Workshop MongoDB: Do Básico ao Avançado

Plano de Aula

Curva de Aprendizado

1. **Fundamentos de NoSQL** - Entendendo bancos não relacionais
2. **Instalação e Configuração** - Ambiente de desenvolvimento
3. **CRUD Básico** - Operações fundamentais
4. **Consultas Avançadas** - Busca e filtragem de dados
5. **Índices e Performance** - Otimização de consultas
6. **Modelagem de Dados** - Estruturas e relacionamentos
7. **Agregações** - Processamento de dados complexos
8. **Operações Avançadas** - Backup, exportação e migração
9. **MongoDB em Ambientes Reais** - Docker e cenários de produção

Dataset para o Workshop

Vamos trabalhar com um dataset de uma loja online que inclui:

- Produtos
- Clientes
- Pedidos
- Avaliações

Módulo 1: Fundamentos de Bancos de Dados NoSQL

O que são Bancos de Dados NoSQL?

NoSQL (Not Only SQL) refere-se a bancos de dados não relacionais projetados para armazenar, distribuir e acessar dados usando modelos diferentes do tradicional modelo relacional.

Tipos de Bancos NoSQL

1. **Documentos** (MongoDB, CouchDB)
2. **Chave-valor** (Redis, DynamoDB)
3. **Colunar** (Cassandra, HBase)
4. **Grafos** (Neo4j, JanusGraph)

Comparação: Relacional vs NoSQL

Característica	Relacional (SQL)	NoSQL (MongoDB)
Estrutura	Tabelas, linhas e colunas	Coleções e documentos
Schema	Rígido	Flexível/Dinâmico
Relacionamentos	Joins	Documentos aninhados/Referências
Escalabilidade	Vertical	Horizontal
Consistência	ACID	Eventual (BASE)
Casos de uso	Dados estruturados, transações	Grandes volumes, dados variáveis

Casos de Uso Reais para MongoDB

1. **Aplicações de Conteúdo**: CMS, blogs, redes sociais
2. **E-commerce**: Catálogos de produtos, perfis de usuários
3. **IoT**: Armazenamento de dados de sensores
4. **Aplicações Mobile**: Backend para apps
5. **Analytics em Tempo Real**: Painéis, métricas
6. **Gerenciamento de Catálogos**: Produtos com atributos variáveis

Módulo 2: Instalação e Configuração do MongoDB

Instalação Convencional

```
# Para Ubuntu/Debian
sudo apt-get update
sudo apt-get install -y mongodb-org

# Para MacOS (com Homebrew)
brew tap mongodb/brew
brew install mongodb-community

# Para Windows
# Baixe o instalador em mongodb.com/try/download/community
```

Instalação via Docker

```
# Puxar a imagem oficial do MongoDB
docker pull mongo

# Executar o container MongoDB
docker run --name mongodb -p 27017:27017 -d mongo

# Executar com persistência de dados
docker run --name mongodb -p 27017:27017 -v /caminho/local:/data/db -d mongo
```

Comparação entre instalações

Aspecto	Instalação Convencional	Docker
Facilidade	Média	Alta
Configuração	Manual	Simplificada
Isolamento	Não	Sim
Escalabilidade	Manual	Facilitada
Portabilidade	Baixa	Alta
Recursos	Usa recursos do host diretamente	Containerizado

Conexão ao MongoDB

```
# Conexão local
mongo
# ou
mongosh

# Conexão com autenticação
mongo -u usuario -p senha --authenticationDatabase admin

# Conexão via string de conexão
mongo "mongodb://usuario:senha@localhost:27017/database"
```

Verificação da instalação

```
// Mostra os bancos existentes
show dbs

// Usa (ou cria) um banco específico
use ecommerce

// Mostra coleções no banco atual
show collections
```

Módulo 3: CRUD Básico

Criação (Create)

```
// Inserir um único documento
db.produtos.insertOne({
  nome: "Tablet Pro X",
  preco: 799.99,
  estoque: 25,
  categoria: "Eletrônicos"
})

// Inserir múltiplos documentos
db.produtos.insertMany([
  {
    nome: "Mouse sem fio",
    preco: 49.99,
    estoque: 100,
    categoria: "Acessórios"
  },
  {
    nome: "Teclado mecânico",
    preco: 129.99,
    estoque: 50,
    categoria: "Acessórios"
  }
])
```

Leitura (Read)

```
// Buscar todos os documentos
db.produtos.find()

// Buscar com formatação melhorada
db.produtos.find().pretty()

// Buscar com critérios
db.produtos.find({ categoria: "Eletrônicos" })

// Buscar um documento específico
db.produtos.findOne({ nome: "Tablet Pro X" })

// Buscar por ID
db.produtos.findOne({ _id: ObjectId("5f8d0d55b54764429c1e1111") })

// Limitar resultados
db.produtos.find().limit(3)

// Pular resultados (paginação)
db.produtos.find().skip(10).limit(10)
```

Atualização (Update)

```
// Atualizar um documento
db.produtos.updateOne(
  { nome: "Tablet Pro X" },
  { $set: { preco: 849.99, desconto: true } }
)

// Atualizar múltiplos documentos
db.produtos.updateMany(
  { categoria: "Acessórios" },
  { $inc: { estoque: -5 } }
)

// Substituir completamente um documento
db.produtos.replaceOne(
  { nome: "Mouse sem fio" },
  {
    nome: "Mouse Bluetooth Premium",
    preco: 79.99,
    estoque: 30,
    categoria: "Acessórios"
  }
)
```

Remoção (Delete)

```
// Remover um documento
db.produtos.deleteOne({ nome: "Tablet Pro X" })

// Remover múltiplos documentos
db.produtos.deleteMany({ estoque: { $lt: 10 } })

// Remover todos os documentos
db.produtos.deleteMany({})

// Remover uma coleção inteira
db.produtos.drop()
```

Módulo 4: Consultas Avançadas

Operadores de Consulta

Operador	Explicação	Exemplo
`$eq`	Corresponde a valores iguais ao especificado	`db.produtos.find({ preco: { $eq: 299.99 } })`
`$ne`	Corresponde a valores diferentes do especificado	`db.produtos.find({ categoria: { $ne: "Eletrônicos" } })`
`$gt`	Maior que o valor especificado	`db.produtos.find({ preco: { $gt: 1000 } })`
`$gte`	Maior ou igual ao valor especificado	`db.produtos.find({ estoque: { $gte: 20 } })`
`$lt`	Menor que o valor especificado	`db.produtos.find({ preco: { $lt: 500 } })`
`$lte`	Menor ou igual ao valor especificado	`db.produtos.find({ estoque: { $lte: 15 } })`
`$in`	Corresponde a qualquer valor do	`db.produtos.find({ categoria: { $in: ["Eletrônicos", "Acessórios"] } })`

Operador	Explicação	Exemplo
	array especificado	
`$nin`	Não corresponde a nenhum valor do array especificado	`db.produtos.find({ categoria: { $nin: ["Vestuário", "Alimentos"] } })`
`$exists`	Corresponde a documentos que possuem o campo especificado	`db.produtos.find({ desconto: { $exists: true } })`
`$type`	Corresponde a documentos com o tipo BSON especificado	`db.produtos.find({ preco: { $type: "double" } })`
`$regex`	Permite usar expressões regulares para buscar	`db.produtos.find({ nome: { $regex: /^Smart/i } })`
`$text`	Realiza busca de texto em campos indexados para texto	`db.produtos.find({ $text: { $search: "smartphone android" } })`
`$all`	Corresponde a arrays que contêm todos os elementos especificados	`db.produtos.find({ tags: { $all: ["bluetooth", "audio"] } })`
`$elemMatch`	Corresponde a documentos que contêm um array	`db.pedidos.find({ itens: { $elemMatch: { quantidade: { $gt: 1 } } } })`

Operador	Explicação	Exemplo
	com pelo menos um elemento que satisfaz todos os critérios	
`$size`	Corresponde a arrays com o tamanho especificado	`db.produtos.find({ tags: { $size: 4 } })`

Exemplos de Buscas Avançadas

Busca exata por valor

```
// Busca produtos com preço exatamente 299.99
db.produtos.find({ preco: 299.99 })
```

Busca por intervalo de valores

```
// Produtos com preço entre 500 e 3000
db.produtos.find({
  preco: { $gte: 500, $lte: 3000 }
})
```

Busca com múltiplos critérios (AND implícito)

```
// Produtos da categoria Eletrônicos com estoque > 15
db.produtos.find({
  categoria: "Eletrônicos",
  estoque: { $gt: 15 }
})
```

Operador OR

```
// Produtos que são Eletrônicos OU custam menos de 500
db.produtos.find({
  $or: [
    { categoria: "Eletrônicos" },
    { preco: { $lt: 500 } }
  ]
})
```

Busca por texto (com e sem acentuação)

```
// Primeiro, criar um índice de texto
db.produtos.createIndex({ nome: "text", descricao: "text" })

// Busca por texto ignorando acentuação
db.produtos.find({
  $text: {
    $search: "camera",
    $caseSensitive: false,
    $diacriticSensitive: false
  }
})
```

Busca em campos aninhados

```
// Busca produtos com memória de 16GB
db.produtos.find({
  "especificacoes.memoria": "16GB"
})
```

Busca em arrays

```
// Produtos com tag "bluetooth"
db.produtos.find({
  tags: "bluetooth"
})

// Produtos com TODAS as tags listadas
db.produtos.find({
  tags: { $all: ["bluetooth", "audio"] }
})
```

Busca por expressão regular

```
// Produtos que começam com "Smart"
db.produtos.find({
  nome: { $regex: /^Smart/i }
})

// Produtos com nome contendo "Pro" em qualquer posição
db.produtos.find({
  nome: { $regex: /Pro/i }
})
```

Projeção de campos

```
// Retornar apenas nome e preço
db.produtos.find(
  { categoria: "Eletrônicos" },
  { nome: 1, preco: 1 }
)

// Excluir campos específicos
db.produtos.find(
  { categoria: "Eletrônicos" },
  { especificacoes: 0, tags: 0 }
)
```

Módulo 5: Índices e Performance

Criação de Índices

```
// Índice simples
db.produtos.createIndex({ nome: 1 })  // 1 ascendente, -1 descendente

// Índice composto
db.produtos.createIndex({ categoria: 1, preco: -1 })

// Índice único
db.clientes.createIndex({ email: 1 }, { unique: true })

// Índice de texto
db.produtos.createIndex({
  nome: "text",
  descricao: "text"
})

// Índice esparso (só para documentos que têm o campo)
db.produtos.createIndex(
  { promocao: 1 },
  { sparse: true }
)

// Índice TTL (expiração após período)
db.sessoes.createIndex(
  { ultimoAcesso: 1 },
  { expireAfterSeconds: 3600 }
)

// Índice geoespacial
db.lojas.createIndex({ localizacao: "2dsphere" })
```

Gerenciamento de Índices

```
// Listar índices
db.produtos.getIndexes()

// Remover um índice
db.produtos.dropIndex("nome_1")

// Remover todos os índices (exceto _id)
db.produtos.dropIndexes()
```

Análise de Consultas

```
// Explicação da consulta
db.produtos.find({ categoria: "Eletrônicos" }).explain()

// Explicação detalhada
db.produtos.find({ categoria: "Eletrônicos" }).explain("executionStats")

// Verificar uso de índices
db.produtos.find({
  categoria: "Eletrônicos",
  preco: { $gt: 1000 }
}).hint({ categoria: 1, preco: -1 }).explain()
```

Módulo 6: Modelagem de Dados em MongoDB

Abordagens de Modelagem

1. **Documentos Aninhados (Embedding)**
   - Melhor para relações 1:1 e 1:poucos
   - Acesso em uma única operação
   - Limitado pelo tamanho máximo do documento (16MB)
2. **Referências (Linking)**
   - Melhor para relações 1:muitos e muitos:muitos
   - Evita duplicação de dados
   - Requer múltiplas operações para acessar dados relacionados

Exemplos de Modelagem

Modelo com documentos aninhados (Embedding)

```
// Cliente com endereços aninhados
db.clientes.insertOne({
  nome: "Carlos Mendes",
  email: "carlos@email.com",
  telefone: "(47) 98765-4321",
  enderecos: [
    {
      tipo: "residencial",
      rua: "Rua das Palmeiras",
      numero: "123",
      cidade: "Florianópolis",
      estado: "SC"
    },
    {
      tipo: "trabalho",
      rua: "Av. Beira Mar",
      numero: "1500",
      cidade: "Florianópolis",
      estado: "SC"
    }
  ]
})
```

Modelo com referências (Linking)

```
// Cliente referenciando pedidos
db.clientes.insertOne({
  nome: "Carlos Mendes",
  email: "carlos@email.com",
  pedidos: [
    ObjectId("7a8b9c0d1e2f3g4h5i6j1112"),
    ObjectId("7a8b9c0d1e2f3g4h5i6j1113")
  ]
})

// Pedidos com referência ao cliente
db.pedidos.insertOne({
  _id: ObjectId("7a8b9c0d1e2f3g4h5i6j1112"),
  clienteId: ObjectId("6a1b2c3d4e5f6a7b8c9d4444"),
  data: ISODate("2023-11-15T10:30:00Z"),
  total: 599.98,
  produtos: [
    { id: ObjectId("5f8d0d55b54764429c1e3333"), qtd: 2 }
  ]
})
```

Consultas em Dados Relacionados

```
// Consulta em dados aninhados
db.clientes.find({
  "enderecos.cidade": "Florianópolis"
})

// Consulta juntando cliente e pedido (manual)
const cliente = db.clientes.findOne({
  email: "carlos@email.com"
})
const pedidosCliente = db.pedidos.find({
  clienteId: cliente._id
}).toArray()

// Agregação para juntar dados
db.pedidos.aggregate([
  {
    $match: {
      clienteId: ObjectId("6a1b2c3d4e5f6a7b8c9d4444")
    }
  },
  {
    $lookup: {
      from: "clientes",
      localField: "clienteId",
      foreignField: "_id",
      as: "cliente"
    }
  }
])
```

Módulo 7: Agregações

Estágios de Agregação Comuns

```
// $match - Filtra documentos
db.produtos.aggregate([
  { $match: { preco: { $gt: 1000 } } }
])

// $group - Agrupa documentos
db.produtos.aggregate([
  { $group: {
      _id: "$categoria",
      count: { $sum: 1 },
      mediaPreco: { $avg: "$preco" },
      minPreco: { $min: "$preco" },
      maxPreco: { $max: "$preco" }
  }}
])

// $project - Seleciona campos específicos
db.produtos.aggregate([
  { $match: { categoria: "Eletrônicos" } },
  { $project: {
      nome: 1,
      preco: 1,
      valorComImposto: { $multiply: ["$preco", 1.15] }
  }}
])

// $sort - Ordena documentos
db.produtos.aggregate([
  { $sort: { preco: -1 } }  // ordem descendente
])

// $limit - Limita número de resultados
db.produtos.aggregate([
  { $sort: { preco: -1 } },
  { $limit: 5 }
])

// $unwind - "Explode" um array em múltiplos documentos
db.produtos.aggregate([
  { $unwind: "$tags" },
  { $group: {
      _id: "$tags",
      count: { $sum: 1 }
  }}
])
```

```
// $lookup - Join entre coleções
db.pedidos.aggregate([
  { $match: { status: "Entregue" } },
  { $lookup: {
      from: "clientes",
      localField: "clienteId",
      foreignField: "_id",
      as: "cliente"
  }},
  { $unwind: "$cliente" }
])
```

Exemplo de Relatório Completo

```
// Relatório de vendas por categoria
db.pedidos.aggregate([
  // Apenas pedidos concluídos
  { $match: { status: { $in: ["Entregue", "Em Transporte"] } } },

  // Desdobrar itens de cada pedido
  { $unwind: "$itens" },

  // Juntar com produtos para obter categoria
  { $lookup: {
      from: "produtos",
      localField: "itens.produtoId",
      foreignField: "_id",
      as: "produto"
  }},
  { $unwind: "$produto" },

  // Calcular valor total por item
  { $project: {
      categoria: "$produto.categoria",
      valorItem: { $multiply: ["$itens.quantidade", "$itens.precoUnitario"] },
      data: "$dataPedido"
  }},

  // Agrupar por categoria
  { $group: {
      _id: "$categoria",
      totalVendas: { $sum: "$valorItem" },
      quantidadePedidos: { $sum: 1 }
  }},

  // Ordenar do maior para o menor valor
  { $sort: { totalVendas: -1 } }
])
```

Módulo 8: Operações Avançadas

Exportação para CSV

```
// No terminal (não no shell MongoDB)
mongoexport --db=ecommerce --collection=produtos --type=csv --fields=nome,categoria,preco,estoque --out=pr

// Com autenticação
mongoexport --db=ecommerce --collection=produtos --type=csv --fields=nome,categoria,preco,estoque --out=pr
```

Geração de PDF com MongoDB e Node.js

```
// Usando Node.js com MongoDB e PDFKit
const { MongoClient } = require('mongodb');
const PDFDocument = require('pdfkit');
const fs = require('fs');

async function gerarRelatorioPDF() {
  const client = await MongoClient.connect('mongodb://localhost:27017');
  const db = client.db('ecommerce');

  // Obter dados para o relatório
  const produtos = await db.collection('produtos')
    .find({ categoria: "Eletrônicos" })
    .sort({ preco: -1 })
    .toArray();

  // Criar PDF
  const doc = new PDFDocument();
  doc.pipe(fs.createWriteStream('relatorio_produtos.pdf'));

  // Adicionar título
  doc.fontSize(20).text('Relatório de Produtos Eletrônicos', {
    align: 'center'
  });

  // Adicionar dados ao PDF
  doc.moveDown();
  produtos.forEach(produto => {
    doc.fontSize(14).text(produto.nome);
    doc.fontSize(12).text(`Preço: R$ ${produto.preco.toFixed(2)}`);
    doc.fontSize(12).text(`Estoque: ${produto.estoque} unidades`);
    doc.moveDown();
  });

  doc.end();
  client.close();
}

gerarRelatorioPDF();
```

Backup e Restauração

```
# Backup de um banco de dados
mongodump --db=ecommerce --out=/caminho/backup/

# Backup de uma coleção específica
mongodump --db=ecommerce --collection=produtos --out=/caminho/backup/

# Backup com compressão
mongodump --db=ecommerce --out=/caminho/backup/ --gzip

# Restauração de um banco de dados
mongorestore --db=ecommerce /caminho/backup/ecommerce/

# Restauração de uma coleção específica
mongorestore --db=ecommerce --collection=produtos /caminho/backup/ecommerce/produtos.bson
```

Movendo Dados Entre Bancos

```
// Via MongoDB Shell
// Copiar dados de uma coleção para outra
db.produtos.find().forEach(function(doc) {
  db.getSiblingDB('ecommerce_backup').produtos.insert(doc);
});

// Via mongoexport/mongoimport
// No terminal
mongoexport --db=ecommerce --collection=produtos --out=produtos.json
mongoimport --db=ecommerce_novo --collection=produtos --file=produtos.json
```

Gerenciamento de Erros Comuns

1. Conexão Recusada

```
// Erro
MongoConnectionError: connect ECONNREFUSED 127.0.0.1:27017

// Solução
// Verificar se o serviço MongoDB está rodando
sudo systemctl status mongod
// ou no Docker
docker ps | grep mongo
```

2. Violação de Índice Único

```
// Erro ao inserir documento com valor duplicado em campo com índice único

// Solução - usar upsert
db.clientes.updateOne(
  { email: "maria.silva@email.com" },
  { $set: { nome: "Maria Silva", telefone: "11987654321" } },
  { upsert: true }
)
```

3. Consulta com ID Inválido

```
// Erro
// Uncaught exception: Error: invalid object id: length

// Solução - verificar o formato do ObjectId
if (ObjectId.isValid(idString)) {
  db.produtos.findOne({ _id: ObjectId(idString) });
} else {
  // lidar com ID inválido
}
```

Módulo 9: MongoDB em Ambientes Reais

MongoDB com Docker Compose

```yaml
# docker-compose.yml
version: '3'
services:
  mongodb:
    image: mongo:latest
    container_name: mongodb
    ports:
      - "27017:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: admin
      MONGO_INITDB_ROOT_PASSWORD: senha123
    volumes:
      - mongo_data:/data/db
    networks:
      - mongo_network

  mongo-express:
    image: mongo-express:latest
    container_name: mongo-express
    depends_on:
      - mongodb
    ports:
      - "8081:8081"
    environment:
      ME_CONFIG_MONGODB_ADMINUSERNAME: admin
      ME_CONFIG_MONGODB_ADMINPASSWORD: senha123
      ME_CONFIG_MONGODB_SERVER: mongodb
      ME_CONFIG_BASICAUTH_USERNAME: admin
      ME_CONFIG_BASICAUTH_PASSWORD: senha123
    networks:
      - mongo_network

volumes:
  mongo_data:

networks:
  mongo_network:
    driver: bridge
```

MongoDB com Replicação

```
# Iniciar um conjunto de réplicas com Docker
docker run -d --name mongo1 -p 27017:27017 mongo --replSet rs0
docker run -d --name mongo2 -p 27018:27017 mongo --replSet rs0
docker run -d --name mongo3 -p 27019:27017 mongo --replSet rs0

# Configurar o conjunto de réplicas
docker exec -it mongo1 mongosh --eval "
rs.initiate({
  _id: 'rs0',
  members: [
    { _id: 0, host: 'localhost:27017' },
    { _id: 1, host: 'localhost:27018' },
    { _id: 2, host: 'localhost:27019' }
  ]
})
"
```

MongoDB Atlas (Serviço na Nuvem)

```
// Conexão com MongoDB Atlas
const { MongoClient } = require('mongodb');

const uri = "mongodb+srv://usuario:senha@cluster0.mongodb.net/ecommerce?retryWrites=true&w=majority";
const client = new MongoClient(uri);

async function conectar() {
  try {
    await client.connect();
    console.log("Conectado ao MongoDB Atlas!");

    const db = client.db('ecommerce');
    const produtos = await db.collection('produtos').find().limit(5).toArray();
    console.log(produtos);
  } finally {
    await client.close();
  }
}

conectar().catch(console.error);
```

Segurança e Autenticação

```
// Criar usuário com permissões específicas
db.createUser({
  user: "app_user",
  pwd: "senha_segura",
  roles: [
    { role: "readWrite", db: "ecommerce" },
    { role: "read", db: "analytics" }
  ]
})

// Iniciar MongoDB com autenticação
// mongod --auth

// Conectar com autenticação
mongo -u app_user -p senha_segura --authenticationDatabase admin
```