Projeto para a vaga de Analytics Enginner

Introdução

Esse documento tem como objetivo apresentar a resolução dos problemas e solicitações propostas nas questões 1 e 2 para a vaga de Analytics Enginner.

Neste documento falarei das decisões técnicas e de negocio que me levaram a chegar a conclusão de que as ferramentas utilizadas e modelagens atendem ao que me foi proposto.

Todo o projeto esta versionado no Github e Docker Hub e podem ser consultados através dos links:

https://hub.docker.com/repository/docker/thiagoadrsantos/pyspark/general
https://github.com/ThiagoAsantos/proj_pyspark
https://github.com/ThiagoAsantos/proj_pyspark_code

O projeto foi feito em um sistema operacional Linux e o banco de dados utilizado é o Postgres SQL

Na sequencia abordaremos os seguintes temas:

Problema 1

- Estrutura do projeto (Ferramentas utilizadas na construção da estrutura do projeto)
- Estrutura de dados (Ferramentas e lógica utilizada na construção do projeto)
- Disponibilidade (Resiliência e flexibilidade da aplicação)
- Escalabilidade (Estratégia utilizada levando em consideração a escalabilidade)
- Performance (Estratégia utilizada levando em consideração a performance)
- Conclusão

Problema 2

- Solução proposta

Problema 1 - Estrutura do projeto

O projeto foi estruturado em contêineres Docker. Um contêiner Docker é uma unidade de software que é executado de forma independente e pode ser executado em diversos ambientes de forma consistente e isolada.
O GitHub é uma plataforma de desenvolvimento e versionamento de código.
O Docker Hub é um serviço aonde podemos guardar e compartilhar contêineres.

Para o problema 1 utilizei 2 projetos com os nomes proj_pyspark e proj_pyspark_code.
O proj_pyspark é responsável pela preparação da estrutura do projeto (imagem) aonde são instaladas todas as dependências, variáveis de ambientes e parametrizações no sistema operacional da imagem (Linux ubuntu) para o projeto.
O projeto proj_pyspark_code é utilizado para definir configurações a partir da imagem que foi criada no projeto proj_pyspark. Nesse projeto utilizo um segundo contêiner com a imagem do Postgres SQL que é o banco de dados que escolhi para a resolução do problema.

Dentro do projeto **proj_pyspark** temos:

- jdk-17_linux-x64_bin.tar.gz - Java que será instalado na imagem do contêiner para utilização do pyspark
- postgresql-42.7.1.jar – Jar que será instalado na imagem para que o pyspark consiga interagir com o Postgres SQL
- docker-compose.yaml – arquivo de configuração do contêiner
- Dockerfile – Arquivo de definição da imagem docker e scripts

Dentro do projeto **proj_pyspark_code** temos:

- pg_data – Pasta que serve para persistir os dados do contêiner do Postgres em minha maquina local
- pyspark_data - Pasta que serve para persistir os dados do contêiner do pyspark em minha maquina local
- backup_bd_postgress – Banco de dados completo com estrutura e dados que foram utilizados no projeto. Para utilizar basta realizar um restore com esse arquivo em uma instalação do postgres
- docker-compose.yaml - Arquivo de configuração dos contêineres
- Docker-file - Arquivo de definição da imagem docker e scripts
- main.py – Pipeline pyspark utilizado para realizar a ingestão dos dados no postgres
- result_script_jane.csv – Resultado do script gerado que foi solicitado pela Jane
- script_criar_banco.sql – Script SQL utilizado para criar o banco de dados, tabelas e relacionamentos
- script_inserir_dados.sql – Script utilizado para pegar os dados das tabelas de stage e inserir nas tabelas definitivas
- script_sql_jane.sql – Query SQL desenvolvida para atender a solicitação da Jane

Estrutura de dados

O pipeline de dados foi feito com pyspark e Postgres SQL

Disponibilidade

O projeto utiliza de ferramentas e arquitetura que pode ser configurado para ter alta disponibilidade. Podemos colocar replicas do banco de dados em locais físicos próximos um dos outros ou em outras regiões e até espalhadas pelo mundo e se por algum motivo o banco de dados cair, outro poderá assumir. Podemos utilizar uma ferramenta orquestradora de contêineres como o Kubernets.

Escalabilidade

Com um projeto em contêineres ou micro serviços, estaremos prontos para trabalhar com Big Data. Caso ocorra um aumento de dados ou ter a necessidade de trabalhar com muitos dados, podemos trabalhar com a ingestão dos dados de forma distribuída com clusters com o pyspark e utilizar um sistema de arquivo que armazenará os dados também de forma distribuída como um HDFS e/ou dados em estrutura de arquivos como o parque e uma estrutura de dados de Data Lake

Performance

Com a utilização da estrutura descrita no item de Escalabilidade, podemos guardar e processar dados em paralelo e consequentemente ganhar performance

Conclusão

Concluo que mesmo com poucos dados no dataset e pensando no volume e na variedade de dados que são gerados diariamente, optei por uma solução mais flexível e robusta para execução do problema 1

Com os links dos repositórios você poderá executar todo o projeto com poucas linhas de código.

O projeto está pronto para ser baixado e configurado para criar o ambiente o zero em qualquer Serviço de Nuvem, OnPremisse ou maquina local.

O Projeto esta pronto para ser adaptado a um modelo de CI/CD e execução em Orquestradores de contêineres.

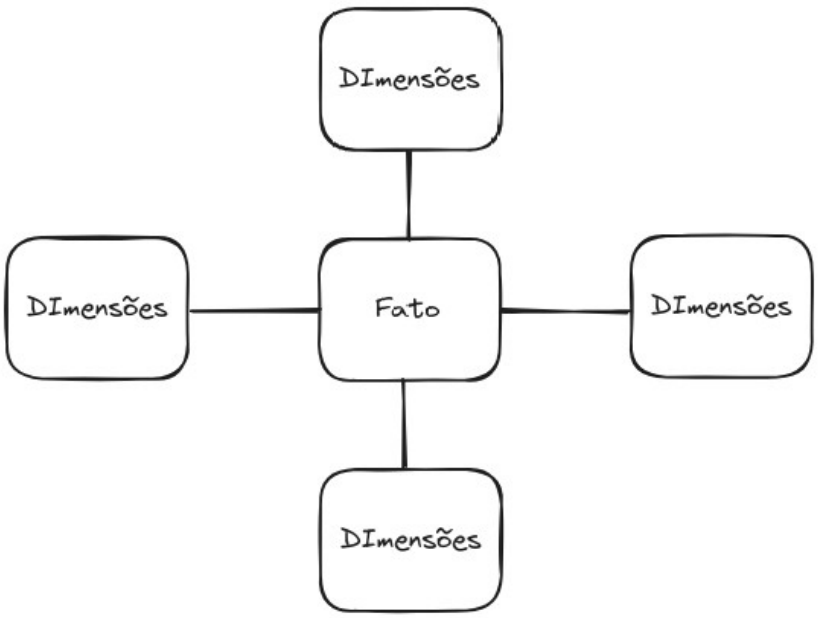
Problema 2 - Solução proposta

Para um ambiente analítico e que será utilizado para ferramentas de Data Visualization, a estrutura que está montada não é apropriada.

Em uma estrutura de analise de dados o foco será a leitura de muitos dados e que são históricos, a estrutura mostrada no problema 1 está apropriada para ambientes OLTP aonde a preocupação não é somente com a leitura e sim com a escrita, com a integridade referencial e evitar duplicidades e inconsistências de dados.

A estrutura mostrada tem muitos relacionamentos o que torna o processo de analise mais difícil e com menos performance. As queries ficam mais complexas e lentas pois são necessárias agregações e muitos joins para obter o resultado como no exemplo do problema 1, e também a query acaba sendo muito complexa assim diminuindo a produtividade tanto do Engenheiro quanto a do analista.

Para resolver essa questão proponho a criação de um modelo OLAP aonde teremos uma ou mais tabelas fatos e dimensões conforme exemplo a seguir



Esse exemplo conhecido como Star Schema possui os dados com os níveis de granularidade já processados e possuem menos relacionamentos fazendo com que a query de analise seja mais rápida e fácil de desenvolver

Para essa solução eu proponho que seja coletado os dados do ambiente produtivo e colocados em uma Stage e na sequencia um segundo pipeline atualizará as tabelas fatos e dimensões do projeto conforme exemplo a seguir:

O processo de migração para nova estrutura poderá ser feito em paralelo com a estrutura atual.
Após a finalização o processo rodará por um tempo em paralelo com a estrutura atual, os valores são batidos e as analises de ganhos de performance e estratégicas levantadas.
Apontaremos os relatórios para a nova estrutura e por fim a estrutura antiga sera desligada.