
Universidade Federal de Sergipe
Departamento de Computação
Campus São Cristóvão – UFS

# Projeto Lógico - NoSQL

Discentes:
CÉZAR AUGUSTO NASCIMENTO DIAS
GABRIEL RODRIGUES DOS SANTOS

Docente:
ANDRE BRITTO DE CARVALHO

São Cristóvão – SE
Fevereiro de 2026

# 1 Relatório

Este relatório apresenta a etapa de projeto lógico em NoSQL a partir do DER desenvolvido anteriormente. O trabalho foi organizado em três partes: pesquisa e escolha do SGBD, mapeamento das estruturas para o modelo NoSQL e validação das restrições definidas no projeto conceitual.

## 1.1 Pesquisa e escolha do SGBD

Foram analisados modelos NoSQL de chave-valor, documentos e grafos. O modelo de chave-valor prioriza acesso rápido por chave; o modelo de grafos é mais indicado para cenários com relacionamentos altamente conectados; e o modelo de documentos oferece maior flexibilidade para dados semiestruturados.

Considerando o domínio do projeto (futebol), o MongoDB foi escolhido por permitir modelagem orientada a documentos, uso combinado de referências e subdocumentos, consultas por agregação para recuperação de relacionamentos e evolução de esquema com menor rigidez estrutural.

| SQL | MongoDB | DynamoDB | Cassandra | Couchbase |
|---|---|---|---|---|
| Tabela | Coleta | Tabela | Tabela | Bucket de dados |
| Linha | Documento | Item | Linha | Documento |
| Coluna | Campo | Atributo | Coluna | Campo |
| Chave primária | ObjectId | Chave primária | Chave primária | ID do documento |
| Índice | Índice | Índice secundário | Índice | Índice |
| Visualização | Visualização | Índice secundário global | Visualização materializada | Visualização |
| Tabela ou objeto aninhado | Documento incorporado | Mapa | Mapa | Mapa |
| Matriz | Matriz | Lista | Lista | Lista |

Figura 1: Comparação geral entre SQL e abordagens NoSQL.

## 1.2 Mapeamento do DER para o MongoDB

O mapeamento foi conduzido com três diretrizes principais: uso de coleções para entidades principais, referências por identificadores para relacionamentos entre entidades

e subdocumentos para dados dependentes da entidade principal.

### 1.2.1 Método de mapeamento adotado

Para manter consistência no processo de conversão, foram adotadas as seguintes regras:

- cada entidade forte do DER foi mapeada para uma coleção própria;
- relacionamentos 1:N foram representados por campos `id_*` no lado N;
- relacionamentos N:M sem atributos próprios foram representados por vetores de IDs;
- relacionamentos N:M com atributos próprios podem ser representados por coleção associativa;
- especialização/generalização foi implementada por coleções de subtipos;
- atributos compostos foram representados como subdocumentos;
- atributos multivalorados foram representados como arrays.

No banco `futebol_db`, foram criadas as coleções `Federacao`, `Campeonato`, `Clube`, `Contrato`, `Estadio`, `Executivo`, `Jogador`, `Partida`, `Tecnico`, `Temporada` e `Arbitro`.

A especialização de Pessoa foi implementada por meio de coleções separadas (`Jogador`, `Tecnico`, `Arbitro` e `Executivo`), sem criação de uma coleção única `Pessoa`. Os relacionamentos 1:N foram representados por campos `id_*`. O relacionamento entre árbitro e partida foi representado por `arbitros_ids` em `Partida`. A súmula e seus eventos foram modelados como subdocumentos em `Partida.sumula.eventos`.

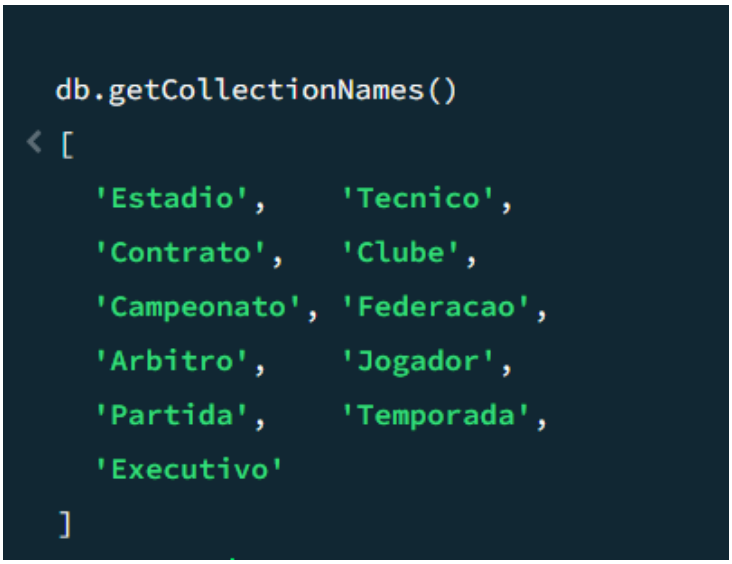


Figura 2: Coleções criadas no banco `futebol_db`.

```
> db.Partida.findOne({ id_partida: 5001 }, { _id: 0, id_partida: 1, sumula: 1, arbitros_ids: 1 })
< {
    id_partida: 5001,
    arbitros_ids: [
      44,
      45,
      46
    ],
    sumula: {
      hora_inicio: '20:03',
      hora_fim: '21:56',
      eventos: [
        {
          id_evento: 90001,
          descricao: 'Gol',
          minuto_ocorrido: 18,
          id_jogador: 101
        },
        {
          id_evento: 90002,
          descricao: 'Cartao amarelo',
          minuto_ocorrido: 71,
          id_jogador: 101
        }
      ]
    }
  }
```

Figura 3: Documento de partida com subdocumento de súmula e lista de eventos.

```
> db.Partida.aggregate([
    { $match: { id_partida: 5001 } },
    { $lookup: { from: "Clube", localField: "id_clube_mandante", foreignField: "id_clube", as: "mandante" } },
    { $lookup: { from: "Clube", localField: "id_clube_visitante", foreignField: "id_clube", as: "visitante" } },
    { $project: {_id: 0,id_partida: 1,mandante: { $arrayElemAt: ["$mandante.nome", 0] },visitante: { $arrayElemAt: ["$visitante.nome", 0] },placar_ma
    }}
  ])
< {
    id_partida: 5001,
    placar_mandante: 2,
    placar_visitante: 1,
    mandante: 'Lagarto',
    visitante: 'Confiança'
  }
```

Figura 4: Agregação com `$lookup` para recuperar mandante e visitante na partida.

```
            in: { id_arbitro: "$$a.id_arbitro", nome: "$$a.primeiro_nome", sobrenome: "$$a.sobrenome" }
          }
        }
    }}
  ])
< {
    id_partida: 5001,
    arbitros: [
      {
        id_arbitro: 44,
        nome: 'Wilton',
        sobrenome: 'Pereira'
      },
      {
        id_arbitro: 45,
        nome: 'Edina',
        sobrenome: 'Alves'
      },
      {
        id_arbitro: 46,
        nome: 'Anderson',
        sobrenome: 'Daronco'
      }
    ]
  }
```

Figura 5: Agregação para recuperar dados dos árbitros a partir de `arbitros_ids`.

## 1.3 Restrições do modelo e validação

As restrições foram tratadas, dentro dos limites do MongoDB, com foco em três pontos:

- unicidade por meio de índices únicos em campos como `id_jogador`, `cpf`, `licenca`, `id_partida`, `id_clube`, `cnpj` e `id_contrato`;
- integridade referencial por uso consistente de IDs e validações na aplicação;
- domínio de atributos respeitando tipos de dados (texto, número, data e vetor).

A validação prática incluiu a consulta dos índices criados e o teste de inserção duplicada, que retornou erro `E11000`, confirmando a restrição de unicidade.

```
> db.Jogador.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { id_jogador: 1 }, name: 'id_jogador_1', unique: true },
    { v: 2, key: { cpf: 1 }, name: 'cpf_1', unique: true },
    { v: 2, key: { licenca: 1 }, name: 'licenca_1', unique: true }
  ]
> db.Partida.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { id_partida: 1 }, name: 'id_partida_1', unique: true }
  ]
> db.Clube.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { id_clube: 1 }, name: 'id_clube_1', unique: true },
    { v: 2, key: { cnpj: 1 }, name: 'cnpj_1', unique: true }
  ]
> db.Contrato.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { id_contrato: 1 }, name: 'id_contrato_1', unique: true }
  ]
```

Figura 6: Índices existentes nas coleções para garantir unicidade.

```
> db.Jogador.insertOne({
    id_jogador: 101,
    licenca: 50001,
    posicao: "Atacante",
    altura: 181,
    peso: 77,
    primeiro_nome: "Caça",
    sobrenome: "Rato",
    data_nascimento: new Date("2004-07-01"),
    cpf: "12345678901",
    telefone: ["79999990000", "79999991111"],
    id_federacao: 1
  })

MongoServerError: E11000 duplicate key error collection: futebol_db.Jogador index: id_jogador_1 dup key: { id_jogador: 101 }
futebol_db >
```

Figura 7: Erro de chave duplicada ao tentar inserir `id_jogador` já existente.

### 1.3.1 Limitações do SGBD e garantias adotadas

O MongoDB não impõe automaticamente todas as restrições do modelo conceitual com o mesmo rigor de um SGBD relacional, sobretudo em integridade referencial forte e cardinalidades mínimas. Para reduzir esse impacto, o projeto combina índices únicos, padronização da estrutura dos documentos e validações no nível da aplicação antes de inserções e atualizações.

# 2 Conclusão

A etapa foi concluída com o mapeamento do DER para o modelo NoSQL em MongoDB, com representação das entidades e dos relacionamentos do domínio de futebol. As estruturas foram implementadas e testadas com dados de exemplo. As restrições foram discutidas e validadas por índices únicos e por testes práticos no ambiente de execução.

# Material complementar

Pasta do projeto no Google Drive (contém o arquivo `futebol_db.json`, com a exportação dos dados do banco `futebol_db` em formato JSON, juntamente com as coleções de cada entidade utilizada no projeto):

https://drive.google.com/drive/folders/1AU0AC5IpNoRW8T1M6gsDfXDCXiSwRXCj?usp=drive_link

# Outros modelos

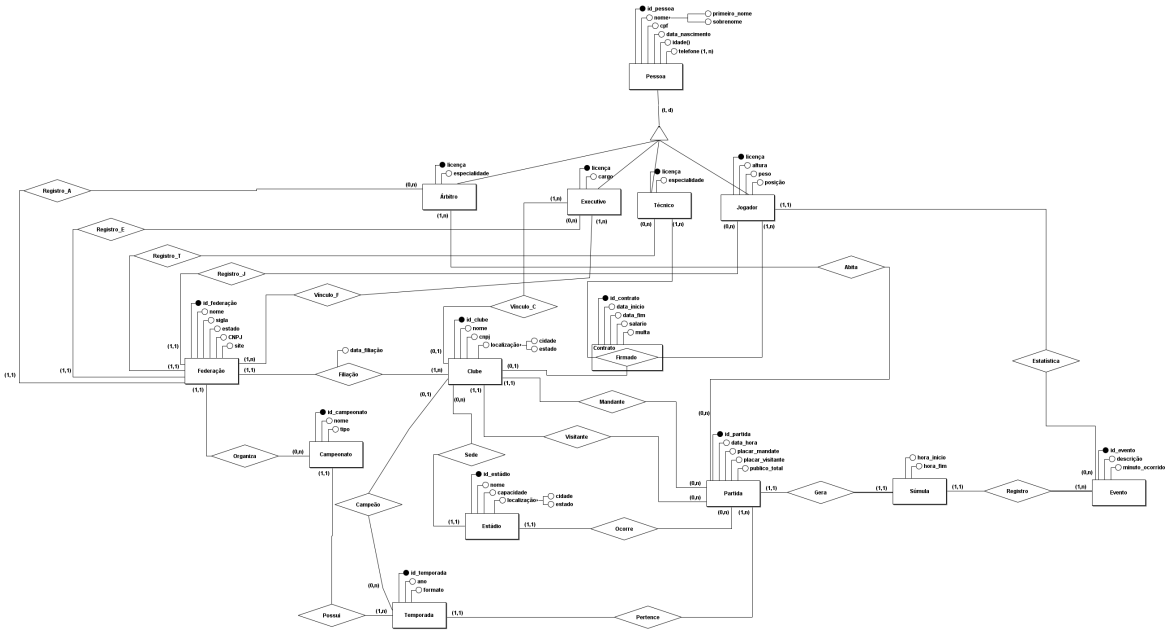


Figura 8: Modelo conceitual (DER) utilizado como base para o projeto.

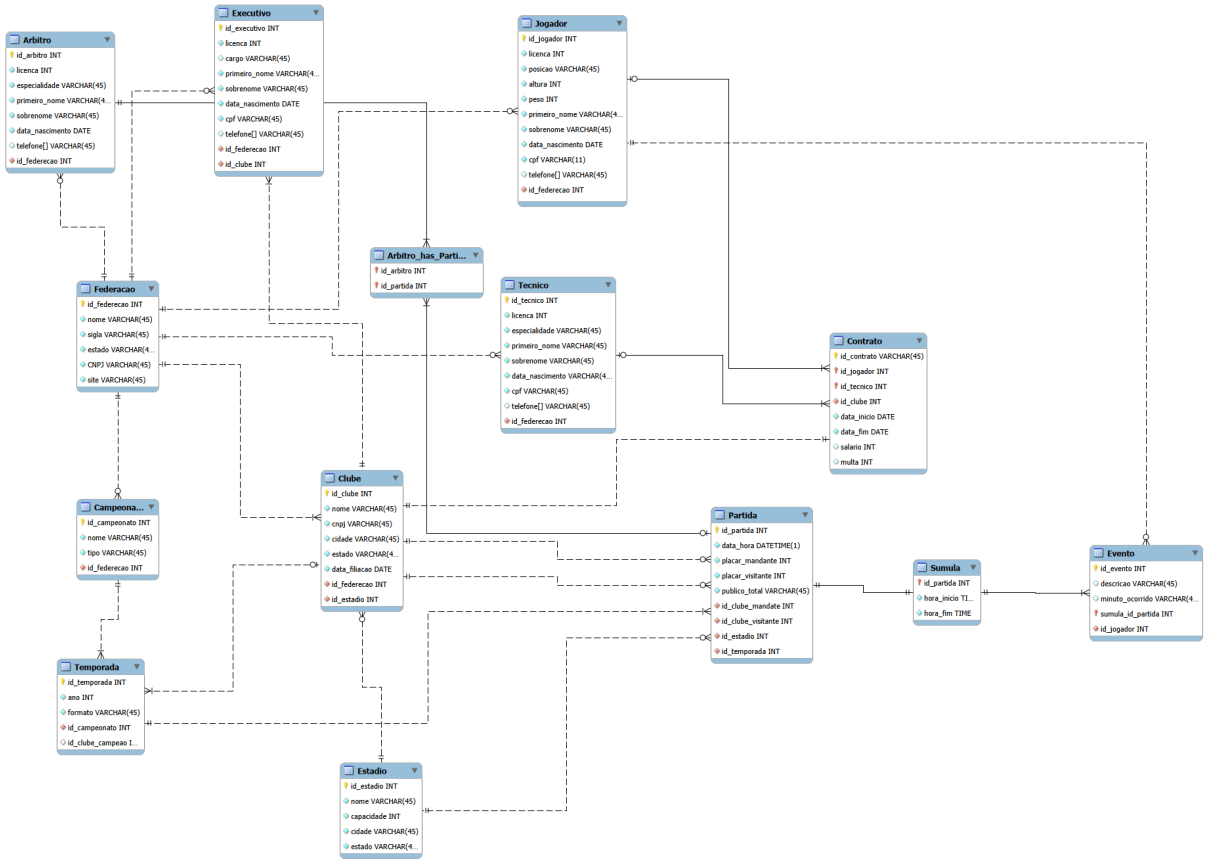


Figura 9: Modelo lógico relacional derivado do modelo conceitual.

# Referências

[1] MONGODB. Documentação MongoDB. Disponível em: https://www.mongodb.com/pt-br/docs/. Acesso em: 09 fev. 2026.

[2] AMAZON WEB SERVICES (AWS). NoSQL. Disponível em: https://aws.amazon.com/pt/nosql/. Acesso em: 09 fev. 2026.

[3] IBM. MongoDB. Disponível em: https://www.ibm.com/br-pt/think/topics/mongodb. Acesso em: 09 fev. 2026.