Designing and Implementing a Data Pipeline to Analyze Scientific Publications

Abstract

In this project we focus on creating a powerful data foundation by combining a structured Data Warehouse (DWH) and a Graph Database. The Data Warehouse uses a Star Schema in MySQL and organizes data around authors, papers, journals and more. Business Intelligence queries also help for ranking authors and analyzing publication trends. The Graph Database helps understand relationships among authors, papers, and journals. Data transformations made in the pipeline involve finding citations, cleansing, and enrichment of data.

Introduction

The combination of structured Data Warehousing (DWH) and the representation of relationships in a Graph Database can be a very useful strategy to understand and visualize data. This project establishes a robust data foundation for effective organization and exploration of relationships within the dataset.

With a Star Schema within the MySQL environment, our Data Warehouse structures information related to authors, papers, journals, and various dimensions in the domain. Business Intelligence queries expand our analytical capacities, making a ranking of authors among many other analytics. The Graph Database also offers a dynamic way of understanding relationships among authors, papers, and journals.

The pipeline that has been created for the project also applies transformations in the data involving finding citations, cleansing, and enriching data. Data ingestion is simplified by working with a subset of the data and making partitions. Cleansing steps involve resolving ambiguous author and venue names, excluding less relevant features and filtering out papers that don't meet specific criteria. We also get affiliations and citations of the paper by using some APIs, enriching the data that we currently have. The transformed and enriched data is then loaded into both a MySQL database and Graph database.

Description

The dataset that we are using for the project originated from the ArXiv repository. The dataset offers a large amount of scholarly articles across various domains, ranging from physics to computer science, mathematics, statistics, electrical engineering, quantitative biology, and economics.

Given the vastness of this dataset, we have chosen to work with a subset of the dataset, to make the data more manageable. To simulate an incremental update to the target Data Warehouse (DW) and Graph Database, we have partitioned these records into smaller sets, feeding them iteratively into our pipeline.

The ArXiv dataset contains very interesting data, including article titles, authors, categories, Digital Object Identifiers (DOIs) and more in JSON format. The metadata file contains essential information for each paper:

- **ArXiv ID (id):** Unique identifier for accessing the paper.
- **Submitter (submitter):** Individual or entity who submitted the paper.
- **Authors (authors):** List of authors contributing to the paper.
- **Title (title):** The title of the paper.
- **Comments (comments):** Additional information such as the number of pages and figures.
- **Journal Reference (journal-ref):** Details about the journal in which the paper was published.
- **DOI (doi):** Digital Object Identifier for the paper.
- **Abstract (abstract):** A summary of the paper's content.
- **Categories/Tags (categories):** Classification in the ArXiv system.
- **Versions (versions):** Version history of the paper.

Our objective is to make this extensive dataset more accessible and analyze the relations between authors, papers, journals and more.

Data Warehouse Analysis

In the DWH we firstly designed a snowflake schema; however, we soon realized that it had a complex normalization and we have changed to a simpler star schema. To create the schema we had to fulfill the following structure:

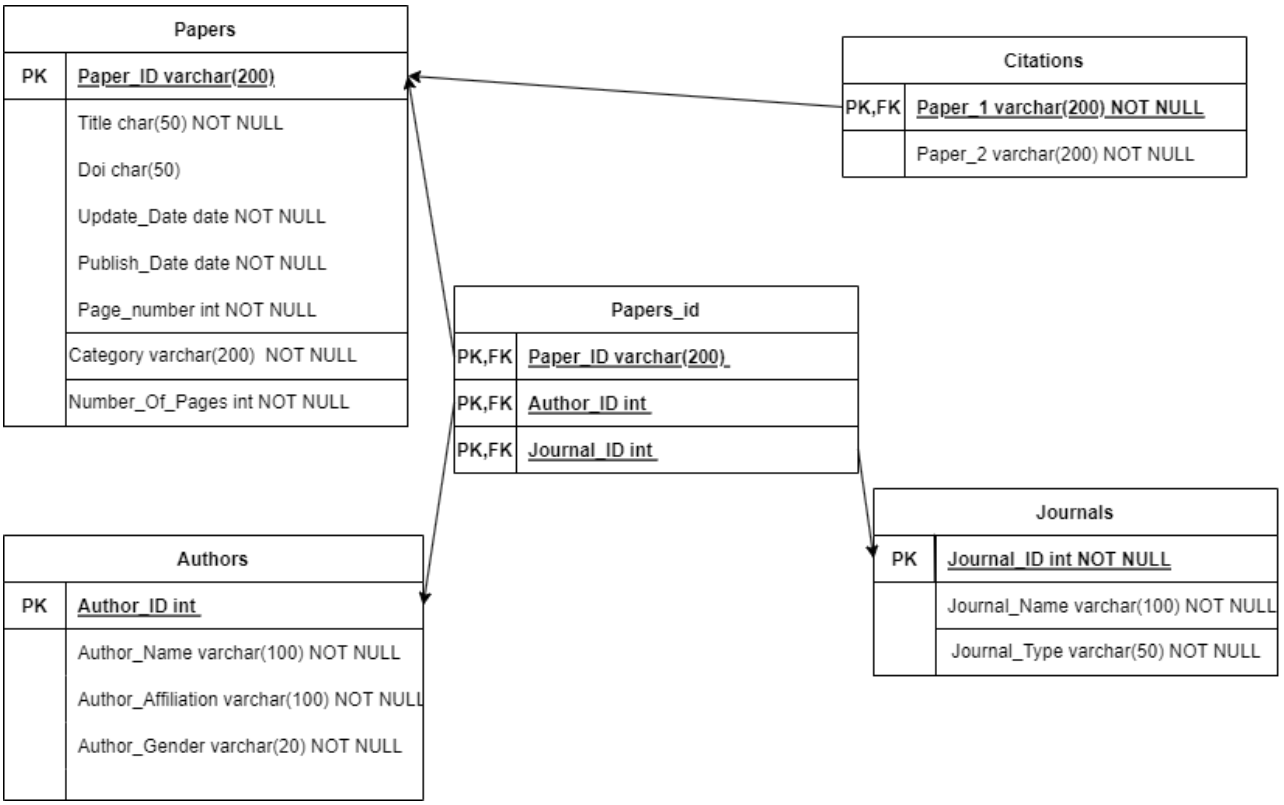
Dimension Tables:

1. **Authors**
2. **Papers**
3. **Journals**
4. **Citations**

Fact Table:

- **Papers_ID**

That structure was used to create this design:



Although we have a star schema we can see a table (Citations) that is not directly linked to the fact table. This happens because the table is used to reference papers of the table Papers, it is an auxiliary table. This table is necessary because the papers cited can be more than one. Moreover, we thought of Paper2 using a foreign key but we realized that most of the papers will not be in the database. Thus, we removed the foreign key.

Business Intelligence Queries:

1. **Rank Authors in a Scientific Discipline:**
    - Identify top authors based on the number of publications in a specific domain.
2. **Compute H-Index:**
    - Calculate the H-Index for authors based on citations and publications.
3. **Histogram of Publications Over Time:**
    - Analyze the distribution of publications over a given time period in a specific domain.

SQL Queries:

1-**Rank Authors in a Scientific Discipline**

SELECT A.Author_ID, A.Author_Name, COUNT(PA.Paper_ID) AS Publication_Count

FROM

```
   Authors A JOIN Papers_ID PA ON A.Author_ID = PA.Author_ID JOIN Papers P ON
PA.Paper_ID = P.Paper_ID

WHERE
   P.Category = 'Science' –This is changed for each discipline
GROUP BY
   A.Author_ID, A.Author_Name
ORDER BY
   Publication_Count DESC;
;
```

2-**Compute H-Index**

```
SELECT A.Author_ID, A.Author_Name, COUNT(PA.Paper_ID) AS Publication_Count,
COUNT(C.Citation_ID) AS Citation_Count

FROM
   Authors A JOIN Papers_ID PA ON A.Author_ID = PA.Author_ID JOIN Papers P ON
PA.Paper_ID = P.Paper_ID LEFT JOIN Citations C ON P.Paper_ID = C.Paper1

GROUP BY
   A.Author_ID, A.Author_Name
HAVING
   Publication_Count >= Citation_Count
ORDER BY
   Publication_Count DESC;
```

3-**Histogram of Publications Over Time**

```
SELECT YEAR(P.Publish_Date) AS Publication_Year, COUNT(P.Paper_ID) AS
Publication_Count

FROM
   Papers P

WHERE
   P.Category = 'Science' –Here you choose the discipline
GROUP BY
   YEAR(P.Publish_Date)
ORDER BY
   Publication_Year;
```

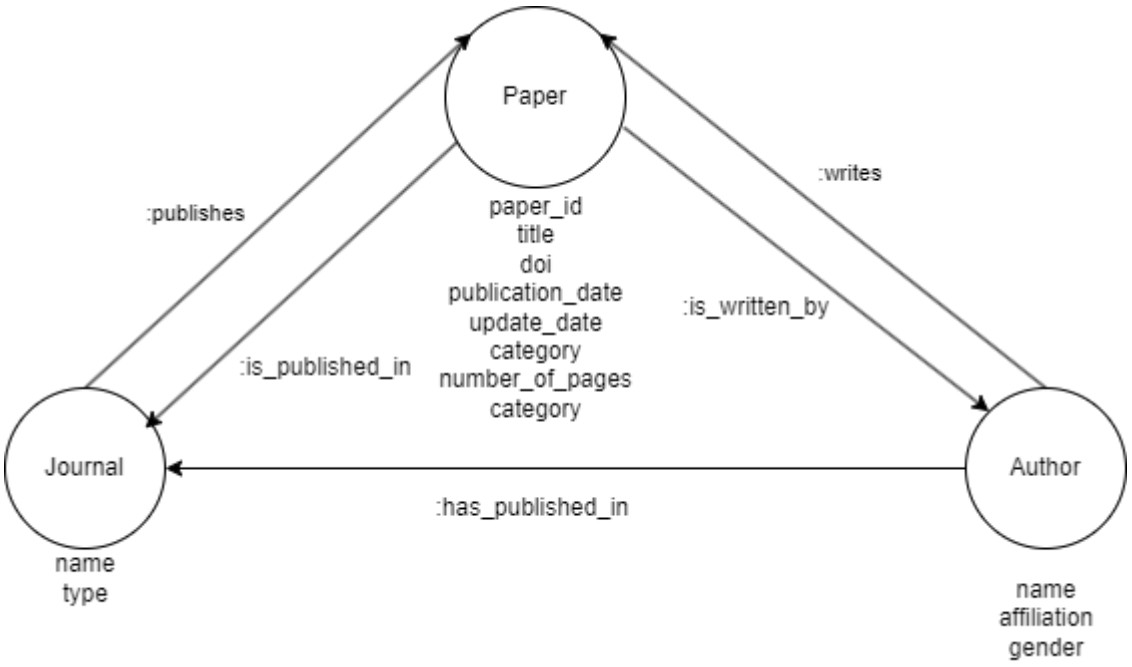
Graph Database Analysis

As in the DWH we followed a structure to create the graph.

1. **Authors**
2. **Papers**
3. **Journal**

Relationships:

1. **Publishes:**
    - Relates the journal with all the papers that are included in it
2. **Is published in:**
    - Relates the paper with the journal where it appears
3. **Writes:**
    - Says who is the author of a paper
4. **Is written by:**
    - Shows which papers has an author written
5. **Has published in:**
    - Establishes in which journals an author appears
6. **References:**
    - Points to the different papers that are referenced in a paper



We performed some analytics but we couldn't do a complex analysis as we initially planned.

Graph Analytics Tasks:

1. **Co-author analysis:**

    - Given an author shows with which other authors it has collaborated with
2. **Shortest link analysis:**
    - Shows how “far” an author is from other. This is the shortest path between two authors linking co-authors.

Graph BI queries:

1- Co-Author Analysis

```
MATCH (a:Author{name:'Name of the author '}) -[:Writes] -> (p:Paper)
<-[:Writes]-(coAuthor:Author)
return coAuthor
```

2- Shortest Link

```
MATCH path = shortestPath((:Author {name: ‘Name of the Author 1’}) -[:Writes*]- (:Author
{name: 'Name of the Author 2'}))
RETURN nodes(path) as coAuthors;
```

The Pipeline

In this section, we will explain our pipeline with all the transformations that we apply to our data:

- ### Data Ingestion

For the data ingestion, we collect a small subset of the data that we had in the initial database (5000 entries to be specific) from the whole .json file and feed that data into our pipeline.

Due to the way our whole pipeline is put together, feeding more subsets of data should be very easy (just feed other entries from the .json file to the pipeline and the databases should populate in an incremental way).

- ### Data Cleansing

For the data cleansing, we implemented some functions to know if each entry in the dataset meets the criteria that we wanted. The main goal of this phase is to filter the dataset, making it easier for data analysis.

First of all, we check if the paper has a comment indicating the number of pages that it has, and if so, we only preserve the ones that have more than 4 pages. We also check if the

paper has authors and DOI (Digital Object Identifier), as these are very important features for our analysis later on.

The attributes that we remove are: "report-no", "license", "created", "submitter", "versions", "abstract" and "authors_parsed".

While all of this is checked and removed from the papers, we create a set to store unique DOIs and a list to store cleaned data. This way, we are able to loop over each paper in the input data, cleaning everything and returning the data that meets the criteria that we imposed.

- ## Data Enrichment

Regarding the data enrichment, we've used a different API in order to get more information about the data we already had. These are the steps we took:

1. Author name normalization: We normalize author names using the Crossref API. This way, we concatenate names in a consistent format.
2. Number of pages: Using the comments section in the database we got the number of pages of each paper. Some papers don't have this information.
3. Get type of the paper: Using the cross ref API we have concluded the type of each publication. Different types of publications are Journal, conference, book …

- ## Data Augmentation

During data augmentation, we get the information about the affiliation of each author, the citations that each paper does and the publication date. Here is a description for each step:

1. Get the affiliation of each author: Even though there's not a since API available where we could get the affiliation information about every author, we've decided to use DBLP and search for the author name. Since our name normalization is not perfect either, this has resulted in many authors having Unknown affiliation.
2. Citations: Since Crossref API does have reference information, we've used that to get the DOI of all the papers that each work cites.
3. Publication date: Publication date is other information that Crossref has. We've used this API to also get the publication date. We found out that this date can sometimes be in the format of year-month-day and other times only year is provided.

Conclusions

To conclude, as we have encountered several problems during the project we have adapted the solutions in the best way that we were able to. Despite the problems mentioned, we managed to put together the pipeline, so that we have both databases with the data, prepared to be analyzed.

We have learned about the different steps of data processing and their difficulties. We have also learned about the technology used in this field.

Individual Contribution Statement

In the following list we put together the team members and their contribution in the project:

- Lukas Arana
  - Implementation of the pipeline
  - MySql BI queries
  - Final report
- Gonzalo Añua
  - Creation of the scripts to populate both databases
  - Neo4J BI queries
  - Final report
- Oier Ijurco
  - Implementation of the pipeline
  - MySql BI queries
  - Final report
- Joanes De Miguel
  - Creation of the scripts to populate both databases
  - MySql BI queries
  - Final report

Appendix

GitHub Link to the repository: https://github.com/GonzaloAnuaRecio/DE_Project