INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO
CÂMPUS CAMPINAS

DEIVE AUDIERES LEAL

AVALIAÇÃO DE ESTRATÉGIAS PARA A CRIAÇÃO DE PIPELINE DE DADOS VOLTADO PARA CIÊNCIAS HUMANAS

CAMPINAS
2021

DEIVE AUDIERES LEAL

AVALIAÇÃO DE ESTRATÉGIAS PARA A CRIAÇÃO DE PIPELINE DE DADOS VOLTADO PARA CIÊNCIAS HUMANAS

Trabalho de Conclusão de Curso apresentado como exigência parcial para obtenção do diploma do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia Câmpus Campinas.

Orientador: Prof. Dr. Samuel Botter Martins.

CAMPINAS

2021

Dados Internacionais de Catalogação na Publicação

Deive Audieres Leal

Avaliação de estratégias para a criação de pipeline de dados voltado para ciências humanas

Trabalho de Conclusão de Curso apresentado como exigência parcial para obtenção do diploma do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo Câmpus Campinas.

Aprovado pela banca examinadora em: _____ de _____ de _____.

BANCA EXAMINADORA

_____

Prof. Dr. Samuel Botter Martins (orientador)

IFSP Câmpus Campinas

_____

Prof.ª Dr.ª Bianca Maria Pedrosa

IFSP Câmpus Campinas

_____

Prof. Me Danilo Douradinho Fernandes

IFSP Câmpus Campinas

Dedico este trabalho a todos aqueles que
não desistem de continuar sonhando,
mesmo perdidos em caminhos tortuosos.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pelo dom da vida e pela oportunidade de concluir mais uma etapa de minha experiência acadêmica. Agradeço a todos os professores e servidores do IFSP Campus Campinas, que contribuíram direta e indiretamente para a conclusão desse trabalho em especial ao Samuel, meu orientador, que entre conversas amenas e direcionamentos assertivos me conduziu até aqui. Agradeço aos meus pais que sempre me ensinaram a seguir em frente, independente do desafio, com sua intensa força de vida e amor. Agradeço aos meus familiares que me deram o suporte necessário: minha irmã Walquíria e meu cunhado Adriano, minha sobrinhas Isabela, Gabriela, Maria Eduarda e Marianne, meus tios Donizete e Madalena e meus primos Milena e José Pedro. Agradeço à Maria, parceira de sonhos reais, por me instigar a pensar fora da zona de conforto, com sua inteligência, sensibilidade e arte de viver. À Patrícia por tornar momentos difíceis suportáveis com sua poesia de muitos tons e conversas sobre o cotidiano e "hipocondisney". Agradeço à Rosemeire, que entre nossas aulas de inglês foi minha psicóloga indireta e amiga confidente. Aos meus amigos e companheiros de curso: Carolina, que me acompanha desde a matrícula e já viu o melhor, o pior e o comum; Bianca, que se juntou a nós na briga cotidiana entre surtos e levezas; Guilherme e Vitória, que chegaram por último, mas que tornaram os momentos vividos mais intensos e memoráveis. E a todos os outros que participaram dessa caminhada meu muito obrigado!

"A força não provém da capacidade física.
Provém de uma vontade indomável."
Mahatma Gandhi

RESUMO

O presente trabalho tem o propósito de realizar o desenvolvimento de *pipeline* de dados tendo como foco a disponibilização desses para análises das ciências humanas a partir da observação de estratégias ferramentais que possibilitam a implementação desses fluxos de dados. Parte-se da premissa que todas as ferramentas e ambiente devem ser de código aberto ou com disponibilização de versão em código aberto, assim como espera-se que o ambiente final de disponibilização dos dados seja de Big Data, para tanto utiliza-se o Hadoop e Hive para armazenamento e interface para a manipulação dos dados. Utiliza-se o modelo de Extração, Carregamento e Transformação, ELT, para caracterizar as tarefas a serem realizadas e o sistema de camadas baseado em *delta lake* é utilizado para armazenamento. Os pipelines foram construídos com Pentaho Data Integration e Airflow com Python utilizando dados públicos brasileiros em três diferentes cenários de extração de dados. O objetivo deste trabalho é diminuir a distância entre as áreas de estudo, criando possibilidades de conectar questões a respostas mais precisas disponíveis nas fontes de dados.

Palavras-chave: ELT, Big Data, Engenharia de Dados, Software Livre.

ABSTRACT

This current work aims at producing the development in data pipeline and the main goal here is to support human science analysis observing tool strategies which make available their implementation. It is based on the premise that every tool and environment must be open source or with an open-source version available and expecting that the final environment of data availability is like Big Data, consequently using Hadoop and Hive to store and do interface for data manipulation. The model of Extract, Load, Transform, ELT, is used to feature the tasks that will be done, and delta lake layers system is used to storage. The pipelines were developed with Pentaho Data Integration and Airflow with Python using Brazilian public data in three different scenarios of data extraction. The goal of this work is to lessen the distance between the areas of studying, creating possibilities for connecting questions to more accurate answers available in data sources.

Keywords: ELT, Big Data, Data Engineer, Software Open Source.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

API	*Application Programming Interface*
BI	*Business Intelligence*
CSV	*Comma-separated values*
DAG	*Directed Acyclic Graph*
DW	*Data Warehouse*
ELT	*Extract, Load, Transform*
ETL	*Extract, Transform, Load*
GB	*Giga Byte*
HDFS	*Hadoop Distributed File System*
HQL	*Hive Query Language*
OLAP	*Online Analytical Processing*
PDI	*Pentaho Data Integration*
SQL	*Structured Query Language*

SUMÁRIO

1 INTRODUÇÃO

Com o aumento acelerado de produção de dados e o amplo uso dentro nos mais diversificados cenários e objetivos, a engenharia de dados se destaca no sentido de manter e criar uma base sólida de dados, que então poderão ser consumidos, seja por cientistas de dados, engenheiros de aprendizado de máquina ou analistas de dados. Dentro desse cenário de crescente uso de dados o engenheiro de dados aparece abarcando entre suas atribuições a construção e manutenção de arquiteturas direcionadas, tais como banco de dados e sistemas de processamento em grande escala. Adentrando essa gama de possibilidades, o ELT — do inglês, *Extract, Load, Transform* — ou extração, carregamento e transformação é uma etapa onde são combinadas diversas fontes de dados (SAS, [20--]).

A extração corresponde ao agrupamento, ou mineração, das mais diversas fontes de dados. Nela os dados são recuperados em sua forma crua, seja de bancos relacionais, planilhas, arquivos, ou outras fontes de armazenamento.

O carregamento dos dados é realizado em um local que possibilita o acesso ao consumidor dos dados – por exemplo, o cientista de dados. Exemplos de locais de acesso são API – do inglês, *Application Programming Interface* – e *data warehouse*.

Data warehouse é um banco de dados estruturado voltado para consultas avançadas, com grande quantidade de dados históricos e direcionado para suportar análises. Nele são centralizados uma grande quantidade de dados advindos de diferentes fontes que passaram pelo processo (RASLAN & CALAZANS, 2014).

A última fase de um processo de ELT, mas que pode ser repetida inúmeras vezes, a transformação visa tornar os dados aptos para a utilização. Assim, são utilizadas técnicas de limpeza, classificação — categóricos e contínuos — e normalização sobre os dados.

Sendo uma etapa tão importante e que tem uma massa de dados que os sistemas comuns não conseguem processar ou armazenar, foram surgindo diferentes ferramentas que auxiliassem no processo de ELT e no armazenamento. Assim, temos desde ferramentas de código aberto, *open source*, a ferramentas proprietárias, mas todas voltadas para maximizar e melhorar as informações que são angariadas e guardadas. Visando o uso de ferramentas *open source* para a execução do ELT, há uma ausência de estudos que comparem as diferentes soluções existentes quando objetiva ambiente de Big Data, bem como quais cenários de extração cada ferramenta é indicada. E aqui apresentamos esse trabalho para auxiliar nesse quesito.

2 OBJETIVOS

2.1 Objetivo Geral

Desenvolver e avaliar diferentes estratégias *open source* para a criação de pipeline de Dados para *big data*, via processo de ELT, com enfoque em ciências humanas.

2.2 Objetivos Específicos

- Desenvolver um processo de ELT aplicável financeiramente e acessível para manutenção e reprodução;
- Armazenar os dados ingeridos pelo processo de ELT em um ambiente de *big data;*
- Utilizar unicamente ferramentas *open source;*
- Integrar dados para que estejam apropriados para análise ou consumo final.

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, detalharemos os principais tópicos e conceitos que fundamentam o desenvolvimento de nosso projeto. As Seções 3.1 a 3.5 focam nos conceitos sobre engenharia de dados e pipeline de dados, enquanto a Seção 3.6 em diante apresentam as tecnologias tipicamente utilizadas para a criação de pipeline de dados.

3.1 Engenharia de Dados

Muitas são as possibilidades para quem deseja se aventurar com dados e desenvolver uma carreira na área, como mostrado na Figura 1. Entre a gama de oportunidades advindas, a engenharia de dados é um campo que se consolida como sendo essencial em ambientes direcionados a dados, e que o cenário exige tratamento de grandes volumes de dados brutos e disponibilidade.

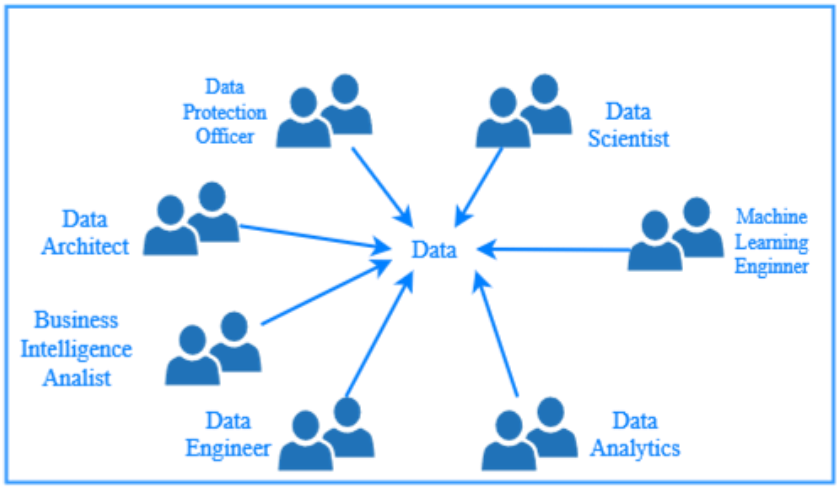

Figura 1 - Carreiras em dados

O *engenheiro de dados* trabalha na garantia de que os dados coletados possam ir da fonte ao consumidor através de fluxos, automáticos ou não, mas que democratizam o acesso aos dados pelas diferentes áreas do negócio tornando-as mais independentes e produtivas. De um profissional dedicado à área é esperado que tenha conhecimento de banco de dados relacionais, não-relacionais, linguagem de programação, em especial SQL, e entendam como otimizar e recuperar dados. Seu trabalho é direcionado de acordo com as regras de negócio para o qual desenvolve o fluxo de dados.

White (2018) observa que o engenheiro de dados pode estar alocado em ao menos três funções principais. A primeira, generalista, é responsável desde a aquisição de dados até a análise e visualização. A segunda é centralizada em pipeline, e trabalha em proximidade com os cientistas de dados na utilização dos dados coletados. Por fim, há o que é centralizado no banco de dados, sua dedicação é centrada no banco de dados analítico, trabalha com o desenvolvimento de esquemas, tabelas, data warehouses e armazenamento distribuído.

Tamir *et al.* (2015) dizem que o engenheiro de dados trabalha na extração e construção de uma grande base de dados possibilitando o desenvolvimento de *insights*. Para eles, engenharia de dados, não se resume à simples manutenção de um repositório para grande volume de dados, mas sobre possibilitar com que todos, desenvolvedores ou não, possam consumi-los nas tarefas diárias, aumentando a base de usuários e dialogando com a universalização e democratização dos dados.

Entre as atividades desempenhadas pelo engenheiro de dados, destaca-se a extração, transformação e carregamento de dados. Na bibliografia é comum encontrarmos a sigla em inglês ETL, – *Extract, Transform, Load* –, assim como sua variante ELT – *Extract, Load, Transform* –, assim sendo, também usaremos as siglas ao invés das variações completas com seus significados. Mas ressaltamos que antes daremos um entendimento geral sobre esse processo, que nos servirá de base para o fluxo de dados proposto na arquitetura deste trabalho.

3.2. Extract, Transform, Load (ETL)

O processo de ETL (Figura 2) é a etapa em que os dados são extraídos da fonte inicial, seja elas um banco ou diversos bancos, com alta ou baixa granularidade, recebem então tratamento, se necessário, e é realizado o carregamento no repositório (Garcia, 2020). No formato ETL, o dado chega pronto ao repositório, imobilizando qualquer intervenção posterior.

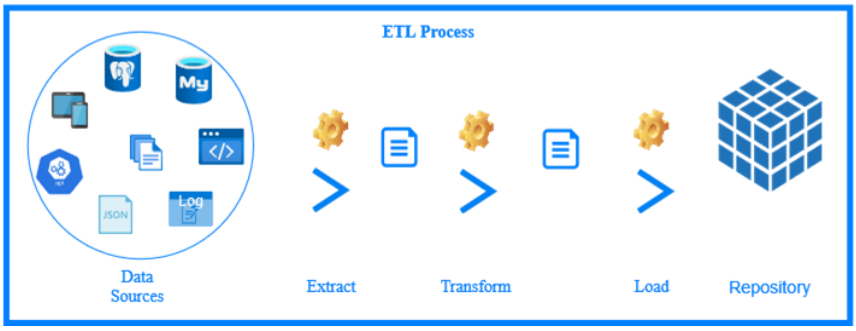

Figura 2 - Processo de ETL

3.2.1. Extração

A extração, *Extract,* é a primeira fase do ETL. Nela é realizada a coleta de dados de diferentes fontes. As origens têm características próprias que são gerenciadas e só então realiza-se a extração. Deve-se observar a estrutura das fontes para que possam ser ajustadas ao repositório que as receberá.

Os dados são recuperados em sua forma crua, seja em bancos relacionais, arquivos estruturados como planilhas e csv, fontes on-line, como API's, e assim por diante. É um processo que, por sua vez, pode ser separado em dois outros, a importação dos dados em si e a atualização dos dados periodicamente.

3.2.2. Transformação

Por sua vez, a transformação, *transform*, se ocupa em tornar os dados aptos para a utilização. Assim, são utilizadas técnicas de limpeza, classificação – categóricos e contínuos – e normalização sobre os dados. Assim sendo, tudo o que não condiz com o que é esperado passam por um procedimento de exclusão ou tratamento levando em conta as regras de negócio definidas.

Tais regras estão em conformidade com o destino, garantindo a confiabilidade do processo. De maneira geral, os dados podem ser convertidos, padronizados e tipificados para se enquadrarem em seu uso. Assim, para análise de dados um formato pode ser especificado enquanto para um projeto que envolva aprendizado de máquina, outra abordagem pode ganhar destaque.

3.2.3. Carga

Por fim, a última fase do ETL, é a carga, *load*, dos dados. Nela os dados são gravados em um repositório de destino que permite o acesso ao consumidor. No caso do *data warehouse*, os dados podem ser carregados em tabelas dimensão e tabelas fato, que fazem parte do cubo de dados OLAP, *Online Analytical Processing*.

O cubo OLAP é um tipo de estrutura que expande os limites dos bancos de dados relacionais amplamente utilizados, melhorando a velocidade para a análise dos dados (MICROSOFT, 2021). Eles têm a capacidade de mostrar e executar cálculos de grandes quantidades de dados ao mesmo tempo que fornecem ao usuário meios de trabalhar tais dados.

Assim sendo, as possibilidades de utilização analítica aumentam consideravelmente através da concentração, divisão e inclusão de dados de acordo com a demanda e os questionamentos que o uso dos dados pode vir a dar suporte para responder.

3.3. Extract, Load, Transform (ELT)

Com relação ao ETL há uma variação, o ELT, onde inverte-se a carga dos dados com a transformação, Figura 3. Nesse modelo, os dados são extraídos da fonte primária e carregados em uma base que aglutinará os diversos tipos de dados.

Com os dados já armazenados e acessíveis o sistema poderá então realizar as transformações e normalizações necessárias de modo mais otimizado, pois os dados estarão no mesmo ambiente.

Para o trabalho com Big Data convencionou-se a utilização do modelo ELT, pois os dados podem ser reestruturados para gerar novas tabelas que o processo no formato ETL não permite. Tendo em vista que gravaremos em um sistema de armazenamento voltado para Big Data, nossa escolha, então, recaiu sobre o ELT.

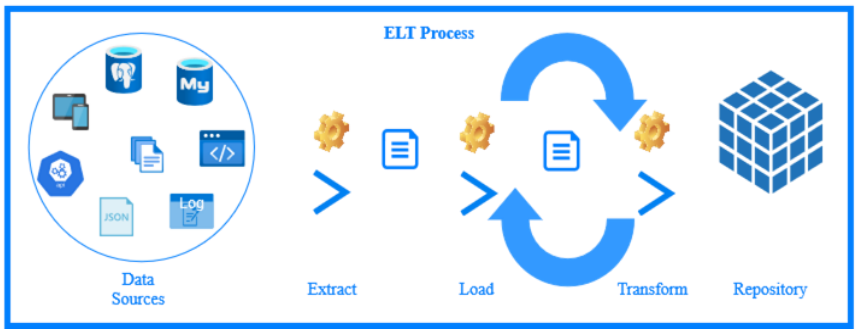

Figura 3 - Processo de ELT

3.4. Data Warehouse

Surgindo de um conceito acadêmico da década de 1980, o *Data Warehouse* (DW) vem suprir a necessidade de organizar e obter dados de bases transacionais, que realizam apenas a ação de inserir, atualizar e excluir nos bancos de dados relacionais, como registros de vendas de mercadorias, e favorecer a análise de dados de negócio ou criação de relatórios de gestão por parte das equipes de *Business Intelligence*, BI, ou *Data Analytics* (Gouveia e Freitas, 2018).

O DW é compreendido como parte de um planejamento para a organização eficiente dos dados e direcionado para dar suporte às estratégias empresariais. Os dados armazenados

são mantidos por um longo período, permitindo que as análises desses dados, previamente consolidados, agrupados e indexados, possam ser estendidas na dimensão tempo. Proporciona o gerenciamento de um grande volume de dados que estariam, a princípio, dispersos em múltiplas fontes, sendo então concentrados e possibilitando a análise de dados coletados através de sistemas transacionais.

Por sua vez, complementando no que se relaciona a análise de dados, o OLAP, *Online Analytical Processing*, favorece a visualização e exploração dos dados. Tais tecnologias são voltadas para sustentar consultas sofisticadas e incomuns, assim como combinar informações por meio de análises comparativas e visualizações customizadas através do tempo. Tais tecnologias possibilitam observações claras de dados multidimensionais, através de compreensões múltiplas e variadas.

Os DW são modelados pensando que o sistema está em constante crescimento e que os dados devem estar integrados, são orientados por assunto e posições históricas. Os dados não são atualizados ou excluídos, mas apenas inseridos iterativamente ou substituídos por uma nova inserção. Os DW podem ser ainda divididos em data mart, que são subconjuntos voltados para um tema ou assunto específico.

As abordagens normalizada e dimensional são as mais usadas para o armazenamento. Na perspectiva normalizada as tabelas são agrupadas por zonas temáticas que refletem as categorias dos dados em geral. Já na perspectiva dimensional os dados são divididos em fatos, os dados numéricos da transação, e em dimensões, informações de referência que contextualizam os fatos.

Vale pontuar rapidamente outro conceito de armazenamento de dados utilizado: o Data Lake. O data lake é uma forma de armazenamento de dados que se propõe a guardar as informações em formato original (RAU, 2021). Assim teremos no mesmo repositório:

- Dados estruturados – facilmente reconhecidos por sua estrutura bem definida, aqui se enquadram tabelas de banco de dados e arquivos tabulares como o csv;
- Dados não estruturados – representados por vídeos, arquivos de áudio, fotos e demais tipos relacionados;
- Semiestruturados – dados que apresentam algum tipo de estrutura são aqui enquadrados. Tais como arquivos de web, json e assim por diante.

Como método para trabalhar nesse modelo de armazenamento é implementado o tratamento por camadas. O delta lake, camada de armazenamento de software livre, é uma excelente proposta para esse objetivo, contendo uma camada para o dado bruto, outra para a

preparação de dados mais detalhados e prontos e uma terceira camada para dados comerciais, enriquecidos ou para consumo final (MICROSOFT, 2021).

Por fim, ao se falar em Big Data, o DW convencional, não se mostra capaz de armazenar ou processar essa massa de dados, somos então levados a utilizar sistemas distribuídos para melhorar os resultados nessa abordagem. Nesse sentido, o Apache Hadoop é o *framework*, conjunto de ferramentas, bibliotecas e funcionalidades inseridas em uma base estrutural única, mais utilizado no momento com essa finalidade e onde se desenvolve o data lake. Passamos assim a apresentar um pouco do Hadoop.

3.4. Hadoop

O Apache Hadoop é um *framework* que surge da necessidade de utilização da computação distribuída para a análise de dados de grande volume e diversificado. Inicialmente criado dentro do Google para melhorar a performance das pesquisas, através da tecnologia MapReduce, foi concretizado e nomeado para Hadoop dentro do Yahoo, alguns anos depois, sendo também disponibilizado como software *open source* pela Apache Software Foundation desde então.

Entre as suas principais características, e que o tornaram amplamente utilizado, Goldman et all (2012) destacam:

- **Código Aberto:** como software livre, ele tem uma grande comunidade por trás, dando suporte ao desenvolvimento de suas ferramentas, encontrando problemas, melhorando a estabilidade e mantendo em desenvolvimento constante.
- **Economia:** Por ser um software de código aberto ele possibilita a análise e adaptação de suas funcionalidades a qualquer interessado. A análise e o processamento de dados podem ser feitos por máquinas e rede convencionais, diminuindo o custo com poderosos sistemas computacionais. Empresas – como Amazon, Google e Azure que são especializadas em computação em nuvem, implementaram versões próprias do Hadoop – podem “alugar” seu parque de computadores, diminuindo os custos com manutenção e compra por parte do cliente.
- **Robustez:** Possui mecanismos que garantem a integridade dos dados: replicando, armazenando metadados e informações de processamento.
- **Escalabilidade:** O Hadoop é configurável, permitindo que, independentemente da quantidade de máquinas que serão utilizadas, todas trabalhem em conjunto em um esforço menor de trabalho técnico e com efetividade.

- **Simplicidade:** Questões relativas à computação paralela, tais como tolerância a falhas, escalonamento e balanceamento de carga não ficam na responsabilidade do desenvolvedor, permitindo que ele direcione sua atenção ao problema que será processado no modelo *Map* e *Reduce*.

O Hadoop é um framework bem extenso com relação aos seus componentes, porém nos ateremos nessa apresentação básica à dois componentes principais YARN e HDFS.

O YARN é o componente responsável por gerenciar e alocar recursos que permitam a ação do cluster para processamento de grande volume de dados (Shenoy, 2014).

Por sua vez, o sistema de armazenamento de arquivos *Hadoop Distributed File System*, HDFS, é distribuído, altamente tolerante a falhas e aplicado sobre máquinas e rede de baixo custo geral (Goldman et all, 2012). Um arquivo é distribuído em blocos de 64 MB, nos sistemas convencionais o bloco é de 512 bytes. Os blocos são replicados em diferentes *clusters*, assim, se um deles apresentar falhas, outro é utilizado, garantindo a integridade. Possui o mesmo tipo de transparência que um sistema convencional, o usuário não identifica diferenças na utilização. Possui algumas estruturas de controle exclusivas como: segurança no tráfego de informações, tolerância a falhas de nós, integridade no arquivo, consistência para os usuários acessarem o mesmo arquivo e o desempenho geral é similar aos convencionais.

O sistema HDFS em si não é relacional e, portanto, pode receber arquivos com qualquer tipo de extensão. Porém, como trabalharemos em um modelo de armazenamento relacional, faz se necessário a utilização de uma interface que compreenda o banco de dados convencional, assim como comandos em linguagem SQL, que é amplamente utilizada nesses tipos de bancos. Para isso, empregaremos o Apache Hive, detalhado na próxima seção.

3.5. Hive

O Apache Hive é uma ferramenta open source presente dentro do ecossistema Hadoop e mantida pela Apache Foundation, que intenciona facilitar o trabalho com SQL e, portanto, com bancos de dados relacionais como o MySQL, MariaDB, Oracle entre tantos outros, mas contendo por “debaixo dos panos” o Hadoop e sua estrutura. O SQL utilizado pelo Hive, assim como em outros bancos proprietário ou open source, é uma versão adaptada do original. No caso do Hive, o SQL utilizado foi batizado como HQL, Hive Query Language.

O Hive foi desenvolvido para atuar como um DW tendo no HDFS, do Hadoop, o sistema de armazenamento em execução. Com ele é possível criar grandes estruturas de dados que tem

na clusterização, distribuição em grupos, sua base, mas que ainda assim é transparente para o desenvolvedor ou usuário.

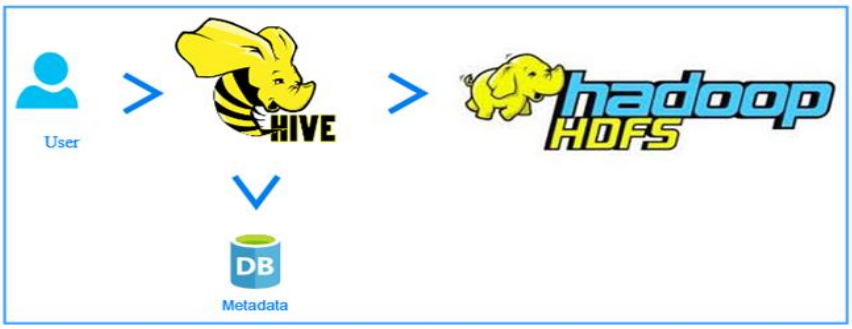

Figura 4 - Arquitetura Hive

O Hive realiza a interface entre o usuário, recebendo uma *query* em formato HQL, por exemplo, e o Hadoop, realizando a tradução da consulta para um job map e reduce, Figura 4. O Hive mantém as estruturas das tabelas e demais dependências em metadados dentro de um banco relacional, possibilitando preservar as construções como em outros DW. Por fim, ele também provê uma conexão jdbc, proporcionando que seja acessado e trabalhado como outros bancos existentes no mercado.

3.6. Pentaho Data Integration

Nascido de uma suíte para inteligência empresarial com ferramentas que propõe soluções desde a extração dos dados até a disponibilização, análise, e criação de *dashboards.*

O Pentaho Data Integration, ou Kettle, é uma das ferramentas mais conhecidas para integração de dados. Contém versões proprietária e *Open Source*, Figura 5.

	Pentaho Enterprise Edition	Pentaho Community Project
CODELESS PIPELINE DEVELOPMENT	✓	✓
ENTERPRISE-SCALE LOAD BALANCING & SCHEDULING	✓	-
STREAMING DATA SUPPORT	✓	-
EASY KETTLE TO SPARK EXECUTION	✓	-
EXPANDED LIBRARY OF CONNECTORS	✓	-
DATA SCIENCE TOOLKIT	✓	-
PROFESSIONAL SERVICES PACKAGES	✓	-
IP PROTECTION	✓	-
TECHNICAL SUPPORT OPTIONS	24/7 availability; Assigned Architect; Mentoring	Forums, Community, How-to Videos
RELEASE SCHEDULE	Major releases, and monthly patches	Only major releases

Figura 5 - Versões do Pentaho. Fonte: Hitachi Vantara

A suíte vem sendo desenvolvida desde 2004 pela própria Pentaho Corporation, sendo que em 2015 ela foi adquirida pela Hitachi Data Systems e em 2017 se tornou a Hitachi Vantara, enquanto marca, mas ainda é reconhecida pelo seu nome inicial.

O Pentaho Data Integration é desenvolvido em linguagem Java e apresenta ao menos três componentes principais:

- Spoon: ferramenta de modelagem gráfica que permite ao usuário usar o *drag* e *drop*, arrastar e soltar, e é onde se define entradas, transformações e saída de dados. A Figura 6 mostra a tela inicial do Spoon
- Pan: aplicação por linha de comando que executa as transformações criadas na interface Spoon.
- Kitchen: executa os jobs partindo de um sistema de arquivos ou banco de dados.

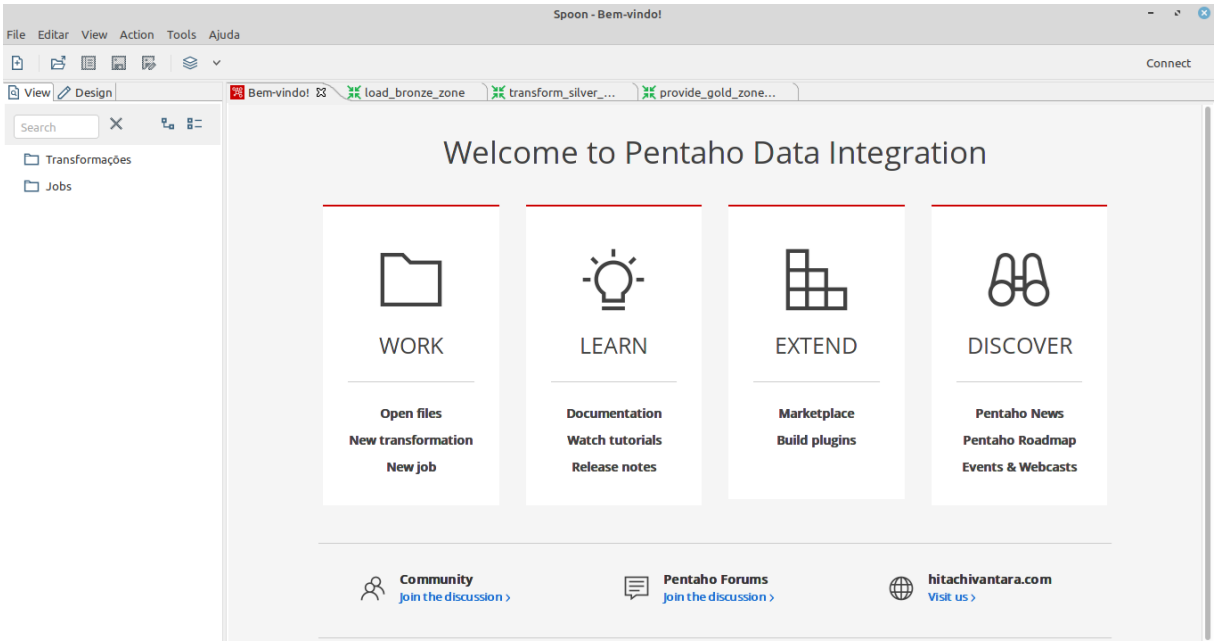

Figura 6 - Tela inicial do Pentaho

Há de se pontuar que ao falarmos em jobs e transformações, no Pentaho *Data Integration*, estamos nos referindo a uma sequência de operações que são realizadas em sequência e completas e assim a automatização de uma dada tarefa, enquanto a transformação é uma operação sobre os dados em si: leitura, seleção de campos, criação de tabela, agregação e assim por diante.

A integração de dados do Pentaho permite que o usuário realize diversas atividades rotineiras de engenharia de dados. A ferramenta possui conexão com inúmeros bancos de dados. Embora ela se destaque com bancos relacionais, não relacionais também podem ser trabalhados com a ferramenta. Há integração com serviços distribuídos e o Hadoop é suportado diretamente.

Em caso de falha em algum ponto do pipeline uma mensagem de e-mail pode ser configurada para gerar um alerta. Assim uma rápida reação ao problema ocorrido pode ser posta em prática. Por outro lado, possibilita também que informes de sucesso sejam enviados a um destinatário.

Outro ponto que vale pontuar, é a possibilidade de agendar uma atividade para ser realizada. Deixando a automação da execução aplicável para quando se desejar. Um arquivo que é disponibilizado mensalmente poderá então ser recuperado sem chamadas desnecessárias.

Pessotti da Rocha Magdaleno (2015) faz uma análise das características da ferramenta completa de BI em sua versão *Community*, *open source*, concluindo que ela obteve resultados

positivos, o que nos traz tranquilidade na sua utilização em nosso experimento de desenvolvimento de pipeline de dados.

Por sua vez, Lira Filho (2013), realiza uma comparação com outra ferramenta que também possui versão *open source*, o Talend Open Studio. Ele conclui que o Pentaho possui maior flexibilidade e velocidade em determinados cenários, mesmo sendo insuficiente no quesito hardware.

3.7. Python

Criada por Guido van Rossum em idos de 1991, segundo informações da página oficial, a linguagem Python hoje ocupa o primeiro lugar no ranking do Instituto de Engenheiros Eletricistas e Eletrônicos ou Instituto de Engenheiros Eletrotécnicos e Eletrônicos dos Estados Unidos, ou IEEE (Figura 7), organização voltada para o fomento da tecnologia.

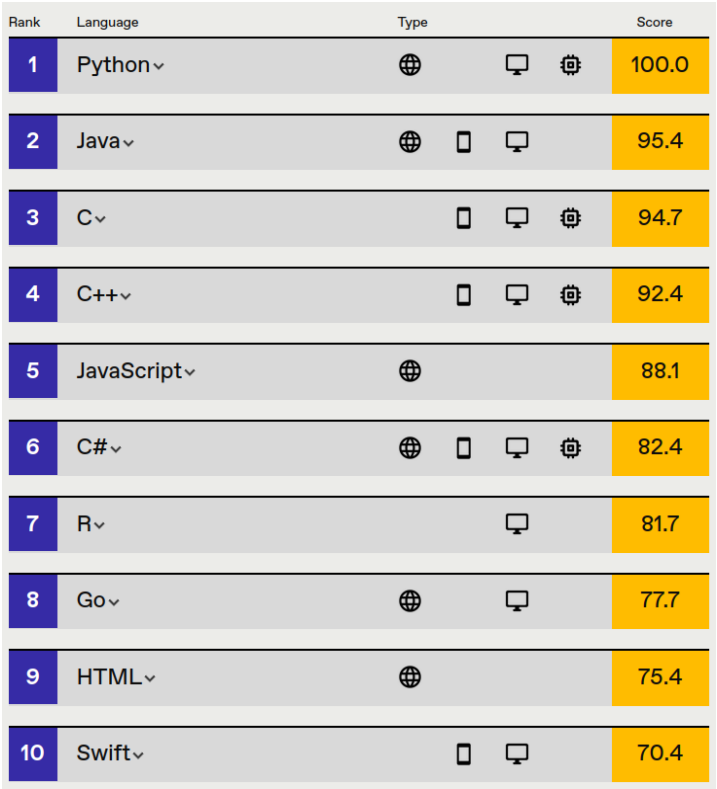

Figura 7 - 10 linguagens de programação mais utilizadas 2021. Fonte: IEEE

Apoiada sobre a premissa de ter uma curva de aprendizado rápida e uma sintaxe mais enxuta e simples, avançou em passos largos dentro do campo dos dados, sendo adotada por cientistas de dados, engenheiros e analistas.

Python é uma linguagem *open source* de propósito geral, ou seja, com ela é possível criar aplicações web, programas de computador e sistemas dos mais variados. Seus módulos, bibliotecas, estendem ainda mais seu poder de ação.

A comunidade é um ponto forte na sua adoção. Ela é responsável, não só pela divulgação, mas implementam e desenvolvem não só os módulos que a compõem, mas a própria linguagem em si.

Atualmente é facilmente observável que usuários não convencionais de linguagem de programação estão se dedicando a aprender e aplicá-la em suas atividades cotidianas, tais como: cientistas sociais, psicólogos, biólogos, geógrafos, jornalistas e muitos outros.

3.8. Airflow

Segundo informações do site do projeto, o Apache Airflow foi desenvolvido inicialmente dentro do Airbnb, em idos de 2014, para gerenciar o complexo fluxo de dados da empresa e em 2016 foi incorporada à Apache Foundation. Como software livre, se tornou um dos projetos mais utilizados da organização, se convertendo em 2019 em um *Top-Level Project*.

Foi desenvolvido em Python e utiliza scripts da linguagem na criação dos *workflows*, fluxo de trabalho. Tem uma interface web por onde o usuário pode gerenciar as atividades que estão sendo realizadas.

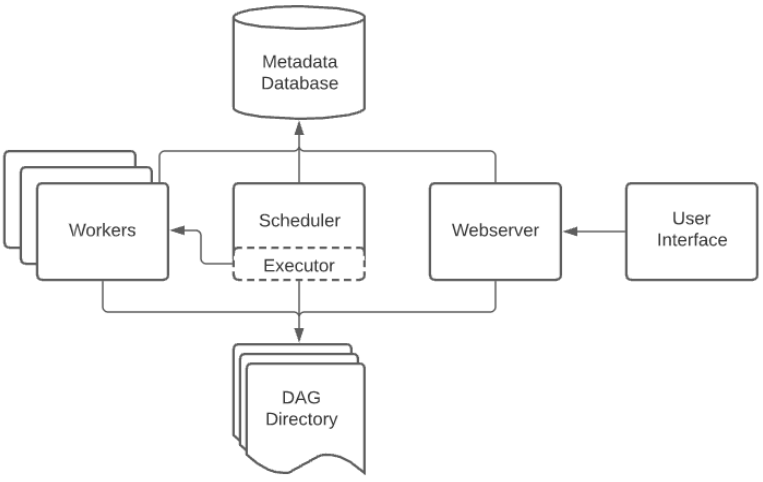

Figura 8 - Arquitetura do Airflow. Fonte: Apache Airflow

Seguindo sua arquitetura, Figura 8, apontamos os seguintes componentes essenciais:

- **DAG, *Directed Acyclic Graph*:** coleta e organiza a série de tarefas. Suas dependências e relacionamentos são observados, gerenciando qual deverá executar;

- ***Webserver*:** propicia uma interface visual ao usuário. Através dela, além de acompanhar o andamento das tarefas, entre outras coisas, podemos observar os logs com a execução do processo, facilitando a resolução de problemas, quando eles ocorrem. A Figura 9 mostra a tela inicial do Airflow com sua interface web focada nas DAGs. Nesta tela, entre outras informações, já é possível verificar se uma DAG está ativa, executando, o intervalo de agendamento e a próxima execução.
- ***Scheduler*:** monitora a ação das tarefas, DAGs e realiza a sincronização delas. A cada minuto verifica no diretório de DAGs se há alteração no código de alguma e se precisa preparar para execução de alguma que esteja ativa.
- ***Metadata Database*:** é onde ficam armazenados os dados relacionados a cada instância de tarefa de uma DAG. Hora da execução, situação de cada tarefa, tempo e agendamento são armazenados aqui.
- ***Executors*:** são a forma pelo qual as instâncias das tarefas são executadas.
- ***Workers*:** são responsáveis pela execução em si de uma tarefa.

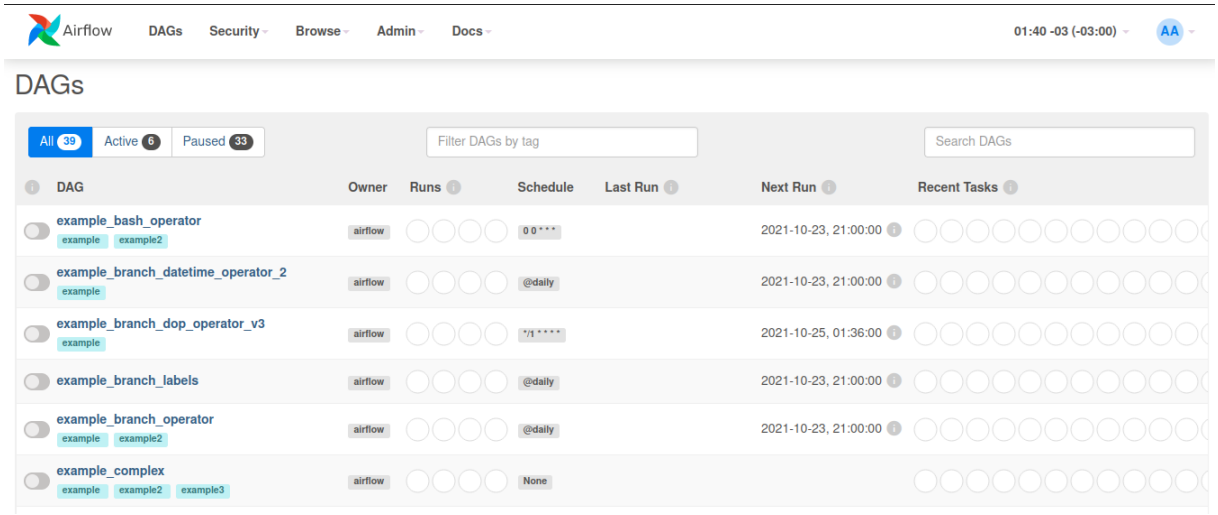

Figura 9 - Tela inicial do Airflow

A integração de dados no Airflow segue a lógica e é desenvolvida em código pelo engenheiro de dados. Todas as etapas são pensadas e modeladas segundo as regras que chegam até o desenvolvedor.

A relação entre as tarefas é dada pelo uso do sinal duplo de menor ou maior. Assim, a etapa 1 seguida pela etapa 2 é apontada da seguinte forma: etapa 1 >> etapa 2.

4 METODOLOGIA

Aqui apresentamos a metodologia proposta para o desenvolvimento dos pipelines de dados de ELT e os três cenários de extração: um arquivo em formato csv, uma API com autenticação e uma tabela armazenada em banco de dados relacional. A Figura 10 ilustra a arquitetura geral do pipeline a ser empregada.

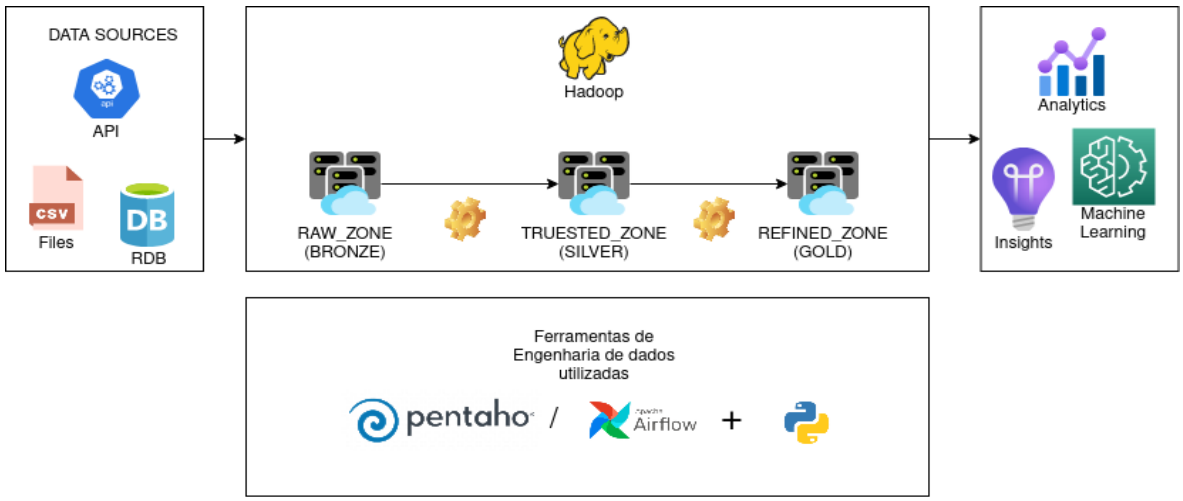

Figura 10 - Arquitetura do pipeline

O desenvolvimento do pipeline evidenciado pela arquitetura acima e baseado no modelo de tratamento do delta lake abarcará as seguintes etapas:

1. Extração e carga na camada bronze dos dados sem tratamento, ou com o mínimo possível, dentro do Hadoop através de conexão com o Hive, no formato de arquivo de texto.
2. Tratamento de dados e arquivamento das tabelas prontas com particionamento e com os elementos no formato parquet será feito na camada silver.
3. Agrupamento de dados e armazenamento de forma que estejam disponibilizados para usuários será na camada gold.

Para fins de comparação entre as diferentes soluções adotadas, consideraremos os seguintes pontos relacionados às ferramentas para a criação do pipeline: instalação, documentação oficial, aprendizagem e execução do desenvolvimento. Destaca-se que a arquitetura de armazenamento empregada se baseia no formato de camadas utilizado em *data lake*, onde os dados, independente do formato, são primeiro armazenados e só então se inicia

as análises, porém a implementação será em ambiente de data warehouse com o Apache Hive, já que são dados tabulares ou mantidos em tabelas do tipo relacionais.

As bases de dados usadas foram retiradas de portais abertos. Para o nosso experimento foram usados dados da campanha de vacinação contra a COVID 19, importantes para a tomada de decisão pelo poder público e em pesquisas econômicas e sociológicas que podem versar sobre os impactos da vacinação nas diferentes regiões com relação à produção e ao trabalho, tais informações foram acessados por API.

O arquivo em formato csv contêm informações de escolas brasileiras e conta com dados de estudantes, professores, tipo de disciplina, se a escola é pública ou privada e assim por diante. Assim, entendo ser útil para análises pedagógicas.

Por fim, foi utilizado o MariaDB como fonte de dados relacional. O MariaDB é um banco de dados *open source* e surgiu como uma segmentação do MySQL. Para o nosso experimento foi previamente populado com os dados de latitude e longitude de todas as cidades brasileiras e podem servir de suporte em análises geoespaciais quando em conjunto com outras fontes para melhor entendimento de efeitos regionais de ações do poder público.

Os softwares utilizados são listados na Tabela 1 enquanto um resumo do hardware pode ser observado na Tabela 2 logo abaixo.

Softwares utilizados	
Nome	Versão
Airflow	2.0.2
Docker	20.10.9
Hadoop	3.3.1
Hive	3.1.2
MariaDB	10.6.4
MySQL	8.0.26
Pentaho Data Integration	9.2.0
Python	3.8.10
Visual Studio Code	1.61.2

Tabela 1 - Softwares utilizados

Configurações do dispositivo utilizado para laboratório	
Sistema Operacional:	Linux Mint 20.2 - “Uma”, Kernel 5.4.0
Processador:	Intel i3-3110M, 64 bits, Dual Core, 4 Threads, 2400 MHz
Memória:	8GB
SSD:	240GB
HDD:	500GB

Tabela 2 - Hardware utilizado

5. APLICAÇÃO

5.1 Pentaho

5.1.1 Instalação

A instalação o Pentaho se mostra de fácil execução. Não contém um instalador específico para cada sistema operacional, mas executáveis em geral, e têm como dependência o Java sdk 8. O arquivo de download é grande, na versão 9.2.0.0 que foi utilizada no desenvolvimento dos pipelines e lançada em 02 de agosto de 2021, contém 1,9 GB compactado, Figura 11.

pdi-ce-9.2.0.0-290.zip	2021-06-02	1.9 GB
pme-ce-9.2.0.0-290.zip	2021-06-02	1.7 GB
pad-ce-9.2.0.0-290.zip	2021-06-02	28.4 MB
prd-ce-9.2.0.0-290.zip	2021-06-02	1.7 GB

Figura 11 - Arquivos para download compactados do Pentaho

Incorporado à pasta compactada, há todos os drivers, executáveis e arquivos necessários para execução do programa em ambiente Linux, Windows e Mac. Para a utilização do programa é necessário descompactar os arquivos, garantir que o Java está configurado nas variáveis de ambiente e executar o arquivo relacionado a cada sistema operacional.

5.1.2 Documentação Oficial

Em questão de documentação, diretamente no site do desenvolvedor do Pentaho encontramos os principais pontos necessários para início de aprendizagem: instalação, primeiros passos e de consulta. Porém, o que ganha ênfase, foi o apoio por outros grupos em diferentes mídias. No site do projeto há apontamento para fontes externas como: blogs, vídeos, fóruns entre outros.

Há de se destacar os vídeos de usuários do software e da comunidade que são hospedados no YouTube. Enquanto on-line, no sítio do projeto, encontramos a documentação apenas em inglês, há uma farta gama de facilitadores que disponibilizam seus vídeos em português na plataforma.

5.1.3 Aprendizagem

No quesito aprendizagem, o Pentaho se mostra bastante prático. Utilizando a ideia de arrastar, soltar e ligar, possibilita o desenvolvimento de um fluxo de dados diretamente em interface interativa, diminuindo o quanto de programação é necessário como conhecimento prévio. O usuário, para um melhor aproveitamento da ferramenta deve conhecer os tipos de dados, inteiro, decimal, string e sua estrutura, mas para a utilização dos componentes não é necessária uma grande bagagem técnica. Em geral SQL pode ser bastante útil para algumas atividades com dados. Se houver necessidade de utilizar outras linguagens, é possível a realização da atividade através da instalação de plugins. Desta maneira, é viabilizado o trabalho com Python, PHP e outras linguagens disponíveis com a implementação desses extensores.

A comunidade é o grande diferencial. Contribuindo com inúmeras dicas, tutoriais e artigos que ajudam o usuário iniciante a conseguir montar seus primeiros jobs e cadeias de transformação.

5.1.4 Desenvolvimento

5.1.4.1 Camada bronze

As três fontes de dados adotadas neste trabalho – tabela de banco de dados com informações de geolocalização, API com dados de vacinação contra COVID19 e csv com referência à educação – foram bem atendidas na primeira etapa de importação. Na Figura 12 temos na primeira linha a migração de dados de origem de banco de dados relacional, na segunda com dados de API com conexão e leitura do arquivo json que é baixado e na terceira linha a origem dos dados é do arquivo em csv.

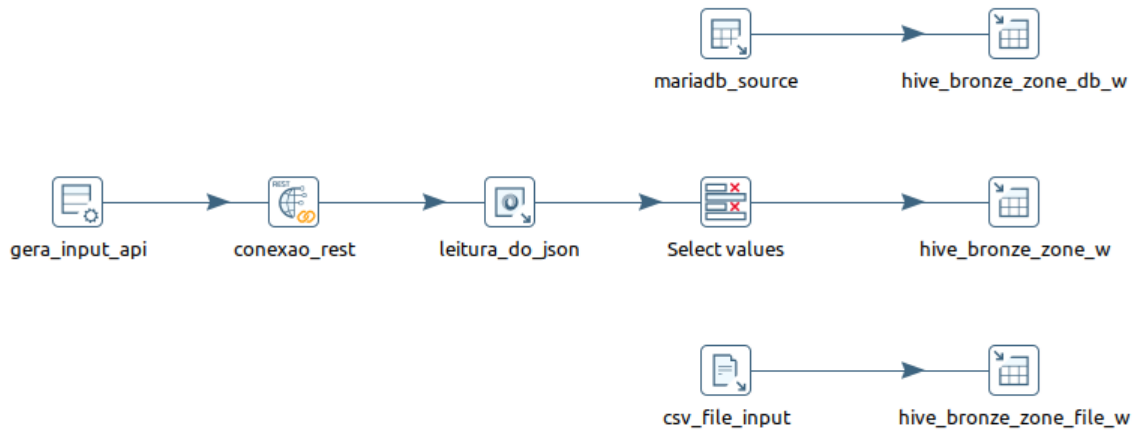

Figura 12 - Extração de dados para a camada bronze

No cenário de importação com dados originados de banco de dados com a base de geolocalização, a migração foi um pouco lenta, porém aceitável dentro da proposta. Há diversos conectores disponíveis, assim, não foi um problema realizar a conexão com o banco de dados utilizado para teste, MariaDB. E da mesma forma que em outros cenários, criar a tabela era facilitado com sugestão em SQL (Figura 13).

Com relação ao cenário com dados oriundos de acesso via API acessando informações da campanha de vacinação contra COVID19, ele tem em seu arcabouço as ferramentas necessárias para acessar a endereço de acesso para conexão, realizar o login e acessar os dados em formato json. Porém, cabe observar que a interpretação dos dados não era feita instantaneamente, mas foi necessário a utilização de um amostral do arquivo de dados fonte para que ele pudesse verificar a estrutura e assim realizar a interpretação da base acessada através da API.

No cenário de extração que conta com fonte advinda de arquivo de dados educacionais, o Pentaho entendeu rapidamente o formato dos dados do arquivo csv, a edição dos formatos também foi facilitada, por fim disponibiliza uma sugestão em SQL para a criação da tabela. Assim, a população da tabela foi rápida e sem impedimentos técnicos.

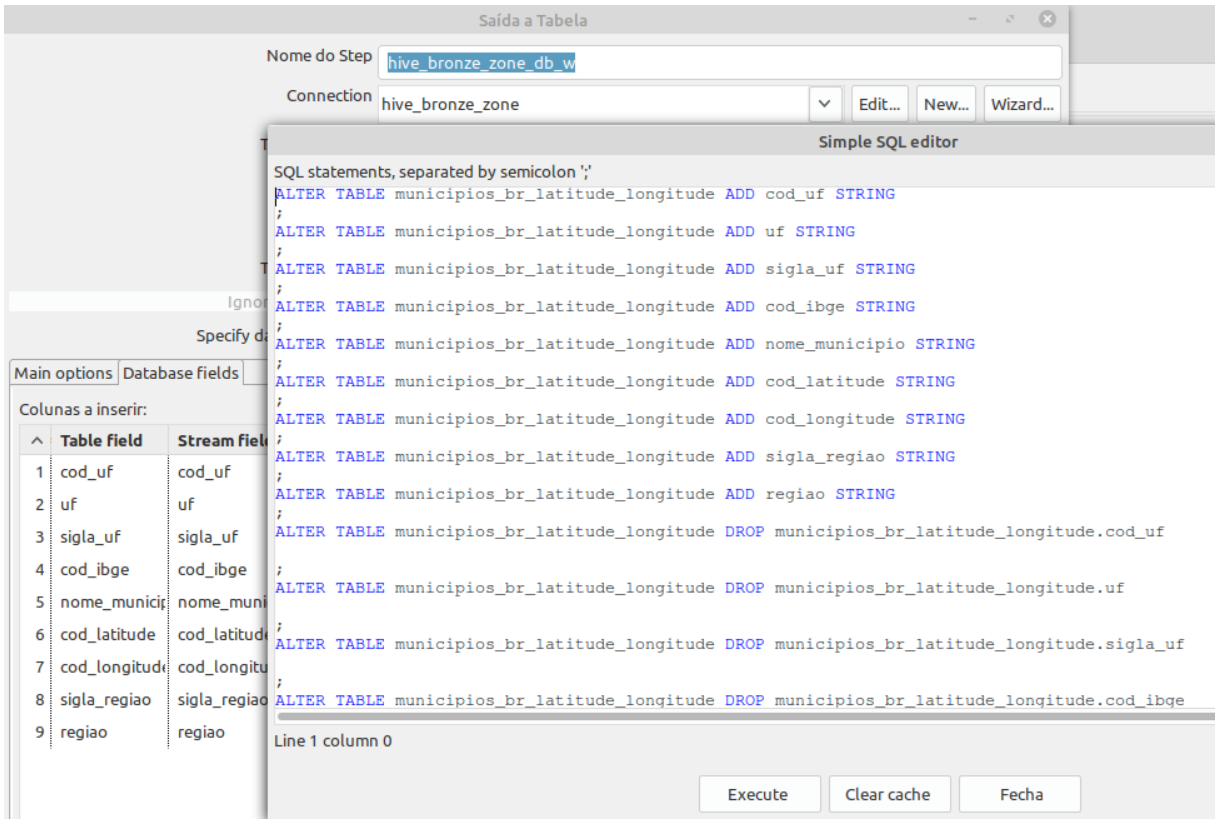

Figura 13 - Sugestão do Pentaho para criação de tabela

5.1.4.2. Camada silver

Adentrando a camada silver, o experimento passou a realizar conexão apenas com o Hadoop em seu modo pseudo distribuído, assim não houve um acréscimo no quesito tempo entre as diferentes tarefas, mas todas ocorriam em tempo de execução, Figura 14.

Enquanto tratamento, os dados de vacinação da COVID19, linha um da Figura 14, foram primeiro ajustados enquanto nomenclatura, depois foram selecionados porque nem todos os campos iriam ser mantidos para a inserção na camada silver. Todos os três cenários passaram pela etapa de seleção, seja para sua realização ou para homologação de que os dados que seriam inseridos condiziam com o que era esperado.

O Pentaho Data Integration permite que os arquivos que forem preencher as tabelas em formato parquet sejam mantidas, ainda viabilizam a inserção sobre escrita de modo direto, *Insert Overwrite*. Com os dados na camada silver, passemos então à camada gold da arquitetura.

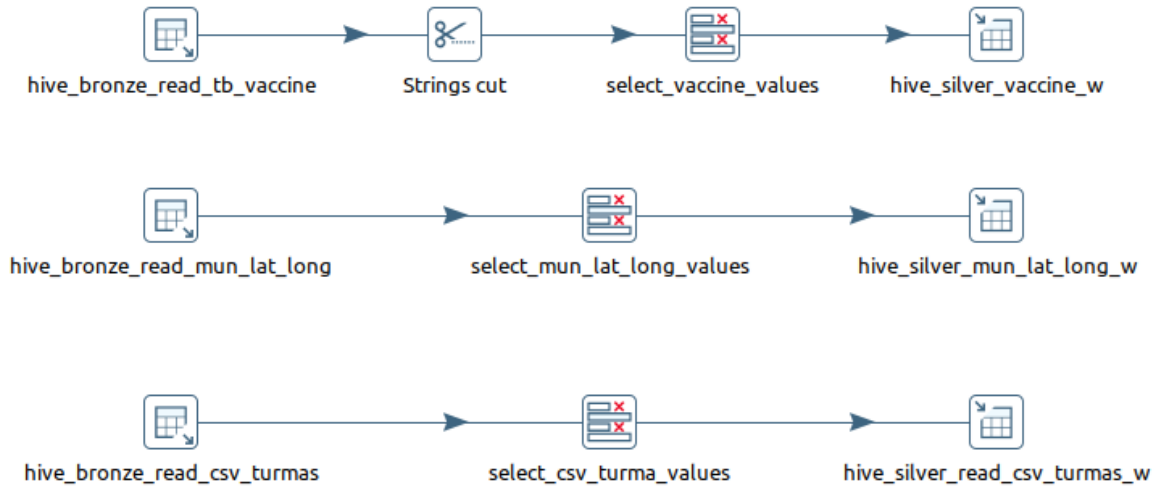

Figura 14 - Execução da camada silver

5.1.4.3. Camada gold

Nessa etapa, resolvemos utilizar de exemplo duas tabelas para demonstrar uma ação na camada gold, assim foi realizada a ação de juntar, *join*, tabelas diferentes – a base educacional que teve campos tratados e selecionados e os dados de geolocalização – através de um identificador comum e ordenado, no caso exposto foi utilizado o código de cidade com o código de identificação do IBGE. Fazendo a junção de dados escolares com as localizações geográficas, Figura 15, é viabilizado ao analista a criação de visualizações geográficas com granularidade de diferentes níveis. No Pentaho, após a junção dos dados, eles ficam disponíveis e novamente podemos escolher quais serão armazenados na tabela final. Assim como, com base neles, podemos criar a tabela que será utilizada para a população dos dados e consequente disponibilização. Após finalização do pipeline com Pentaho, passamos a seguir a mostrar o mesmo processo com Airflow e Python.

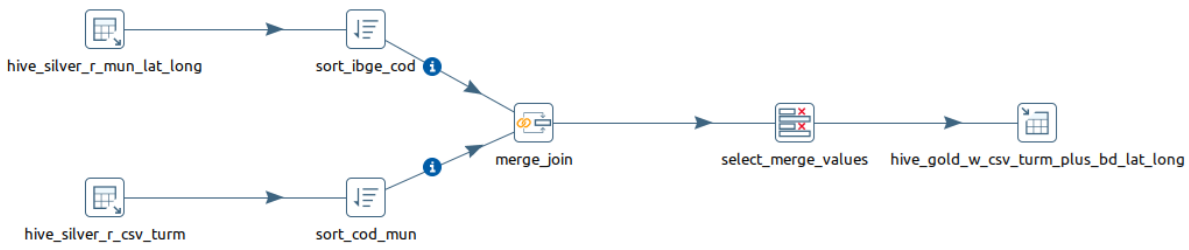

Figura 15 - Execução na camada gold

5.2 Airflow e Python

5.2.1 Instalação

Em questão de instalação, o Airflow oferece duas formas principais, Figura 16: uma delas através de Docker, criando uma virtualização com a imagem, e outra localmente com o Python, pela instalação do módulo através do repositório oficial para desenvolvedores terceiros, o Pypi.

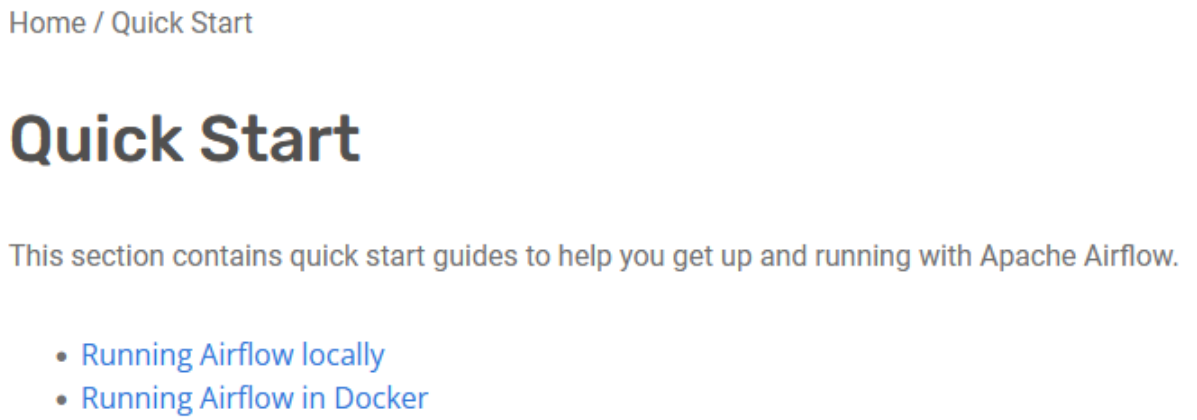

Figura 16 - Tipos de instalação do Airflow

Ambas as opções são muito bem documentadas. Como verificação, realizamos a instalação das duas formas disponíveis e, seguindo o passo a passo disponibilizado no site, o resultado foi um processo tranquilo.

Com relação ao Python não tivemos que realizar a instalação pois ele já vem integrado em várias distribuições linux e na que usamos, Linux Mint, ele veio em sua versão 3.8. Da mesma maneira, usuários de MAC OS não precisarão realizar a instalação pois também vem com uma instalação padrão já realizada. Para sistemas operacionais Windows, ele conta com executável que pode ser baixado no site oficial e que segue os padrões costumeiros de instalação com botões de confirmação.

Como salientado, a documentação é bastante completa, assim, passemos a tratar dela no próximo tópico.

5.2.2 Documentação oficial

Nesse quesito o Airflow é muito bem documentado, Figura 17, contendo respostas ou apontando soluções para a maioria das questões que poderia ter o usuário iniciante, seja relacionado à instalação, conceitos para o entendimento do funcionamento do programa, utilização dos componentes ou exemplos de como fazer ou empregar as opções disponíveis.

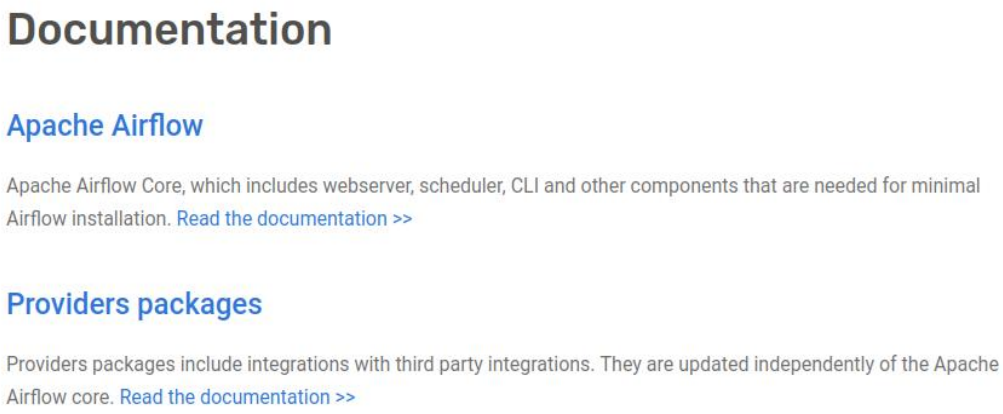

Figura 17 - Detalhe da página de documentação

Para usuários mais experientes de Python, podem ser visualizados os códigos fonte das implementações. A documentação indica os passos para que o desenvolvedor também possa criar seus próprios pacotes caso não encontre o que precisa ou possa melhorar algum existente. É possível iniciar a aprendizagem apenas tendo como base a documentação oficial, sem muita procura em outras mídias. Aproveitando o ensejo dado, passemos a falar um pouco sobre o observado sobre possibilidades de aprendizagem.

5.2.3 Aprendizagem

Como uma ferramenta voltada para gerenciamento de fluxo de dados, a aprendizagem também é direcionada para esse público de pessoas que já trabalham com desenvolvimento. Embora seja bem fácil encontrar tutoriais para as mais diferentes categorias de usuários, o entendimento é que o tipo principal de utilizador para o qual são direcionados os tutoriais é para o de desenvolvedores de preferência com conhecimento em Python.

Observa-se que as DAGs, parte do Airflow responsável pela coleta e organização da série de tarefas com observância das dependências e relacionamentos, são criadas por script Python, ou seja, aqui se entendo o porquê que o conhecimento prévio da linguagem é necessário

para a execução apropriada das etapas orquestradas pelo software e, portanto, para a sua aprendizagem.

A interface web disponibilizada é para acompanhamento, gerenciamento e atividades relacionadas, não sendo para o desenvolvimento. Nesse caso, utilizamos outros softwares, como será apontado na próxima seção onde falaremos sobre os cenários de extração e carga de dados para a camada bronze com o Airflow. Sigamos com o desenvolvimento do pipeline.

5.2.4 Desenvolvimento

5.2.4.1 Camada bronze

Na realização das tarefas de desenvolvimento foram utilizados o Python puro e módulos como o *request*, para chamadas API, o pandas, para manipulação e criação de dados tabulares. Os códigos foram estruturados em funções que executam apenas uma parte específica de cada etapa. Dessa maneira, há funções direcionadas ao download de dados, outras que tratam, selecionam e assim por diante. Por seu turno, as tabelas foram desenvolvidas em scripts SQL e HQL. O editor de código escolhido foi o Visual Studio Code da Microsoft, Figura 18, que permite tanto o desenvolvimento em Python, em SQL e HQL quanto testes em *Jupyter Notebook*.

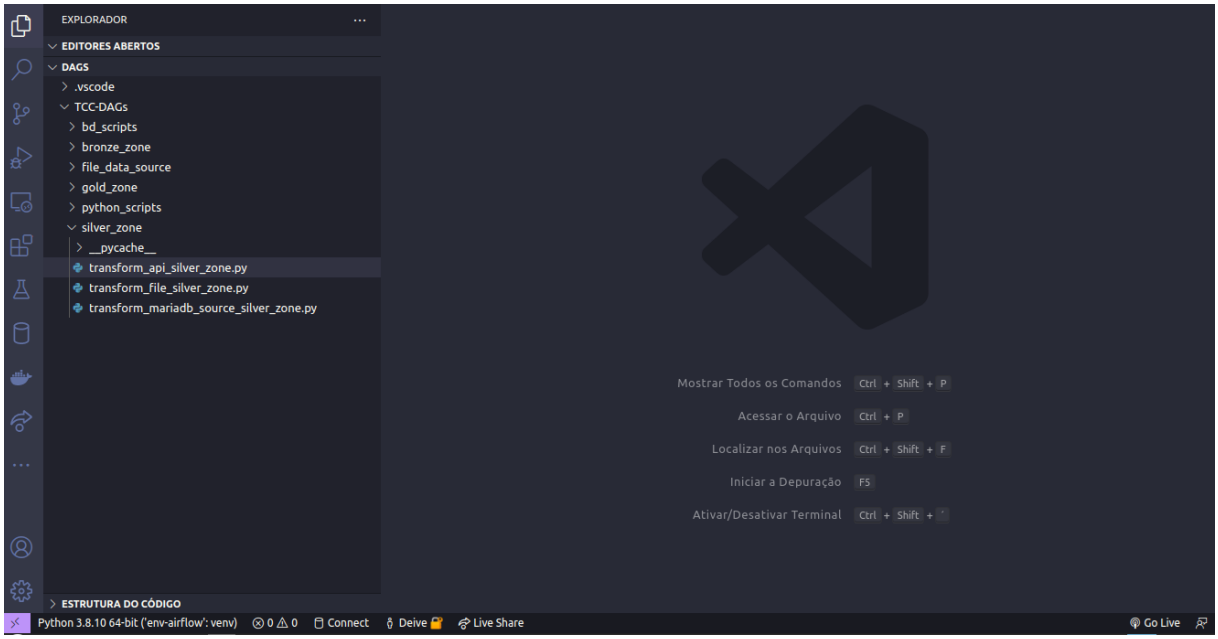

Figura 18 - Tela de desenvolvimento do Visual Studio Code

O arquivo csv, de dados educacionais, foi previamente carregado no Jupyter para verificação de integridade e desenvolvimento das etapas pelo qual deveria passar. O Hive permite que os dados de um arquivo sejam carregados diretamente em uma tabela e essa foi a abordagem seguida. Primeiro criamos a tabela, depois realizamos inserção dos dados diretamente do arquivo.

Sendo uma etapa temporária, já vinculamos a exclusão da tabela na DAG da camada silver, e sua contínua criação sendo realizada na bronze, garantindo que os dados não serão duplicados e o sistema sobrecarregado com dados desnecessários. Na Figura 19 mostramos uma DAG com a visualização das tarefas na interface disponível no webserver. Nela observamos a criação da tabela e a carga dos dados na camada bronze.

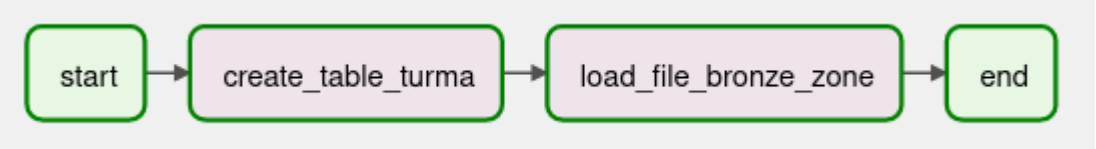

Figura 19 - DAG de extração de dados de arquivo csv na camada bronze

No caso da migração da base de dados armazenado no MariaDB, dados de geolocalização, foi uma etapa facilitada pelo Airflow. Após a instalação do pacote para trabalhar com bancos MySQL, e consequentemente MariaDB, entre os componentes disponíveis há um especializado para esse tipo de processo. Bastando realizar o preenchimento com as informações de origem e destino, Figura 20. Vale ressaltar que também já havia sido modelado a tabela que receberia esses dados.

```
mariadb_to_hive = MySqlToHiveOperator(
    task_id='mariadb_to_hive',
    sql="select * from db_source.municipios_br_latitude_longitude",
    hive_table="bronze_zone.tb_bronze_airflow_mun_br_lat_long",
    mysql_conn_id='mariadb_lab',
    hive_cli_conn_id='hive_connection',
    dag=dag,
)
```

Figura 20 - Migração de dados de base relacional para camada bronze com Airflow

No cenário com dados oriundos de acesso via API acessando informações da campanha de vacinação contra COVID19 foram realizadas quatro tarefas diferentes. Primeiramente o download dos dados em formato json e a transformação dos mesmos em formato tabular, csv,

com uma função Python, Figura 21. Depois o mesmo arquivo é recarregado e uma etapa de escolha de campos é realizada gerando um novo arquivo tabular, essa etapa de escolha de campos só foi realizada aqui para facilitar a manipulação, porém, normalmente o arquivo inteiro é carregado sem alterações. Por fim, a tabela é criada no Hive e a inserção dos dados feita da mesma forma como no carregamento de arquivo csv já apresentado, finalizando a extração e carga dos dados na camada bronze e seguindo então para a camada silver.

```
def json_to_df():
    df = pd.DataFrame(api_con_transf())
    change_json = [df['hits'][5][i]['_source'] for i in range(0,len(df['hits'][5]))]
    df2 = pd.DataFrame()
    for i in range(0,len(df['hits'][5])):
        df2 = pd.concat([pd.DataFrame(change_json[i],index=[i]), df2])
    return df2
```

Figura 21 - Detalhe da conversão de json para dataframe

5.2.4.2 Camada silver

Após a ingestão de dados da camada bronze, o trabalho com os dados passou a ser realizado através de scripts HQL. Em geral, consistindo em no mínimo 3 tarefas orquestradas pela DAG:

- Criação da tabela que será populada no Hadoop pelo Hive e que depois de realizada poderia ser descartada ou pulada;
- Transição de dados de uma camada para a outra, bronze para silver, sendo que aqui também caberiam outras tarefas de tratamento de dados e transformação;
- Exclusão da tabela que ficava na camada bronze, Figura 22.

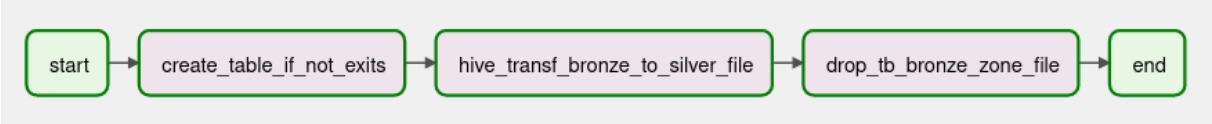

Figura 22 - Detalhe da DAG de transição de bronze para silver

Finalizando o processo dessa camada, passamos a fazer a mesma união de tabelas praticadas pelo Pentaho, como será observado na próxima e última seção finalizando o pipeline com Airflow. Seguimos então para a camada gold.

5.2.4.3 Camada gold

Na camada gold executamos mais uma vez a tarefa de exemplo de realizar a união de duas tabelas para demonstrar uma ação, procedendo novamente a junção, *join*, de tabelas diferentes – a base educacional que teve campos tratados e selecionados e os dados de geolocalização foram mantidas para que o processo fosse o mais próximo possível um do outro – a concretização da junção das tabelas deu-se por um único script que já agregava o necessário, Figura 23.

```
INSERT OVERWRITE TABLE `gold_zone`.`tb_gold_airflow_file_join_bd`PARTITION (`ano_censo`)
SELECT
  `num_matriculas`,
  `id_historia` ,
  `id_geografia` ,
  `id_filosofia` ,
  `id_ensino_religioso` ,
  `id_estudos_sociais` ,
  `id_sociologia` ,
  `id_informatica_computacao` ,
  `id_educacao_indigena` ,
  `cod_uf` ,
  `sigla_uf` ,
  `nome_municipio` ,
  `cod_latitude` ,
  `cod_longitude` ,
  `sigla_regiao`,
  `ano_censo`
FROM `silver_zone`.`tb_silver_airflow_turma`
JOIN `silver_zone`.`tb_silver_airflow_mun_br_lat_long`
ON `tb_silver_airflow_turma`.`fk_cod_municipio` = `tb_silver_airflow_mun_br_lat_long`.`cod_ibge`
```

Figura 23 - Script de união das tabelas de educação e localização geográfica

Nele verificamos, então, que primeiro escolhemos as variáveis que farão parte da nova tabela, garantimos o campo que utilizado para particionamento, no caso pelo ano do censo, e unidas pelo campo “cod_municipio” e “cod_ibge”, que não farão parte do produto final, servindo apenas de elo de ligação.

Aqui finalizamos o processo de elaboração do pipeline de dados e partimos então para uma pequena análise do que pôde ser observado enquanto da realização do trabalho.

5.3 Análise dos resultados

Após o desenvolvimento de pipeline de dados tendo como ferramentas de desenvolvimento de um lado o Pentaho e de outro o Airflow em conjunto com o Python podemos observar alguns pontos de destaque.

De uma forma geral o resultado é a realização das tarefas pretendidas, mas que são executadas de maneiras bem distintas. Observa-se que podemos até ter públicos-alvo diferentes: de um lado uma parcela de usuários mais generalista e de outro um grupo com uma carga de conhecimentos específicos.

Assim, seguiremos a Tabela 3 a fim de organizar a discussão passando pelos principais pontos que foram levantados durante o desenvolvimento.

Tabela de Contraposição		
Requisito	Pentaho	Airflow + Python
Instalação	Arquivo único zipado	Python ou Docker
Documentação	Documentação básica	Documentação abundante
Aprendizado geral	Muitos tutoriais e comunidade ativa	Muitos tutoriais e comunidade ativa
Principal forma de interação de desenvolvimento	Arrasta e solta direto na interface	Scripts em linguagem de programação
Execução na camada bronze	Importação direta dos dados das três fontes analisadas	Importação direta dos dados das três fontes analisadas
Execução na camada silver	Executada por tarefas programáveis graficamente	Tarefas executadas por scripts HQL
Execução na camada gold	Executada por tarefas programáveis graficamente	Tarefa executada por scripts HQL

Tabela 3 - Principais ponto observados durante o desenvolvimento

No requisito de instalação o Pentaho traz tudo pronto em uma pasta compactada, mas esse mesmo ponto também pode ser visto de forma negativa, visto que o usuário tem um arquivo muito grande para realizar o download. Por sua vez o Airflow é instalado via Python ou Docker, necessitando, desde a instalação, algum conhecimento na linguagem.

A documentação de ambos é adequada, mas nesse quesito o Airflow é muito superior, contendo informação sobre todos os pontos necessários e indo além com exemplos e dicas de como criar seus próprios componentes. Ambas as documentações estão apenas em inglês.

Embora a curva de aprendizado do Python seja excelente, para quem está iniciando o desenvolvimento de pipelines o Pentaho se mostra mais fácil. Fato relacionado ao tipo de

interface *drag* e *drop* disponível no Pentaho, enquanto o Airflow funciona baseado em scripts Python.

Na execução do desenvolvimento na camada bronze o Pentaho facilita para o desenvolvedor contendo conexões para os principais tipos de dados e fontes de dados, ajudando na inferência de tipos de dados e criação de tabelas. Por outro lado, o Airflow propicia uma maior manipulação dos dados, já que quem faz todo o processo de modelagem é o desenvolvedor.

Na camada silver ambos se mostraram práticos. Mesmo um atuando por scripts e outro por interface gráfica. Vale observar que os scripts usados pelo Airflow nessa etapa foram do tipo HQL.

A união realizada na camada gold finalizou o processo e disponibilizou a tabela para análise via conexão jdbc. Nesse caso, os dados poderiam ser reestruturados em dashboards. Pentaho e Airflow realizaram satisfatoriamente a tarefa.

6 CONSIDERAÇÕES FINAIS

Por fim, observa-se que Pentaho e Airflow com Python são ferramentas bem diferentes, mas que realizam bem as ações que se propõem a fazer e que podem até mesmo fazer etapas diferentes do desenvolvimento ganhando tempo no resultado. Dessa forma, a sugestão é que etapas que podem ser realizadas por um usuário comum possam ser realizadas no Pentaho, enquanto manipulações mais específicas podem ser orquestradas com o Airflow.

Uma outra sugestão possível é que o Pentaho seja a porta de entrada para o mundo da engenharia de dados, mas que após o ganho de mais conhecimento ou experiência, que o desenvolvedor experimente o Airflow, pois provavelmente irá gostar de ter maior poder de manipulação sobre os dados.

Com relação aos dados que foram utilizados entendemos que após a execução do pipeline de dados temos uma base adequada para ser utilizada no entendimento de questões escolares com granularidade suficiente para comparações regionais, já que na camada gold unimos uma base educacional com dados de geolocalização. Conseguindo alcançar o objetivo de dispor de dados úteis para humanas, nesse caso pedagogia.

Também entendemos que alcançamos ao objetivo de implementar uma solução financeiramente viável por conseguir empregar apenas ferramentas open source, e dessa forma o custo de investimento cai muito. Viabilizando sua execução em diferentes tipos de organizações.

Por fim, como próximo passo, há a expectativa de criar uma interface web e assim utilizar pipelines para a composição de uma base grande que possa servir de fonte de informações para inúmeras áreas das ciências humanas.

REFERÊNCIAS

APACHE AIRFLOW. **Apache Airflow**. Disponível em: https://airflow.apache.org. Site oficial. Acesso em: 23 out. 2021.

APACHE HADOOP. **Apache Hadoop**. Disponível em: https://hadoop.apache.org. Site oficial. Acesso em: 10 out. 2021.

APACHE HIVE. **Apache Hive TM**. Disponível em: https://hive.apache.org. Site oficial. Acesso em: 09 out. 2021.

ARMBRUST, Michael; DAS, Tathagata; SUN, Liwen; YAVUZ, Burak; ZHU, Shixiong; MURTHY, Mukul; TORRES, Joseph; VAN HOVELL, Herman; IONESCU, Adrian; ŁUSZCZAK, Alicja. **Delta lake: high-performance acid table storage over cloud object stores**. Proceedings Of The Vldb Endowment, [S.L.], v. 13, n. 12, p. 3411-3424, ago. 2020. VLDB Endowment. http://dx.doi.org/10.14778/3415478.3415560. Disponível em: https://databricks.com/wp-content/uploads/2020/08/p975-armbrust.pdf. Acesso em: 29 out. 2021.

COSTA, Marcelo. **Pentaho Data Integration - ETL em Software Livre**. 2017. Disponível em: https://www.infoq.com/br/articles/pentaho-pdi/. Accesso em: 01 set. 2021.

EL-SAPPAGH, Shaker H. Ali; HENDAWI, Abdeltawab M. Ahmed; BASTAWISSY, Ali Hamed El. **A proposed model for data warehouse ETL processes**. Journal of King Saud University - Computer and Information Sciences, [S.L.], v. 23, n. 2, p. 91-104, Jul. 2011. Elsevier BV. Disponível em: https://www.sciencedirect.com/science/article/pii/S131915781100019X. Acesso em: 27 de maio de 2021.

GARCIA, Marco. **Data Enginner ou Engenheiro de Dados - Conheça mais sobre**. 2020. Disponível em: https://www.cetax.com.br/blog/data-engineer-ou-engenheiro-de-dados/. Acesso em: 24 de maio de 2021.

GEHRING, Cristiano. **Big Data : análise e a avaliação de ferramentas para gerenciamento de grande volume de dados**. 2018. [18] f. Artigo de conclusão de curso (Bacharel em Ciência da Computação). Curso de Ciência da Computação. Universidade de Passo Fundo, Passo Fundo, RS, 2018.

Goldman, A., Kon, F., Pereira Junior, F., Polato, I. & Pereira, R. (2012). **Apache Hadoop: Conceitos teóricos e práticos, evolução e novas possibilidades**. XXXI Jornadas de atualizações em informática.

GOUVEIA, Roberta Macêdo Marques; FREITAS, Charles Nicollas Cavalcante. **Implementação de um Data Warehouse para Análise de Dados Abertos Governamentais da Educação à Distância**. Revista de Educação Ciência e Tecnologia, Canoas, v. 7, n. 2, 2018. Disponível em: https://periodicos.ifrs.edu.br/index.php/tear/article/view/3037/2112. Acesso em: 25 maio 2021.

HARENSLAK, Bas; RUITER, Julian de. **Data Pipelines with Apache Airflow**. Shelter Island, Ny: Manning Publications Co., 2021.

HITACHI VANTARA. 2021. Disponível em: https://community.hitachivantara.com/s/. Acesso em: 01 set. 2021.

IEEE. **Top Programming Languages 2021**. 2021. Disponível em: https://spectrum.ieee.org/top-programming-languages/. Acesso em: 20 out. 2021.

LIRA FILHO, Hermanny Alexandre dos Santos. **Análise comparativa das Ferramentas de ETL – Kettle e Talend**. 2013. 79 f. TCC (Graduação) - Curso de Bacharelado em Sistemas de Informação, Universidade Federal da Paraíba, Rio Tinto, 2013. Disponível em: https://repositorio.ufpb.br/jspui/handle/123456789/17132?locale=pt_BR. Acesso em: 29 maio de 2021.

MATTHES, Eric. Curso Intensivo de Python: uma introdução prática e baseada em projetos à programação. São Paulo: Novatec, 2016.

MICROSOFT. **Visão geral dos cubos OLAP do Service Manager para análise avançada**. 2021. Disponível em: https://docs.microsoft.com/pt-br/system-center/scsm/olap-cubes-overview?view=sc-sm-2019. Acesso em: 30 maio de 2021.

MICROSOFT. **Descrever a arquitetura do Delta Lake do Azure Databricks**. 2021. Disponível em: https://docs.microsoft.com/pt-br/learn/modules/describe-azure-databricks-delta-lake-architecture. Acesso em: 29 out. 2021.

PESSOTTI DA ROCHA MAGDALENO, Rafael. **Analisando o Pentaho como Ferramenta de BI**. 2015. 55 f. TCC (Graduação) - Curso de Bacharelado em Sistemas de Informação, Centro Universitário Eurípides de Marília, Marília, 2015. Disponível em: https://aberto.univem.edu.br/bitstream/handle/11077/1383/Analisando%20o%20Pentaho%20como%20ferramenta%20de%20BI.pdf?sequence=1&isAllowed=y. Acesso em: 15 set. 2021.

PYTHON.ORG. **Site Oficial**. Disponível em: https://python.org.br. Site oficial. Acesso em: 12 out. 2021

RAMPAZO, Paolo. **Apache Airflow: conceitos iniciais**. 2021. Disponível em: https://medium.com/datarisk-io/apache-airflow-conceitos-iniciais-e09c0dd18141. Acesso em: 23 out. 2021.

RASLAN, Daniela Andrade; CALAZANS, Angélica Toffano Seidel. **Data Warehouse: conceitos e aplicações**. Universitas: Gestão e TI, Brasília, ago. 2014. Centro de Ensino Unificado de Brasília. Disponível em: https://www.researchgate.net/publication/287517542. Acesso em: 25 de maio 2021.

RAU, Isabele Aurora Cândido Vitorino. **Data Lake: uma nova abordagem para o armazenamento de dados.** 2021. 72 f. TCC (Graduação) - Curso de Bacharel em Sistemas de Informação, Universidade do Sul de Santa Catarina, Florianópolis, 2021. Disponível em: https://repositorio.animaeducacao.com.br/handle/ANIMA/13790. Acesso em: 27 out. 2021.

SAS. **ETL: o que é e qual sua importância?** [20--]. Disponível em: https://www.sas.com/pt_br/insights/data-management/o-que-e-etl.html. Acesso em: 30 maio 2021.

SENE, Allan. **O que faz um Engenheiro de Dados?** 2018. Disponível em: https://medium.com/data-hackers/o-que-faz-um-engenheiro-de-dados-fdcb0bca966b. Acesso em: 25 de maio 2021.

SHENOY, Aravind. **Hadoop Explained.** Birmingham: Packt Publishing, 2014.

TAMIR, Mike; MILLER, Steven; GAGLIARDI, Alessandro. **The Data Engineer**. Ssrn Electronic Journal, [S.L.], 2015. Elsevier BV. Disponível em: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2762013. Acesso em: 29 maio 2021.

WHITE, Sarah K. **O que é e o que faz um engenheiro de dados?** 2018. Disponível em: https://cio.com.br/tendencias/o-que-e-e-o-que-faz-um-engenheiro-de-dados/. Acesso em: 24 de maio 2021.