Guia Explicativo: Implementação e Consultas MongoDB

Este guia explica como realizar as etapas de implementação (A2) e consultas (A3) no MongoDB, utilizando o esquema não-relacional definido na primeira parte do trabalho.

A2. Implementação do Banco de Dados Não-Relacional

A implementação envolve a criação do banco de dados, a aplicação do *Schema Validation* nas coleções e a inserção dos dados.

1. Configuração no MongoDB Atlas (A2.1)

A primeira etapa é criar um cluster no **MongoDB Atlas**, que é a plataforma de banco de dados como serviço (DBaaS) do MongoDB.

1. **Criação do Cluster:** Acesse o MongoDB Atlas, crie uma conta (se necessário) e siga o assistente para criar um novo *Cluster*. Escolha a opção *M0 Sandbox* (gratuita) para fins de estudo.
2. **Configuração de Acesso:** Configure o *IP Access List* para permitir conexões do seu endereço IP e crie um *Database User* com permissões de leitura e escrita.
3. **Conexão:** Obtenha a *Connection String* (string de conexão) do seu cluster. Ela será usada para conectar o *MongoDB Shell* (`mongosh`) ou o *MongoDB Compass*.

2. Criação das Coleções com Schema Validation (A2.2)

O *Schema Validation* é crucial para garantir que os documentos inseridos nas coleções `pedidos` e `produtos` sigam a estrutura definida (JSON Schema).

Comando para Criar o Banco de Dados e a Coleção `produtos` :

No `mongosh` , após conectar-se ao seu cluster:

JavaScript

```
// 1. Conecte-se ao seu banco de dados (substitua 'BD2_Projeto2' pelo nome que você escolheu)
use BD2_Projeto2

// 2. Crie a coleção 'produtos' aplicando o Schema Validation
db.createCollection("produtos", {
   validator: {
      $jsonSchema: {
```

```
         bsonType: "object",
         required: ["id_produto_original", "nome", "quantidade_estoque",
"preco_unitario_atual", "categoria"],
         properties: {
            "id_produto_original": { "bsonType": "int" },
            "nome": { "bsonType": "string" },
            "quantidade_estoque": { "bsonType": "int", "minimum": 0 },
            "preco_unitario_atual": { "bsonType": "double", "minimum": 0 },
            "categoria": {
               "bsonType": "object",
               "required": ["id_categoria", "nome"],
               "properties": {
                  "id_categoria": { "bsonType": "int" },
                  "nome": { "bsonType": "string" }
               }
            }
         }
      }
   }
})
```

Comando para Criar a Coleção `pedidos`:

JavaScript

```
// Crie a coleção 'pedidos' aplicando o Schema Validation
db.createCollection("pedidos", {
   validator: {
      $jsonSchema: {
         bsonType: "object",
         required: ["data_pedido", "status", "cliente", "endereco_entrega",
"itens"],
         properties: {
            "data_pedido": { "bsonType": "date" },
            "status": { "bsonType": "string", "enum": ["Em processamento",
"Enviado", "Entregue", "Cancelado"] },
            "valor_total": { "bsonType": "double", "minimum": 0 },
            "cliente": {
               "bsonType": "object",
               "required": ["id_cliente", "nome", "tipo", "documento"],
               "properties": {
                  "id_cliente": { "bsonType": "int" },
                  "nome": { "bsonType": "string" },
                  "tipo": { "bsonType": "string", "enum": ["Física",
"Jurídica"] },
                  "documento": { "bsonType": "string" },
                  "telefone": {
                     "bsonType": "array",
```

```
                    "items": {
                       "bsonType": "object",
                       "required": ["numero"],
                       "properties": {
                          "numero": { "bsonType": "string" }
                       }
                    }
                 }
              }
           },
           "endereco_entrega": {
              "bsonType": "object",
              "required": ["logradouro", "numero", "cidade", "estado"],
              "properties": {
                 "logradouro": { "bsonType": "string" },
                 "numero": { "bsonType": "string" },
                 "cidade": { "bsonType": "string" },
                 "estado": { "bsonType": "string" }
              }
           },
           "itens": {
              "bsonType": "array",
              "minItems": 1,
              "items": {
                 "bsonType": "object",
                 "required": ["id_produto", "nome_produto", "quantidade",
"preco_unitario"],
                 "properties": {
                    "id_produto": { "bsonType": "int" },
                    "nome_produto": { "bsonType": "string" },
                    "quantidade": { "bsonType": "int", "minimum": 1 },
                    "preco_unitario": { "bsonType": "double", "minimum": 0 }
                 }
              }
           }
        }
     }
  }
})
```

3. Inserção de Dados (A2.3)

A inserção de dados deve ser feita utilizando o método `insertMany()` para inserir um conjunto de documentos de uma só vez.

Exemplo de Inserção na Coleção `produtos`:

JavaScript

```
db.produtos.insertMany([
   {
      "id_produto_original": 101,
      "nome": "Smartphone X",
      "descricao": "Modelo de última geração.",
      "quantidade_estoque": 50,
      "preco_unitario_atual": 2500.00,
      "categoria": {
         "id_categoria": 1,
         "nome": "Eletrônicos"
      }
   },
   {
      "id_produto_original": 102,
      "nome": "Notebook Gamer",
      "descricao": "Alto desempenho para jogos.",
      "quantidade_estoque": 15,
      "preco_unitario_atual": 7800.00,
      "categoria": {
         "id_categoria": 1,
         "nome": "Eletrônicos"
      }
   }
   // Adicione mais documentos aqui
])
```

Exemplo de Inserção na Coleção `pedidos`:

JavaScript

```
db.pedidos.insertMany([
   {
      "data_pedido": new Date("2025-10-20T10:00:00Z"),
      "data_remessa": new Date("2025-10-21T14:00:00Z"),
      "status": "Entregue",
      "valor_total": 5000.00,
      "cliente": {
         "id_cliente": 1,
         "nome": "João da Silva",
         "tipo": "Física",
         "documento": "123.456.789-00",
         "telefone": [{ "numero": "51999998888" }]
      },
      "endereco_entrega": {
         "logradouro": "Rua das Flores",
         "numero": "123",
         "cidade": "Porto Alegre",
```

```
            "estado": "RS"
        },
        "itens": [
            {
                "id_produto": 101,
                "nome_produto": "Smartphone X",
                "quantidade": 2,
                "preco_unitario": 2500.00
            }
        ]
    },
    {
        "data_pedido": new Date("2025-10-25T15:30:00Z"),
        "data_remessa": new Date("2025-10-26T10:00:00Z"),
        "status": "Em processamento",
        "valor_total": 7800.00,
        "cliente": {
            "id_cliente": 2,
            "nome": "Empresa Alfa Ltda",
            "tipo": "Jurídica",
            "documento": "00.000.000/0001-00",
            "telefone": [{ "numero": "5133334444" }]
        },
        "endereco_entrega": {
            "logradouro": "Av. Principal",
            "numero": "500",
            "cidade": "Canoas",
            "estado": "RS"
        },
        "itens": [
            {
                "id_produto": 102,
                "nome_produto": "Notebook Gamer",
                "quantidade": 1,
                "preco_unitario": 7800.00
            }
        ]
    }
    // Adicione mais documentos aqui
])
```

A3. Consultas no MongoDB

As consultas no MongoDB são realizadas usando o *Aggregation Pipeline* (Pipeline de Agregação) para operações complexas como agrupamento, cálculo e ordenação.

1. Comandos de Consulta (A3.1)

a) Listar a quantidade total de produtos em cada categoria

Esta consulta usa o estágio `$group` para agrupar os documentos da coleção `produtos` pela categoria e somar a quantidade em estoque.

JavaScript

```
db.produtos.aggregate([
   {
      $group: {
         _id: "$categoria.nome", // Agrupa pelo nome da categoria
         total_produtos: { $sum: "$quantidade_estoque" } // Soma a
quantidade em estoque
      }
   },
   {
      $sort: { total_produtos: -1 } // Opcional: Ordena do maior para o menor
   }
])
```

b) Apresentar as entregas com os dados de endereço, nome do cliente e ordenados por data de entrega

Esta consulta usa `$project` para selecionar os campos desejados e `$sort` para ordenar.

JavaScript

```
db.pedidos.aggregate([
   {
      $match: { status: "Entregue" } // Filtra apenas pedidos entregues
   },
   {
      $project: {
         _id: 0, // Exclui o _id do resultado
         nome_cliente: "$cliente.nome",
         data_entrega: "$data_entrega",
         endereco: "$endereco_entrega"
      }
   },
   {
      $sort: { data_entrega: 1 } // Ordena pela data de entrega ascendente
   }
])
```

c) Calcular o valor total de cada pedido e exibir somente os pedidos com total acima de 1000

Como o campo `valor_total` já está presente no nosso esquema desnormalizado, a consulta é simplificada. Se não estivesse, usaríamos `$unwind` e `$group` para calcular.

JavaScript

```
db.pedidos.find(
   { valor_total: { $gt: 1000 } }, // Filtra pedidos com valor total maior que 1000
   { _id: 1, valor_total: 1, "cliente.nome": 1 } // Projeta apenas os campos necessários
).sort({ valor_total: -1 }) // Opcional: Ordena pelo valor total
```

d) Listar o nome dos 10 produtos mais vendidos por quantidade em ordem decrescente

Esta é uma consulta complexa que exige o *Aggregation Pipeline* para desmembrar o array de itens (`$unwind`), agrupar e somar as quantidades.

JavaScript

```
db.pedidos.aggregate([
   {
      $unwind: "$itens" // Desmembra o array 'itens' para tratar cada item individualmente
   },
   {
      $group: {
         _id: "$itens.nome_produto", // Agrupa pelo nome do produto
         total_vendido: { $sum: "$itens.quantidade" } // Soma a quantidade vendida
      }
   },
   {
      $sort: { total_vendido: -1 } // Ordena do mais vendido para o menos vendido
   },
   {
      $limit: 10 // Limita aos 10 primeiros
   }
])
```

e) Apresentar a quantidade de clientes em cada região

Esta consulta agrupa os clientes pelo estado do endereço de entrega.

JavaScript

```
db.pedidos.aggregate([
   {
      $group: {
         _id: "$endereco_entrega.estado", // Agrupa pelo estado
         quantidade_clientes: { $addToSet: "$cliente.id_cliente" } // Cria
um conjunto único de IDs de clientes por estado
      }
   },
   {
      $project: {
         _id: 0,
         estado: "$_id",
         quantidade_clientes: { $size: "$quantidade_clientes" } // Conta o
tamanho do conjunto (clientes únicos)
      }
   },
   {
      $sort: { quantidade_clientes: -1 }
   }
])
```

2. Duas Consultas Relevantes Adicionais (A3.2)

Consulta Adicional 1: Clientes que Compraram Mais de 5 Produtos Diferentes

Esta consulta identifica clientes que demonstram alta variedade de compra, o que pode ser útil para campanhas de fidelidade.

JavaScript

```
db.pedidos.aggregate([
   {
      $group: {
         _id: "$cliente.nome", // Agrupa pelo nome do cliente
         produtos_diferentes: { $addToSet: "$itens.id_produto" } // Cria um
conjunto de IDs de produtos únicos
      }
   },
   {
      $project: {
         _id: 0,
```

```
        cliente: "$_id",
        total_produtos_diferentes: { $size: "$produtos_diferentes" } //
Conta o número de produtos únicos
      }
   },
   {
      $match: { total_produtos_diferentes: { $gt: 5 } } // Filtra clientes
com mais de 5 produtos diferentes
   },
   {
      $sort: { total_produtos_diferentes: -1 }
   }
])
```

Consulta Adicional 2: Valor Médio de Pedido por Categoria de Produto

Esta consulta ajuda a entender quais categorias geram o maior valor médio de transação.

JavaScript

```
db.pedidos.aggregate([
   {
      $unwind: "$itens" // Desmembra os itens do pedido
   },
   // Usa $lookup para buscar o nome da categoria na coleção 'produtos'
   // Nota: Em um modelo ideal, a categoria estaria embutida no item do
pedido para evitar o lookup,
   // mas como o id_produto está embutido, podemos fazer o lookup para fins
de demonstração.
   {
      $lookup: {
         from: "produtos",
         localField: "itens.id_produto",
         foreignField: "id_produto_original",
         as: "dados_produto"
      }
   },
   {
      $unwind: "$dados_produto" // Desmembra o array de lookup
   },
   {
      $group: {
         _id: "$dados_produto.categoria.nome", // Agrupa pelo nome da
categoria
         valor_total_pedidos: { $sum: "$valor_total" }, // Soma o valor
total dos pedidos
         total_pedidos: { $addToSet: "$_id" } // Conta o número de pedidos
```

```
únicos
      }
   },
   {
      $project: {
         _id: 0,
         categoria: "$_id",
         valor_medio_pedido: { $divide: ["$valor_total_pedidos", { $size:
"$total_pedidos" }] } // Calcula a média
      }
   },
   {
      $sort: { valor_medio_pedido: -1 }
   }
])
```