Carlos Henrique Hideki Koti da Silva
Gabriel da Silva Pereira
Lucas Soares Fabricio

ETAPA 3
Modelagem e Implementação de Banco de Dados

Bauru
2025

Carlos Henrique Hideki Koti da Silva
Gabriel da Silva Pereira
Lucas Soares Fabricio

ETAPA 3
Modelagem e Implementação de Banco de Dados

Professor: Luis Alexandre da Silva.
FATEC BAURU – BANCO DE DADOS

Bauru
2025

Sumário

1 Introdução

O presente relatório documenta a terceira etapa do projeto de desenvolvimento de banco de dados baseado nos microdados do Censo Escolar (INEP). O objetivo principal desta fase foi a transição dos dados brutos e tratados (CSV), provenientes da Etapa 2, para um modelo relacional robusto, implementado em Sistema Gerenciador de Banco de Dados (SGBD).

As atividades descritas englobam a modelagem conceitual e lógica, a aplicação de regras de normalização para garantir a integridade dos dados e o desenvolvimento de scripts automatizados para a carga (população) das tabelas, garantindo eficiência e escalabilidade.

2 Metodologia de Modelagem

A estratégia de modelagem adotada partiu da análise dos dados tratados na Etapa 2, visando a eliminação de redundâncias e inconsistências. O processo seguiu as diretrizes da arquitetura relacional clássica.

2.1 Normalização

Para assegurar a eficiência do armazenamento e a integridade referencial, o modelo foi submetido às regras de normalização até a **Terceira Forma Normal (3FN)**.

- **1ª Forma Normal (1FN):** Garantida a atomicidade dos valores e a existência de chave primária.
- **2ª Forma Normal (2FN):** Todos os atributos não-chave dependem totalmente da chave primária.
- **3ª Forma Normal (3FN):** Eliminação de dependências transitivas.

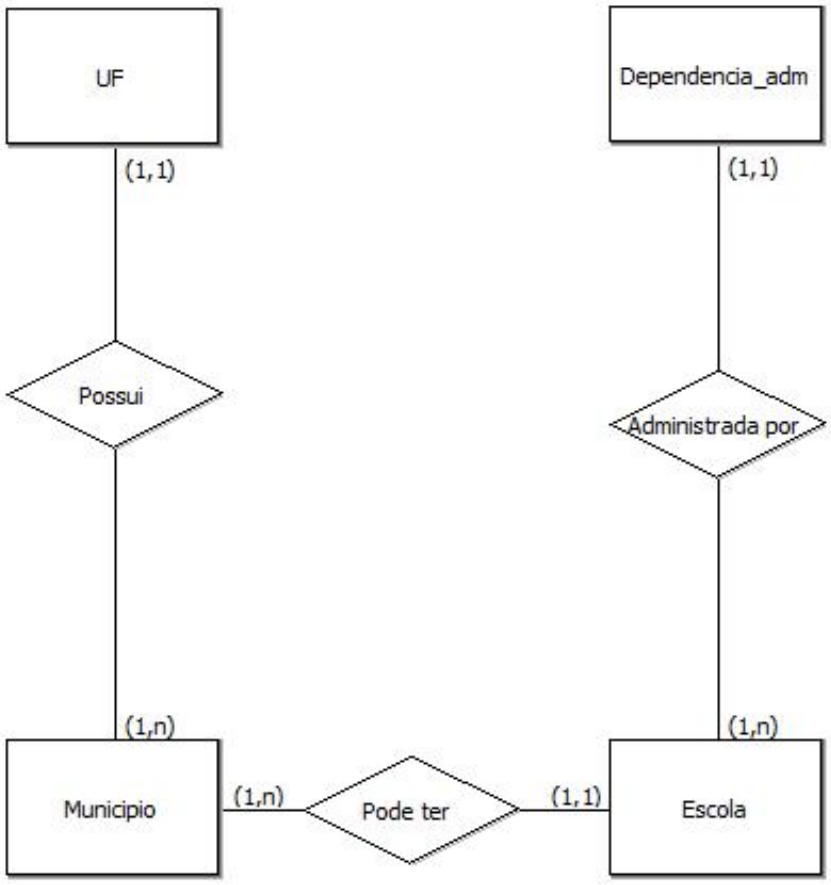
Aplicação Prática no Projeto: No arquivo original, atributos como `nome_municipio`, `estado` e `dependencia_adm` repetiam-se textual e exaustivamente para cada registro de escola. Para satisfazer a 3FN e otimizar o armazenamento, estes dados foram segregados em tabelas de domínio (*lookup tables*), sendo referenciados por chaves estrangeiras na tabela fato (`escola`).

3 Modelo de Dados

3.1 Modelo Conceitual (Diagrama ER)

O diagrama Entidade-Relacionamento (ER) foi estruturado com as entidades UF, Município, Dependência Administrativa e Escola. A visualização gráfica das entidades e seus relacionamentos é apresentada na Figura 1.

Figura 1 – Diagrama Entidade-Relacionamento (Conceitual)

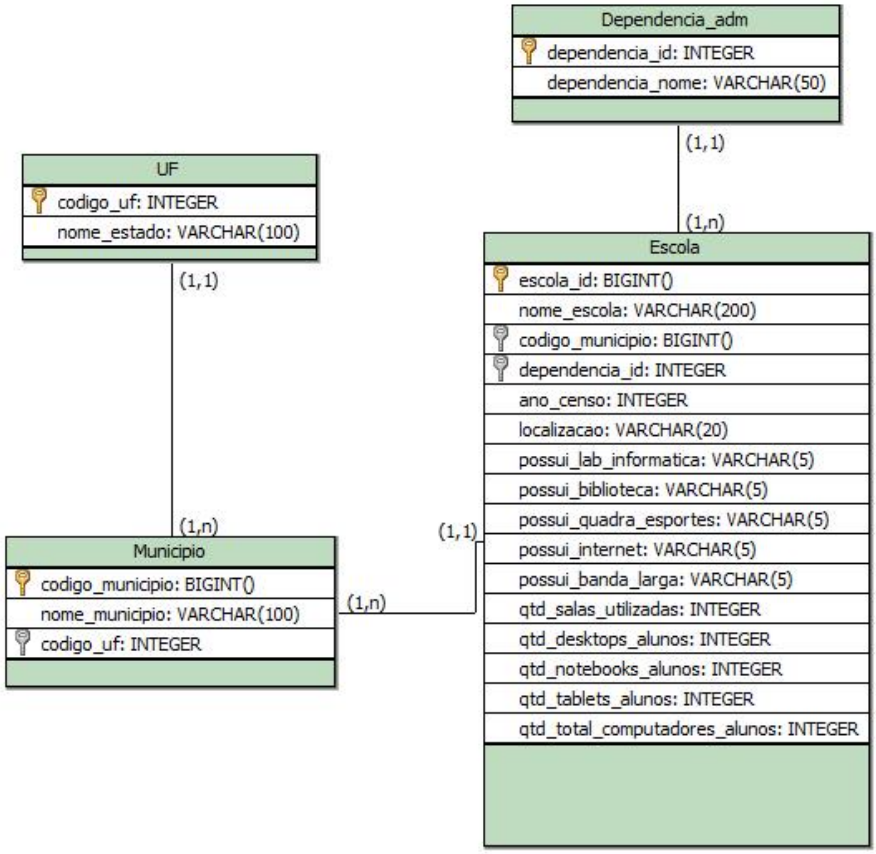


Fonte: Elaborado pelos autores (2025).

3.2 Modelo Lógico e DDL

O esquema lógico relacional (Figura 2) detalha a estrutura das tabelas, chaves primárias (PK) e estrangeiras (FK).

Figura 2 – Diagrama Lógico (Esquema Relacional)



Fonte: Elaborado pelos autores (2025).

Abaixo apresenta-se o script de definição de dados (DDL) desenvolvido para o SGBD MySQL.

Listing 3.1 – Script DDL de Criação do Banco

```sql
-- 1. Tabela UF
CREATE TABLE uf (
  codigo_uf INT NOT NULL,
  nome_estado VARCHAR(100) NOT NULL,
  PRIMARY KEY (codigo_uf)
);

-- 2. Tabela Depend ncia
CREATE TABLE dependencia_adm (
  dependencia_id INT NOT NULL,
  nome_dependencia VARCHAR(50) NOT NULL,
  PRIMARY KEY (dependencia_id)
);
```

```
14
15 -- 3. Tabela Munic pio
16 CREATE TABLE municipio (
17   codigo_municipio BIGINT NOT NULL,
18   nome_municipio VARCHAR(150) NOT NULL,
19   codigo_uf INT NOT NULL,
20   PRIMARY KEY (codigo_municipio),
21   KEY idx_municipio_uf (codigo_uf),
22   CONSTRAINT fk_municipio_uf FOREIGN KEY (codigo_uf) REFERENCES uf(
       codigo_uf)
23 );
24
25 -- 4. Tabela Escola (Fato)
26 CREATE TABLE escola (
27   escola_id BIGINT NOT NULL AUTO_INCREMENT,
28   ano_censo INT NOT NULL,
29   nome_escola VARCHAR(255) NOT NULL,
30   codigo_municipio BIGINT NOT NULL,
31   dependencia_id INT NOT NULL,
32   localizacao VARCHAR(20),
33   -- Infraestrutura
34   possui_lab_informatica VARCHAR(5),
35   possui_biblioteca VARCHAR(5),
36   possui_quadra_esportes VARCHAR(5),
37   possui_internet VARCHAR(5),
38   possui_banda_larga VARCHAR(5),
39   -- M tricas
40   qtd_salas_utilizadas INT DEFAULT 0,
41   qtd_desktops_alunos INT DEFAULT 0,
42   qtd_notebooks_alunos INT DEFAULT 0,
43   qtd_tablets_alunos INT DEFAULT 0,
44   qtd_total_computadores_alunos INT DEFAULT 0,
45   PRIMARY KEY (escola_id),
46   KEY idx_escola_municipio (codigo_municipio),
47   CONSTRAINT fk_escola_municipio FOREIGN KEY (codigo_municipio)
48       REFERENCES municipio(codigo_municipio),
49   CONSTRAINT fk_escola_dependencia FOREIGN KEY (dependencia_id)
50       REFERENCES dependencia_adm(dependencia_id)
51 );
```

4 Processo de Carga (ETL)

Dada a volumetria dos dados, a inserção manual seria inviável e propensa a erros. Desenvolveu-se um script em Python utilizando a biblioteca `Pandas` para automatizar a geração dos comandos SQL (`INSERT`).

4.1 Estratégia de Automação

O script realiza as seguintes operações:

1. Leitura do CSV tratado na Etapa 2.
2. Extração de valores únicos para povoar as tabelas auxiliares (`uf`, `municipio`, `dependencia_adm`).
3. Mapeamento lógico (*Mapping*) para substituir os textos de dependência administrativa pelos seus respectivos IDs gerados.
4. Geração de um arquivo `.sql` contendo todos os comandos de inserção, garantindo a ordem correta para não violar restrições de chave estrangeira.

Listing 4.1 – Script de Geracao de Carga (ETL) - Adaptado

```
import pandas as pd

# Carregando o CSV tratado da Etapa 2
arquivo_csv = "dados_censo_escolar_tratados.csv"
df = pd.read_csv(arquivo_csv)

# Funcao para tratar aspas simples em nomes (ex: D'Oeste)
def esc(valor):
    if pd.isna(valor): return ""
    return str(valor).replace("'", "''")

print("Iniciando geracao dos scripts SQL...")

# ------------------------------------------------------------
# 1. GERACAO DE IDs PARA DEPENDENCIA (Transformation)
# ------------------------------------------------------------
# Extrai as dependencias unicas do CSV (Estadual, Municipal, etc.)
dep_unicas = df['dependencia_adm'].unique()
# Cria um dicionario para mapear Texto -> ID (Ex: 'Estadual': 1)
map_dep = {nome: i+1 for i, nome in enumerate(dep_unicas)}

# ------------------------------------------------------------
```

```
# 2. ESCRITA DOS ARQUIVOS SQL (Load)
# ------------------------------------------------------------

# A) Tabela UF (Usa o codigo_uf ja existente no CSV)
ufs = df[['codigo_uf', 'estado']].drop_duplicates()
with open("insert_uf.sql", "w", encoding="utf-8") as f:
    for _, row in ufs.iterrows():
        f.write(f"INSERT INTO uf (codigo_uf, nome_estado) VALUES "
                f"({row['codigo_uf']}, '{esc(row['estado'])}');\n")

# B) Tabela Dependencia (Usa o ID gerado pelo dicionario)
with open("insert_dependencia.sql", "w", encoding="utf-8") as f:
    for nome, id_new in map_dep.items():
        f.write(f"INSERT INTO dependencia_adm (dependencia_id,
    nome_dependencia) "
                f"VALUES ({id_new}, '{esc(nome)}');\n")

# C) Tabela Municipio (Usa codigo_municipio existente)
muns = df[['codigo_municipio', 'municipio', 'codigo_uf']].
    drop_duplicates()
with open("insert_municipio.sql", "w", encoding="utf-8") as f:
    for _, row in muns.iterrows():
        f.write(f"INSERT INTO municipio (codigo_municipio,
    nome_municipio, codigo_uf) "
                f"VALUES ({row['codigo_municipio']}, '{esc(row['
    municipio'])}', {row['codigo_uf']});\n")

# D) Tabela Escola (Tabela Fato)
# - Mapeia a dependencia para ID
# - Nao insere escola_id (deixa o banco gerar via Auto Increment)
with open("insert_escola.sql", "w", encoding="utf-8") as f:
    for _, row in df.iterrows():
        # Busca o ID numerico correspondente ao texto da dependencia
        id_dep_fk = map_dep[row['dependencia_adm']]

        # Monta o INSERT
        f.write(
            "INSERT INTO escola (ano_censo, nome_escola,
    codigo_municipio, "
            "dependencia_id, localizacao, possui_lab_informatica, "
            "possui_biblioteca, possui_quadra_esportes, possui_internet,
     "
            "possui_banda_larga, qtd_salas_utilizadas,
    qtd_total_computadores_alunos) "
            "VALUES ("
            f"{row['ano_censo']}, '{esc(row['nome_escola'])}', "
            f"{row['codigo_municipio']}, {id_dep_fk}, "
```

```
63              f"'{esc(row['localizacao'])}', '{esc(row['
    possui_lab_informatica'])}', "
64              f"'{esc(row['possui_biblioteca'])}', '{esc(row['
    possui_quadra_esportes'])}', "
65              f"'{esc(row['possui_internet'])}', '{esc(row['
    possui_banda_larga'])}', "
66              f"{int(row['qtd_salas_utilizadas'])}, {int(row['
    qtd_total_computadores_alunos'])});\n"
67          )
68
69 print("Processo concluido! Arquivos .sql gerados.")
```

5 Validação e Resultados

Após a execução dos scripts no SGBD, foram realizados testes de validação para assegurar a consistência do banco de dados:

- **Integridade Referencial:** Tentativas de inserir escolas com códigos de município inexistentes ou códigos de dependência inválidos foram rejeitadas pelo banco, confirmando o funcionamento correto das *Foreign Keys*.
- **Consistência de Dados:** Consultas SQL de agregação (ex: `COUNT(*)` agrupado por estado) retornaram valores idênticos aos do CSV original, validando que não houve perda de dados durante a migração.
- **Eficiência:** A criação de índices nas colunas `codigo_municipio` e `dependencia_id` otimizou significativamente o tempo de resposta em junções (*JOINs*) durante as consultas de teste.

6 Conclusão

A conclusão da Etapa 3 marca um ponto crucial no projeto da disciplina. A transformação de arquivos planos (CSV) para um modelo relacional normalizado (3FN) resultou em uma estrutura de dados mais segura, escalável e alinhada às boas práticas de engenharia de dados.

A implementação bem-sucedida das tabelas de domínio e o uso rigoroso de chaves estrangeiras garantem que as próximas etapas de visualização e divulgação (Etapa 4) consumirão dados consistentes e íntegros, permitindo a geração de *insights* confiáveis sobre a infraestrutura escolar brasileira.