Map Reduce, Geoposicionamento e GridFS

QXD0099 - Desenvolvimento de Software para Persistência

Universidade Federal do Ceará - *Campus* Quixadá

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

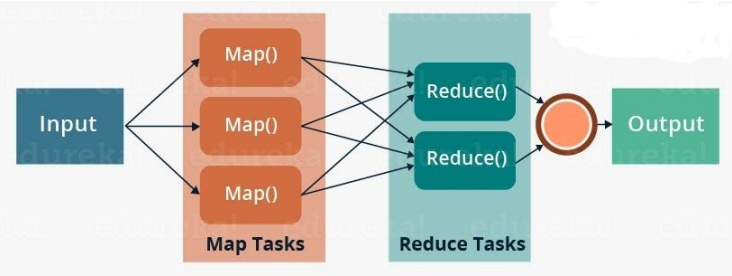
Agenda

- O que é MapReduce?
- Exemplo de MapReduce
- Geoposicionamento no MongoDB
- O que é Geoposicionamento?
- Criando Índices Geoespaciais
- Exemplo de Dados Geoespaciais
- Consultas Geoespaciais
- Encontrar locais dentro de uma área (polígono)
- GridFS: Armazenamento de Arquivos no MongoDB

O que é MapReduce?

- MapReduce é um modelo de programação usado para processar e analisar grandes volumes de dados de forma distribuída. No MongoDB, ele é usado para processamentos em larga escala, como análise de logs e cálculos estatísticos.

- **O MapReduce opera em duas fases:**
  - Map: Extrai e transforma os dados.
  - Reduce: Agrega e processa os resultados.




```
Collection
        db.employee.mapReduce(
Map     function() { emit(this.firstName, this.salary); },
Reduce  function(key, values) {return Array.sum(values)}, {
query       query:{salary: {$lt : 5000}},
output      out:"result"
        }
    ).find()
```

Exemplo de MapReduce

- O MongoDB permite executar MapReduce através do método .mapReduce().
- Suponha que temos uma coleção de vendas (sales) com a seguinte estrutura:

```
{ "_id": 1, "produto": "Notebook", "categoria": "Eletrônicos", "valor": 3500, "quantidade": 2 }
{ "_id": 2, "produto": "Celular", "categoria": "Eletrônicos", "valor": 2000, "quantidade": 1 }
{ "_id": 3, "produto": "Fone de Ouvido", "categoria": "Acessórios", "valor": 150, "quantidade": 5 }
```

- Queremos calcular o total de vendas por categoria. Podemos usar MapReduce:

```
resultado = db.sales.map_reduce(
    "function() { emit(this.categoria, this.valor * this.quantidade); }",
    "function(chave, valores) { return Array.sum(valores); }",
    "total_vendas_por_categoria"
)
for doc in resultado.find():
    print(doc)
```

Exemplo de MapReduce

- Saída esperada:

```
{ "_id": "Eletrônicos", "value": 9000 }
{ "_id": "Acessórios", "value": 750 }
```

- **MapReduce vs Aggregation Framework**
- O MapReduce pode ser substituído pelo Aggregation Framework, que é mais eficiente para a maioria dos casos. Exemplo:

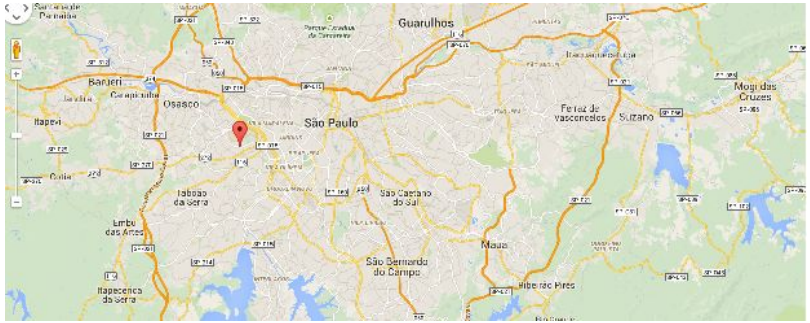
```
resultado = db.aggregate([
    {"$group": {"_id": "$categoria", "total": {"$sum": {"$multiply": ["$valor", "$quantidade"]}}}}
])

for doc in resultado:
    print(doc)
```

O que é Geoposicionamento?

- O MongoDB suporta dados geoespaciais, permitindo armazenar e consultar localizações no formato GeoJSON.

- Ele oferece suporte a tipos de índice espacial como:
  - 2dsphere (para consultas em um globo)
  - 2d (para projeções bidimensionais)



Criando Índices Geoespaciais

- Vamos criar um índice para armazenar coordenadas de estabelecimentos:
  - db.create_index([("localizacao", "2dsphere")])

- Exemplo de Dados Geoespaciais
  - Vamos inserir alguns estabelecimentos com coordenadas:

```
db.insert_many([
    {"nome": "Supermercado X", "localizacao": {"type": "Point", "coordinates": [-46.6356, -23.5505]}},
    {"nome": "Farmácia Y", "localizacao": {"type": "Point", "coordinates": [-46.6300, -23.5550]}},
    {"nome": "Restaurante Z", "localizacao": {"type": "Point", "coordinates": [-46.6200, -23.5400]}}
])
```

Consultas Geoespaciais

- Encontrar locais próximos
- Podemos encontrar os estabelecimentos próximos a uma coordenada específica:

```
# Coordenada de referência
ponto_referencia = {"type": "Point", "coordinates": [-46.6350, -23.5500]}

# Consulta para encontrar locais próximos dentro de um raio de 5000 metros
resultado = db.find({
    "localizacao": {
        "$near": {
            "$geometry": ponto_referencia,
            "$maxDistance": 5000  # Distância máxima em metros
        }
    }
})

# Exibir resultados
for doc in resultado:
    print(doc)
```

Encontrar locais dentro de uma área (polígono)

- Podemos buscar todos os locais dentro de uma região definida:

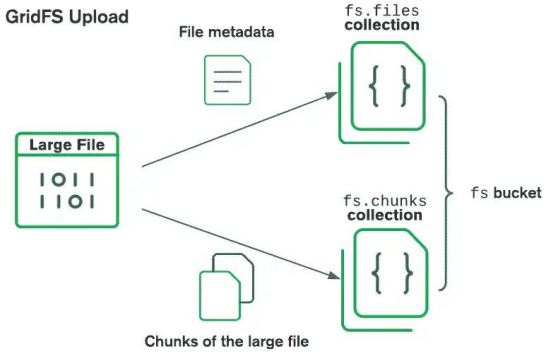
```
# Definir o polígono de busca
poligono = {
    "type": "Polygon",
    "coordinates": [[
        [-46.6400, -23.5600],
        [-46.6200, -23.5600],
        [-46.6200, -23.5400],
        [-46.6400, -23.5400],
        [-46.6400, -23.5600]  # Fechando o
polígono
    ]]
}
```

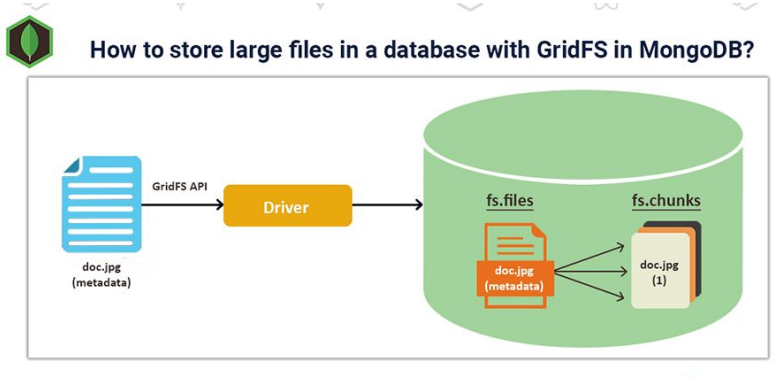
```
# Consulta para encontrar locais dentro do polígono
resultado = db.find({
    "localizacao": {
        "$geoWithin": {
            "$geometry": poligono
        }
    }
})

# Exibir resultados
for doc in resultado:
    print(doc)
```

GridFS: Armazenamento de Arquivos no MongoDB

- O MongoDB não armazena arquivos diretamente, mas usa o GridFS para dividir grandes arquivos em partes menores chamadas de chunks (padrão: 255 KB cada). Isso permite armazenar arquivos grandes de forma eficiente.





Como Funciona o GridFS?

- **Ele divide os arquivos em duas coleções:**
  - fs.files: Metadados sobre o arquivo (nome, tamanho, tipo, etc.).
  - fs.chunks: Partes do arquivo armazenadas.

- **Upload de Arquivo via MongoDB Shell**
  - mongofiles -d meuBanco put meuArquivo.pdf

- **Download de Arquivo**
  - mongofiles -d meuBanco get meuArquivo.pdf

Usando GridFS

```python
# Upload de arquivo para GridFS
@app.post("/upload/")
async def upload_arquivo(file: UploadFile = File(...)):
    file_id = fs.put(file.file, filename=file.filename)
    return {"mensagem": "Arquivo enviado com sucesso", "file_id": str(file_id)}
```

Usando GridFS

```python
@app.get("/download/{file_id}")
async def download_arquivo(file_id: str):
    try:
        object_id = ObjectId(file_id)  # Converter file_id para ObjectId
        file = fs.get(object_id)

        return StreamingResponse(file, media_type="application/octet-stream", headers={
            "Content-Disposition": f"attachment; filename={file.filename}"
        })
    except gridfs.NoFile:
        raise HTTPException(status_code=404, detail="Arquivo não encontrado")
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Usando GridFS

```
@app.get("/arquivos/")
async def listar_arquivos():
    arquivos = db["uploads.files"].find({}, {"_id": 1, "filename": 1})  # Buscar arquivos com ID e nome

    return [{"file_id": str(doc["_id"]), "filename": doc["filename"]} for doc in arquivos]
```

Usando GridFS

```python
@app.delete("/delete/{file_id}")
async def deletar_arquivo(file_id: str):
    try:
        object_id = ObjectId(file_id)  # Converter string para ObjectId
        fs.delete(object_id)  # Deletar arquivo do GridFS
        return {"mensagem": "Arquivo deletado com sucesso"}
    except gridfs.NoFile:
        raise HTTPException(status_code=404, detail="Arquivo não encontrado")
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

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 Graham. 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