re_S10_A2_SL6_backend

Roteiro de Atividade Prática

Nome: _____ Turma: _____

Título da atividade: Otimização de consultas

Objetivo:

Otimizar operações de leitura e escrita em um banco de dados NoSQL para melhorar a performance e garantir que as operações sejam realizadas de forma eficiente, reduzindo o tempo de resposta e o consumo de recursos.

Conceito técnico: otimização de consultas em NoSQL

A otimização de consultas em bancos de dados NoSQL, como MongoDB, envolve melhorar a performance das operações de leitura e escrita, especialmente em aplicações que interagem com grandes volumes de dados. Assim como nos bancos de dados relacionais, consultas mal estruturadas podem resultar em lentidão, alto consumo de recursos e tempo de resposta elevado. Algumas das principais técnicas de otimização em bancos NoSQL incluem:

- índices: criar índices nas propriedades mais consultadas permite buscas mais rápidas, reduzindo o tempo necessário para localizar documentos dentro das coleções;
- filtragem adequada: utilizar operadores como $match e $limit no MongoDB para restringir os resultados e evitar a sobrecarga de retorno de dados desnecessários;
- evitar operações de join entre coleções: em MongoDB, as operações de join podem ser simuladas com $lookup, mas devem ser usadas com cuidado, pois podem prejudicar a performance em grandes coleções;
- normalização e desnormalização: em NoSQL, a desnormalização é comum, pois pode melhorar a performance, especialmente quando os

dados relacionados são frequentemente acessados juntos, reduzindo a necessidade de operações adicionais;
- projeção de campos: utilizar projeções para retornar apenas os campos necessários, evitando que documentos inteiros sejam carregados quando apenas uma parte dos dados é relevante.

Objetivo da atividade:

O participante irá otimizar consultas em um banco de dados NoSQL (MongoDB) para melhorar a performance. A atividade envolve identificar consultas ineficientes e aplicar técnicas como criação de índices e filtragem para otimizar o desempenho das operações.

Enunciado:

Você foi encarregado de otimizar algumas operações em MongoDB que estão degradando a performance de um sistema de e-commerce. A aplicação utiliza um banco de dados NoSQL para armazenar informações sobre produtos, pedidos e clientes. Suas tarefas são:

1. identificar consultas que podem ser otimizadas;
2. criar índices nas propriedades que são frequentemente consultadas;
3. reduzir o volume de dados retornado pelas consultas aplicando filtragens e projeções apropriadas.

Implemente a otimização das consultas utilizando as melhores práticas de performance para NoSQL.

Código de programação:
Consulta original ineficiente:

```
produtos = db.produtos.find({})
```

Explicação da ineficiência: a consulta acima retorna todos os documentos da coleção **produtos**, o que pode causar sobrecarga no banco de dados, especialmente se houver muitos registros.

Consulta otimizada com filtragem:

```
produtos = db.produtos.find({"preco": {"$gt": 100}}, {"nome": 1, "preco": 1})
```

Explicação da otimização: essa consulta otimizada usa o operador **$gt** (greater than) para filtrar produtos com preço superior a 100. Além disso, ela utiliza projeção para retornar apenas os campos **nome** e **preco**, reduzindo o volume de dados transferidos, o que melhora a performance.

Exemplo de criação de índice:

```
db.produtos.create_index([("preco", 1)])
```

Explicação da criação de índice: criar um índice na propriedade **preco** melhora a performance de consultas que filtram produtos por preço, já que o banco de dados pode localizar os documentos mais rapidamente.

Consulta original com lookup ineficiente:

```
pedidos = db.pedidos.aggregate([
    {"$lookup": {
        "from": "clientes",
        "localField": "cliente_id",
        "foreignField": "_id",
        "as": "cliente_info"
    }}
])
```

Consulta otimizada:

```
pedidos = db.pedidos.aggregate([
    {"$lookup": {
        "from": "clientes",
```

```
        "localField": "cliente_id",
        "foreignField": "_id",
        "as": "cliente_info"
    }},
    {"$match": {"cliente_info.ativo": True}}
])
```

Explicação da otimização: a consulta foi otimizada para incluir um **$match** que restringe o resultado apenas a clientes ativos. Isso reduz a quantidade de dados retornados e evita a busca por clientes inativos. Além disso, o uso do operador **$match** próximo ao **$lookup** garante que o filtro seja aplicado cedo no pipeline, melhorando a eficiência.

Perguntas:

- Explique como a criação de índices melhora a performance das consultas em bancos de dados NoSQL.
- Por que a projeção de campos, como {"nome": 1, "preco": 1}, pode melhorar a performance de uma consulta em MongoDB?
- Descreva uma situação em que o uso de $lookup pode prejudicar a performance em MongoDB, e como isso pode ser otimizado.