Informações e Recursos Adicionais sobre MongoDB

QXD0099 - Desenvolvimento de Software para Persistência

Universidade Federal do Ceará - *Campus* Quixadá

Prof. Francisco Victor da Silva Pinheiro
victorpinheiro@ufc.br

Agenda

- Recursos Avançados do MongoDB
- Replica Sets
- Sharding (Escalabilidade Horizontal)
- Aggregation Framework
- MongoDB Atlas
- Transações Multi-Documento
- Modelagem de Dados no MongoDB
  - Embed vs. Referência
- Desempenho no MongoDB
- Segurança no MongoDB
- Ferramentas e Extensões
- Casos de Uso Típicos do MongoDB – Exemplos Reais

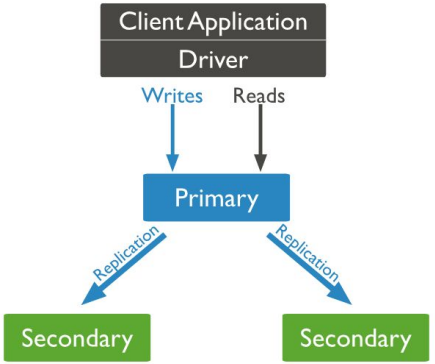
Recursos Avançados do MongoDB

- MongoDB não é apenas um banco de dados NoSQL, mas um ecossistema completo com funcionalidades robustas que atendem a cenários variados. Aqui estão algumas delas:
  - Replica Sets
  - Sharding
  - Aggregation Framework
  - Transactions (Transações Multi-Documento)
  - MongoDB Atlas

Replica Sets

- **O que é?**
  - Replica Sets são conjuntos de instâncias do MongoDB que mantêm os mesmos dados, garantindo alta disponibilidade e tolerância a falhas.
    - Apenas um nó é primário (realiza operações de escrita).
    - Outros nós são secundários (replicam os dados e podem ser usados para leitura).
    - Se o primário falhar, um novo primário é eleito automaticamente.

- **Benefícios**
  - Alta disponibilidade: Se um nó falha, outro assume automaticamente.
  - Backup automático: Os dados são replicados continuamente.



Replica Sets

- **Exemplo de Configuração**
- Criar um arquivo de configuração mongod.conf para cada nó

```
replication:
  replSetName: "meuReplicaSet"
```

- Iniciar os servidores MongoDB com os arquivos de configuração

```
mongod --config /caminho/para/mongod.conf
```

- Conectar ao MongoDB e inicializar o Replica Set no Shell

```
rs.initiate({
  _id: "meuReplicaSet",
  members: [
    { _id: 0, host: "localhost:27017" },
    { _id: 1, host: "localhost:27018" },
    { _id: 2, host: "localhost:27019" }
  ]
})
```

Sharding (Escalabilidade Horizontal)

- **O que é?**
  - Sharding distribui grandes conjuntos de dados em múltiplos servidores, permitindo escalabilidade horizontal.

- **Arquitetura:**
  - **Shard Servers:** Armazenam os dados.
  - **Config Servers:** Guardam informações sobre os shards.
  - **Query Routers (mongos):** Direcionam as consultas.

Sharding (Escalabilidade Horizontal)

- **Configuração**
  - **Iniciar os servidores de configuração**
    - mongod --configsvr --replSet configReplSet --port 27019 --dbpath /dados/config
  - **Iniciar o roteador (mongos):**
    - mongos --configdb configReplSet/localhost:27019
  - **Adicionar shards no Shell:**
    - sh.addShard("shard1/localhost:27017")
    - sh.addShard("shard2/localhost:27018")
  - **Habilitar Sharding para um banco de dados:**
    - sh.enableSharding("meuBanco")
    - sh.shardCollection("meuBanco.minhaColecao", { campoChave: "hashed" })

Aggregation Framework

- **O que é?**
- O Aggregation Framework permite executar consultas complexas no MongoDB.

- **Principais Operadores**
  - **$match:** Filtra documentos.
  - **$group:** Agrupa dados.
  - **$project:** Seleciona campos específicos.
  - **$unwind:** Transforma arrays em documentos individuais.

Aggregation Framework

- **Exemplo de Agregação**

```python
from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017/")
db = client.meuBanco

pipeline = [
    {"$match": {"categoria": "eletrônicos"}},  # Filtra produtos eletrônicos
    {"$group": {"_id": "$marca", "total": {"$sum": 1}}},  # Conta produtos por marca
    {"$sort": {"total": -1}}  # Ordena pelo total
]

resultado = db.produtos.aggregate(pipeline)
for doc in resultado:
    print(doc)
```

MongoDB Atlas

- **O que é?**
  - Uma plataforma de banco de dados como serviço (DBaaS) da MongoDB.

- **Benefícios:**
  - Gerenciamento simplificado de clusters.
  - Suporte para réplicas e sharding com poucos cliques.
  - Escalabilidade automática.
  - Ferramentas integradas, como backups, monitoramento e análise.

- Ideal para desenvolvedores que querem evitar a complexidade de configurar infraestrutura manualmente.

Transações Multi-Documento

- **O que é?**
  - Desde a versão 4.0, o MongoDB passou a suportar transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade) para operações multi-documento em um único shard.

  - Já na versão 4.2, o suporte foi estendido para transações distribuídas em múltiplos shards (Sharded Clusters), tornando o MongoDB uma opção mais robusta para aplicações que exigem garantias transacionais.

Transações Multi-Documento

- **Como Funciona?**
  - As transações no MongoDB permitem agrupar múltiplas operações de leitura e escrita (CRUD) em uma única unidade atômica. Isso significa que todas as operações da transação são confirmadas juntas (commit) ou desfeitas (rollback) caso alguma falhe.

  - As transações utilizam um modelo similar ao de bancos relacionais, incluindo:
    - Início da transação (startSession() e startTransaction())
    - Execução das operações dentro da transação
    - Commit ou Rollback, garantindo que os dados permaneçam consistentes.

Transações Multi-Documento

- **Exemplo 1**

```
with client.start_session() as session:
    with session.start_transaction():
        db.usuarios.insert_one({"nome": "João"}, session=session)
        db.pedidos.insert_one({"usuario": "João", "pedido_id": 123},
session=session)
```

Transações Multi-Documento

- **Exemplo 2**

```python
try:
    # Iniciar a transação
    session.start_transaction()

    # Inserir um documento na coleção "clientes"
    db["clientes"].insert_one({"_id": 1, "nome": "João", "saldo": 1000}, session=session)

    # Inserir um documento na coleção "transacoes"
    db["transacoes"].insert_one({"_id": 1, "cliente_id": 1, "valor": -500}, session=session)

    # Confirmar a transação
    session.commit_transaction()
    print("Transação confirmada com sucesso.")

except PyMongoError as e:
    # Se ocorrer um erro, reverter a transação
    session.abort_transaction()
    print("Erro na transação, alterações revertidas:", e)

finally:
    # Encerrar a sessão
    session.end_session()
```

Transações Multi-Documento

- **Principais Benefícios**
  - Atomicidade: Todas as operações são aplicadas ou nenhuma.
  - Consistência: O estado do banco não fica corrompido em caso de falha.
  - Isolamento: As transações são isoladas até serem confirmadas.
  - Durabilidade: Após o commit, as mudanças são persistentes.

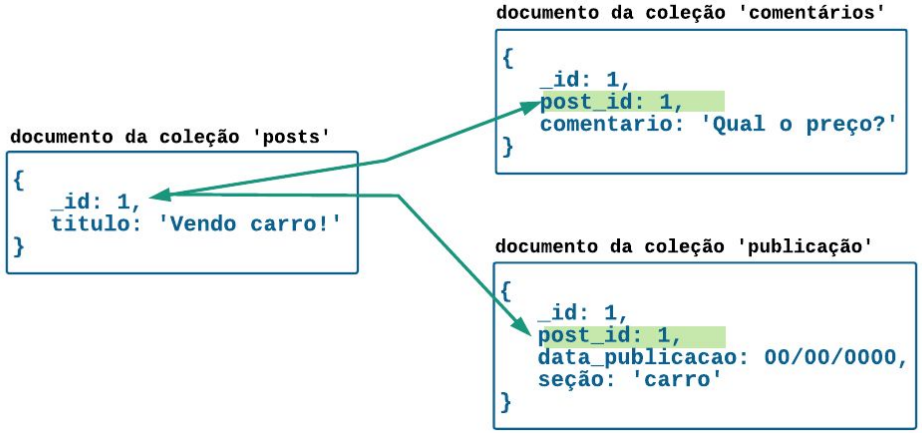
- **Limitações**
  - Desempenho: O uso de transações pode impactar a performance, pois exige mais recursos.
  - Tempo máximo: As transações devem ser curtas para evitar bloqueios de recursos.
  - Operações em Sharded Clusters: A partir da versão 4.2, suportado apenas com readConcern e writeConcern adequados.

Modelagem de Dados no MongoDB

- **Embed (Incorporação)**
  - Quando usar? Relacionamentos 1:1 ou 1:N pequenos.
  - **Vantagem:** Acesso mais rápido.

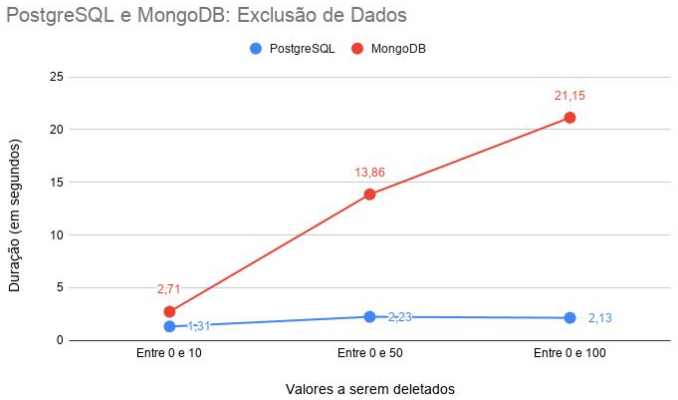
Modelagem de Dados no MongoDB

- **Referência**
  - Quando usar? Relacionamentos N:N ou 1:N grandes.
  - **Vantagem:** Evita duplicação de dados.



Desempenho no MongoDB

- O desempenho do MongoDB é um fator crucial para garantir consultas rápidas, armazenamento eficiente e boa escalabilidade. A performance do banco pode ser otimizada com técnicas específicas, desde modelagem de dados até configuração do servidor.
  - Modelagem de Dados
  - Índices
  - Consultas Eficientes
  - Otimização de Escrita
  - Gerenciamento de Memória e Cache
  - Monitoramento e Performance



Desempenho no MongoDB

- **Otimização de Escrita**
  - Desative o Journaling se não precisar de alta durabilidade (não recomendado para produção crítica).
  - Use operações em lote (bulkWrite) para inserir ou atualizar vários documentos de forma eficiente:

```
db.usuarios.bulk_write([
    InsertOne({"nome": "Ana", "idade": 25}),
    InsertOne({"nome": "Carlos", "idade": 30})
])
```

- Ajuste os Write Concerns dependendo da necessidade de durabilidade vs. desempenho.

Desempenho no MongoDB

- **Gerenciamento de Memória e Cache**
  - O MongoDB usa Memory-Mapped Files (MMF) para acessar dados rapidamente.
  - Evite swapping, garantindo que o banco tenha RAM suficiente para armazenar índices e dados frequentemente acessados.
  - Monitoramento com ferramentas como mongostat e mongotop ajuda a identificar gargalos de memória.

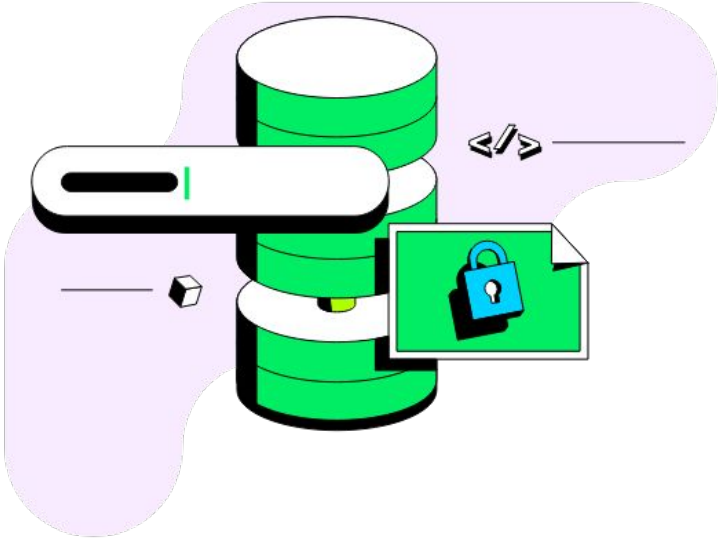
Desempenho no MongoDB

- **Monitoramento e Performance**
  - Ferramentas essenciais para análise de desempenho:
    - db.serverStatus() → Exibe estatísticas gerais do servidor.
    - db.collection.stats() → Exibe estatísticas de uma coleção.
    - MongoDB Atlas Performance Advisor → Analisa automaticamente as consultas e sugere índices.

Segurança no MongoDB

- A segurança no MongoDB é um aspecto crucial para proteger os dados contra acessos não autorizados, vazamentos e ataques.

- O MongoDB oferece diversos mecanismos para reforçar a segurança, como autenticação, controle de acesso, conexões seguras e criptografia.



Segurança no MongoDB

- **Criando Usuário com Permissões**
  - Por padrão, o MongoDB não exige autenticação (modo sem segurança ativado), o que significa que qualquer pessoa com acesso ao banco pode ler e modificar os dados.
  - Para evitar isso, devemos habilitar autenticação e criar usuários com permissões específicas.

Segurança no MongoDB

- Criando um usuário no banco de dados admin com permissões de leitura e escrita no banco meu_banco:

```
use admin
db.createUser({
    user: "meu_usuario",
    pwd: "minha_senha",
    roles: [{ role: "readWrite", db: "meu_banco" }]
})
```

- **Explicação dos campos:**
  - user: Nome do usuário.
  - pwd: Senha do usuário.
  - roles: Lista de permissões do usuário.
    - readWrite: Permite ler e escrever no banco meu_banco.
    - Outros roles disponíveis:
      - read: Apenas leitura.
      - dbOwner: Controle total do banco de dados.
      - clusterAdmin: Acesso administrativo ao cluster MongoDB.

Segurança no MongoDB

- Após criar o usuário, sempre conecte-se autenticando as credenciais:

```
mongo -u "meu_usuario" -p "minha_senha" --authenticationDatabase "meu_banco"
```

Segurança no MongoDB

- **Conexões Seguras**
  - O MongoDB permite reforçar a segurança por meio da autenticação de usuários, conexões criptografadas com SSL/TLS e configurações seguras no servidor.
- Para que a autenticação de usuários funcione corretamente, é necessário ativá-la no arquivo de configuração do MongoDB (mongod.conf).
- Edite o arquivo mongod.conf e adicione:

```
security:
     authorization: "enabled"
```

- Reinicie o servidor MongoDB para aplicar as mudanças:
  - systemctl restart mongod

Segurança no MongoDB

- **Usar conexões criptografadas com SSL/TLS**
- Para evitar que os dados sejam interceptados durante a comunicação entre o cliente e o servidor, recomenda-se o uso de SSL/TLS.

- Passos para habilitar SSL/TLS no MongoDB:
  - Gerar um certificado SSL/TLS:

```
openssl req -newkey rsa:4096 -nodes -keyout certificado.key -x509 -days 365 -out certificado.pem
```

- Configurar o MongoDB para exigir conexões seguras: No arquivo mongod.conf, adicione:

Segurança no MongoDB

- Configurar o MongoDB para exigir conexões seguras: No arquivo mongod.conf, adicione:

```
net:
  ssl:
    mode: requireSSL
    PEMKeyFile: /caminho/certificado.pem
```

- Iniciar o MongoDB com SSL ativado
  - mongod --sslMode requireSSL --sslPEMKeyFile /caminho/certificado.pem

- Conectar com um cliente MongoDB usando SSL:
  - mongo --ssl --sslPEMKeyFile /caminho/certificado.pem --host meu_servidor

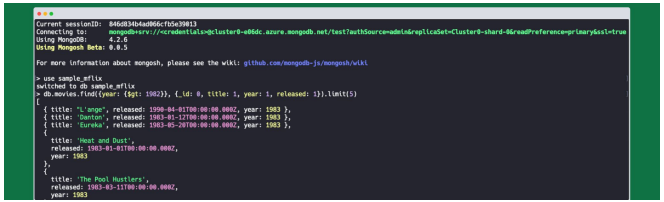
Segurança no MongoDB

- Para proteger seu MongoDB, siga estas melhores práticas:
  - Crie usuários com permissões restritas (readWrite, dbOwner, etc.).
  - Habilite autenticação (security.authorization: "enabled").
  - Use conexões criptografadas com SSL/TLS para impedir interceptação de dados.
  - Restrinja acessos via firewall, limitando conexões por IP.
  - Evite injeção de código, sempre validando as entradas do usuário.
  - Monitore logs e acessos, utilizando ferramentas como mongostat e mongotop.

Ferramentas e Extensões

- MongoDB Compass: Interface gráfica para explorar dados.
- Mongoose: ODM para Node.js.
- Mongosh: Shell interativo.
- Backup e Restauração
  - mongodump --db meuBanco --out /backup
  - mongorestore /backup/meuBanco

Mongoose { }



Casos de Uso Típicos do MongoDB – Exemplos Reais

- **iFood**
- Caso de uso: Um sistema de pedidos precisa processar milhares de transações simultaneamente e armazenar dados de clientes, pedidos, status de entrega e geolocalização.

- **Por que usar MongoDB?**
  - Modelagem de dados flexível, permitindo armazenar pedidos com diferentes combinações de produtos.
  - Escalabilidade horizontal para lidar com picos de tráfego em horários de alta demanda.
  - Integração com GeoJSON, permitindo rastrear entregadores em tempo real.

Casos de Uso Típicos do MongoDB – Exemplos Reais

- **Netflix**
- Caso de uso: Plataformas de streaming precisam armazenar bilhões de registros sobre usuários, filmes, músicas e preferências de consumo para recomendar conteúdos personalizados.

- **Por que usar MongoDB?**
  - Suporte a grandes volumes de dados.
  - Consultas rápidas utilizando índices e agregações complexas.
  - Integração com Machine Learning para análise de padrões de comportamento.

Casos de Uso Típicos do MongoDB – Exemplos Reais

- **Friend**
- Caso de uso: Plataforma autoadaptativo de monitoramento de idosos utilizando internet das coisas médicas e microsserviços

- **Por que usar MongoDB?**
  - Suporte a grandes volumes de dados.
  - Integração com Machine Learning para análise de padrões de movimentos.

Referências

- MongoDB – Site oficial
  - http://www.mongdb.com
- MongoDB Manual
  - http://docs.mongodb.org/manual/
  - http://docs.mongodb.org/manual/MongoDBmanual.pdf
- Slides: Building your first app: an introduction to MongoDB. Norman Grah
  Consulting Engineer, 10gen.
- Slides: mongoDB.
  - Júlio Monteiro (julio@monteiro.eti.br).
- Slides Why MongoDB Is Awesome
  - John Nunemaker - Ordered List (john@orderedlist.com)

Obrigado!

Dúvidas?

Universidade Federal do Ceará - *Campus* Quixadá

Prof. Francisco Victor da Silva Pinheiro
victorpinheiro@ufc.br