**Atividade de Banco de Dados Não Relacional – GeoInsight**

**1. Criação do Banco de Dados**

* **Nome do Banco de Dados:** geoinsight\_db
* **Estrutura das Coleções:**
  + **locations**: Armazenará informações sobre os locais que os usuários pesquisam ou salvam. Cada documento pode conter as coordenadas, o nome de exibição (ex: "São José dos Campos, Brasil") e, possivelvelmente, um ID de usuário associado.
  + **stac\_items**: Guardará os metadados dos itens STAC (Spatiotemporal Asset Catalog) retornados pela API do INPE. Estes são os registros de dados de satélite para diferentes coleções (Landsat, Sentinel, etc.).
  + **users**: Para armazenar dados dos usuários da aplicação, como nome, email e, futuramente, preferências ou locais salvos.

**2. Modelagem de Relacionamentos**

No MongoDB, a escolha entre incorporar (embedding) e referenciar (referencing) depende dos padrões de acesso aos dados e da natureza do relacionamento entre eles.

**A. Exemplo de Embedding (Documentos Incorporados)**

Usaremos embedding na coleção users para armazenar os locais favoritos de cada usuário.

* **Relação:** Um usuário pode ter *vários* locais favoritos, mas essa quantidade é geralmente pequena e os dados dos locais são acessados sempre junto com o perfil do usuário.
* **Estrutura do Documento users:**

JSON

{

"\_id": ObjectId("67c5e2b2a273e8a353e21248"),

"name": "Alisson Franco",

"email": "alisson.franco@email.com",

"favorite\_locations": [

{

"display\_name": "São José dos Campos, SP, Brasil",

"latitude": -23.1791,

"longitude": -45.8872,

"added\_at": ISODate("2025-10-19T21:00:00Z")

},

{

"display\_name": "Campinas, SP, Brasil",

"latitude": -22.9056,

"longitude": -47.0608,

"added\_at": ISODate("2025-10-20T11:30:00Z")

}

]

}

**B. Exemplo de Referencing (Documentos Referenciados)**

Usaremos referencing para ligar um location aos múltiplos stac\_items (dados de satélite) que podem existir para ele.

* **Relação:** Um único local no mapa pode corresponder a *milhares* de registros de satélite ao longo do tempo. Incorporar todos esses registros dentro do documento do local o tornaria imenso e ineficiente.
* **Estrutura do Documento locations:**

JSON

{

"\_id": ObjectId("a8d6f3c3b384f9b464f32359"),

"display\_name": "São José dos Campos, SP, Brasil",

"latitude": -23.1791,

"longitude": -45.8872

}

* **Estrutura do Documento stac\_items (com referência):**

JSON

{

"\_id": ObjectId("b9e7a4d4c495aa0575a43460"),

"stac\_id": "S2-16D\_V2\_21LZH\_20251015",

"collection": "S2-16D-2",

"datetime": ISODate("2025-10-15T13:45:00Z"),

"cloud\_cover": 15.5,

// Referência ao ID do local correspondente

"location\_id": ObjectId("a8d6f3c3b384f9b464f32359")

}

**C. Justificativa da Escolha**

* **Use Embedding (Incorporar) quando:**
  + A relação é "um para poucos" (um usuário para alguns locais favoritos).
  + Os dados são acessados juntos. Raramente você vai querer os dados de um usuário sem seus locais favoritos.
  + A consistência dos dados é importante e eles não mudam com frequência.
* **Use Referencing (Referenciar) quando:**
  + A relação é "um para muitos" (ou "um para milhares", como no nosso caso de um local para inúmeros registros de satélite).
  + Os documentos "filhos" (os stac\_items) são muito grandes ou em grande número. Incorporá-los criaria documentos gigantes, excedendo o limite de 16MB do MongoDB e degradando a performance.
  + Os dados são acessados de forma independente. Por exemplo, podemos querer consultar todos os stac\_items de uma determinada data, independentemente do local.

**3. Scripts MongoDB**

JavaScript

// --- 1. Selecionando o Banco de Dados ---

// Este comando cria o banco de dados se ele não existir.

use geoinsight\_db;

// --- 2. Inserção de Documentos (Embedding) ---

// Usando insertOne para adicionar um usuário com seus locais favoritos incorporados.

db.users.insertOne({

"name": "Georgia Mantchev",

"email": "georgia.m@email.com",

"favorite\_locations": [

{

"display\_name": "Jacareí, SP, Brasil",

"latitude": -23.3054,

"longitude": -45.9659,

"added\_at": new Date()

}

]

});

// --- 3. Inserção de Documentos (Referencing) ---

// Primeiro, inserimos um local e guardamos seu ID.

const sjcLocation = db.locations.insertOne({

"display\_name": "São José dos Campos, SP, Brasil",

"latitude": -23.1791,

"longitude": -45.8872

});

const sjcId = sjcLocation.insertedId;

// Agora, usamos insertMany para adicionar vários STAC items referenciando o ID do local.

db.stac\_items.insertMany([

{

"stac\_id": "LC08\_L1TP\_219076\_20251001",

"collection": "LANDSAT-8-L1",

"datetime": new Date("2025-10-01T13:30:00Z"),

"cloud\_cover": 5.2,

"location\_id": sjcId // Referência

},

{

"stac\_id": "S2A\_MSIL2A\_20251003",

"collection": "S2-16D-2",

"datetime": new Date("2025-10-03T13:50:00Z"),

"cloud\_cover": 25.8,

"location\_id": sjcId // Referência

},

{

"stac\_id": "S2B\_MSIL2A\_20251008",

"collection": "S2-16D-2",

"datetime": new Date("2025-10-08T13:50:00Z"),

"cloud\_cover": 10.1,

"location\_id": sjcId // Referência

}

]);

// --- 4. Consultas de Validação ---

// Consulta básica para ver todos os usuários.

db.users.find();

// Consulta com projeção: mostrar apenas o nome do usuário e o nome de seus locais favoritos.

db.users.find({}, { "name": 1, "favorite\_locations.display\_name": 1, "\_id": 0 });

// Consultando todos os STAC items para um local específico usando seu ID.

// (Substitua o ObjectId pelo ID real gerado na sua inserção)

db.stac\_items.find({ "location\_id": ObjectId("a8d6f3c3b384f9b464f32359") });

// --- 5. Consultas com Operadores Lógicos ---

// Encontrar itens da coleção 'S2-16D-2' que tenham poucas nuvens (cloud\_cover < 20)

// E que sejam de uma data específica ou posterior.

db.stac\_items.find({

$and: [

{ "collection": "S2-16D-2" },

{ "cloud\_cover": { $lt: 20 } },

{ "datetime": { $gte: new Date("2025-10-05T00:00:00Z") } }

]

});

**4. Comparação com SQL (PostgreSQL)**

Se estivéssemos modelando essa estrutura em um banco de dados relacional como o PostgreSQL, usaríamos tabelas e chaves estrangeiras.

* Modelagem em SQL:

Teríamos três tabelas principais: users, locations, e stac\_items. Para o relacionamento "um-para-muitos" dos locais favoritos do usuário, criaríamos uma tabela de junção (ou pivô) chamada user\_favorite\_locations.

SQL

-- Tabela de usuários

CREATE TABLE users (

id SERIAL PRIMARY KEY,

name VARCHAR(100),

email VARCHAR(100) UNIQUE

);

-- Tabela de locais

CREATE TABLE locations (

id SERIAL PRIMARY KEY,

display\_name VARCHAR(255),

latitude DECIMAL(10, 6),

longitude DECIMAL(10, 6)

);

-- Tabela de junção para locais favoritos

CREATE TABLE user\_favorite\_locations (

user\_id INTEGER REFERENCES users(id),

location\_id INTEGER REFERENCES locations(id),

added\_at TIMESTAMP,

PRIMARY KEY (user\_id, location\_id)

);

-- Tabela para os dados de satélite

CREATE TABLE stac\_items (

id SERIAL PRIMARY KEY,

stac\_id VARCHAR(100) UNIQUE,

collection VARCHAR(50),

datetime TIMESTAMP,

cloud\_cover REAL,

location\_id INTEGER REFERENCES locations(id) -- Chave estrangeira

);

* Exemplo de Consulta Equivalente:

A consulta MongoDB para buscar itens com poucas nuvens da coleção S2-16D-2 para um local específico seria assim em SQL, exigindo um JOIN:

SQL

SELECT

s.stac\_id,

s.collection,

s.datetime,

s.cloud\_cover

FROM

stac\_items s

JOIN

locations l ON s.location\_id = l.id

WHERE

l.display\_name = 'São José dos Campos, SP, Brasil'

AND s.collection = 'S2-16D-2'

AND s.cloud\_cover < 20;

Isso demonstra como o SQL requer JOINs explícitos para combinar dados de diferentes tabelas, enquanto o MongoDB pode fazer isso com consultas mais diretas ou através do design do schema (embedding).

**5. Prints de Execução**

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.