Data Engineering Project Report

I. Topics

1. Motivation

Our goal was to gain insight into the crimes committed in the cinema industry.
Indeed, following the numerous recent cases of sexual harassment perpetrated by well-known actors and producers, we initially wanted to focus on sexual harassment data in the cinema sector. We subsequently extended it to any type of assault or crime due to lack of data. As the legal problems of actors and crew members are often discussed, we wanted to see if there were real consequences on the popularity of films for example.

2. Questions

We wanted to answer the following questions:

- Is the popularity of movies affected by the legal troubles of its cast members?
- Are there genres that attract more people with legal troubles ?
- Is there a cinema profession that attracts more people with legal problems?

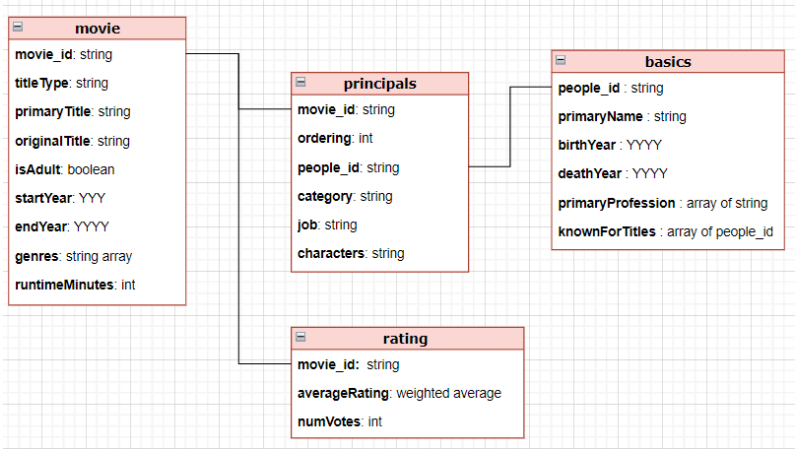
II. Data Source

Our ingestion pulls data from 2 different data sets:

1. IMDB

We get some data from the Internet Movie Database (IMDB) website which provides an online database of information related to movies, actors, producers…
The dataset can be found here : https://developer.imdb.com/non-commercial-datasets/

We downloaded 4 tables from the website:



- The movie table gives information about movies, their title, if the movie is PEGI 18, the startdate and end dates of filming, the genre, and how long the movie is.

- The basics dataset gives informations about the people that works in the cinema: we have their name, date of birth and their profession (actors, producers, writers)

- The principals dataset associates a person with a film in which they worked, with their job in that movie.

- Finally, the rating database give the average rating and the number of people who rated that movie

2. Wikidata

Regarding our justice data, we retrieved it from WIKIDATA : a graph database that normalizes part of the data available on wikipedia.
We used the following request to get our data from this database:

```
SELECT DISTINCT ?personLabel ?offenceLabel ?offence WHERE {
  ?person p:P106 _:b168.
  _:b168 ps:P106 ?metier.
  ?metier p:P31 _:b169.
  _:b169 ps:P31 wd:Q4220920.
  {
    ?person p:P1399 _:b171.
    _:b171 ps:P1399 ?offence.
  }
  UNION
  {
    ?person p:P1344 ?statement0.
    ?statement0 ps:P1344 ?offence.
    ?offence p:P31 _:b167.
    _:b167 (ps:P31/(wdt:P279*)) wd:Q1456832.
  }
  UNION
  {
    ?person p:P793 ?statement0.
    ?statement0 ps:P793 ?offence.
    ?offence p:P31 _:b187.
    _:b187 (ps:P31/(wdt:P279*)) wd:Q8016240.
  }
  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "en,fr".
    ?person rdfs:label ?personLabel.
    ?offence rdfs:label ?offenceLabel.
  }
}
ORDER BY (?personLabel)
```

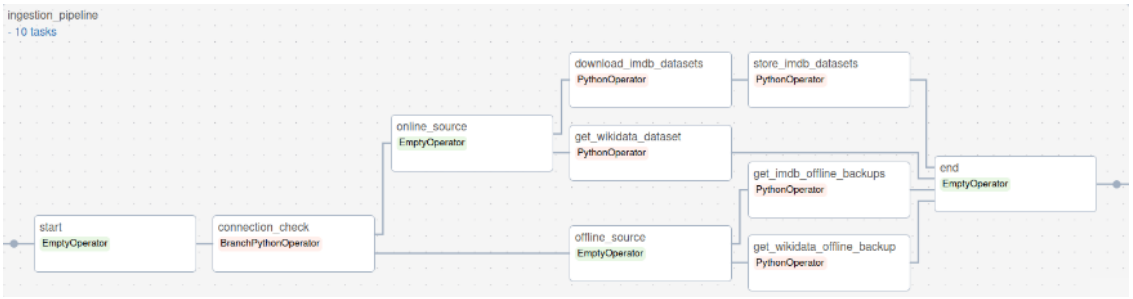
We first check if ?person has an occupation related to movies.
We then identified several cases (separated by 'UNION') in order to maximize the data returned:

- ?person was convicted of ?offence.
- ?person has important event ?offence and ?offence is an instance of legal case
- ?person participates in ?offence and ?offense is an instance of legal offence

Thanks to this request, we have around 190 lines returned.

III. Data ingestion

Here is what our data ingestion pipeline looks like on airflow :



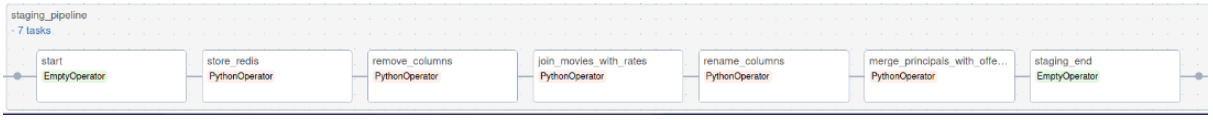
We first check our connection with a PythonBranchOperator. If we are connected to the internet, we will then download our dataset directly from the internet. Otherwise, we will get our data from sampled CSV that we have previously stored in a folder.

In both cases, we get our data from csv files that we open with panda and store in MongoDB.

We downloaded our data from wikidata using the query presented previously. And we also get 4 csv files for each table from the IMDB dataset. Those two steps are done simultaneously, because they don't depend on each other.

IV. Data Staging

Our staging pipeline contains 7 differents tasks:



Our staging pipeline represents the main part of our pipeline. It is supposed to clean the data and store it into a nice SQL database, so that it can be used by the production overlords.

1. Development process

We mainly used pandas to extract our data from the csv files and to do the transformations. Pandas is indeed an easy tool to use and to understand. We also used REDIS throughout the whole wrangling phase, in order to reduce data access time, improve performance, and facilitate, accelerate storage between each task in our wrangling pipeline.

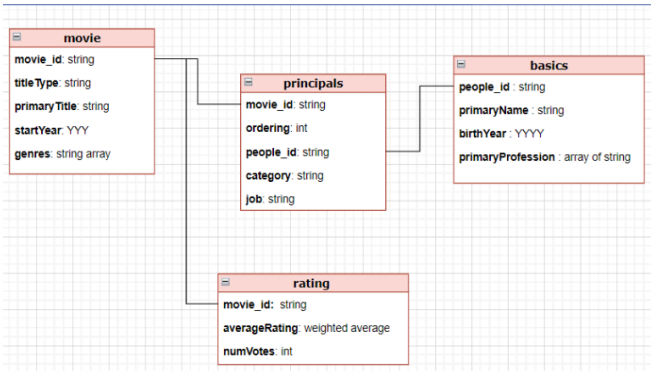
Finally, we store our data in MongoDB, which helps to implement persistence.

2. Data cleaning and wrangling

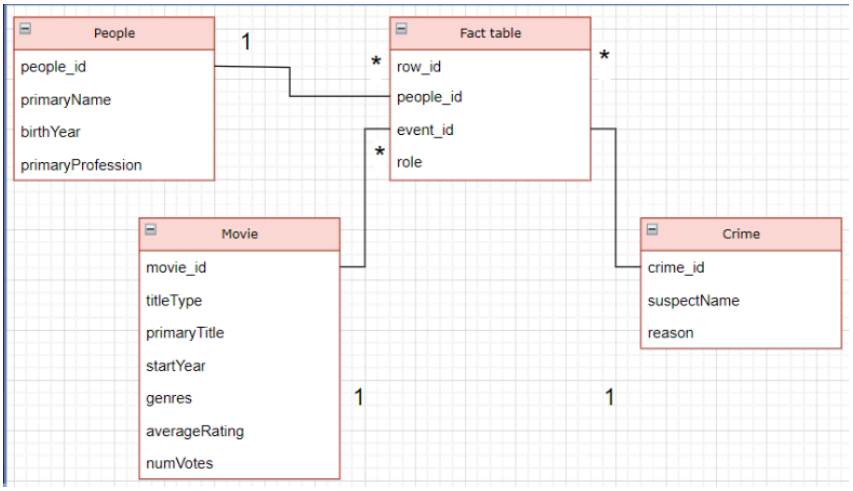
The nuclear and thermal plant data was already very clean so only a minimum amount of cleaning was done:

The IMDB data needed some preprocessing, we first removed some unimportant columns in order to improve performance. We removed for instance the deathYear column, the runtime of the movie, if the movie was pegi 18… We also had to add some indexes, rename some columns for better understanding…

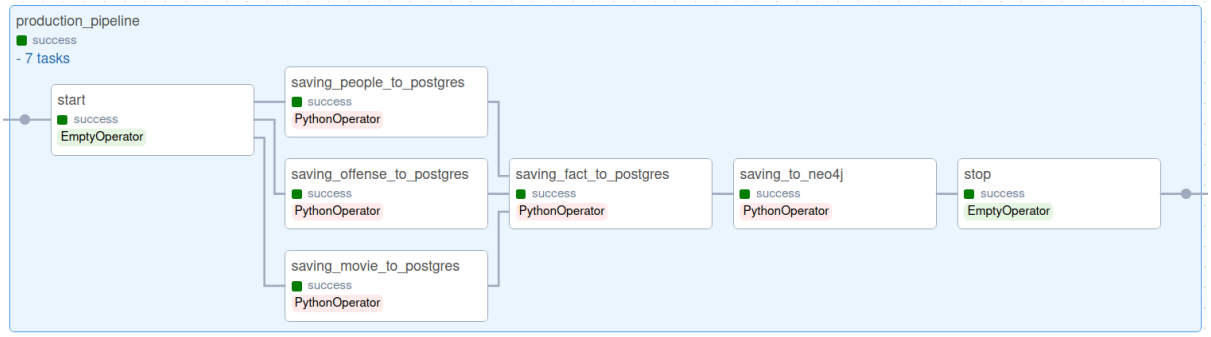
This lead us to the following schema :



Then we joined our movie and rating tables using the movie_id field. Finally we merged our two dataset using the primary name of the actors. We finally got the following star schema:



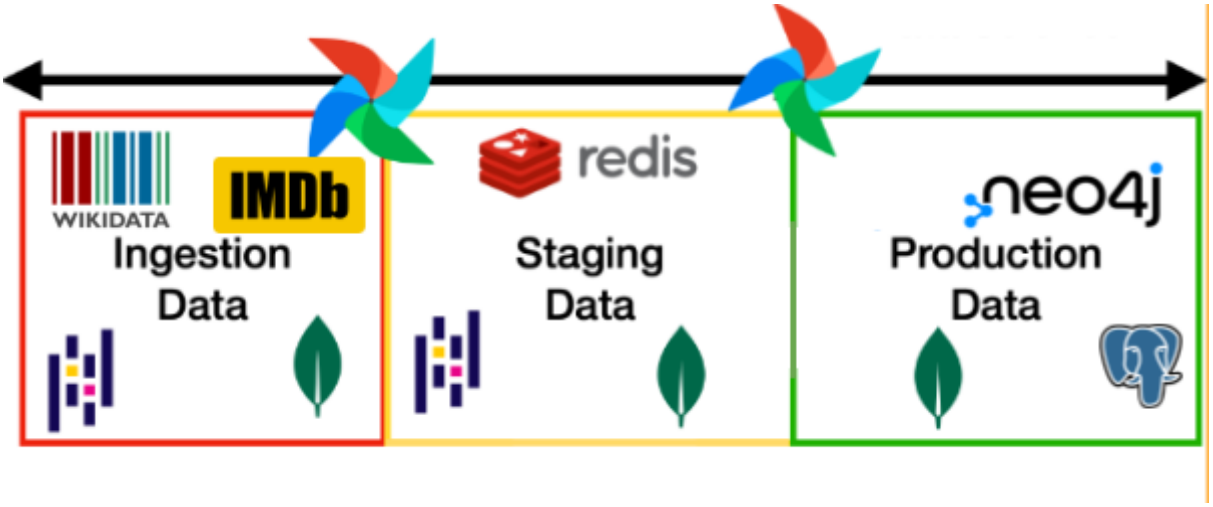
V. Production Phase



In the production phase we exported the data to a Postgresql database to make the data persistent. We separated the work into 4 nodes, one for each table of our star schema, so that the work can be parallelized. The Fact table is created afterwards to get the foreign keys from the events and people table. We had to define a polymorphic entity Event, to be able to create a foreign key field for Crime and Movie

We also exported the data from Postgresql to neo4j to be able to access the data in a graph database because that would make it easier to query and visualize it considering the nature of the information stored.

VI. Tool used



We used the following tools during the project:

- MongoDB to store and query the cleaned data
- pandas to extract and stage the data
- Redis for better performance during the staging phase
- SPARQL to query wikidata
- Postgresql for data persistence
- Neo4j to have a queryable graph

VII. Difficulties

1. Data collection difficulties:

We faced multiple problems trying to get legal data from wikidata. Indeed, it was hard to find extensive, reliable and structured data on the matter. There are also rarely manichean legal issues.

If we are only interested in the facts that have been condemned by the courts, we have very little data. Indeed, few cases which concern such influential men are really condemned by the courts. On the other hand, if we extend the research to social networks we may then be confronted only with rumors. It was hard to find a balance that gave us sufficient data to analyze.

We had some ideas to collect our legal dataset:

- ★ Exploring social media data
  However, as we said it is really not reliable. It was also a large amount of unstructured data that required a lot of processing.
- ★ Using the data available on wikidata
  The data was structured and easy accessed, however it was not extensive and we had few results.
- ★ Data from newsPapers
  We couldn't find a usable source.

We then decided to use data from wikidata despite the few results returned.
We have enough data to try to analyze it, however it is too few to be representative.

2. Hardware related difficulties:

- We both had a hard time trying to install and use docker. Margaux managed to install it on her computer late in the semester by setting up an unstable dual boot that crashed frequently and unpredictably .
- Félicie's laptop also struggled with docker due to a memory problem, the pipeline would then take forever to run, without finishing, even when the pipeline worked well on Margaux's computer.