Projeto Lógico de Banco de Dados

Luis Felipe Ramalho Carvalho

Universidade Federal de Sergipe - Prof. André Britto

1. Projeto Lógico – Modelo NoSQL

1.1 Introdução

Nesta etapa do trabalho, foi realizado o mapeamento do projeto conceitual previamente desenvolvido para um modelo de dados NoSQL. O objetivo desta fase é estudar e aplicar os conceitos de bancos de dados NoSQL, analisando suas características, diferenças em relação ao modelo relacional e formas de representação de entidades e relacionamentos definidos no DER.

Para esta implementação, foi escolhido um Sistema Gerenciador de Banco de Dados NoSQL orientado a documentos, permitindo maior flexibilidade estrutural e melhor adaptação a cenários com dados semiestruturados.

1.2 Pesquisa e Estudo do SGBD NoSQL

Após pesquisa e análise de diferentes SGBDs NoSQL, como Cassandra, Redis, DynamoDB e MongoDB, o grupo optou pelo **MongoDB**.

O MongoDB é um banco de dados NoSQL orientado a documentos, no qual os dados são armazenados em documentos no formato BSON (Binary JSON). Diferentemente do modelo relacional, o MongoDB não utiliza tabelas e registros, mas sim **collections** e **documents**, possibilitando estruturas flexíveis e hierárquicas.

Entre as principais características do MongoDB, destacam-se:

- Modelo orientado a documentos;
- Esquema flexível;
- Suporte a documentos aninhados;
- Alto desempenho em operações de leitura;
- Facilidade de integração com aplicações modernas.

1.3 Estratégia de Mapeamento para NoSQL

O mapeamento do DER para o modelo NoSQL foi realizado considerando as boas práticas do MongoDB, adotando as seguintes estratégias:

- **Entidades principais** foram mapeadas como **collections**;
- **Relacionamentos fortes** foram representados por **documentos embutidos**;
- **Relacionamentos com atributos próprios** foram representados como **agregações** dentro de um documento principal;
- A herança presente no modelo relacional foi substituída por um atributo discriminador.

1.4 Estruturas Criadas no MongoDB

1.4.1 Collection `usuarios`

A collection `usuarios` representa a entidade Usuário e suas especializações (Usuário Comum, Administrador da Empresa e Administrador do Sistema). A herança do modelo relacional foi substituída pelo atributo `tipo`.

A entidade Empresa foi representada como um **subdocumento embutido**, uma vez que possui forte dependência funcional em relação ao usuário no contexto da aplicação.

Exemplo de estrutura:

```
{
  "_id": ObjectId,
  "nome": "João Silva",
  "email": "joao@email.com",
  "senha": "hash_da_senha",
  "tipo": "usuario_comum",
  "empresa": {
    "nome": "Empresa X",
    "cnpj": "11.111.111/0001-11",
    "quantidade_funcionarios": 50
  },
  "data_criacao": "2026-01-30T10:00:00Z"
}
```

1.4.2 Collection `leads`

A collection `leads` representa a entidade Lead, modelada como uma entidade independente. O endereço foi representado como um documento embutido, aproveitando a flexibilidade do modelo orientado a documentos.

Exemplo de estrutura:

```
{
  "_id": ObjectId,
  "nome": "Empresa Y",
  "email": "contato@empresa.com",
  "cnpj": "22.222.222/0001-22",
  "endereco": {
    "rua": "Rua X",
    "cidade": "Cidade Y",
    "estado": "PB",
    "cep": "00000-000"
  },
  "data_cadastro": "2026-01-30T11:00:00Z"
}
```

1.4.3 Collection `pesquisas` (Relacionamento / Agregação)

A collection `pesquisas` representa a entidade Pesquisa e o relacionamento entre Pesquisa e Lead. No modelo NoSQL, o relacionamento foi representado através de **documentos embutidos**, eliminando a necessidade de tabelas intermediárias, comuns no modelo relacional.

Os atributos do relacionamento, como ordem e relevância, foram armazenados juntamente com cada lead associado à pesquisa.

Exemplo de estrutura:

```
{
  "_id": ObjectId,
  "termo_busca": "clínicas médicas",
  "data_pesquisa": "2026-01-30T12:00:00Z",
  "usuario_id": ObjectId,
  "quantidade_leads": 2,
  "resultados": [
```

```
    {
      "lead_id": ObjectId,
      "ordem": 1,
      "relevancia": 0.95
    },
    {
      "lead_id": ObjectId,
      "ordem": 2,
      "relevancia": 0.82
    }
  ]
}
```

1.5 Garantia das Restrições do Modelo Conceitual

Embora o MongoDB não imponha restrições de integridade da mesma forma que um SGBD relacional, as restrições do projeto conceitual foram garantidas da seguinte forma:

- **Chave Primária (PK):**
  Garantida automaticamente pelo atributo `_id` do MongoDB.

- **Unicidade:**
  Garantida através de índices únicos, como nos atributos `email` e `cnpj`.

- **Chaves Estrangeiras (FK):**
  Representadas por referências através de `ObjectId`, com controle realizado pela aplicação.

- **Integridade Referencial:**
  Garantida pela lógica implementada na aplicação Python.

- **Cardinalidade:**
  Controlada pela estrutura dos documentos e dos arrays embutidos.

- **Ações de exclusão e atualização (ON DELETE / ON UPDATE):**
  Implementadas no nível da aplicação, conforme boas práticas em bancos NoSQL.

1.6 Aplicação CRUD

Foi desenvolvida uma aplicação em Python utilizando a biblioteca `pymongo`, responsável por realizar operações CRUD (Create, Read, Update e Delete) sobre três estruturas do banco:

- `usuarios` (entidade);
- `leads` (entidade);
- `pesquisas` (relacionamento/agregação).

A aplicação permite a inserção, leitura, atualização e remoção de dados, demonstrando o correto funcionamento das estruturas criadas no MongoDB.

1.7 Considerações Finais

A modelagem NoSQL realizada demonstrou-se adequada ao domínio do problema, permitindo maior flexibilidade na representação dos dados e simplificação das consultas. A utilização de documentos embutidos possibilitou a eliminação de junções complexas, comuns no modelo relacional, além de facilitar a implementação das operações CRUD.

Dessa forma, os objetivos da etapa NoSQL do trabalho foram plenamente atendidos, respeitando as restrições do projeto conceitual e explorando as características do SGBD escolhido.