Olá Arquiteto, tudo bem?

Estamos com um projeto novo e gostaria de discutir uma proposta para implementação, ok?

Precisamos fazer a ingestão de uma base fria, composta por arquivo CSV/TSV zipados. Ao todos temos 8 arquivos, sendo todos com tamanho inferior a 1GB (zipado) cada.

Devemos fazer a ingestão e, depois de transformar esses dados, disponibilizá-los em um banco para que o time de BI do cliente possa consumir nossas tabelas.

Eu proponho que inicialmente a gente suba esses arquivos, no S3 da AWS. Se possível, seria interessante já subir de forma descompactada. O arquivo maior vai ter em torno de 3.5GB e o restante em torno de 1GB. Caso não seja possível, fazemos a descompactação dentro do nosso ETL mesmo.

Proponho utilizar o Databricks, tanto para executar o nosso ETL como para guardar as tabelas.

Para que o processamento seja mais ágil, vamos ler os arquivos CSV e transformar em PARQUET.

Como vários serviços são compatíveis com o PARQUET, ficaria mais fácil qualquer alteração alteração de arquitetura depois. Além disso ele é mais compacto e vamos ganhar performance no nosso processamento.

Sugiro que a gente utilize o Unity Catalog do Databricks, que irá facilitar para a equipe de governança.

Apesar de nossa carga inicial ser fria, acredito que possamos utilizar o Delta Lake. Com ele podemos indexar e otimizar os arquivos.

------------------------------------------------------------------------------------------------------------------------

Eu já transformei os dados localmente e você pode encontrá-los em: <https://github.com/lucieness/movies_data/tree/main/db_files>.

Já deixei preparado os arquivos em PARQUET e DELTA, ficando a sua escolha qual deles já carregar.

O modelo desses arquivos estão abaixo:

1. https://github.com/lucieness/movies\_data/tree/ecore/db\_files/main/movie\_star\_rating

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **campo** | **tipo** |  |  |  |
| customer\_id | INT |  |  |  |
| product\_id | STRING |  |  |  |
| product\_title | STRING |  |  |  |
| rating\_date | STRING |  |  |  |
| star\_rating | INT |  |  |  |
| platform | STRING |  |  |  |
|  | | | | |
| platform: |  |  |  |  |
| NETFLIX é identificada com a sigla "NFLX" | | | | |
| AMAZON é identificada com a sigla "AMZ" | | | | |

Esses arquivos foram carregados com uma tabela de nome “movie\_star\_rating”.

1. <https://github.com/lucieness/movies_data/tree/main/db_files/delta/netflix_movie_data>

|  |  |
| --- | --- |
| **campo** | **tipo** |
| product\_id | STRING |
| product\_title | STRING |
| year\_of\_release | STRING |

Esses arquivos foram carregados com uma tabela de nome “netflix\_movie\_data”.

Pensei em já carregar uma outra tabela, mas não consegui carregar localmente. Como os dados ainda não estão sendo utilizados (conteúdo da review), acredito que no momento não vá ter necessidade.

-----------------------------------------------------------------------------------------------------------------------------

O cliente já solicitou alguns dados e gerei algumas Queries SQL. O documento <https://github.com/lucieness/movies_data/blob/main/sql/Queries%20SQL.docx> contem todas as questões, com suas respectivas queries.

Disponibilizei o notebook que rodei no Databricks: <https://github.com/lucieness/movies_data/blob/main/sql/databricks_notebook.py>.

Importante: não fiz ainda nenhum tipo de otimização e particionamento dos dados. Acredito que possamos utilizar o Z-ORDER para melhorar performance.

Outro ponto importante: Acho que podemos utilizar a arquitetura Medalhão para os dados.

Como precisamos manipular bastante os dados iniciais, podemos ter uma STAGING AREA para gerar o PARQUET.

Apesar da carga ser fria, podemos criar um workflow dentro do próprio Databricks. Assim deixamos documentado o que foi feito.

Arquiteto, espero que tenha entendido a minha proposta. Por favor, caso não tenha entendido algo, só me chamar.

Existem dados adicionais no repositório <https://github.com/lucieness/movies_data/>.

Obrigada,

Luciene Silva

Engenheira de Dados