Universidade Federal de Sergipe
Departamento de Computação
Disciplina: Banco de Dados 1

Projeto Lógico - Banco de Dados NoSQL: Steam Ecosystem

Alunos: Rafael Gomes Oliveira Santos
Alícia Vitória Sousa Santos

Professor: André Britto

Resumo

Este relatório apresenta o projeto lógico de um sistema de ecossistema de jogos (Steam) mapeado para o SGBD NoSQL MongoDB. Contempla a fundamentação da escolha tecnológica, a estratégia de mapeamento de 17 unidades lógicas (tabelas e relações) para documentos, os schemas JSON de validação implementados e a discussão sobre como as restrições de integridade foram tratadas no novo paradigma.

São Cristóvão
9 de fevereiro de 2026

1 Introdução

O objetivo deste trabalho é apresentar o projeto lógico, no modelo NoSQL, de um sistema de gerenciamento de loja de jogos originalmente modelado em DER e implementado em ambiente relacional (PostgreSQL). A migração do modelo relacional para um modelo orientado a documentos exige decisões conscientes sobre embutimento (*embedding*) de dados, uso de referências (*referencing*) e a definição de onde garantir as restrições de integridade, visando otimizar a performance de leitura e a localidade de dados.

2 Pesquisa e escolha do SGBD

Pela simplicidade, compatibilidade nativa com documentos JSON e alta escalabilidade, selecionamos o **MongoDB** para a modelagem. No contexto do Teorema CAP, o MongoDB é classificado como um sistema **CP** (Consistente e Tolerante a Partições), garantindo que todos os nós visualizem os mesmos dados simultaneamente, o que é fundamental para a gestão de ativos financeiros (Carteira) e licenças de software. Além disso, sua natureza *schema-less* favorece a representação de polimorfismo entre diferentes tipos de produtos (Jogos, Softwares e DLCs).

3 Estratégia de mapeamento Relacional → NoSQL

As decisões estratégicas adotadas para a consolidação das 17 tabelas em 4 coleções foram:

- **Entidades independentes:** As entidades principais, como `usuarios`, `produtos`, `compras` e `avaliacoes`, foram traduzidas diretamente como coleções raiz.
- **Embutir documentos (Embedding):** Dados que possuem forte dependência existencial ou que são consultados de forma agregada foram embutidos. Exemplos incluem a `Carteira` e a relação N:M `Biblioteca_has_Produto` dentro de usuários, e a `Nota_Fiscal` dentro de compras.
- **Polimorfismo:** Unificamos as tabelas de especialização (*Jogos, Software, DLC*) na coleção `produtos`, utilizando um discriminador de tipo para diferenciar os atributos específicos de cada um.
- **Referências (ObjectId):** Traduzimos a noção de chaves estrangeiras transacionais como referências. A coleção `compras`, por exemplo, referencia `id_usuario` e `id_produto`.
- **Chaves Naturais:** Seguindo o refinamento do modelo físico, utilizamos nomes (Strings) como identificadores únicos para `desenvolvedora` e `publicadora`.

Tabela SQL (Origem)	Coleção NoSQL (Destino)	Estratégia
Usuario, Carteira	`usuarios`	Embedding (1:1)
Biblioteca, Bibl_has_Produto	`usuarios`	Embedding (Array de Itens)
Produto, Jogos, Software, DLC	`produtos`	Polimorfismo / Unificação
Genero, Jogos_has_Genero	`produtos`	Denormalização (Array)
Conquista, RequisitoSistema	`produtos`	Embedding (Atomicidade)
Desenvolvedora, Publicadora	`produtos`	Extended Reference (Nomes)
Compra, Nota_Fiscal	`compras`	Embedding 1:1 e Linking
Avaliacao	`avaliacoes`	Referencing (ID Linking)

Tabela 1: Rastreabilidade das 17 Tabelas Relacionais para as 4 Coleções NoSQL.

4 Discussão

A migração do modelo relacional para o orientado a documentos envolveu *trade-offs* significativos:

- **Vantagens:** Consultas de perfil e catálogo tornaram-se mais rápidas, pois a localidade de dados elimina *Joins* complexos. O uso de *Arrays* para idiomas e gêneros otimiza a filtragem.
- **Desvantagens:** Ausência de restrições de integridade referencial rígidas (como o *Restrict*). Regras de integridade complexas e validações de tipos foram garantidas através do **$jsonSchema** no SGBD e devem ser complementadas na camada de aplicação.

5 Integridade e Restrições de Esquema ($jsonSchema)

Implementamos validações estritas para garantir a consistência estrutural.

5.1 Coleção: Usuarios

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["nome_completo", "email", "senha", "nivel", "carteira",
        "biblioteca"],
    "additionalProperties": false,
    "properties": {
      "_id": { "bsonType": "objectId" },
      "nome_completo": { "bsonType": "string", "maxLength": 100 },
      "email": { "bsonType": "string", "pattern": "^.+@.+$" },
      "senha": { "bsonType": "string", "maxLength": 32 },
      "nivel": { "bsonType": "int", "minimum": 1 },
```

```
      "carteira": {
        "bsonType": "object",
        "required": ["moeda", "saldo_atual"],
        "additionalProperties": false,
        "properties": {
          "moeda": { "bsonType": "string" },
          "saldo_atual": { "bsonType": "double", "minimum": 0 }
        }
      },
      "biblioteca": {
        "bsonType": "array",
        "items": {
          "bsonType": "object",
          "required": ["produto_id", "status_instalacao"],
          "additionalProperties": false,
          "properties": {
            "produto_id": { "bsonType": "objectId" },
            "tempo_jogado": { "bsonType": ["string", "null"] },
            "status_instalacao": { "bsonType": "bool" }
          }
        }
      }
    }
  }
}
```

5.2 Coleção: Produtos

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["tipo", "titulo", "preco", "data_lancamento", "generos"
        , "idiomas", "desenvolvedora", "publicadora"],
    "additionalProperties": false,
    "properties": {
      "tipo": { "enum": ["Jogo", "Software", "DLC"] },
      "desenvolvedora": { "bsonType": "string" },
      "publicadora": { "bsonType": "string" },
      "conquistas": {
        "bsonType": "array",
        "items": {
          "bsonType": "object",
          "required": ["titulo", "descricao", "imagem_icone"],
          "properties": {
            "imagem_icone": { "bsonType": "binData" }
          }
```

```
18            }
19          }
20        }
21    }
22 }
```

5.3 Coleção: Compras e Avaliações

Para compras, embutimos a nota fiscal. Para avaliações, permitimos campos nulos (*texto_analise* e *data_postagem*) conforme definido no modelo físico.

6 Conclusão

As decisões de modelagem e as validações via *schema* garantem que a flexibilidade do NoSQL não comprometa a integridade dos dados essenciais. A consolidação em documentos agregados respeita o paradigma orientado a documentos, delegando a gestão de relacionamentos transversais à camada de aplicação ou a índices secundários.