**Relatório do Projeto de Processamento e Análise de Dados de Desmatamento do Bioma Cerrado**

**Data:** 13 de Abril de 2025

**Local:** Piracicaba, Estado de São Paulo, Brasil

**Autor:** Fernando Nunes da Silva

**1. Fluxo de Trabalho e Abordagem Utilizada na Solução**

O fluxo de trabalho implementado para processar e analisar os dados de desmatamento do bioma Cerrado seguiu as seguintes etapas principais:

1. **Configuração e Inicialização:**
   * Criação de uma classe TerraBrasilisWFS para encapsular a lógica de interação com o serviço WFS do TerraBrasilis. Esta classe é responsável por construir as URLs das requisições e realizar o download dos dados, implementando mecanismos de retry para lidar com falhas de conexão.
   * Definição dos parâmetros da requisição WFS, incluindo o bioma de interesse (prodes-cerrado-nb), o tipo de dado (yearly\_deforestation), o período desejado e o formato de saída (application/json).
2. **Download dos Dados:**
   * Utilização da função download\_data da classe TerraBrasilisWFS para obter os dados de desmatamento do GeoServer do TerraBrasilis, aplicando os filtros de data e paginação definidos.
   * Verificação da resposta do servidor para garantir que os dados foram baixados com sucesso.
3. **Estruturação e Validação Inicial dos Dados:**
   * Conversão da resposta JSON para uma estrutura de dados Python (lista de dicionários representando as feições).
   * Iteração sobre cada feição para extrair as propriedades relevantes (data de emissão, área, outros atributos) e a geometria.
4. **Processamento e Garantia da Qualidade Geoespacial:**
   * **Validação de Geometrias:** Implementação de uma função validate\_geometry para verificar a validade das geometrias utilizando a biblioteca Shapely. **Aqui é necessário a ajuda de algum especialista em GEO para orientar quais técnicas e prováveis erros que necessitam de tratamento.**
   * **Manipulação de Coordenadas:** As coordenadas das geometrias (originalmente em formato GeoJSON) são extraídas e formatadas como strings WKT (Well-Known Text) para armazenamento no banco de dados espacial.
   * **Consistência do Sistema de Coordenadas:** Todas as geometrias são explicitamente definidas com o SRID 4326 (WGS 84), garantindo a consistência para análises geoespaciais futuras.
5. **Armazenamento no Banco de Dados PostgreSQL:**
   * **Criação do Esquema:** Criação de um esquema específico (raw\_data) no banco de dados PostgreSQL para organizar os dados brutos. A criação do esquema é feita verificando se ele já existe antes de tentar criá-lo novamente.
   * **Definição da Tabela:** Utilização do SQLAlchemy para definir o esquema da tabela deforestation dentro do esquema raw\_data, incluindo colunas.
   * **Inserção de Dados:** Os dados processados (incluindo as geometrias validadas e corrigidas) são inseridos na tabela deforestation utilizando o SQLAlchemy Core para garantir a correta adaptação dos tipos de dados, especialmente para a coluna de geometria.
   * **Prevenção de Redundância:** Implementação de uma lógica básica para verificar a existência de registros com base em um identificador único (se disponível nas propriedades) antes da inserção, evitando a duplicação de dados.
6. **Otimização de Consultas:**
   * **Criação de Índices:** Criação de índices nas colunas publish\_year (índice simples) e geom (índice espacial GIST) para otimizar a performance de consultas futuras que envolvam filtros por data ou operações espaciais. Um índice adicional foi criado na coluna year como exemplo de otimização para outros atributos.

**2. Dificuldades Encontradas e Como Foram Resolvidas**

* **Desconhecimento do ambiente GEO:** Inicialmente com consultas na web, realização de alguns downloads e entendimento da estrutura dos dados de retorno do ambiente.
* **Erro de Decodificação JSON (JSONDecodeError):** Inicialmente, o script encontrou erros ao tentar decodificar a resposta do GeoServer como JSON. Isso foi resolvido através da verificação do código de status da resposta HTTP e da inspeção do conteúdo da resposta como texto (response.text) para identificar se o servidor estava realmente retornando JSON válido. A causa geralmente era uma resposta vazia ou um erro do servidor não formatado em JSON.
* **Problemas com Geometrias Inválidas ('NoneType' object has no attribute 'startswith'):** Erros na função de validação de dados GEO. Isso ocorreu porque a lógica inicial de extração da geometria não tratava casos onde naõ se tinha certeza do tipo de dado GEO utilizado na estrutara de dados da API. A solução envolveu adicionar verificações mais robustas para garantir a existência e o tipo correto das partes da geometria antes de processá-las. A utilização da biblioteca Shapely para criar objetos geométricos a partir do GeoJSON também ajudou a lidar com diferentes estruturas de coordenadas de forma mais robusta.

**3. Otimizações Aplicadas para Melhorar a Performance**

As seguintes otimizações foram aplicadas para melhorar a performance do processo:

* **Indexação de Colunas:** Criação de índices nas colunas publish\_year e year, que são prováveis de serem usadas em filtros de consultas. Isso acelera a busca por registros com base nesses atributos.
* **Indexação Espacial:** Criação de um índice espacial do tipo GIST na coluna geom. Índices espaciais são cruciais para otimizar consultas que envolvem operações espaciais, como encontrar áreas de desmatamento dentro de uma determinada região ou que se intersectam com outras feições geográficas.
* **Download Paginação (Implementação Futura):** Embora não totalmente implementado no código base gerado neste versão, a estratégia de usar o parâmetro maxFeatures na requisição WFS sugere a intenção de implementar paginação para o download de grandes volumes de dados, evitando sobrecarregar a memória e melhorando a eficiência do download.
* **Utilização de Tipos de Dados Eficientes:** O uso do tipo JSONB no PostgreSQL para armazenar as propriedades permite consultas eficientes dentro dos dados JSON. O tipo GEOMETRY com a extensão PostGIS garante o armazenamento otimizado e a capacidade de realizar operações espaciais eficientes no banco de dados.

**4. Sugestões de Melhorias para Futuras Versões do Processo**

* **Implementação Completa de Paginação:** Para lidar com grandes volumes de dados, implementar um loop completo para percorrer todas as páginas de resultados do WFS, utilizando parâmetros como startIndex e potencialmente analisando o número total de feições retornadas para determinar quantas requisições são necessárias.
* **Paralelização do Processamento:** Para acelerar o processamento de um grande número de feições, considerar a utilização de técnicas de paralelização (exemplo usando as bibliotecas multiprocessing ou asyncio em Python) para validar e preparar os dados para inserção no banco de dados.
* **Tratamento Mais Robusto de Diferentes Tipos de Geometria:** Expandir a lógica para lidar com outros tipos de geometria que podem estar presentes nos dados do TerraBrasilis (e.g., Point, LineString), definindo os tipos de coluna apropriados no banco de dados e implementando a lógica de validação e correção correspondente.
* **Consultar um especialista em dados Geo:** Para trazer maior robustez ao código, verificar pontos de atenção necessários nos processamentos de dados GEO.
* **Mecanismos de Detecção e Tratamento de Erros Mais Detalhados:** Implementar um sistema de logging mais abrangente para registrar informações sobre o processo, incluindo erros, avisos e estatísticas de processamento. Isso facilitaria o diagnóstico de problemas e o monitoramento da execução.
* **Configuração Flexível:** Adaptar o código para trabalhar com parâmetros de configuração salvos em arquivo de configuração (exemplos, URL do GeoServer, nome do bioma, período), como que o realizado para os parâmetros de banco de dados tornando o processo mais flexível e fácil de adaptar a diferentes cenários.
* **Validação de Dados Não Espaciais:** Além da validação geométrica, implementar validações para os outros atributos dos dados (e.g., verificar se as datas estão em um formato correto, se os valores de área estão dentro de limites razoáveis).
* **Implementação de Testes Unitários e de Integração:** Criar testes automatizados para verificar o perfeito funcionamento das diferentes partes do processo (exemplos, download de dados, validação geométrica, inserção no banco de dados).
* **Criação de classes / Interfaces adicionais:** padronização de métodos que possam ser reutilizados, como criar classes e interfaces para operações com o banco de dados e arquivos. Garantindo padrão e manutenibilidade do código.

**5. Soluções para Corrigir Geometrias Inválidas**

Em consultas na Internet, diversas soluções podem ser utilizadas para corrigir geometrias inválidas, e a escolha da melhor solução depende da natureza específica das invalidades encontradas nos dados.

Neste projeto, optei pela solução abaixo, pois foi a que mais me ajudou a trazer resultados e dar garantia de uniformidade para o processamento dos dados, conforme detalhamento:

* **Converter geometria para WKT:** Em breve pesquisa na Internet verifiquei que o formato WKT traz interoperabilidade, sendo um formato padrão, facilitando a troca de informações geométricas entre diferentes sistemas e softwares SIG.
* **shape() (Shapely):** A função shape() da biblioteca Shapely é utilizada para criar um objeto geométrico Shapely a partir da estrutura GeoJSON da geometria baixada. Durante esse processo de criação, a Shapely tenta interpretar a geometria e pode, em alguns casos, normalizar ou corrigir pequenas inconsistências na estrutura da geometria original, resultando em um objeto Shapely válido.
* **WKTElement() (GeoAlchemy2):** Ao criar um objeto WKTElement a partir da representação WKT da geometria Shapely, o GeoAlchemy2 prepara a geometria para ser armazenada na coluna GEOMETRY do PostgreSQL. Embora o WKTElement em si não realize uma correção ativa da geometria, o fato de a geometria ter sido previamente processada pela Shapely (através da função shape()) significa que algumas invalidades podem já ter sido tratadas.

.

**6. Número Total de Áreas Encontradas de Desmatamento com Geometrias Válidas para todo o Bioma Cerrado**

O número total de áreas encontradas de desmatamento com geometrias válidas para todo o bioma Cerrado, para o ano de 2024, com base nos dados processados e armazenados no banco de dados, é de **54.743**

Consulta realizada no banco de dados:

**SELECT** **year**, **COUNT**(\*)

**FROM** ***raw\_data***.***deforestation***

**WHERE** **ST\_IsValid**(***geom***) = **true**

**group** **by** **year**;

Resultado do banco de dados para a query:

