1) aluno

Normalização:

1FN – Primeira Forma Normal: uma entidade está na primeira forma normal se:

- Possuir chave-primária.
- Todos os valores forem atômicos.
- Não possuir atributos multivalorados.
- Não possuir atributos compostos.

Alterações:

- Desmembramento de endereço em partes (rua, número, CEP, …).
- Telefone foi dividido em residencial e celular.

2FN – Segunda Forma Normal: uma entidade está na segunda forma normal se:

- Estiver na 1FN.
- Atributos que não sejam chaves (PK ou FK) forem dependentes de todas as partes das chaves-primárias.
- Não existirem dependências parciais.

Alterações:

- Endereço: Rua → tipo de logradouro
  - alunos (1,n) possui (1,1) endereco_aluno (1,n) determinado por (1,1) tipo_logradouro
- Telefone: Residencial/fixo → tipo de telefone
  - alunos (1,n) possui (0,1) telefone_aluno (0,1) determinado (1,1)tipo_telefone

Tabelas Alteradas/Novas:

- Aluno / aluno
  - Cadastro aluno/ id_aluno / int / PK
  - Nome / nome_aluno / varchar (20) / NN
  - Sobrenome / sobrenome_aluno / varchar (20) / NN
  - Nascimento / nasc_aluno / date / NN
  - Email / email_aluno / varchar (40) / NULL
  - Idioma / id_idioma / int / FK / NN

- Endereço do Aluno / endereco_aluno:
  - id_endereco_aluno / int / PK
  - id_aluno / int / FK / NN
  - id_tipo_logradouro / int / FK / NN
  - logradouro / varchar(50) / NN
  - numero / int / NN
  - complemento / varchar (20) / NULL
  - bairro / varchar (20) / NULL
  - cep / char(8) / NN
  - cidade / varchar (30) / NULL

- Tipo de Logradouro / tipo_logradouro
  - Código do tipo de logradouro: id_tipo_logradouro / id / PK
  - Nome do tipo de logradouro: tipo_logradouro / varchar (10) / NN

- Telefone do Aluno / telefone_aluno:
  - id_telefone_aluno / int / PK
  - id_aluno / int / FK / NN
  - id_tipo_telefone / int / FK / NN
  - num_telefone / char(11) / NN

- Tipo de telefone / tipo_telefone
  - Código do tipo de logradouro: id_tipo_telefone/ id / PK
  - Nome do tipo de logradouro: tipo_telefone/ varchar (15) / NN

3FN – Terceira Forma Normal: uma entidade está na terceira forma normal se:

- Estiver na 2FN.
- Não existem dependências transitivas, ou seja, nenhuma coluna não-chave não pode depender de outra coluna não-chave.

Bairro, Cidade, Estado: interdependentes, mas não iremos alterar nesse momento.

Implementação:

Criar modelo lógico no MySQL Workbench e usar o Forward Engineer.

Inserir dados:

INSERT INTO tabela(*campo_1, campo_2,…*) VALUES (*valor_1, valor_2,…*);

Consultas:

Use JOIN para fazer combinações de tabelas:

JOIN: combina dados de duas ou mais tabelas, de acordo com seu relacionamento.

- INNER JOIN: retorna quando houver ao menos 1 correspondência.
- OUTER JOIN: retorna mesmo que não houver correspondência.
  - LEFT JOIN: retorna dados relacionados de duas tabelas e também os dados não relacionados encontrados na tabela à esquerda da cláusula JOIN.
  - RIGHT JOIN: retorna dados relacionados de duas tabelas e também os dados não relacionados encontrados na tabela à direita da cláusula JOIN.
  - FULL JOIN: todas as linhas de dados da tabela à esquerda de JOIN e da tabela à direita serão retornadas.

Exemplos:

```
SELECT colunas
FROM tabela1
INNER JOIN tabela2
ON tabela1.coluna=tabela2.coluna;

SELECT *
FROM tabela1 AS A
JOIN tabela2 AS B ON A.coluna = B.coluna;
```