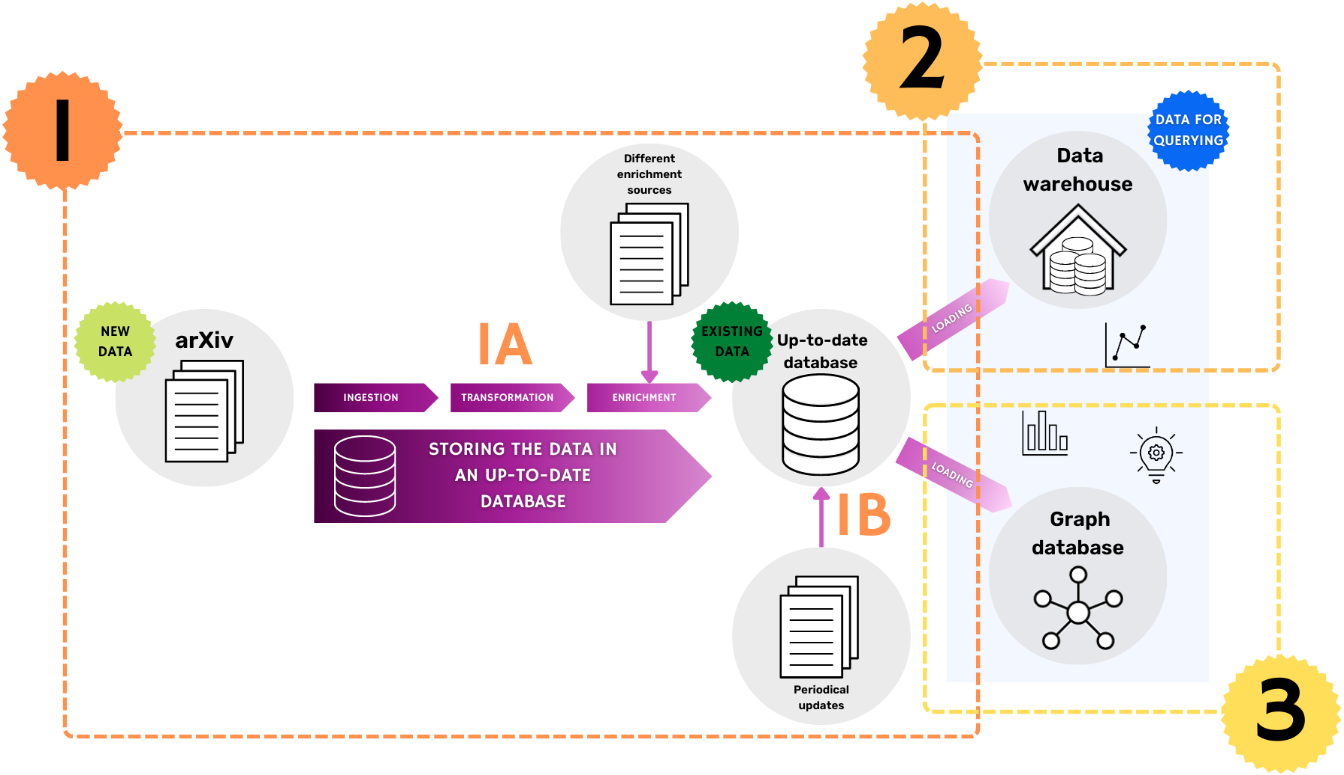
FINAL REPORT

"*Data is like garbage. You'd better know what*

*you are going to do with it before you collect it.*"

— Mark Twain

In this project, a data pipeline to analyse data about scientific publications was built. In figure 1, the overall pipeline design is shown. As one can see, the pipeline can be divided into three main parts/stages and the first part additionally into two subparts (1A so-called transformation pipeline and 1B so-called updating pipeline). In the first part, 1A, the new raw data in JSON format is ingested into the pipeline, where it is transformed and enriched and then loaded into the up-to-date database. In subpart 1B, the data in the up-to-date database is updated periodically. Also, in the first part of the pipeline, the data is prepared in the correct form (the CSV files are written), so it can be uploaded into the data warehouse (DWH) (part 2) and the graph database (DB) (part 3). In the final report, all these stages are thoroughly covered, including the DWH and graph DB designs, together with relevant queries that DWH and graph DB will answer. And last but not least, the guidelines for running the built data pipeline are given.



**Figure 1** Design of the overall data pipeline

**Part 1**

**Dataset and pre-pipeline processing**

ArXiv is the open-access archive for scholarly articles from a wide range of different scientific fields. The dataset given for the project contains metadata of the original arXiv data, *i.e.* metadata of papers in the arXiv. The metadata is given in JSON format and contains the following fields:

* id – publication arXiv ID
* submitter – the name of the person who submitted the paper/corresponding author
* authors – list of the names of the authors of the paper (in some cases, this field includes additional information about the affiliations of the authors)
* title – the title of the publication
* comments – additional information about the paper (such as the number of figures, tables and pages)
* journal-ref – information about the journal where the article was published
* doi – Digital Object Identifier (DOI) of the paper
* report-no – institution's locally assigned publication number
* categories – categories/tags in the arXiv system, i.e. field of the current study
* license – license information
* abstract – abstract of the publication
* versions – history of the versions (version number together with timestamp/date)
* update\_date – timestamp of the last update in arXiv
* authors\_parsed – previous authors field in the parsed form

In this project's scope, the fields of submitter, title, journal-ref, doi, categories, versions and authors\_parsed were used (all the others were dropped).

The original dataset contains information about more than 2 million publications. To simulate a real-life situation where new data is coming in continuously, the original dataset was split into smaller parts (... parts containing information about ... publications). Also, before ingesting the data into the pipeline, publications, which data will be enriched, were selected. This preselection was made because the enrichment process via API calls is very time-consuming; therefore, enriching all the publications' data is out of this project's scope. The preselection was made by filtering out all the publications with DOI (needed for API calls) and then grouping them by categories (major field + sub-category, see table 1). (The details about how publications have been divided into these categories are given in the next chapter.) From each group, 20 publications, which data will be enriched, were selected, and their DOIs were written into the DOIs\_for\_enrichment.csv file. This file is used in the transformation pipeline, as explained in the next chapter.

**Table 1** Categories (major field together with one sub-category forms one category) that are used for selecting the publications that are going to be enriched

|  |  |
| --- | --- |
| **Major field** | **Sub-category** |
| natural science | * mathematics |
| * computer sciences |
| * information sciences |
| * earth sciences |
| * biology science |
| * physical sciences |
| * chemical sciences |
| engineering and technology | * electrical, electronic and information engineering |
| * mechanical engineering |
| social sciences | * economics, finance and business |

**Transformation pipeline (1A)**

As mentioned before, the first part of the overall pipeline can be divided into two subparts: transformation pipeline (1A) and updating pipeline (1B). This chapter covers the concepts of the transformation pipeline. The details about the updating pipeline are given in the next chapter.

In the transformation pipeline, the raw data in JSON format (as explained in the previous chapter) is ingested into the overall pipeline; then, this data is cleaned and enriched if needed and loaded into the up-to-date database. Because this project uses the approach where the data will be stored in the up-to-date database to simulate the real-life setup where constant data updates are required because of its volatile nature (the number of citations of the publications changes and so on) before discussing all the details about transformation pipeline, main concepts of this database are given. The schema of this DB is shown in figure 2.

Diagram

Description automatically generated

**Figure 2** Schema of an up-to-date database

As one can see, the up-to-date DB consists of 9 relations. The table "AUTHORS" contains all the relevant information, such as last name, first name, the abbreviation of the first name, extra titles, positions, and real-life h-index, about the publications' authors in the DB. In the table "AFFILIATIONS", the institution names and places where authors work are gathered. In the table "PUBLICATIONS", the data about publications (such as DOI, title, publishing date of first and last version, name of the submitter, type and language of publication, volume, issue and page numbers, number of references and citations) are recorded. The relation "ARXIV\_CATEGORIES" contains information about publications' arXiv categories. In the relation "VENUES", all the data about venues (full name, abbreviation, print and electronic ISSN) where publications were published are collected. All other relations are created to deal with the entities' m:n relationships.

The Airflow DAG "create\_DB\_tables\_and\_SQL\_statements" (see figure 3) was written to create this Postgres database. This DAG should be run only once at the very beginning of the overall pipeline. During the run, all the database tables are generated. Besides, all the other SQL statements needed in the first part of the overall pipeline are also generated. These SQL statements include the ones used for populating the tables with new data or updating the existing data (considering all the constraints that are present in the DB), together with the statements used for generating the views (authors' view and venues' view) that are needed for getting the data in the correct form to load into the DWH and graph DB. (The mentioned authors' view contains authors' IDs, last, first, and full names, first name abbreviations, positions, real-life h-indices, and the h-indices calculated based on the data present in the DB. In the venues' view, besides the data that this in the "VENUES" table (venue ID, full name, abbreviation, print and electronic ISSN), also the calculated h-index for each venue is given.)

Diagram

Description automatically generated

**Figure 3** create\_DB\_tables\_and\_SQL\_statements Airflow DAG

The transformation pipeline itself is implemented as an Airflow DAG "articles2DB". The graph representation of this DAG is shown in the following image.

Diagram

Description automatically generated

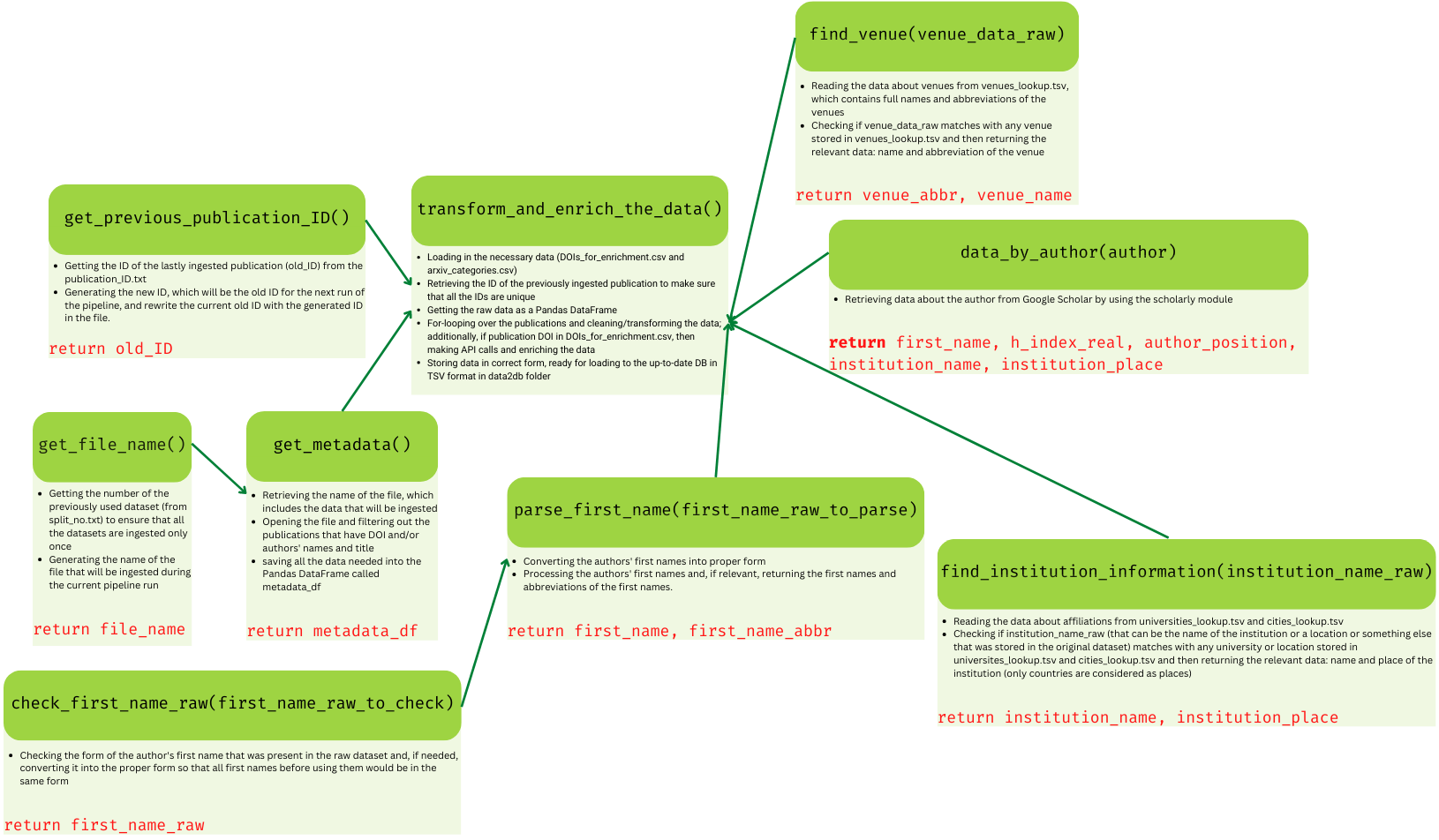
**Figure 4** create\_DB\_tables\_and\_SQL\_statements Airflow DAG

To run this DAG, the "data" folder should contain six subfolders: "inputs", "setups", "lookup\_tables", "data2db", "sql", and "final\_data" (see table 2).

**Table 2** Data folders required for running the first part of the pipeline

|  |  |
| --- | --- |
| **Subfolder** | **Explanation** |
| inputs | The raw JSON datasets generated during pre-pipeline processing by splitting the original arXiv dataset are stored in this folder. | |
| setups | It contains three files: "DOIs\_for\_enrichment.csv", "publication\_ID.txt", and "split\_no.txt". "DOIs\_for\_enrichment.csv" contains DOIs of the articles, which data will be enriched during the transformation pipeline (see chapter "Dataset and pre-pipeline processing"). The file "publication\_ID.txt" is used for storing the last publication ID to ensure that all the publication IDs are unique. Before the first run of the pipeline, the ID in this file should be 0. The file "split\_no.txt" is used for a similar purpose – storing the number of the previously used dataset to ensure that all datasets are ingested only once. Before the first run of the pipeline, the split number should be 0. | |
| lookup\_tables | It contains four tables: "cities\_lookup.tsv", "lookup\_table\_domains.csv", "universities\_lookup.tsv", and "venues\_lookup.tsv". All these tables (except "lookup\_table\_domains.csv") are used for enriching and transforming the raw data.  For enriching and transforming the:   * affiliations' data the "cities\_lookup.tsv" and "universities\_lookup.tsv" are used.  In the "cities\_lookup.tsv", the names of 42905 world's largest cities (including the Unicode and ASCII strings) together with names of countries are given. This data was retrieved from SimpleMaps.com (<https://simplemaps.com/data/world-cities>).  In the "universities\_lookup.tsv", the data (names, countries, cities/locations and abbreviations of locations) of about 1152 world universities are stored. This data was retrieved by merging and preprocessing the World University Rankings (<https://www.kaggle.com/datasets/mylesoneill/world-university-rankings>) dataset and University Rankings 2017 (<https://data.world/education/university-rankings-2017>) dataset. * venues' data the "venues\_lookup.tsv" is used. This dataset contains information such as the name and abbreviation (with and without dots) of 87224 venues. This dataset was retrieved from the Web Of Science (<https://images.webofknowledge.com/images/help/WOS/A_abrvjt.html>). To make the enrichment process easier and faster, the dataset was preprocessed, and therefore the lookup table also contains fields abbrev\_check and full\_check. The lowercase abbreviations and full names of venues without spaces are stored in these fields. | |
| data2db | The data that will be loaded into the up-to-date database are stored in this folder. Before the first run, it should contain two files: "arxiv\_categories.csv" and "venues\_df.tsv". By the way, the latter should contain only the header row before the first run of the pipeline. In the "arxiv\_categories.csv", the IDs and arXiv category names of 156 categories are stored. | |
| sql | This folder is used for storing all the SQL statements. Before the first run, the folder should be empty, but after the run of "create\_DB\_tables\_and\_SQL\_statements" DAG, the .sql files are generated and saved in this folder. | |
| final\_data | This folder should contain the data in the correct form, ready to be loaded into the DWH and graph DB. Before the first run of the pipeline, this folder should be empty. | |

The transformation pipeline starts with the task "transform\_the\_data", which can be considered the most important task in this part of the pipeline. In this task, nine different functions (written in Python by one of the authors of this project) are used (see figure 5).



**Figure 5** Schema of the task transform\_the\_data

During this task, the new dataset (a subset of the original dataset) is selected and ingested into the pipeline. It is important to note that all the publications where essential data (like DOI together with authors and/or title) is missing will be dropped because this may lead to inconsistencies in the final data. (It means if it is impossible to identify the publication unambiguously, all the data about it will be discarded.) Then the fields of submitter, title, journal-ref, doi, categories, versions and authors\_parsed are filtered out and used in the following steps. The only fields used without further processing are the submitter and doi (at the end, these are stored in their present form in up-to-date DB in the table "PUBLICATIONS"). The next list explains the steps that will be carried out to transform and enrich the existing data with relevant and essential information.

1. The field versions is used to get the dates of the first and last version and the number of versions in the arXiv. (At the end, these values are stored in up-to-date DB in the table "PUBLICATIONS").
2. To get the information about venues (name and abbreviation), the field journal-ref is used. The data in this field is cleaned and then checked if it matches with any venue in the lookup table venues\_lookup.tsv (by using the function find\_venue(venue\_data\_raw)). (At the end, these values are stored in up-to-date DB in the table "VENUES").
3. The field authors\_parsed is used for several purposes. First of all, this field contains information about the authors' names. Therefore, it is processed (by using functions check\_first\_name\_raw(first\_name\_raw\_to\_check) and

parse\_first\_name(first\_name\_raw\_to\_parse)) to get the last and first names together with the abbreviation of each author's first name. Sometimes, this field also contains some extra suffixes of the name, such as Jr or II *etc*., which are also stored. (At the end, these values are stored in up-to-date DB in the table "AUTHORS"). Secondly, this field can also include data about authors' affiliations. Thus, the function find\_insitution\_information(institution\_name\_raw) is called to find whether this raw data field matches any institution or location stored in the universities\_lookup.tsv or cities\_lookup.tsv. It is important to note that data about location/place after the transformation pipeline is always at the country level. (At the end, these values, institution name and place, are stored in up-to-date DB in the table "AFFILIATIONS").

1. The field categories is used to map each publication with arXiv categories. (At the end, arXiv categories are stored in up-to-date DB in the table "ARXIV\_CATEGORIES").
2. The field doi is used for getting additional information (its type, number of references, citations/number of citations, page numbers, and different attributes relevant to journal articles, such as an issue number) about the publication. This method for enrichment is considered only when the DOI of publication is in the file DOIs\_for\_enrichment.csv, as mentioned before. Two APIs are used to retrieve a piece of extra information: Crossref REST API1 and OpenCitations API2 (see the following examples).

|  |
| --- |
| *# Example of retrieving the publication type based on DOI by using the Crossref REST API*  crossref\_results = crossref\_commons.retrieval.get\_publication\_as\_json('10.1103/PhysRevA.75.043613')  print(crossref\_results['type'])  *# Output:*  journal-article  *# Example of retrieving the number of references based on DOI by using the Crossref REST API*  crossref\_results = crossref\_commons.retrieval.get\_publication\_as\_json('10.1103/PhysReVA.75.043613')  print(crossref\_results['reference-count'])  *# Output:*  23  *# Example of retrieving the number of citations based on DOI by using the OpenCitations API*  client = opencitingpy.client.Client()  open\_citation\_result = client.get\_citation\_count('10.1103/PhysReVA.75.043613')  print(open\_citation\_result)  *# Output:*  13 |

*By using OpenCitations API, also the authors of the publication are received. Names of these authors are used to get more information about them (for example, their real-life h-index or full names). For that purpose, a scholarly3, a module that allows retrieving author and publication information from Google Scholar (and function data\_by\_author(author)), is used.*

|  |
| --- |
| *# Example of retrieving the author's real-life h-index by using a scholarly module*  search\_query = scholarly.search\_author('S. Wichmann')  first\_author\_result = next(search\_query)  author = scholarly.fill(first\_author\_result)  print(author['hindex'])  *# Output:*  39 |

*However, since scholarly has a limited number of times to retrieve the data, this part of the pipeline's code is now commented in to prevent it from running. Therefore, the relevant lines in the function transform\_and\_enrich\_the\_data() should be commented out before running the pipeline if one wants to use scholarly.*

When all the relevant data is gathered and modified, the TSV files needed for populating the up-to-date DB with this data are written. (Additionally, the TSV file (citing\_pub.tsv), where the publication ID and the DOIs of the publications that cite this publication, are recorded. This file is needed for the graph database.)

After that, the data is loaded into the up-to-date DB. In the cases where it is necessary to check that only new data is loaded to ensure that there would be no duplicates in the DB, the additional temporary tables (such as "AFFILIATIOS\_TEMP" and "AUTHORS\_TEMP") are used. (These tables are always emptied before a new batch of data.) For example, the data about authors is bulk inserted into the "AUTHORS\_TEMP" table. Then the data of each author is compared with the data of each author in the "AUTHORS" table. If the author's information is not already present in the DB, a new author is inserted into the "AUTHORS" table. Otherwise, the data about the author is discarded.

In the last step of the transformation pipeline, the data in the database is copied and stored in CSV files. These files are used for loading the data into DWH and graph DB. Appropriate views are generated beforehand to get all the required information (like calculated h-indices of the venues and authors). Also, the field of study is normalised. For that, each arXiv category is mapped against the Scientific Disciplines classification table4. (This table is stored in suitable form in lookup\_table\_comains.csv.) For example, if the value in the arXiv category field is "cs.AI" after the mapping, besides this tag, there are three new tags: major\_field: "natural sciences", sub\_category: "computer sciences" and exact\_category: "artificial intelligence".

**Updating pipeline (1B)**

To fulfil the prerequisites of using the up-to-date DB approach, the data about publications in the database should be updated periodically. For that reason, the Airflow DAG update\_articles\_in\_DB was built (see figure 6).

Graphical user interface

Description automatically generated with medium confidence

**Figure 6** update\_articles\_in\_DB Airflow DAG

During the run of this DAG, the publications with DOIs, that are stored in DB are updated by using their DOIs and OpenCitations API. In this project, only the number of citations is considered as changing field. If the API call returns a new value for this variable, the data about publication is updated. Since venues' and authors' h-indices depend on the number of citations, the relevant views are refreshed after updates.

Similarly to the transformation pipeline, updating the pipeline ends with copying the data. However, at this time, only publications', authors' and venues' data is copied and saved as CSV files ready for the following pipeline parts.

**Part 2**

After thoroughly investigating the data to understand what parts of it are usable for the project, it is possible to phrase the BI queries that would be the basis for developing a data warehouse. In the subsequent sections, the BI queries, the schema of DWH and the technologies that will be used are discussed.

QUERIES

1. Getting authors (or ranking them)

* with the most publications in a given year, scientific domain and/or publication venue
* with the most citations in a given year, scientific domain and/or publication venue
* with the highest h-index in a given time period
* with the broadest horizon (authors who have written papers in the largest amount of different scientific domains)

1. Getting institutions (or ranking them)

* with the most publications in a given year, scientific domain and/or publication venue
* that have the highest impact in the scientific world (institutions that have papers which have been cited the most in a given year, scientific domain and/or publication venue)

1. Getting publications (or ranking them)

* with the most citations in a given year, scientific domain and/or publication venue

1. Getting journals (or ranking them)

* with the highest h-index in a given year and/or scientific domain

1. What are the year's hottest topics (categories of scientific disciplines)?
2. How does the number of publications on a given topic change during a given time frame (histograms of the number of publications on a given topic over a given period of time)?
3. Who is the author whose h-index has increased the most during the given time?
4. Which journal's h-index has increased the most during the given time?
5. Which papers have the most prolonged period between the first and last version? Are there any journals where publishing takes much more time compared to others?

SCHEMA

Based on the formulated BI queries, the proper schema of a data warehouse for storing data about scientific publications would contain a fact table, "PUBLICATIONS", and five dimension tables: "AUTHORS", "AUTHORS' AFFILIATIONS", "PUBLICATION VENUES", "SCIENTIFIC DOMAINS" and "TIME" (see Figure 2).

A picture containing graphical user interface

Description automatically generated

**Figure 2** Schema of DWH

The fact table "PUBLICATIONS" will store the primary keys of dimension tables (or dimension group keys in cases where bridge tables are used) as foreign keys together with additional information about the record (see Table 1 for more details).

**Table 1** Attributes of the fact table together with explanations

|  |  |
| --- | --- |
| **Attribute** | **Explanation** |
| authors\_group\_key | the "AUTHORS" dimension group key; is required to get information about the authors of the publication | |
| affiliations\_group\_key | the "AUTHORS' AFFILIATIONS" dimension group key; is required to get information about the affiliations of the authors of the publication | |
| venue\_ID | the primary key of the "PUBLICATION VENUES" dimension; is required to retrieve data about venues where the paper was published | |
| domain\_group\_key | the "SCIENTIFIC DOMAINS" dimension group key; is required to get information about the field of study | |
| time\_ID | the primary key of the "TIME" dimension; is required to query when the publication was published (time information about the last version in the arXiv dataset at the moment when data was added to the DWH) | |
| DOI | Digital Object Identifier (DOI) of the paper | |
| title | the title of the publication | |
| type | type ("journal article", "conference material", *etc.*) of the publication | |
| number\_of\_authors | number of authors | |
| submitter | the name of the person who submitted the paper/corresponding author | |
| language | the language of the publication | |
| volume | volume number; applicable when the paper is published in the journala | |
| issue | issue number; applicable when the paper is published in the journala | |
| page\_numbers | publication page numbers in the journal or the other published scientific papers collectiona | |
| number\_of\_pages | total number of pages of the publication | |
| number\_of\_references | number of publications that present publication cites | |
| no\_versions\_arXiv | number of versions of the current publication in the arXiv dataset; since the arXiv dataset is updated frequently, this field may change – a new version of the publication may be published | |
| date\_of\_first\_version | date when the first version (version v1 in arXiv) was created; is required for measuring the time interval between the first and current version of the publication | |
| number\_of\_citations | number of publications that cite the present publication; this field may change over time | |
| *The following attributes are added for historical tracking.* | | |
| is\_current\_snapshot | the flag to indicate if the row represents the current state of the fact; is updated when a new row is added | |
| snapshot\_valid\_from | the date this row became effective | |
| snapshot\_valid\_to | the date this row expired; is updated when a new row is added | |

aIt is important to note that not all additional information fields are applicable in all cases. For example, workshop materials do not have an issue number. In these situations, the field will be filled as not-applicable.

As one can notice, the timestamped accumulating snapshots concept5,6 is used for historical tracking of the data. This approach is suitable because the data will change infrequently. For example, the number of citations of one publication may increase very often during some period, but at the same time, there may be long time intervals during which this value remains the same.

In the dimension table "AUTHORS", all the relevant data about the publications' authors (name and h-index) will be stored. The difference between fields "h\_index\_real" and "h\_index\_calculated" is that the first h-index is retrieved by an API call and refers to the real-life h-index that the author has. The second h-index is calculated based on the data added to the DWH. The reason to keep both is that it is one way to immediately see if there is an error in the data pipeline – a calculated h-index could never be higher than a real-life one.

Since one author can have several publications and one publication can have several authors (many-to-many relationship), the bridge table7 will be used to connect the author's dimension with specific facts. Additionally, an author's h-index (both of them) is a variable that changes over time. For BI queries (for example, getting the author whose h-index increased the most during the last year), tracking that change is essential. Therefore, the type 2 slowly changing dimensions concept8,9 is used – when the author's h-index changes, a new dimension record is generated. At the same time, the old record will be assigned a non-active effective date, and the new record will be assigned an active effective date. The bridge table will also contain effective and expiration timestamps to avoid incorrect linkages between authors and publications10.

In the dimension table "AUTHORS' AFFILIATIONS", information about the institutions (name and location) of the authors of the publications will be gathered. Similarly to the authors' dimension, in this case, there could be a many-to-many relationship between the dimension and fact. In other words, there could be many publications from one institution, and authors of the same publication can have different affiliations. Therefore, the bridge table will be used to connect the dimension table records with the fact table records.

In this step, one may notice that there is no connection between the authors and the affiliations. (For example, in the authors' dimension table, there is no information about the author's affiliation.) The reasoning behind this decision is that BI queries (see the previous section) do not require author-level information about affiliations. Based on the current schema, it is possible to fulfil all the queries requiring affiliations information.

Dimension table "PUBLICATION VENUES" will store data about the venues of the publications. In this table, there could be many fields that do not apply to all the records. For example, if the type is "book", the field "h\_index\_calculated" is irrelevant. However, if the field h-index is applicable (for journals), similarly to the "AUTHORS" dimension table, tracking its changes is essential from the BI point of view. Therefore, this table will also use the type 2 slowly changing dimensions concept.

In the dimension table "SCIENTIFIC DOMAINS", the categories (in three levels besides the arXiv tag) of scientific disciplines of publications will be gathered. Again, the bridge table will be used to overcome the shortcomings related to many-to-many relationships between publications and scientific domains (one publication can belong to many scientific domains, and many publications can have the same domain).

The "TIME" dimension will hold all the relevant (from the BI point of view) time information about the publications. Besides the timestamp of the publication, it also has separate fields for year, month and day.

TECHNOLOGIES

**PostreSQL with Citus extension**

The Postgres with Citus extension is chosen as a data warehouse database.

**PostgreSQL**

PostgreSQL11 is an open-source object-relational database solution with a solid background, up-to-date documentation, popularity across top-tech companies and a strong community.

The database supports various data types, including JSON/JSONB; all the innovations come from Postgres extensions. It has different optimisation tools for analytical queries, such as indexes12 and table partitioning13.

**Citus**

However, the PostgresSQL build-in features deliver a powerful tool, but for better analytical query processing and potential database growth, the Citus14 extension would also be used in the solution. Citus is a Postgres extension that supports sharding, paralleling SQL across multiple nodes supported and owned by Microsoft15. It delivers insanely fast performance. Even with real-time data ingest and billions of rows of data. The idea of Citus extension was initially designed as an OLAP solution based on PostgreSQL. Moreover, the focus of development is changes, and today Citus does either: OLTP and HTAP16.

**ClickHouse benchmark (Citus vs MariaDB with ColumnStore)**

According to the ClickHouse benchmark, the main SQL competitor is MariaDB with ColumnStore extension. However, it does not support all SQL operations. The Citus outperforms the MariaDB in DB storage size and loading the data to the database but is less performant in cold and hot data retrieving. In the benchmark, the indexes optimisation was not used for Postgres solution, but with them, it would outperform the main competitor in many scenarios.

**Production implementation facts**

Postgres with Citus extension is used on Azure for a petabyte-scale analytical solution17.

**Part 3**

The labelled property graph model is used instead of RDF to design the graph database. It makes the graph look more concise and allows to specify properties next to nodes and edges.

QUERIES

The database is designed to answer queries about relationships between authors (co-authorship), authors and affiliations (employment), publications and scientific domains, and publications and venues. The following list is a sample list of queries that a user might be interested in:

* Getting an author
* who collaborates with a given author
* who collaborates with a given author in a given year
* who writes in a given scientific domain
* who writes in a given venue
* who writes for a given affiliation
* Getting a publication:
* cited by a given publication
* cited by a given author
* published in a given venue
* affiliated with a given affiliation
* from a given scientific domain
* Getting an affiliation
* that covers a given scientific domain
* publishes in a given publication venue
* employs a given author
* Getting a scientific domain
* that is covered by a given affiliation
* that is covered by a given publication venue
* that is covered by a given author
* Getting a publication venue
* that covers a given scientific domain
* that publishes for a given affiliation
* that publishes for a given author

Besides that, the schema supports more complex analytical questions:

* What is the most influential publication:
* in a given year?
* in a given scientific domain?
* in a given venue?
* in a given affiliation?
* What is a community of authors
* that covers a given scientific domain?
* that publishes in a given publication venue?
* that publishes for a given affiliation?
* Which author has the most self-citations (or citations to other authors from the same affiliation)?
* Which author has the most collaborations?
* Is there a connection between co-authors and where they publish their papers?
* What is the missing link between two authors from different affiliations who have not collaborated yet?

For the analytical questions, several graph algorithms are used. For example, the ArticleRank algorithm18 provided by the Neo4j Graph Data Science Library plugin is used to find the most influential publication or author. Communities can be detected by community detection algorithms, *e.g.*, Louvain19 or K-Means Clustering20. The path-finding algorithms are used to find a connection between two authors or missing link between them, A\*21 or Yen's Shortest Path22.

SCHEMA

The property graph diagram below shows entities of the database and their relationships. The entities are represented as nodes, the relationships are represented as directed edges, node properties are specified inside nodes, and edge properties are displayed as notes on a yellow background.

All entities contain properties relevant to the queries above. One of the edges, (:Author)-[:works\_at {date}]->(:Affiliation), also contains a property to indicate that the relationship is temporal, and that might be important for some queries. Nodes like (:Author), (:Affiliation), and (:Publication) can have self-loops to indicate co-authorship, employment, and self-citations, respectively.

Diagram

Description automatically generated

**Figure 3** Schema of graph database

In Table 2, the entities together with properties are given.

**Table 2** Entities and their properties that are used in the graph database

|  |  |
| --- | --- |
| **Entity** | **Properties** |
| Author | full\_name, h\_index | |
| Affiliation | name, place | |
| Publication | doi, title, year | |
| ScientificDomain | major\_field, sub\_category, exact\_category, arxiv\_category | |
| Venue | issn, name, h\_index | |

Relationships in the graph database:

* AUTHOR\_OF: (:Author)-[:AUTHOR\_OF]->(:Publication)
* COLLABORATES\_WITH: (:Author)-[:COLLABORATES\_WITH]->(:Author)
* WORKS\_AT: (:Author)-[:WORKS\_AT {date}]->(:Affiliation)
* PUBLISHED\_IN: (:Publication)-[:PUBLISHED\_IN]->(:Venue)
* BELONGS\_TO: (:Publication)-[:COVERS]->(:ScientificDomain)
* CITED\_BY: (:Publication)-[:CITED\_BY]->(:Publication)
* COVERS: (:Affiliation)-[:COVERS]->(:ScientificDomain)
* PUBLISHES\_IN: (:Affiliation)-[:PUBLISHES\_IN]->(:Venue)
* COLLABORATES\_WITH: (:Affiliation)-[:COLLABORATES\_WITH]->(:Affiliation)
* COVERED\_BY: (:ScientificDomain)-[:COVERED\_BY]->(:Venue)

TECHNOLOGIES

The Neo4j graph database engines with Cypher23 as the query language will be used to implement the graph model. Neo4j is an ACID-compliant transactional database widely used for graph data with native graph storage and processing. It supports the property graph model and has been widely used in the industry while being developed since 2007 by Neo4j, Inc24.

**Guidelines for running the overall pipeline**

...

**References**

1. Bartell, A. REST API. *Crossref* https://www.crossref.org/documentation/retrieve-metadata/rest-api/.

2. Peroni, S. & Shotton, D. Open Citation: Definition. 95436 Bytes (2018) doi:10.6084/M9.FIGSHARE.6683855.

3. Kannawadi, S. A. C., Panos Ipeirotis, Victor Silva, Arun. scholarly: Simple access to Google Scholar authors and citations https://pypi.org/project/scholarly/.

4. Scientific Disciplines - EGI Glossary - EGI Confluence. https://confluence.egi.eu/display/EGIG/Scientific+Disciplines.

5. Mundy, J. Design Tip #145 Timespan Accumulating Snapshot Fact Tables. *Kimball Group* https://www.kimballgroup.com/2012/05/design-tip-145-time-stamping-accumulating-snapshot-fact-tables/ (2012).

6. Timespan Tracking in Fact Tables | Kimball Dimensional Modeling Techniques. *Kimball Group* https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dimensional-modeling-techniques/timespan-fact-table/.

7. Thornthwaite, W. Design Tip #142 Building Bridges. *Kimball Group* https://www.kimballgroup.com/2012/02/design-tip-142-building-bridges/ (2012).

8. Kimball, R. Slowly Changing Dimensions. *Kimball Group* https://www.kimballgroup.com/2008/08/slowly-changing-dimensions/ (2008).

9. Kimball, R. Slowly Changing Dimensions, Part 2. *Kimball Group* https://www.kimballgroup.com/2008/09/slowly-changing-dimensions-part-2/ (2008).

10. Multivalued Dimensions and Bridge Tables | Kimball Dimensional Modeling Techniques. *Kimball Group* https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dimensional-modeling-techniques/multivalued-dimension-bridge-table/.

11. postgres/postgres. (2022) https://github.com/postgres/postgres.

12. PostgreSQL - INDEXES. https://www.tutorialspoint.com/postgresql/postgresql\_indexes.

13. 5.11. Table Partitioning. *PostgreSQL Documentation* https://www.postgresql.org/docs/15/ddl-partitioning.html (2022).

14. citusdata/citus. (2022) https://github.com/citusdata/citus.

15. Our Story | Citus Data, now part of Microsoft. https://www.citusdata.com/about/our-story/.

16. Hybrid Transaction/Analytical Processing Will Foster Opportunities for Dramatic Business Innovation. *Gartner* https://www.gartner.com/en/documents/2657815.

17. Architecting petabyte-scale analytics by scaling out Postgres on Azure with the Citus extension. *TECHCOMMUNITY.MICROSOFT.COM* https://techcommunity.microsoft.com/t5/azure-database-for-postgresql/architecting-petabyte-scale-analytics-by-scaling-out-postgres-on/ba-p/969685 (2019).

18. Article Rank - Neo4j Graph Data Science. *Neo4j Graph Data Platform* https://neo4j.com/docs/graph-data-science/2.2/algorithms/article-rank/.

19. Louvain - Neo4j Graph Data Science. *Neo4j Graph Data Platform* https://neo4j.com/docs/graph-data-science/2.2/algorithms/louvain/.

20. K-Means Clustering - Neo4j Graph Data Science. *Neo4j Graph Data Platform* https://neo4j.com/docs/graph-data-science/2.2/algorithms/alpha/kmeans/.

21. A\* Shortest Path - Neo4j Graph Data Science. *Neo4j Graph Data Platform* https://neo4j.com/docs/graph-data-science/2.2/algorithms/astar/.

22. Yen's algorithm Shortest Path - Neo4j Graph Data Science. *Neo4j Graph Data Platform* https://neo4j.com/docs/graph-data-science/2.2/algorithms/yens/.

23. Cypher Query Language - Developer Guides. *Neo4j Graph Data Platform* https://neo4j.com/developer/cypher/.

24. Neo4j Open Source Project. *Neo4j Graph Data Platform* https://neo4j.com/open-source-project/.