Enabling the Analysis of Computer Science Research via a Relational Database from DBLP

Raquel Berenguer

Director: Javier Cánovas

June 2024

Universitat Oberta de Catalunya

Creative Commons Rights

Bachelor Thesis	
Title	Enabling the Analysis of Computer Science Research via a Relational Database from DBLP
Author	Raquel Berenguer Mueller
Thesis Director	Jaiver Luís Cánovas Izquierdo
Date	June 2024
Bachelor Degree	Grau en Enginyeria Informàtica
Area	Intel·ligència artificial
Language	English
Keywords	DBLP · Scientometrics · Data Science
Repository	`https://github.com/SOM-Research/dblp-extractor`

Dedication/Quote

"*Only through understanding our past can we shape our future.*"
- Isaac Asimov (The Foundation Trilogy)

Acknowledgements

I would like to thank Dr. Javier Luis Cánovas Izquiero for his guidance throughout the whole process of this work and for being understanding and humane in the setbacks of life itself.

Also, thanks to the SOM-Research group, Dr. Robert Clarisó for facilitating some resources from the group, and Dr. Abel Gómez for their patience and help with the small issues with the server.

I would like to thank my husband for his support, help, and time.

Abstract

Scientometrics is a field aimed at measuring and analyzing scholarly literature. It helps researchers better understand the dynamics of research fields, and it provides a comprehensive approach to the quantitative features and characteristics of science and scientific research. Some existing approaches to performing scientometrics studies are DBLP, Lens.org, Faceted or Alexandria; being DBLP the most representative in the field of computer science. However, the current analysis research relies on outdated datasets and is mainly based on ad-hoc studies that lack semantic information (e.g., explicit links between papers and authors, or between authors and conferences). To address this situation, in this project, we propose an approach to creating a relational database to facilitate the analysis of research data in computer science.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Context

The field of Scientometrics aims to measure and analyze scholarly literature. It provides a comprehensive approach to the quantitative features and characteristics of science and scientific research, and it helps researchers better understand the dynamics of research fields.

Current approaches to performing scientometrics studies suffer from several issues, namely: (I1) They are ad-hoc studies based on existing datasets offering simple query features, such as analysis on a simplified subset of DBLP papers. (I2) The dataset upon which they are based is poorly maintained or abandoned (e.g., FACETED). And (I3) current datasets provide semantically poor data; that is, there are no relationships explicitly identified among the data elements.

This project addresses these issues by developing an extraction and analysis process that will make it easier to perform scientometrics studies, especially when using the DBLP dataset. In order to address I1 and I3, we intend to develop a relational database to facilitate the querying and analysis of those data. Furthermore, the extraction of relationships among elements, which helps to extract non-trivial information, is considered a task. Finally, an automatic process to keep the dataset up-to-date will be incorporated into the project, thus addressing I2.

The main starting resource for this project is the DBLP dataset. The Schloss Dagstuhl Leibniz Center for Informatics at the University of Trier maintains a huge XML with all the data they collect about computer science publications. The project name is DBLP[1], but there is no relational database that stores all the information. DBLP offers a simplified searcher for locating publications or researchers, but it is easy to get a lot of noise and unrelated items when making complicated searches.

Upon searching for alternative compilations of papers, some results would be located; however, certain challenges would be confronted. Other existing resources concerning datasets include Faceted, Alexandria3k, and Lens.org. Faceted [2] is currently abandoned, and the data is out of date. Alexandria3k [3] which content is from *Crossref* [4], Academic Torrents [5] and Aria2 [6] is not focus on specific Computer Science papers like DBLP. Lens[7] is a new tool that is patent-oriented and contains scholarly literature metadata. Similar to Aria 2, it is not restricted to Computer Science papers.

XML markdown has both the convenience and inconvenience of creating a custom tree of elements and their attributes. That means that, at any time, new elements of the tree could be created. Furthermore, it implies that there is no data validation. It is expected that dealing with a large dataset, which is stored in XML, will present a challenge. Some obstacles may include redundant data or data that is stored in different elements. Moreover, it could cause difficulties in creating relationships between models. In order to update the database, a new system will be developed. Finally, it will yield clean and workable data from the DBLP.

When the database is ready, some metrics will be displayed to illustrate their content. Moreover, some complex analyses as examples of usage would be found. For instance, a potential metric could

[1] `https://dblp.uni-trier.de/`
[2] `https://dblp.l3s.de/?q=&newQuery=yes&resTableName=query_resultwsZCVP`
[3] `https://dspinellis.github.io/alexandria3k/schema.html`
[4] `https://www.crossref.org/`
[5] `https://academictorrents.com/`
[6] `https://aria2.github.io/`
[7] `https://www.lens.org/`

involve identifying the typical path new researchers take to publish their article or the main keywords for papers in conferences and journals. This project can be put into practice by applying the metrics defined before to empirically study the current situation in specific research communities.

1.2 Goals

The development of this project will involve three main goals:

G1 Development of a relational database for computer science research. This goal implies getting clean and workable data from DBLP. It is a challenge dealing with a huge dataset like DBLP. Thus, the larger the dataset, the harder it is to maintain the same structure or easier it is to get redundancies. A system to keep the dataset updated will also be developed.

G2 Definition of relevant metrics. A set of metrics will be defined to analyze several scenarios in research publication data. For instance, a possible metric could involve identifying the best path new researchers should take to publish their article in a relevant conference or journal or the best features a paper should meet to be published.

G3 Perform empirical studies To put into practice the project approach, it is interesting to apply the metrics defined in G2 to empirically study the current situation in specific research communities.

1.3 Sustainability, diversity, and ethical/social challenges

The impact of the project would be evaluated in several areas, including sustainability, diversity, and ethical or social changes. As a software project, certain fields have a limited impact based on their application.

Sustainability The impact of this project is minimal beyond its consumption. It is true that some processes are slow, depending on the capacity of the machine, which could be running for several days, causing a greater environmental impact.

Ethical behaviour and social responsibility Working with information regarding their work from others can frequently have an impact on their employment. This project aims to assist researchers who are interested in studying scientometrics. Moreover, the outcomes of those studies may have an impact: identifying the optimal course of action for a novice researcher and obtaining a list of the most pertinent topics or relevant conferences/journals are instances of the potential transformative power in the perception of the academic community.

Diversity, gender and human rights This project has the potential to have no impact on gender, diversity, or human rights on its own. However, it may prove beneficial for future research on diversity within the academic realm. As a first step, DBLP does not provide any information about the gender of researchers, but cross-referencing with other databases or doing a small task like scraping could obtain this information.

1.4 Approach and methodology

The objective of this project is to construct a database and populate it with data pertaining to studies and researchers. This information originates from the DBLP project. Once the database is ready to use, it needs a view of their data in order to find some studies to perform.

The first step is to create the database. To achieve this, a thorough analysis of the data should be performed. From this, it is expected that analysis will be capable of identifying the models and their associated relationships, which will be presented in a UML class diagram.

It could be difficult to insert data from an XML file into a database. The selection of the optimal tool for preprocessing substantial XML requires thorough investigation. A simple way to achieve this is with language programming, and a test with Python was executed. It takes three days to insert only a third of the data in that test. Therefore, it was necessary to find an alternative solution. Upon conducting several tests in AWK, it was determined that the entire procedure was unfeasible. Although AWK was theoretically capable of handling all SQL inserts, implementing it proved to be a daunting task. However, AWK proved to be effective for preprocessing. With simple searches by regex, copy, and replace commands, it was able to accomplish that task. The PostgreSQL XPath tool was finally discovered as a solution. This tool was capable of searching for any data within an XML tree. With the aid of AWK for the preprocessing and SQL scripts for the insertion, the set-up process can be successfully completed.

In order to maintain the database up-to-date, a process of increasing the amount of data should be followed. To do this process, the strategy should be similar to the set-up process. The AWK tool would do the part about preprocessing for preprocessing. SQL scripts would do the insertion using XPath PostgreSQL functions.

A set of metrics should be created to understand the database's size. Furthermore, the metrics should reveal relationships and some aggregated information. Ultimately, from these metrics, we were able to extract the outcomes of two studies. These studies should be defined after the description metrics are clear in order to know what kind of studies they could perform.

1.5 Schedule

In the project, sixteen tasks and seven milestones have been planned to reach. All together are divided into six groups named *PAC*s: from the *PAC0* to *PAC5*. The next sections describe the task and milestones, as well as the timing.

Tasks

Groups of tasks are based on the deliveries of the Bachelor Thesis have to be presented. Thus, there are groups with only one task (*PAC1*) and two groups unified (*PAC2 & PAC3*) The tasks are listed below, divided by group of tasks. All tasks are shown in a Gantt Chart in the Timing subsection (cf. 1.5).

- *PAC0*: Project description
  - Task1: Information gathering and resources selection.
  - Task2: Write project definition.
- *PAC1*: Project planning

  - Task3: Project planning
- *PAC2 & PAC3*: Project development
  - Task4: Evaluate and study the dataset.
  - Task5: Insert researchers and institutions.
  - Task6: Insert publication groups and venues
  - Task7: Insert publications
  - Task8: Data evaluation and data cleaning
  - Task9: Data visualization and metric computation
- *PAC4*: Bachelor Thesis writing
  - Task10: Review word done and restructure bachelor thesis
  - Task11: Write conclusions and future work
  - Task12:Proofread thesis
- *PAC5*: Bachelor Thesis defense
  - Task13: Prepare and record the presentation of the work
  - Task14: Defend and answer committee questions

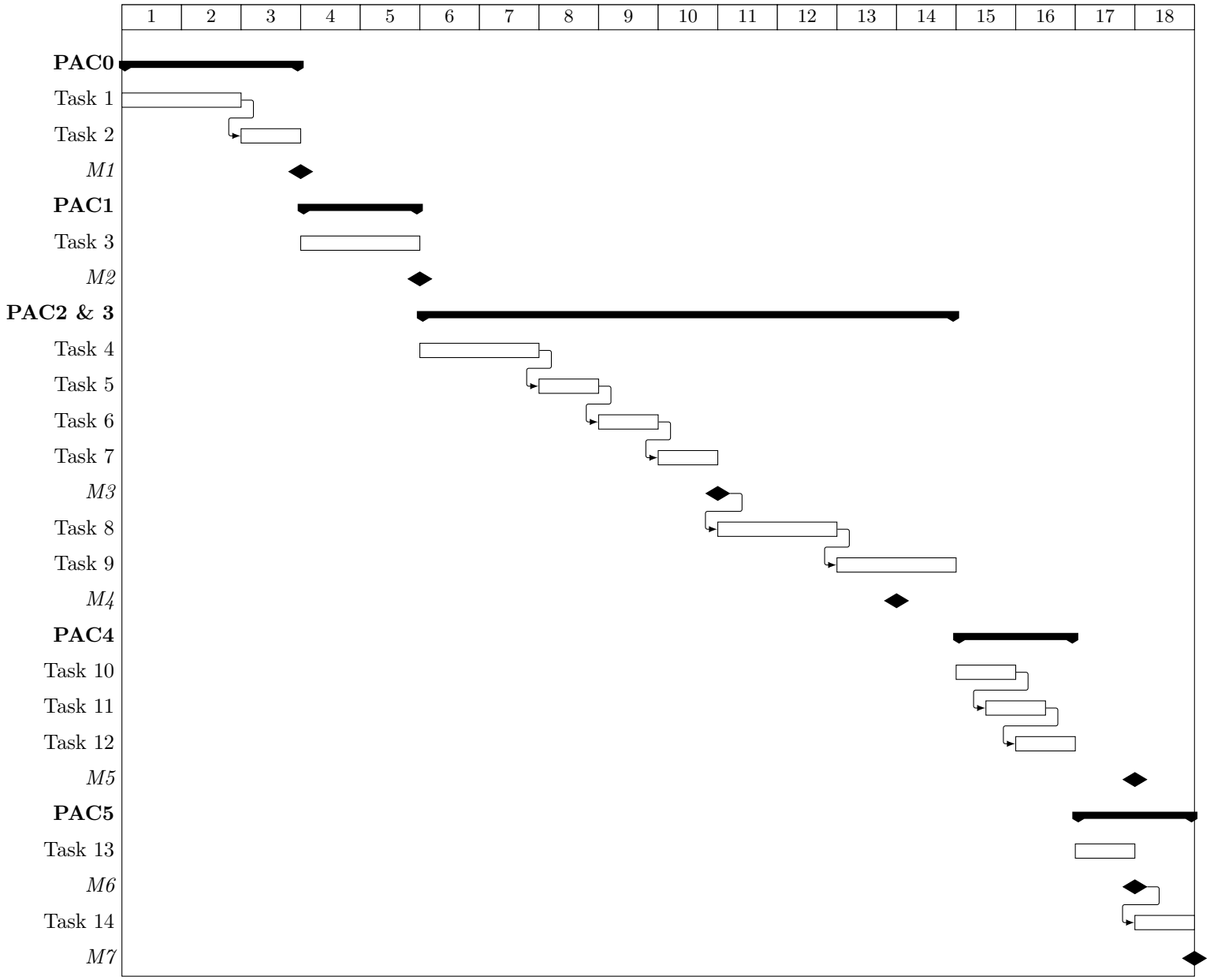
Milestones

To monitor the advance of the project, at the end of each group of tasks there is a milestone. Additionally, there is an intermediate milestone for PAC5. Each milestone corresponds to the delivery of each PAC. The milestones are listed in the following:

- Milestone 1 (*M1*) should be reached at the start of the week three. It should contain the basics to start the project: title, topics, goals, resources, etc.
- Milestone 2 (*M2*) should be reached at the middle of the week five. The content currently will cover the working plan and task definition.
- Milestone 3 (*M3*) should be reached at the middle of the week ten. This milestone represents the half of the project. At this time, the database will be fulfilled and the main models to work will be chosen.
- Milestone 4 (*M4*) should be reached at the middle of week fourteen. The end of the project is delivered in this milestone, when some relations between model elements will be expected to be found.
- Milestone 5 (*M5*) should be reached at the end of the week sixteen. The next three weeks are devoted to writing the Bachelor's thesis document.
- Milestone 6 (*M6*) should be reached at the end of the week seventeen. In this milestone, the first part of the thesis defense: the thesis video presentation, will be expected to be delivered.
- Milestone 7 (*M7*) should be reached at the end of the week eighteen. The final milestone is the second part of the thesis defense: answer the committee questions.

Timing

To determine the time spent on each task, a Gantt chart is created (1.5). The timing is in weeks, representing 1 the first week of the project and 18 the last week. Thus, week 1 is week number 9 in the present year, and week 18 is week 26.


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
PAC0
Task 1
Task 2
M1
PAC1
Task 3
M2
PAC2 & 3
Task 4
Task 5
Task 6
Task 7
M3
Task 8
Task 9
M4
PAC4
Task 10
Task 11
Task 12
M5
PAC5
Task 13
M6
Task 14
M7

1.6 Results summary

The project was stored in the GitHub repository as a public project named dblp-extractor[8].

1.6.1 Set-up process

Since the objective of the project is to create a relational database capable of performing simple and complex queries, a cross-sectional exploratory study was performed to determine which technology to use. The aim of this study is to find the best tool to create a database and process the data to insert it into. PostgreSQL was chosen because it is an open-source SQL database capable of handling a large volume of data. After conducting several tests, it was determined that AWK was the most effective method for preprocessing the dataset. For the insert into PostgreSQL, we used SQL scripting for simplicity and efficiency. It would be employed within a virtual machine to gain control over the environment in which the database runs. Finally, in order to make it easy to replicate the setup process, a couple of bash scripts were written to include all the steps. One is intended for preprocessing the data, which includes the AWK scripts to split and parse the data. The other is intended for insertion into the database and contains only SQL scripts. To summarize, the project can be settled within a day. It was tested on an external Debian machine provided by the SOM Research[9] group.

1.6.2 Auto-increment process

Every month, the DBLP organization uploads a new XML file with all the data. Some of the data is new, other entries have been modified, and others are the same as they were a month ago. In order to maintain the database, an automatic increment process would be required. To accomplish that task, it would be necessary to identify the newly acquired and modified data. A previous examination of the dataset is essential. Resulting in two datasets: old data and new data. After that, the new data should be preprocessed similarly to the setup process. With preprocessed data, SQL scripts should insert new data and modify the old ones that suffered modifications. Similar to the setup process, a couple of scripts encapsulates all the necessary steps. One is for preprocessing, and the other is for inserting and updating the data.

1.6.3 Metrics and usage

When the database is complete and everything is into it, it could generate some metrics for the data to provide an explanation for the database. A visual representation depicts the extent of the database and aids in comprehending its magnitude. To provide a visual aid, it would be helpful to have a table with the number of items for the most important tables. To illustrate the most important relationship, some plots would be generated. Other plots would be generated with some more combined data in order to show some important key metrics. All of these metrics and statistics will aid in comprehending the data and its intricate connections.

Additionally, there would be two complex metrics as examples of usage. Those metrics are aimed at showing non-trivial information. One of the initial intricate metrics would be the examination of the most frequent words in the titles of publications published within a particular group of publications. The topic trend of those publications should be represented in this study. The study

[8] `https://github.com/SOM-Research/dblp-extractor`
[9] `https://som-research.uoc.edu/`

determines the researcher's initial publications based on the types of publications. This implies that the investigation should provide a comprehensive overview of the publication type's itinerary. The addition of filters to display diverse paths, analyze them, and perform comparisons could prove to be a valuable exercise in data compression.

1.7 Brief description of the remaining chapters of the report

This report would also include a section dedicated to the resources and working methods 2, a section explaining the results 3 and a conclusion of the work done 4. A small list of vocabulary is provided at the end of the project, as well as the references used in it. In the Annex, all the figures used in the project will be found in a larger size to help readers read all the details if they need them.

1.7.1 Section of Resources and Working Methods

The section on resources and working methods comprises of four subsections, namely Resources, Working Methods, Problems Found, and Solutions and Alternatives.

The first subsection describes the resources required for the project. Additionally, there is a profound explanation of the architecture and the technologies used. And how the decision was proceeded.

The working methods section provides detailed information about the process of creating the project. Initially, a description of the data present in the dataset is provided. It then proposes a UML class diagram to illustrate the models and their relationships. The process of inserting the data into the database is detailed, accompanied by a description of the scripts employed and a diagram to provide a more comprehensive understanding of the process. Additionally, it explains what data was inserted. Similar to the process of inserting data, there are scripts for the process of incrementing data, which explain which scripts are used. In order to better understand the flow of increment data, a descriptive image is provided. The last point is aimed at describing metrics and statistics, explaining them. Furthermore, it briefly explains the tools used to obtain those metrics.

The subsection on problems found compiled all major issues found during the project execution. Receives a comprehensive description of the issue and its rationale. Furthermore, some examples are provided in order to better understand those issues.

The last subsection is intended to provide solutions to the problems presented in the previous section. Some of them are briefly explained in the Problems subsection, and this section provides a deeper understanding of the solution taken and the results of those decisions.

1.7.2 Section of Results

The section on results is divided into three subsections: Model, system deployment, and autoincrementality, as well as metrics and advanced metrics.

The first subsection describes the models created in the project, what they are, and how they relate to each other. There is a section that explains the system deployment result and how to execute it. Additionally, the result of this execution is explained. The final autoincrement process is also explained here. The points are similar to those in the chapter on Resources and Working Methods, with the difference that there is a summary of the most important points of the process.

The objective of the metrics subsection is to describe the database through metrics. The results of the most important relationships were provided in a set of plots with their explanations. There is

a table that summarizes the database according to the number of elements inserted for each table. To better understand the relationships between models, plots are provided.

Ultimately, the final subsection was targeted towards the Advanced Metrics. These metrics are expected to display non-trivial information from the database. This subsection was divided into two subsubsections, one for each advanced metric: *The most common words from related conferences* and *Journey of a researcher started in 2000 or after*. The first subsubsection was a small study to determine what the most frequent words were in the titles of some publications. In order to gain a better understanding, the study was applied to obtain some metrics divided into temporal windows and compare them. The other intricate metric was a study of a researcher's initial publications' trajectory. To accomplish this task, the researchers relied on the types of publications and determined that the first publication occurred in 2000 or later. By utilizing the publication list mentioned in the initial complex metric as a filter, an additional set of results were obtained, and a comprehensive comparison was conducted.

1.7.3 Section of Conclusion and Future Work

The section on conclusion and future work has two sections with the same name: one is 'Conclusion' and the other is 'Future Work.' As the name of the subsection suggests, this section of the report aims to summarize all the projects with some conclusions. In the last paragraph, it expresses some future work to continue with the project. Examples of other studies and new techniques that could be applied were included.

2 Resources and working methods

2.1 Resources

The DBLP is a comprehensive compilation of academic publications, primarily related to the field of computer science, that is regularly updated, published, and maintained by the Schloss Dagstuhl Leibniz Center for Informatics. It is published on a website[10] that provides a search service that facilitates straightforward queries, such as locating all publications from an author or the total number of publications for an author. However, it does have limitations concerning more intricate queries, such as locating publications from an author as the first author. This website also offers the option to download all the data in a data file in the form of XML, a markup language model.

The XML file defines one-way relationships among data, such as a list of publications of an author, title, year of publication, coauthors, or ISBN in the case of printed publications. However, it lacks the majority of backwards relationships. To obtain digested information, it is necessary to read the whole data file several times, which is a difficult and time-consuming task.

In order to address this situation, in this project, we define an approach to processing the whole XML file and importing it into a relational database, adding the necessary relationships. Figure 1 shows the architecture of the system as a summary.

When choosing a database, there are three main characteristics to consider: it should be a relational database, it should be open-source, and it should be capable of managing large amounts of data. With that in mind, PostgreSQL was chosen instead of MariaDB because it was prepared to manage large amounts of data. PostgreSQL also provides a variety of tools and aids for data in different formats, including XML.

[10] `https://dblp.org`

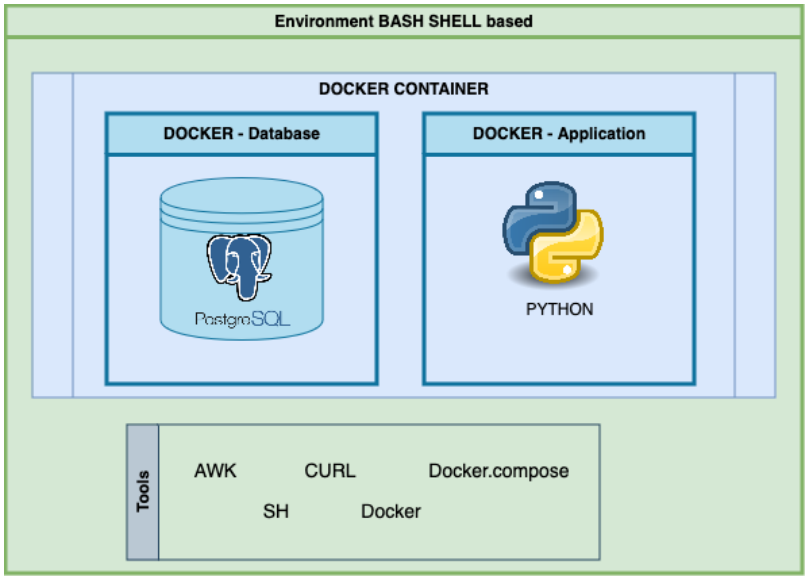


Figure 1: Simple sketch of the system architecture

The project's installation and local management costs have been minimized by selecting a virtualization tool, where we can define all the requisites in one simple file and then leverage all the work to it: download all the tooling, programming languages, code, and sources for the project.

Docker was selected as a tool for virtualization, and Docker Compose manages all the services and images. We have defined two machines, one for the database named db (PostgreSQL) as a service and another for the application itself named app (Python). Additionally, exists a Docker image of PostgreSQL maintained by the PostgreSQL community[11]. This makes the project easier to maintain.

As a prerequisite, everything should be in a bash shell environment. Afterward, tools such as AWK, CURL, and SH could be utilized. AWK was selected to assume the responsibility of preprocessing the data. Initially, parse the data into a UTF-8 format and then insert a blank between the items to facilitate the identification of each item. Finally, AWK splits the file into smaller XML files, which are processed by the SQL scripts. CURL was necessary to download the XML automatically, and SH to execute compile commands, both for the set-up process and the increment-data process.

The second Docker image was of a small Alpine machine with Python 3 on it. Python was chosen as a programming language to resolve the issue with generating metrics. It has numerous libraries to process data and generate plots, which helps to accomplish the task. If the scenario changes and another programming language is required, then it would suffice to change the requirements of this second Docker, keeping the database intact.

[11] `https://github.com/docker-library/postgres`

2.2 Working methods

2.2.1 Analysis and XML structure

To ensure an efficient importation process, we first analyzed in depth the XML file provided by DBLP, which is 4 GB large and contains more than 10 million items. In particular, we studied the model elements, attributes, references, exceptions, and the improvements and changes the file suffered over time. To achieve this aim, we relied on the official documentation provided by DBLP 6, which explains the structure and some of the challenges of maintaining this data. In the documentation, authors explain how they treated the names of researchers, which sometimes can lead to problems. For instance, the same author can be referenced with different names, because of marriages, divorces, or just because their sign has changed over time. The duplication of the most common names and surnames is others of the problems they encountered in the data.

Taking an evaluation of the dblp-2024-01-04 XML data version, we can found 10644596 items classified in nine XML classes. Those classes are grouped into different models on the database, the conversion XML ↔ database used is listed below:

- "article": Publications (3445001 items)
- "book": Publications (17949 items)and PublicationGroups (2380 items), (total: 20443 items)
- "data": Publications (4739 items)
- "proceedings": PublicationGroups (57791 items)
- "inproceedings": Publications (3439269 items)
- "www": Researchers (3477424 items)
- "mastersthesis": Publications (23 items)
- "incollection": Publications (70314 items)
- "phdthesis": Publications (129592 items)

All items have at least two attributes: *mdate* and *key*, and one element: *title* by definition. *Mdate* is the last date the item was modified, and *key* is the unique reference to that item (i.e., it may serve as a primary key in a relational database). *Title* is used mostly as the title of a publication; aside from researchers, (*www* items) have all the same content: "Home Page". The *title* in those it there for a historical reason, nowadays, they are not used.

Every class item is defined by its own subset of attributes and elements, and some of them are optional. For each class item, we can find:

- *article*
  - Attributes: mdate, key, publtype, cdate
  - Elements: title, author, pages, year, journal, number, ee, url, volume, crossref, note, cdrom, editor, cite, tt, booktitle, publnr, month, publisher
- *book*
  - Attributes: mdate, key, publtype

  - Elements: title, author, school, year, publisher, series, Volume, isbn, ee, pages, note, editor, booktitle, url, crossref, month, cite, cdrom

- *data*
  - Attributes: mdate, key
  - Elements: author, title, year, publisher, number, rel, ee, crossref, month
- proceedings
  - Attributes: mdate, key, publtype
  - Elements: author, title, year, booktitle, url, volume, ee, pages, number, note, editor, cite, school, isbn, series, publisher, address, journal
- inproceedings
  - Attributes: mdate, key, publtype
  - Elements: author, title, booktitle, year, url, volume, crossref, ee, pages, number, note, cdrom, editor, cite, tt
- www
  - Attributes: mdate, key, publtype
  - Elements: author, title, url, note, crossref, cite, ee, year, editor
- mastersthesis
  - Attributes: mdate, key
  - Elements: author, title, year, school, note, ee
- incollection
  - Attributes: mdate, key, publtype
  - Elements: author, title, booktitle, year, url, crossref, ee, pages, number, sup, note, cdrom, cite, chapter, publisher
- phdthesis
  - Attributes: mdate, key, publtype
  - Elements: author, title, year, school, publisher, number, note, ee, pages, volume, isbn, month, series

However, that does not mean all items have the same attributes or elements; some of them could not be used. In the worst-case scenario, the same named element could be used for different purposes. The booktitle element could be used to represent the title of a book in a book item, but in a proceeding, it could be used to save the acronym of the conference as shown in Listing 1.

Some of the elements *Author* or *Editor*, which are references to *www* items, have an *orcid* attribute. This attribute is a distinctive identification number within the academic research community.

Listing 1: book item with school element

```
<book mdate="2017-05-16" key="reference/vision/2014">
<editor>Katsushi Ikeuchi</editor>
<title>Computer Vision, A Reference Guide</title>
<publisher>Springer</publisher>
<year>2014</year>
<isbn>978-0-387-30771-8</isbn>
<isbn>978-0-387-31439-6</isbn>
<ee>https://doi.org/10.1007/978-0-387-31439-6</ee>
<booktitle>Computer Vision, A Reference Guide</booktitle>
<url>db/reference/vision/vision2014.html</url>
</book>

<proceedings mdate="2019-05-14" key="conf/uml/2005">
<editor>Lionel C. Briand</editor>
<editor>Clay Williams</editor>
<title>Model Driven Engineering Languages and Systems, 8th International Conference,
MoDELS 2005, Montego Bay, Jamaica, October 2-7, 2005, Proceedings</title>
<volume>3713</volume>
<year>2005</year>
<isbn>3-540-29010-9</isbn>
<booktitle>MoDELS</booktitle>
<series href="db/series/lncs/index.html">Lecture Notes in Computer Science</series>
<publisher>Springer</publisher>
<ee>https://doi.org/10.1007/11557432</ee>
<url>db/conf/uml/models2005.html</url>
</proceedings>
```

2.2.2 UML first approach

To visualize the first approach for a relational database structure, we created a UML diagram, which is shown in Figure 2.

As can be seen, there are seven tables: Publications, Authorships, Awards, Researchers, Institutions, PulicationGroups and PublicationVenues, and two enumerations: PublicationType and VenueType. The *Awards* table and *Institution* and *Publication Group* relationships are shown in gray due to being later removed. The table of *Awards* was removed because the award information was related to *Researchers* instead of *Publications*. That makes the information about awards less interesting to analyze; furthermore, only some awards had the year informed, making it impossible to infer which publication was awarded. On the other hand, the relationship between *Institution* and *PublicationGroup* was removed because there is only one with that information, and it adds extra unnecessary complexity to the database. That information is shown by *school* element. The element *School* could be found in items with the classes *book*, *proceedings*. Listing 2 shows an example.

The information about the relationship between *Researcher* and *Institution* can be found in *www* item, an example is shown in Listing 3.

The element *note* with the attribute *affiliation* indicates the relationship with the *Institution*,

which is the content of the element, in this case "Hong Kong Polytechnic University, Department of Applied Mathematics, Hong Kong".

Listing 2: book item with school element

```
<book mdate="2019-06-19" key="books/daglib/0018643">
<author>Bernhard Hollunder</author>
<title>Subsumption algorithms for some attributive concept description languages.</title>
<pages>1-90</pages>
<school>University of Kaiserslautern, Germany</school>
<year>1989</year>
<series>SEKI Report</series>
<volume>89-16</volume>
</book>
```

Listing 3: www item with affilied to an institution

```
<www mdate="2020-11-11" key="homepages/78/2022-50">
<author>Kai Wang 0050</author>
<title>Home Page</title>
<note type="affiliation">Hong Kong Polytechnic University, Department of Applied
Mathematics, Hong Kong</note>
<url>https://orcid.org/0000-0002-9219-875X</url>
</www>
```

Researcher, Publication, and Publication Group are the most significant models that can be readily identified due to their data being derived from XML itself: xml_key, xml_mdate and xml_item.

Because an XML could have new elements, or some of them could become more important over time, the entire item is saved in a column (xml_item). Furthermore, they have the potential to serve as future references or to verify the original content to validate the object itself. The xml_key works as an identifier inside the XML file, and here it is saved for a search use. Some references between models were made using this key. The mdate attribute is a mandatory attribute for every item within the XML file. It displays the last modification date for the item, and its significance is due to the increasing process.

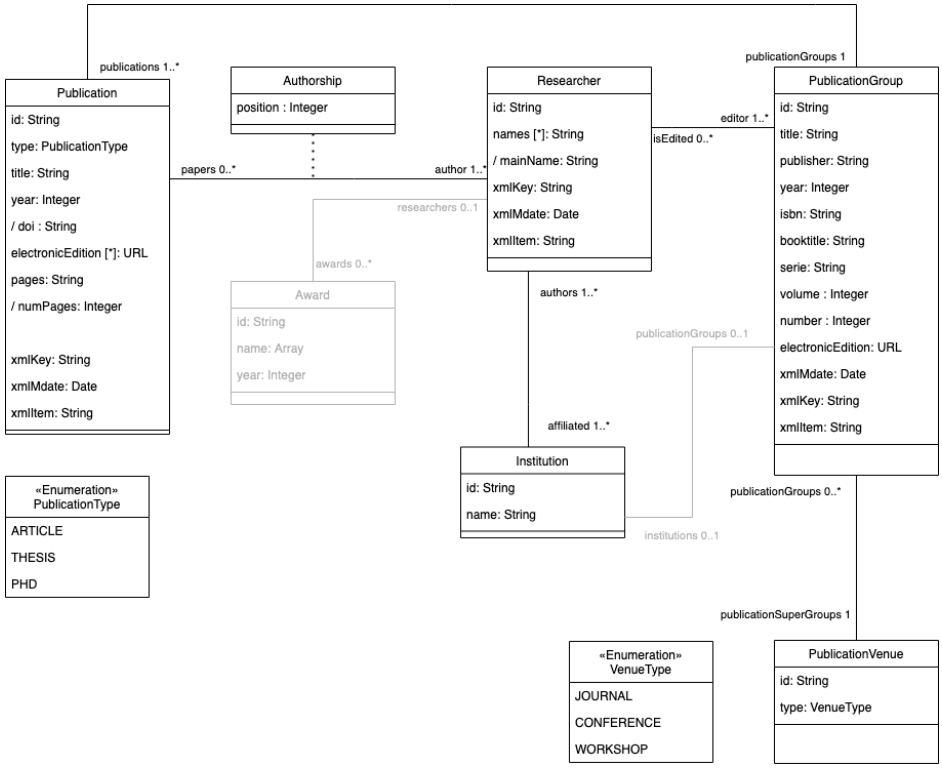


Figure 2: First Approach UML

2.2.3 Process to insert XML items to Database

The process of inserting items from the XML into the database is a comprehensive undertaking. Foregot this task, was necessary to work iterativelly becasuse when a error ocurres, whole process stops and needs to start again. The major errors are listed in the Section 2.3

In order to keep the process more clean and easier to work with, the XML was divided into nine other XML files by sub-items: article, book, data, proceedings, inproceedings, www, mastersthesis, incolletions, and phdthesis. Three of these split XMLs, namely www, article, and inproceedings, have been split again due to their excessive size, thereby limiting their processing to a maximum of 500,000 items per file. Figure 3 depicts the division of the XML into a total of 32 files, of which 28 would be utilized to insert the data.

Following the development of the database model, a database schema was incorporated as a DDL SQL script. Five SQL scripts were developed for inserting models: Three of them were dedicated to the import of researchers, publication groups and publications, respectively, as shown in Figure 4. Another script was devoted to relationships, while the final one was developed to calculate the number of pages of publications. The aforementioned script utilizes the XPath function of PostgreSQL to access the XML files.

Figure 4 shows how each XML file is added to a database table using the SQL scripts described in Figure 3.

A Bash script named set-up.sh was written in order to automate this part. This script downloads the DBLP XML from their repository, preprocesses it, and removes the files used in this process. The preprocessing is done with the AWK tool, which will parse the file from ISO-8859-1 to UTF-8, add blank spaces between items to simplify the split process, and split the file into nine files. As shown in Figure 3, some of the files will then be split again. The files that require further division are also divided using AWK. Finally, files that are no longer required are removed to save space in the machine. The entire procedure is illustrated in Figure 5.

The database schema underwent minor alterations to fix errors and add support information, which was sometimes redundant, to help with querying tasks. The column xml_tag was added to the researchers, publications and publication_groups tables. This may be redundant, but it helps to filter some queries. Additionally, names were added to the researchers table in order to have an array of names for each researcher inside the same object. As well as authors and editors in publications and publication_groups respectively, into obtain the array of researchers who participate in those items. The publication also featured new columns, namely journals, enabling the querying of publications by journal, and the pair of is_corrigendum, corrigendum_of, enabling the correlation between a corrigendum and its publication. The attribute of position was added to editors, which is the relationship between researchers and publication_groups. The enumeration of publication type was changed to journal, thesis, conference, workshop, book, and unknown. The last one was for publications that could not be identified.

The project[12] should be easy to replicate in any environment compatible with Docker.

Figure 3: Representation of the split process

[12] https://github.com/SOM-Research/dblp-extractor

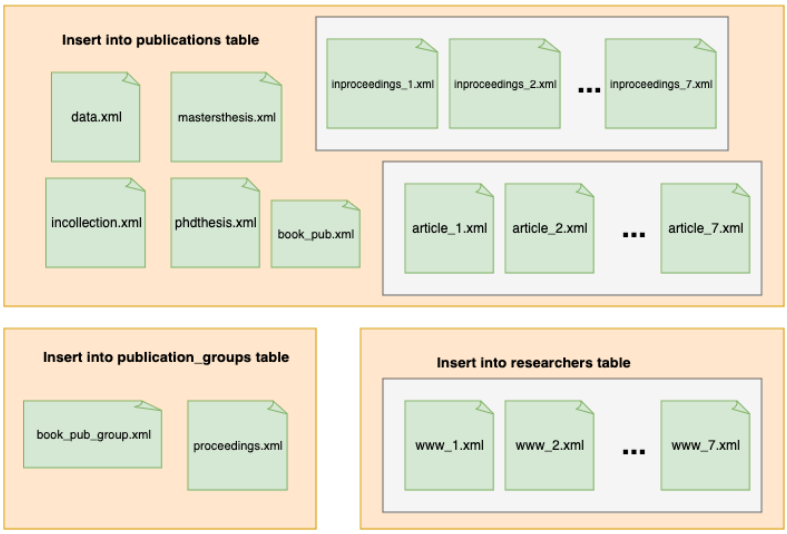


Figure 4: Group of XML files by which table was inserted their data

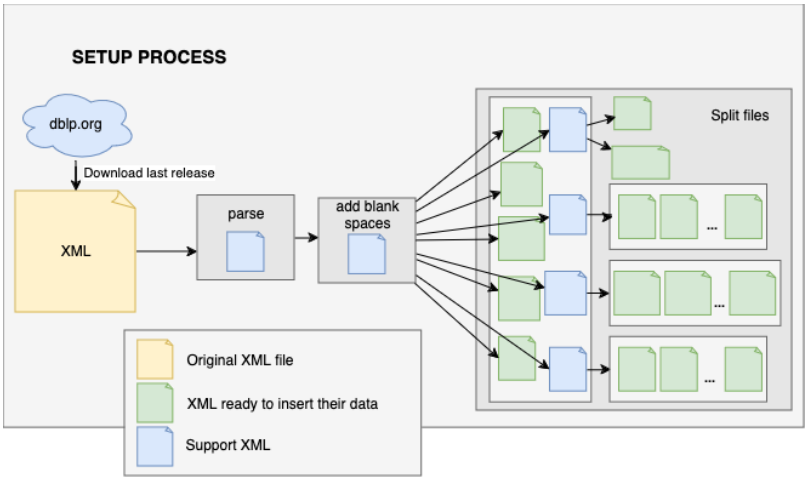


Figure 5: Schema of the set-up process.

2.2.4 Process to maintain data up to date and auto-incrementability

Another of the major problems that this project has the aim of solving is keeping the dataset up to-date. Each month, the DBLP organization uploads a new XML with all the data, that means there is more data each month or some of the data already exists suffered some modifications. As is said in Section 2.2.1, all XML items have an attribute named *mdate* and with this data, it is possible to filter all items subject to update or insert in the database.

Figure 6 would help to understand this process. After an update of tests between the XML from March and April, a total of 99714 was filtered as new or modified items. Of this total, 39087 are modifications of items that already exist.

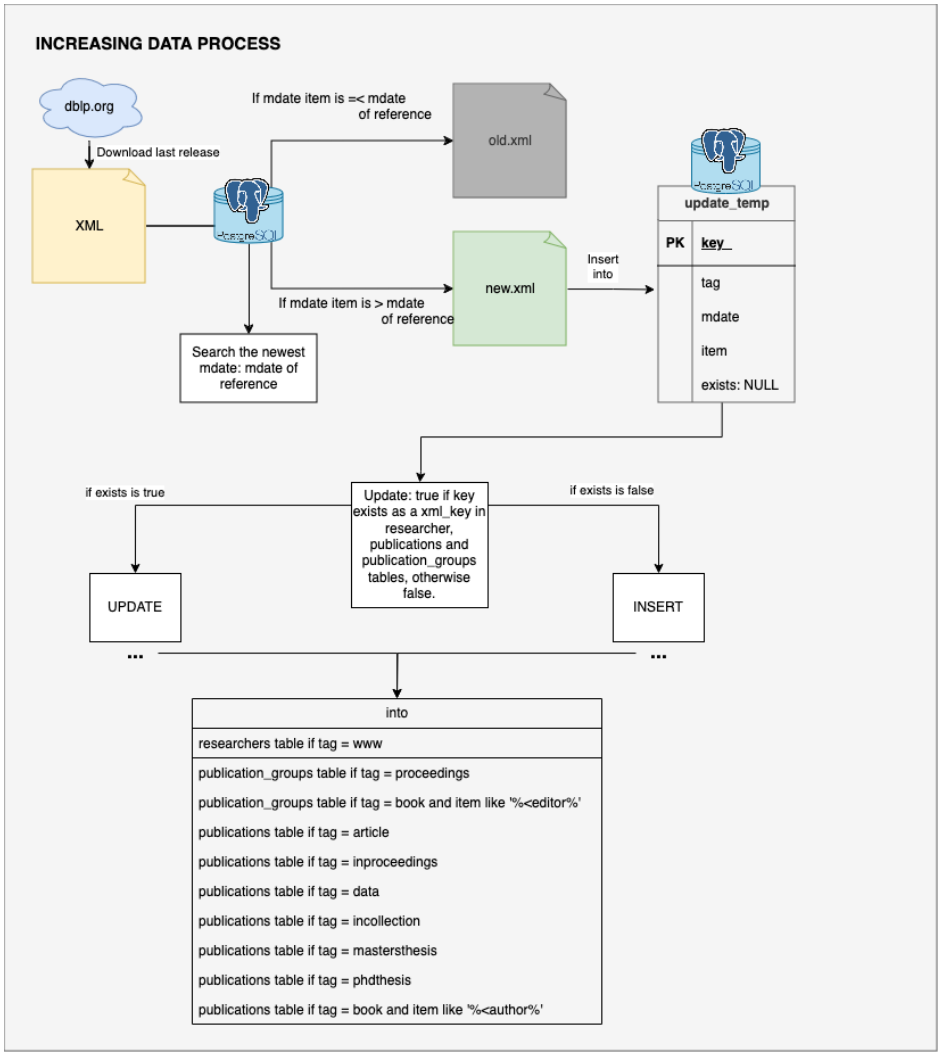


Figure 6: Schema of the increasing data process

2.2.5 Getting the metrics

The basic metrics were taken by querying directly to the database. Those metrics would be descriptive, and getting a summary of how many items has each table should be enough. But to get the box plots or a more complex plots, like the path for the researchers by publications or the most common words in the titles; it was necessary to resort to a programming language: Python was selected for this job because has many libraries to acomplish the task.

SQLAlchemy[13] ORM was selected with the aim of helping the application access the database through Python. *Matplotlib*[14] was the library used to print most of the metrics. NLTK[15] library was used to calculate the most frequent words. And sankeyflow[16] to print sankey diagrams.

2.3 Problems found

During the development of the project, we found some problems or challenges to overcome.

2.3.1 System resources management

Python was used an instance for the insertion data. was a mistake .

A common challenge for a large database is the management of its resources. All the process should be the best optimized that could be to minimize the resources that spent. The release of the DBLP XML in June 2024 has a weight of $4,27$ GB, and each month it increases by around $0,04$ GB. That means hundreds of thousands of new items each month. To process that amount of data, it is important which technologies are used, and many times, the order of the execution could save or aggregate hours of processing.

Several times, in the set-up process, it terminated abruptly because of a lack of space. The XML had to go through reconfiguration scripts, and each created a new file. Furthermore, it was necessary to split the file into small files to get a cleaner and more efficient process, which would generate more files and leave less space in the machine. The same thing happens with the data increment process.

2.3.2 Format and reformatting XML files.

When is inserting information from one system to another, there is strange when no need for any formatting. This is not the case. The initial obstacle encountered in the formatting of DBLP XML was the codification of the file. The original was found in the codification ISO-8859-1, which is a UNICODE, also referred to as Latin 1, used to write languages from Occidental Europe, which includes Germany, the country where the DBLP project is held. The issue lies with the fact that information systems, tools, and more are written in UTF-8. Afterward, it is necessary to perform a parse from ISO-8859-1 to UTF-8.

Other problems found in the format of XML were that, sometimes, items had no spaces between them. That increases the difficulty of splitting the XML and, countless times, throws errors of format, or, in the worst case, inserts some incorrect information about an item. This is a dangerous error, as it did not appear until the wrong data was accessed. When this file is large, the difficulty increases because the time spent fixing errors could be high.

2.3.3 Item *book* as a *Publication* or a *PublicationGroup*

When the *book* item was taken as a *PublicationGorup* something had no sense: some of them had any reference to them. That means any publication was in these items, but other *book* items had

[13] `https://docs.sqlalchemy.org/en/20/index.html`
[14] `https://matplotlib.org/`
[15] `https://www.nltk.org/`
[16] `https://pypi.org/project/sankeyflow/#description`

publications referred to them. Then, the first *book* items group work more like a publication, and the other group more like a publication groups.

The *book* item, which is classified as a *PublicationGroup* mainly has one or more *incollection* XML items referenced to it. In Listing 4 4 are two items with a *crossref* element; this element is a key from a *book* item, shown in Listing 5 5.

Listing 4: Two items incollection class with the same crossref element

```
<incollection mdate="2017-05-16" key="books/daglib/p/Reinhartz-Berger13">
<author>Iris Reinhartz-Berger</author>
<title>When Aspect-Orientation Meets Software Product Line Engineering.</title>
<pages>83-111</pages>
<year>2013</year>
<booktitle>Domain Engineering, Product Lines, Languages, and Conceptual Models</booktitle>
<ee>https://doi.org/10.1007/978-3-642-36654-3_4</ee>
<crossref>books/daglib/0032299</crossref>
<url>db/books/daglib/0032299.html#Reinhartz-Berger13</url>
</incollection>

<incollection mdate="2017-05-16" key="books/daglib/p/KoshimaET13">
<author>Amanuel Alemayehu Koshima</author>
<author>Vincent Englebert</author>
<author>Philippe Thiran</author>
<title>A Reconciliation Framework to Support Cooperative Work with DSM.</title>
<pages>239-259</pages>
<year>2013</year>
<booktitle>Domain Engineering, Product Lines, Languages, and Conceptual Models</booktitle>
<ee>https://doi.org/10.1007/978-3-642-36654-3_10</ee>
<crossref>books/daglib/0032299</crossref>
<url>db/books/daglib/0032299.html#KoshimaET13</url>
</incollection>
```

Listing 5: Book item with a key which some incollection items have as a crossref.

```
<book mdate="2020-11-03" key="books/daglib/0032299">
<editor>Iris Reinhartz-Berger</editor>
<editor orcid="0000-0002-4021-7752">Arnon Sturm</editor>
<editor>Tony Clark 0001</editor>
<editor>Sholom Cohen</editor>
<editor>Jorn Bettin</editor>
<title>Domain Engineering, Product Lines, Languages, and Conceptual Models</title>
<publisher>Springer</publisher>
<year>2013</year>
<isbn>978-3-642-36653-6</isbn>
<isbn>978-3-642-36654-3</isbn>
<ee>https://doi.org/10.1007/978-3-642-36654-3</ee>
<url>db/books/daglib/0032299.html</url>
</book>
```

Other items *book* do not have any publication that could be referenced to them. And typically have a similar structure as an item that could be stores as a *Publication*. The Listing 3 4 is an example of a *book* item as a *Publication*.

The challenge resides in finding a way to know with a simple set of rules when an item should be saved as a *Publication* or a *PublicationGroup*.

After some tests, the key difference in their structure was found: *Publications* have authors, but *PublicationGroups* have editors; this is the criteria to classify them.

2.3.4 Number of pages calculation

In Figure 2, the Publication table has *pages* as a String and *numPages* as an Integer calculated. One of the most consuming challenges in the import phase was identifying the numerous cases to calculate this number of pages. Even so, some of the *pages* are not correctly settled, consequently not saving a correct number of pages. And in this point, the number of pages for all Publications is not possible to ensure. A total of 74456 Publications has not the pages value. Then a $1,22\%$ of publications which has *pages* has not been processed well. An instance is a publication with a *DOI* identified saved as a *pages*. Once a process of inserting ends, we are being able to calculate how many publications we have with a calculated number of pages.

2.3.5 Corrigendum as a publication itself

Some of the publications are corrigendums from other publications, with the same authors, sometimes in the same year, sometimes in later years. The interest in identifying all of corrigendums and relating to their original publication falls in getting an accurate metrics.

2.3.6 Institution names repeated many times

The result of querying the total affiliations from all institutions shows that all institutions have only one affiliate. Looking in the institution names is found some similar names but not at all the same, Table 1 (1) shows the result of the query in Listing 5. The aim of this query is getting all institution names with the word 'UOC'. The two institutions are nearly the same, just one has '(IN3)' after 'Internet Interdisciplinary Institute', but the second result does not have it. Both institutions are the same, but in the database have two different names.

```
1 SELECT name FROM institutions WHERE name LIKE '%UOC%'
```

Listing 6: Query to get insitution names with the word UOC.

	name
1	Open University of Catalonia (UOC), Internet Interdisciplinary Institute (IN3), Barcelona, Spain
2	Open University of Catalonia (UOC), Internet Interdisciplinary Institute, Barcelona, Spain

Table 1: Result of the query shown in the Listing 5

Before to work with *institutions* a process of data cleaning is mandatory.

2.3.7 Other minor errors

- An odd error was a publication with a duplicated author in their list of authors but different position.

2.4 Solutions and alternatives

2.4.1 System resources management

The full definition of the problem about system resources management was in section 2.3.1. Process all the XML along; it is a difficult process that is so huge to do in a comfortable place of time. To solve this, splitting the XML and working in different parts was the best decision. Furthermore, keeping the process in separate parts makes it easier to understand. Although the space in the machine is affected, to solve this, removing files was the key to keeping enough space.

2.4.2 Format and reformatting XML files

The formatting issue was divided into two parts: one was the codification, and the other was the lack of spaces between items. The first problem was solved by parsing the entire XML file from ISO-8859-1 to UTF-8. The whole file was parsed from ISO-8859-1 to UTF-8, replacing all special characters one by one. The second issue was finding the start and ending tags and adding a blank. Both of these problems were resolved using AWK.

2.4.3 Item *book* as a *Publication* or a *PublicationGroup*

The definition of the *book* as a *Publication* or a *PublicationGroup* was in section 2.3.3. The key to solving the *books* problem was looking at their items: which has authors, that means it is a publication, and on the other hand, which has editors was a publication group. To put in practice this, an *awk* script was written to separate in two XML files: one for the books like publications and another for the books like publication groups. Once had the two files, each of them was treated as a publication file (like inproceedings) or a publication group files (like proceedings).

2.4.4 Number of pages calculation

The problem with the calculation of pages was defined in the section 2.3.4. Calculating the number of pages per publication is an easy but long task. Many formats are found in the XML, and sometimes wrong information like a number of the order of hundred of thousands. For the most cases, that can be calculated filtering the data by format and then making the correct calculus.

An example the publication, article, with key "conf/www/CibranVVSJ07" has "211-242" in the field pages. To calculate this is taking the second part "242" and subtract the first one "211", the total given is the number of pages: 31 pages in this case.

More examples of formats are shown in the Listing 7, the first like is the key, the second like is the pages like and each of them are separate with a blank.

Listing 7: Book item with a key which some incollection items have as a crossref.

```
<article mdate="2021-02-17" key="journals/pacmpl/SpathAB19">
<pages>48:1-48:29</pages>
```

```
<inproceedings mdate="2019-05-29" key="journals/jmlr/BubeckCK12">
<pages>41.1-41.14/pages>

<phdthesis mdate="2021-07-17" key="books/daglib/0081602">
<pages>I-XIV, 1-167</pages>

<inproceedings mdate="2017-05-21" key="conf/3dic/WangHCYKCCCLJTL15">
<pages>TS6.3.1-TS6.3.4</pages>

<inproceedings mdate="2017-05-25" key="conf/pimrc/WangCZ02">
<pages>B138-B142</pages>

<inproceedings mdate="2019-11-13" key="conf/itc-asia/Zorian17">
<pages>ix-xi</pages>

<inproceedings mdate="2018-11-06" key="conf/si3d/StateMNCCCF95">
<pages>69-74, 208</pages>
```

2.4.5 Corrigendum as a publication itself

Corrigendum problem was defined in the section 2.3.5. Because of the low number of Corrigendums (2290) was decided to pospone for the future work to improve the system.

2.4.6 Institution names repeated many times

The Institution names problem was defined in the secction 2.3.6. PostgreSQL has some functions like *word_similarity*. These functions return a percentage of similarity. After some tries, no function was found good enough.

The example is in Listing 8, where the current name of an institution was selected as a filter. The result is shown in Table 2. The first row has proximity 1 because it is the same as the example. Then, some results showed more similarity after getting someone known as the same institution or a more related institution.

Listing 8: Select word similarity for Open University of Catalonia.

```
select
    word_similarity('Open University of Catalonia, Barcelona, Spain', name) as similarity,
    name
from institutions
where word_similarity('Open University of Catalonia, Barcelona, Spain', name) > 0.4
order by similarity DESC;
```

1	1.0	Open University of Catalonia, Barcelona, Spain
[..]	[..]	[..]
7	0.84782606	Polytechnic University of Catalonia (UPC), Barcelona, Spain
[..]	[..]	[..]
10	0.82978725	Polytechnic University of Catalonia, ARCO, Barcelona, Spain
[..]	[..]	[..]
13	0.7906977	Polytechnic University of Catalonia, Barcelona Supercomputing Center, Barcelona, Spain
[..]	[..]	[..]
14	0.7291667	International University of Catalunya, Barcelona, Spain
[..]	[..]	[..]
17	0.6976744	University of Barcelona, Spain
18	0.6976744	Autonomous University of Barcelona, Spain
19	0.6976744	Polytechnic University of Catalonia, Spain
[..]	[..]	[..]
23	0.6511628	Open University of Catalonia (UOC), Internet Interdisciplinary Institute, Barcelona, Spain
24	0.6511628	Open University of Catalonia, Department of Computer Science, Multimedia and Telecommunication, Barcelona, Spain
25	0.6511628	Open University of Catalonia, Department of Computer Science, Multimedia and Telecommunication
26	0.6511628	Open University of Catalonia, Computer Science Department
27	0.6511628	Open University of Catalonia (UOC), Internet Interdisciplinary Institute (IN3), Barcelona, Spain
28	0.6511628	Catalan Earth Observation Centre, Cartographic and Geological Institute of Catalonia, Barcelona, Spain
29	0.6511628	Open University of Catalonia, Department of Computer Science, Spain
[..]	[..]	[..]

Table 2: Result of word similarity

Keeping the same query but changing the function to *strict_word_similarity* has a very similar result. The result 'Open University of Catalonia, Department of Computer Science, Spain' has the 21st position from the similarity of 'Open University of Catalonia, Barcelona, Spain' with the same punctuation of 0.6511628.

With these results, it's not trivial to find a way to aggregate institutions. From the 3477424, an $24,33\%$ of the researchers (142940) have enough information to know with which institutions they are affiliated. It must be added that the information provided is limited, since we do not have the relationship with which publication that affiliation was made.

3 Results

3.1 Model, System deployment and autoincrementality

3.1.1 Model

As a result of working with DBLP data, the model suffered some changes, which were reflected in the database. The result database scheme is illustrated in Figure 7. This schema is adapted for a PostgreSQL database, which accepts an array of strings as a field, like names, authors, or editors from researchers, publications or publication groups, respectively.

In the publications table it was necessary to add a *journal* column, some of the publications in the XML have this field. For the majority of the journals, that field shows which journal was published, and there is also no publication group for this journal. That means to query all the publications for a specific journal, it is necessary to filter by this column. Also added were two columns for a future upgrades: *is_corrigendum* and *corrigendum_of* to relate publications with their corrigendum.

In institutions was added the columns *country* and *version_names*. The first was created to filter researchers by country affiliation. The second new column was created to unite all institutions with different exact names but referring to the same institution.

A column *xml_tag* was mandatory to know the origin of the data; this way, it is easier to trace of it. And that column was added into researchers, publications, and publication_groups tables.

Finally, oid_xml is a support table. Its purpose is when the XML filed is loaded to proceed with the inserts. Sometimes it needs to get access again to those data, and then with the OID number, the field does not need to open once more, making the process more efficient.

3.1.2 System deployment

A Docker Composer with two Dockers was settled up to ensure the stability and homogeneity of the systems settings. One docker with a PostgreSQL image, the official one. The second is a simple Alpine, has a small app in Python to get access to the database. It was also used to run a group of scripts to get some metrics.

The requirements for a machine where wants to be settled up the project are: AWK toolkit, CURL, Docker and Docker-Compose. Additionally, the machine should be a bash shell based.

A script in the root of the project, named "*set-up.sh*", downloads the XML from the DBLP website. Reformatted and splits that file into a more workable XML.

At this point, the PostgreSQL docker image should be running to execute the next bash script, inser-data.sh, stored in the database directory. This script should be executed inside Docker and insert all the XML into a ready-to-use database.

3.1.3 Autoincrement process

In the autoincrement process, there is a light description. A similar set of scripts, to the set-up ones, is created for this process. And a similar process should be executed. Outside the Docker and in the root of the project is the "*update.sh*". And inside the database directory is "update-data.sh" which should also be executed inside the PostgreSQL image.

Then the database would be up-to-date, and new research could be done. However, there is no proves to their efficiency. Around the 1% of the items in the DBLP XML original, are new or need to be updated in respect to the last month. But the process takes almost the same time as the

setup process. It is not strange at all because the step of creating the relationships between models takes the major part of the time, and this would be done the same way as the set-up.

Meanwhile, the autoincrement process is object to some optimization, it could use the set-up process as a process of autoincrement.

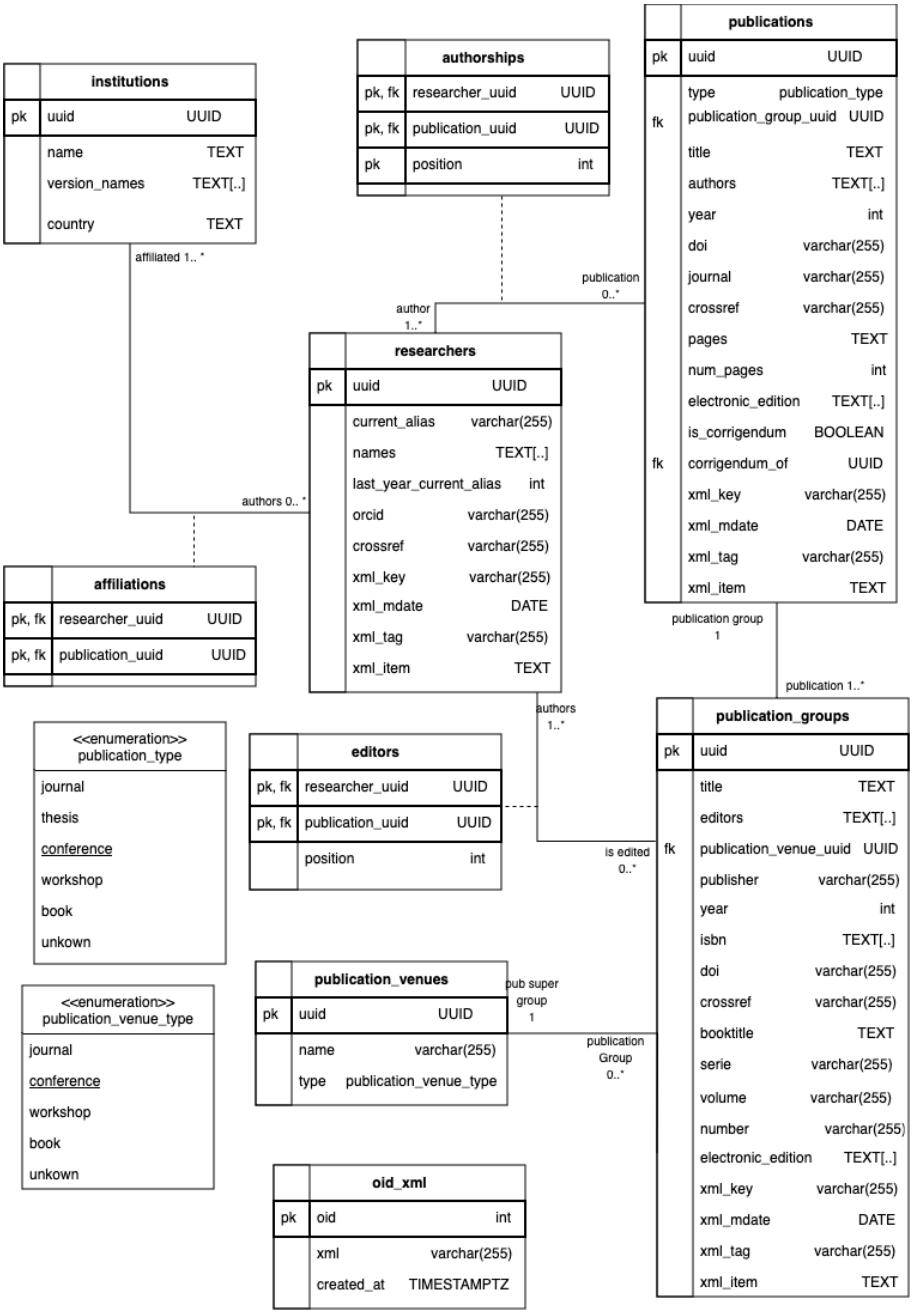


Figure 7: Final DBLP Database

3.2 Metrics

3.2.1 Summary of tables

As a result of the project, some overall numbers are shown in Table 3. One of the data presented is the ratio of affiliations to researchers: Only 4.11 percent of researchers possess some affiliations, with a median of 2.16 percent affiliations per institution. That is very odd, but there are explanations in the section2.3.6. Another interesting statistic is that there are very few workshops; only 0.07% of the publications are workshops. It could be that they are not saved in the DBLP database or that they are not identified as a workshop. Since 0.71% is not identified, the probability that it is not saved in DBLP is high.

Researchers	3.477.424		
Institutions	66.147		
Affiliations	142.940		
Publications	7.088.938	Journals	3.457.709
		Conferences	3.422.188
		Workshops	5.052
		Thesis	128.427
		Books	25.466
		Unknown	50.096
Authorships	23.221.710		
Publication groups	60.171		
Editors	146.423		

Table 3: Number of items per table

3.2.2 Publications per researcher

The relationship between publications and researchers could be a number of publications per researcher or number of researchers (authors) per publication.

In the first case, there is a median of 2 publications per researcher as an overall view. But when it is filtered by type of publication, the median number goes down to 1. In the plot on the left in the Figure 8, all outliers, and that could explain why the median of overall get one point height: there are many outliers, exactly 492.420 that represents an $14,18\%$ of the number of publications per researcher are outliers. The number of outliers is lower in filtered cases: journals has 262.936 that means an $11,31\%$ of them; conferences have 258.788 outliers, and that is the $11,7\%$; and workshops have a lower ratio of outliers with 80 points out of the box, which represents only an $0,68\%$ of them.

The right plot in Figure 8 shows it without the outliers. The common number of publications per researcher oscillates between one and eight.

3.2.3 Authors per publication

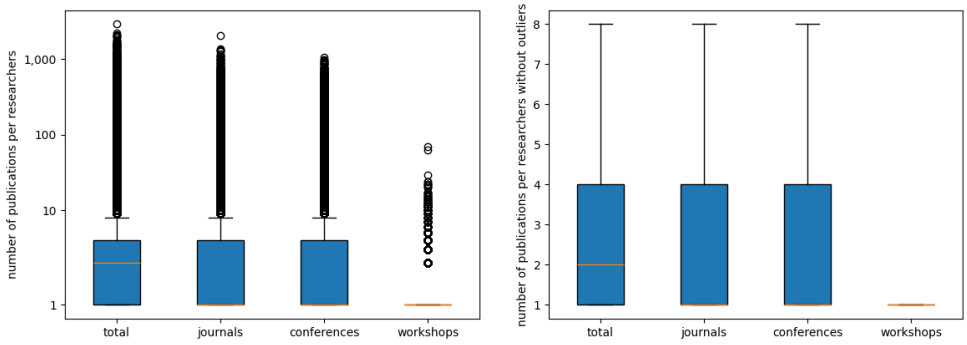
When referring to the number of authors per publication, it means that at least one author is in the position of the main author. A publication typically has more authors and researchers who collaborated on the investigation in some way.

	total	outliers	percent
Total	7.051.199	216.199	$3,07\%$
Journals	3.436.053	119.315	$3,47\%$
Conferences	3.417.038	95.535	$2,8\%$
Workshops	5.020	61	$1,22\%$

Table 4: Outliers removed from plot

As occurs in publications per researcher, this case also has many outliers, as shown by the plot on the left in Figure 9. The median of all four shown cases (the total and those filtered by journals, conferences and workshops) is the same: 3 authors per researcher. The plot on the right of Figure 9 shows that Journals and Conferences usually have up to 7 authors, and workshops increase that to up to 9 (without counting outliers). The total is influenced by conferences and journals because these two types are the 97%.

To understand better, the right plot in Figure 9. Table 4 is the summary of outliers removed per box plot.



(a) Results in Log(y)

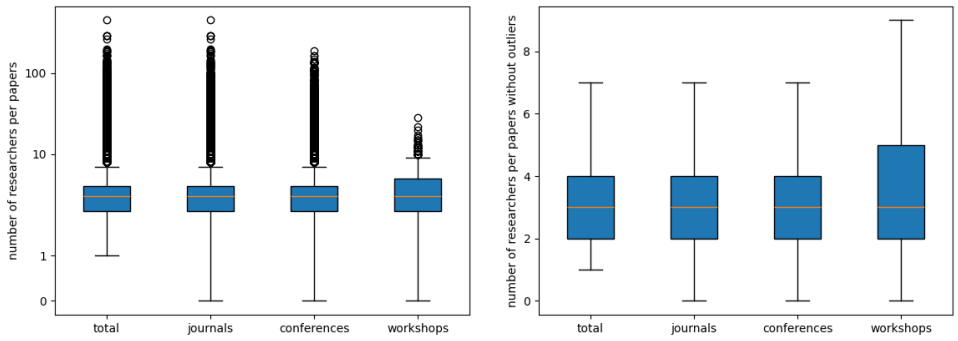
(b) Result without outliers

Figure 8: Publications per researcher

3.2.4 Publications per Group of publication

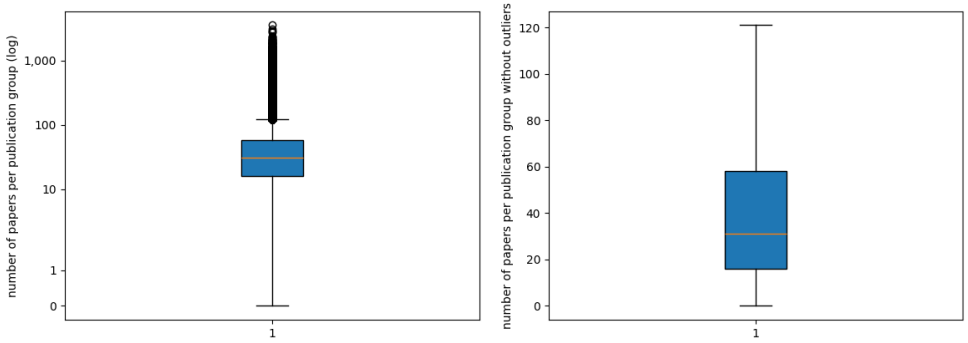
Publication Groups is defined a specific group of publications like a journal or a conference, and not an organization. Usually, the same conference from different years is represented as two unrelated entities.

In the figure 10 is shown two box plots of number of publications per publication group, one with the outliers (left) and calculated their Log(y), and on the left without the outliers. Has a median of 31 publications per publication group. And the outliers removed from the right box plot are 5.114 given an $8,53\%$ of the data.



(a) Results in Log(y) (b) Result without outliers

Figure 9: Author per publication



(a) Results in Log(y) (b) Result without outliers

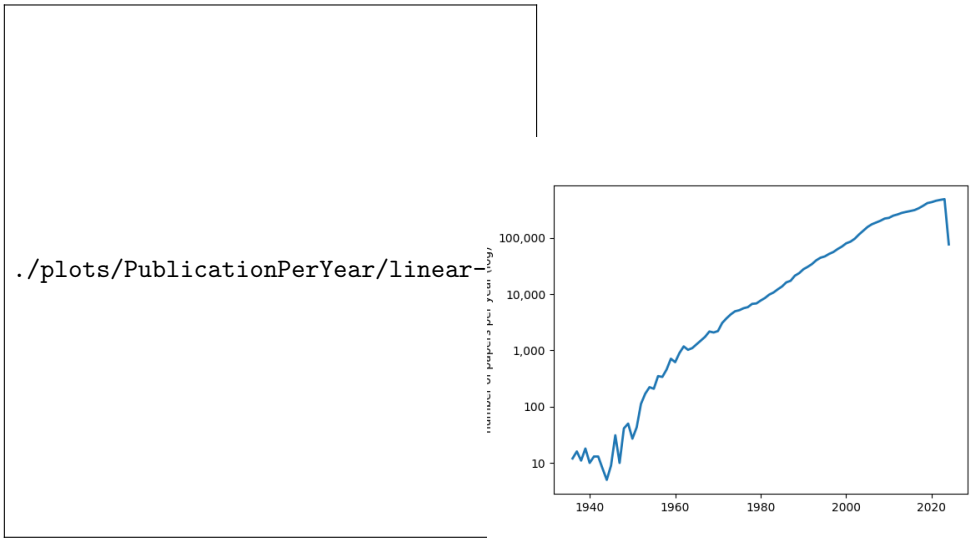
Figure 10: Publications per group of publications

3.2.5 Publications per year

Another noteworthy statistic is the number of publications per year. The figure depicts the number of publications produced annually, revealing the fluctuation of the publication's growth or decline year after year, and identifying the years with the highest or lowest number of publications. Figures 11 and 12 show these metrics. A lineal plot shows how to grow the number of publications per year. In the plot on the left, around the change of the millennial looks like an exponential increment, but on the right plot, where data is shown as a Log(y), shows that the incrementally is stabilizing.

The pronounced fall at the end of the plot is due to the number of publications in 2024. It looks like the publications have collapsed, but is only because this year not finished, and more publications will be published this year.

In the box plots in figure 12, it is observed that the median is very low compared to the third

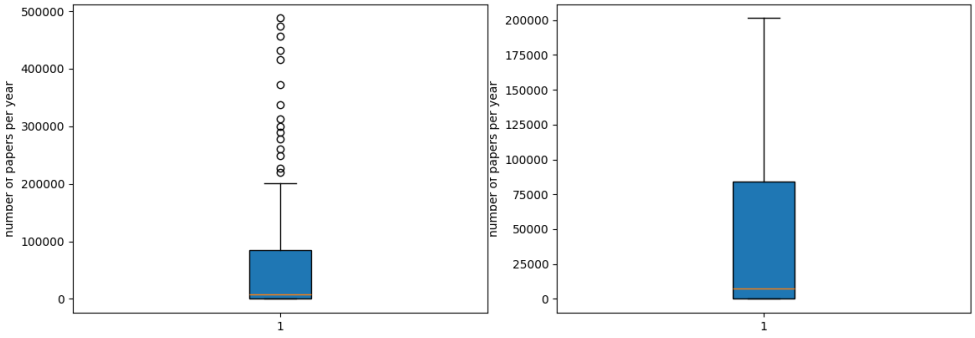


(a) Original result (b) Result in Log(y)

Figure 11: Linear plots of publications per year

quartile. This makes sense, since in the early years, before 2000, there were very few publications compared to the number per year that there are in years after 2000, with experiments an exponential growth.

From the 90 years registered, in the right plot of figure 12, 15 years were removed as there are outliers, that was the 16, 7%.



(a) Result in Log(y) (b) Result without "outliers"

Figure 12: Box plots of publications per year.

3.3 Advanced metrics

3.3.1 Most common words from related conferences and journals

An intriguing metric could be to display the most prevalent words in the titles of publications in order to reveal the trend words within those titles. That words could be compared with other trends in order to see how they influence subgroups of publications. The conferences and journals listed in 3.3.1 are about similar topics. This filter was made to compare them. Also was removed stop-words, nonsense words and the words are in the title of each conference or journal to avoid noise.

The python library "NLTK :: Natural Language Toolkit" was selected to do the work easier. In addition, from the "sop-words" provided by the library, were added punctuation sings (".", ",", ":", "(", ")" ...) and a common word which not give any topic information.

The words cataloged as no-topic are: 'using', 'approach', 'development', 'design', 'process', 'study', 'search', 'service', 'method', 'code', 'case', 'formal', 'based', 'learning', 'applications', 'system', 'program', 'use', 'towards', 'section', 'part', 'introduction', 'guest', 'issue', 'source', 'review', 'systematic', 'workshop', 'analysis', 'editorial'.

This tries to show what are the subtopics most common in the different conferences and journals. Always applying in the publications published in the conferences and journals listed in 3.3.1.

The conferences list:

- International Conference on Software Language Engineering (SLE)
- International Conference on Software Engineering (ICSE)
- International Conference on Advanced Information Systems Engineering (CAiSE)
- International Conference on Model Driven Engineering Languages and Systems (MoDELS)
- International Conference on Conceptual Modeling (ER)
- International Conference on Web Engineering (ICWE)
- The Web Conference (WWW)
- International Conference on Model Transformation (ICMT)
- International Working Conference on Exploring Modeling Methods for Systems Analysis and Development (EMMSAD)
- International Conference on Software Analysis, Evolution and Reengineering (SANER)
- Research Challenges in Information Science (RCIS)
- International Conference on Mining Software Repositories (MSR)
- International Symposium on Empirical Software Engineering and Measurement (ESEM)
- Fundamental Approaches to Software Engineering (FASE)
- Symposium on Applied Computing (SAC)

The jorunals list:

- Software and Systems Modeling
- IEEE Transactions on Software Engineering (TSE)
- ACM Transactions on Software Engineering and Methodology (TOSEM)
- IEEE Software
- IEEE Computer
- Communications of the ACM
- Information & Software Technology
- Empirical Software Engineering
- Journal of Object Technology
- Journal of Systems and Software (JSS)

Getting the premises before described, in figure 13 shows the 5th most frequent words in the 62.350 titles processed. "Data" was the ultimate winner, getting the double of times than the next one, "algorithm". "Algorithm" and "testing" have a very similar number of repetitions. And the last two in the podium also have a similar frequency between them: "framework" and "requirements". That shows "Data" is the most important topic by difference.

To better comprehend the figure 13, the results were divided into journals, left, and conferences, right, in figure 14. The term "Data" retains the first position in both plots. The data almost duplicates the frequency from the next in the pod, so it makes sense. Both plots do not have all the other words present in them, respectively.

Figure 14 (left plot) shows that the five most common words are: data, algorithm, testing, computer, and programming. In contrast to journals, conferences, the right plot of figure 14, shows that "Data" surpasses the double of "framework", which is the next plot. In fact, the five most frequent words in conference titles have a similar usage.

A total of 26,642 titles were given in conferences and 35,708 titles were given in journals. That translates to 42.73% and 57.27%, respectively. The plots in Figure 14 show that the data topic is a trend in conferences, but not in journals. This is an important topic, but other topics are almost the same importance.

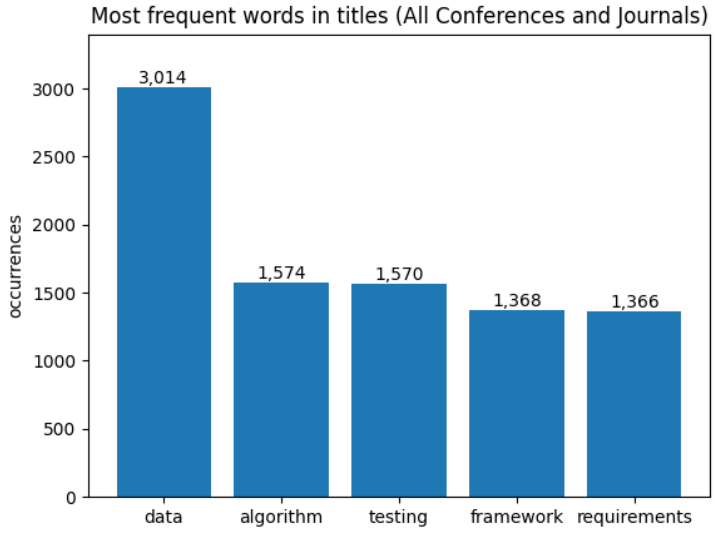
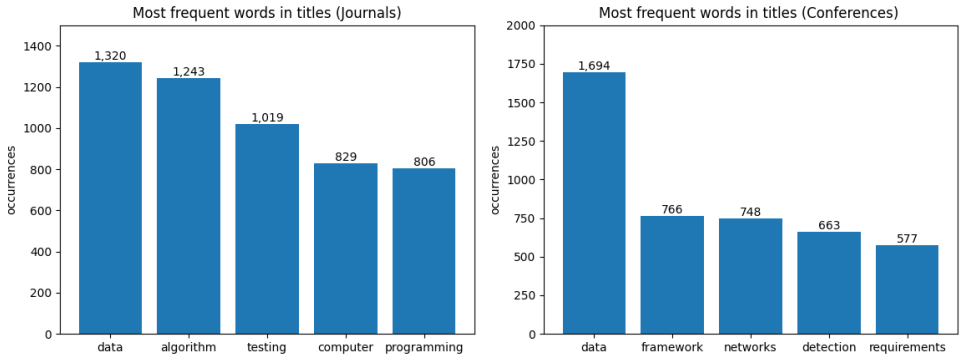


Figure 13: Bar plot with most common words in titles of publications



(a) Bar plot with most common words in titles of journals

(b) Bar plot with most common words in titles of conferences

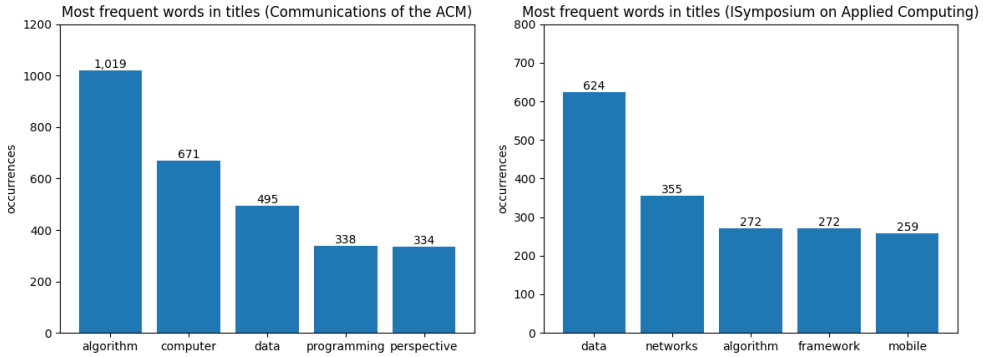
Figure 14: Most common words in titles of journals and conferences

The quantity of titles (publications) may have an impact on the outcome. A search of the conferences and journals with the most number of titles was made using this premise. SAC, "Symposium on Applied Computing" was the conference with the most titles. It has 8,483 titles, and represents a 31.84% of the conferences and a 13.61% of the overall. And "Communications of the ACM" was the journal with the most titles. It has 13,500 titles, representing a 37.81% from the journals and a 21.17% overall.

The left plot on Figure 15 shows the five most common words in titles published in the journal "Communications of the ACM." 'Algorithm' is the commonest word. In a global term, are 81.98% of occurrences between journals, and 64.74% overall.

The left plot on Figure 15 represents the five most common words in the titles published in SAC conference. 'Data' is the word most commonly used: 36.84% of the occurrences between conferences and 20.7% of the overall "data" occurrences.

It is safe to say that publication groups with more publications have a greater influence on the final output. Nonetheless, this is not a bad result. It makes sense that the publication groups with the most published papers influence the others. The popularity of the conferences and journals has an impact on the words used in the titles of the publications.



(a) Bar plot with most common words in Communications of the ACM journal

(b) Bar plot with most common words in SAC conference

Figure 15: The most common words in a specific conference and journal.

3.3.2 Journey of a researcher started in 2000 or after

Another interesting question to be answered from the database is, 'Which is the journey taken by researchers?" To answer it, first it should be defined what a 'Researcher Journey' is. This journey refers to the publications a researcher has made. The aforementioned publications could be categorized based on their respective types to establish an itinerary. In order to achieve this metric successfully, some criteria should be applied:

- Publications are categorized based on their type.
- Only researchers with four or more publications are eligible.
- Researchers began their academic career in 2000 or after that.

Table 3 shows the five publication types, but journals and conferences have 97.05% of them. These two types were selected to simplify the study. Figure 8 illustrates how many publications there are per researcher. The majority of authors only publish two or fewer papers. But that plot shows that the third quartile is in four publications. Subsequently, this number is utilized

to prolong the journey. Figure 11 shows a significant growth in publications around 2000. The statement 'Researchers started in 2000 or later' helps to simplify the research.

Considering all of this information, the questions to answer are:

- **What are the paths of publications for researchers whose first publication was after 2000?**
- **Does this show significant differences if temporary windows are applied?**
- **Does it have similar behavior in a list of related conferences and journals?**

Figure 16 shows the overall journey. In general terms, most researchers start publishing at conferences rather than journals. Some researchers, starting out in a journal, try to publish their work in a conference. In all cases, a large group of researchers will publish a conference, and the next paper will be published in a journal. The researchers, whose last paper was published in a Journal and the next one is at a Conference, are a small group but are large enough to consider. In general, for each stage, there exist a greater number of journals and a lesser number of conferences, culminating in a roughly equal number of conferences and journals.

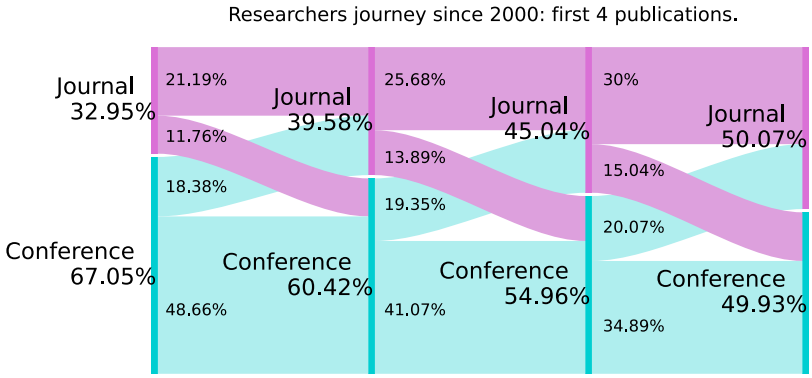
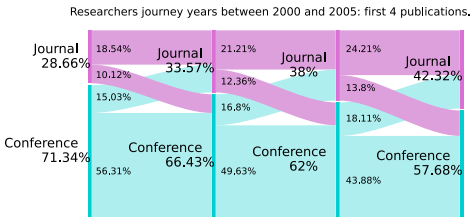


Figure 16: A sankey plot for Journey

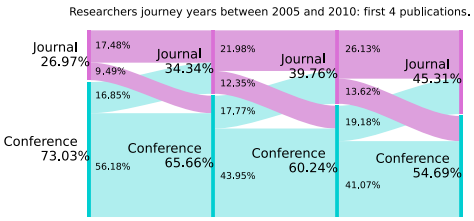
This are the temporal windows are applied to address the second question: [2000–2005), [2005–2010), [2010–2015), [201
Figure 17 shows the five plots as a result.

In the first 15 years, the number of first publications as journals is very low, and the number of second, third, and fourth publications grows gradually. However, since 2015, the number of first publications as conferences has become smaller and smaller. That means there is a trend for new researchers to publish first in journals than in a conference.

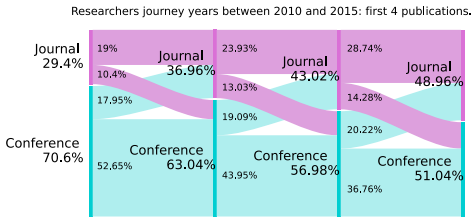
Figure 18 shows the journey taken by researchers, applying the list in Section 3.3.1 as a filter of related journals and publications. The overall indicates a similar behavior; however, in the filter results, there is a noticeable increase in the number of journals within four steps compared to the general.



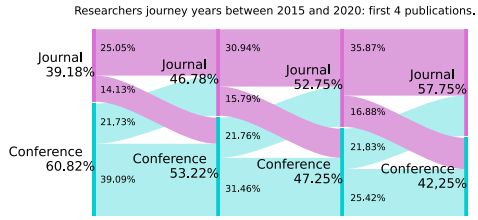
(a) 2000 to 2005



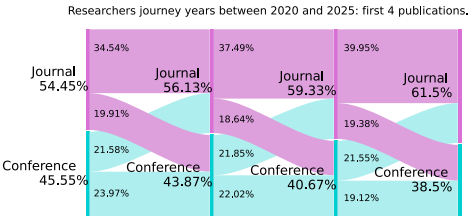
(b) 2005 to 2010



(c) 2010 to 2015



(d) 2015 to 2020



(e) 2020 to 2025

Figure 17: Sankey plots for Journey split by temporal window

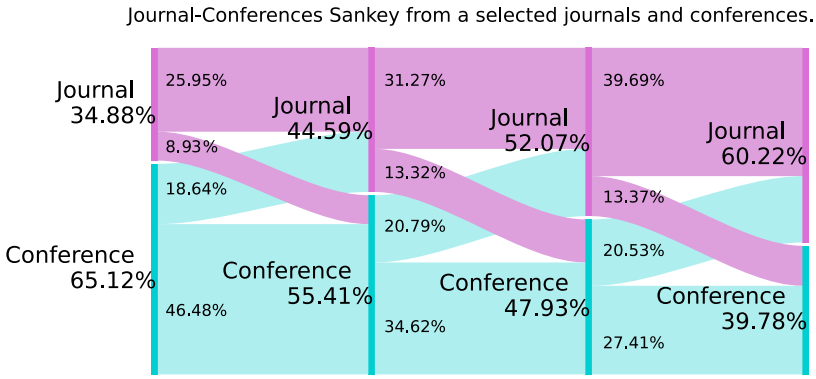
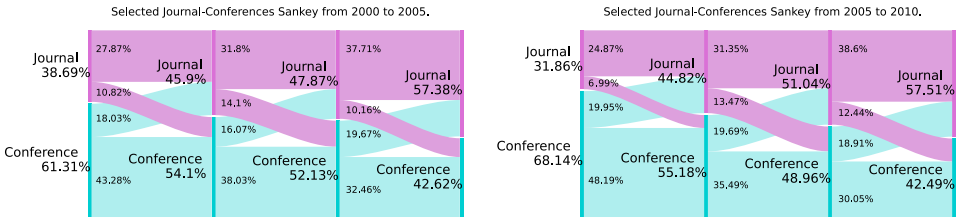


Figure 18: A Sankey plot for Journey applied only to a list of publication groups

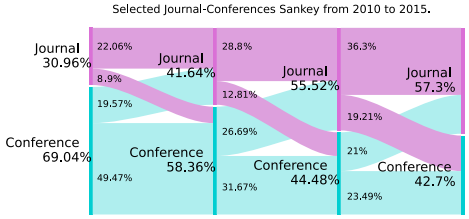
If the comparison is made with the temporal window plots, then the results are different. In the first years, the number of journals as a first publication is slightly higher than the general one. But, between 2015 and 2020, it was almost the same. And the next window, 2020–2025, drastically reversed the results: in the filtered list, there are fewer journals than conferences as a first publication than the general, which has the opposite result.

Another anomaly in this last plot temporal window from 2020 to 2024 from filtered data is that from the second publication to the third and then to the fourth, journals published decreased. This behavior is the opposite to the general view.

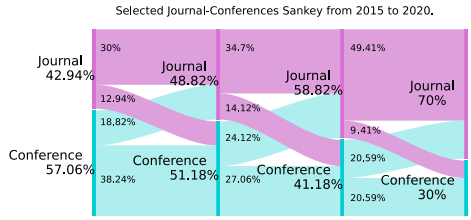


(a) 2000 to 2005

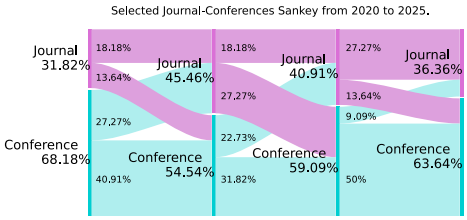
(b) 2005 to 2010



(c) 2010 to 2015



(d) 2015 to 2020



(e) 2020 to 2025

Figure 19: Sankey plots for Journey split by temporal window

4 Conclusion and future work

4.1 Conclusion

It is becoming increasingly common to hear the term scientometrics or metascience, a field aimed at measuring and analyzing scholarly literature. However, the current approaches to performing the studies encountered several issues. These studies are ad hoc and derived from subsets of studies, or the systems are inadequately maintained or abandoned.

The work addresses these issues by developing a relational database that is easy to maintain. The DBLP database will be utilized, which has been recompiled into a substantial XML file and is regularly updated. The relational database is commonly used for the easy way to get relationships, and because this is selected for this project. Furthermore, it would be straightforward to establish and maintain the project by utilizing a few scripts to facilitate the increment of data. After the system was completed, a summary of the data was provided. Additionally, a few examples of usage are described, along with a brief analysis of the data.

Through this project, new research in the field of scientometrics could be conducted. It may also be utilized to identify researchers based on their publications. Furthermore, it could be used to make queries to find the most important publications and researchers.

4.2 Future work

In the future, our approach has the potential to help conduct new studies in the field of scientometrics. As example, certain studies could be conducted:

Try to address some pertinent inquiries, such as identifying the researchers with the greatest influence or identifying the publication group with the greatest impact on the community. Additionally, it would be possible to see which institutions are the most affiliated, or which countries have the most publications, by applying data cleaning. With the help of a scraper, it could also obtain the abstracts of the publications and compare their relevance to the title, as well as keywords. Ultimately, a Regression Analysis may provide recommendations for keywords based on the title and abstract, or it may suggest which journal or conference to publish.

5 Glossary

- attribute: each xml items and xml elements could have attributes.
- doi: Is a uniquely identifier which identifies an article or document, and provides it with a permanent URL.
- element: each xml item has a number of sub-items that we called elements.
- item: first element of the xml tree.
- orcid: Is a unique and open digital identifier that distinguishes a researchere from every other one with the same or a similar name.
- ORM: Object Relational Mapper, is a programming model which allows to map the schema of a relational database.
- XML: eXtensible Markup Language. Allows to define and store data in a compatible format.
- Corrigendum: A document like a publication, but refers to a correction about another publication.

6 Bibliography

- Cánovas, Javier, Cosentino, Valerio & Cabot, Jordi (2016) Analyses of co-authorship graph of CORE-ranked software conference *Scientometrics, 109*, 1665-1693
- Biryukov, Maria & Dong, Cailing (2010) Analysis of Computer Science Communities Based on DBLP *LNISA, 6273*, 228-235
- Wu, Yan, Venkatramanan, Srinivasan & Dah Ming, Chiu (2010) A Population Model for Academia: Case Study of the Computer Science Community Using DBLP Bibliography 1960-2016 *IEEE Transactions on Emerging Topics in Computing, 9*, 258-268
- Ley, Michael DBLP - Some Lessons Learned
- PostgreSQL documentation. Available in: https://www.postgresql.org/docs/current/index.html (Consulted between: 2024 March 10th and 2024 May 28th).
- Docker documentation. Available in `https://docs.docker.com/`(Consulted between: 2024 March 18th and 2024 May 1st)
- Docker compose documentation. Available in `https://docs.docker.com/compose/`(Consulted between: 2024 March 18th and 2024 March 28st)
- StackOverflow: Split a single xml file into 2 files based on tag count. Available in: https://stackoverflow.com/questio a-single-xml-file-into-2-files-based-on-tag-count (Consulted at: 2024 March 18th).
- StackOverflow: Postgres how to pass parameters from command line. Available in: https://stackoverflow.com/quest how-to-pass-parameters-from-command-line (Consulted at: 2024 March 19th).

- Stack Exchange: Selecting only overlapping elements from array of ranges. Available in: https://dba.stackexchange.com/questions/147129/selecting-only-overlapping-elements-from-array-of-ranges (Consulted at: 2024 March 19th).
- Creative Commons. Available in: https://chooser-beta.creativecommons.org/ (Consultat: 2024 April 02nd).
- Google Developers. Available in: https://developers.google.com/public-data/docs/canonical/countries_csv (Consulted at: 2024 April 12th).
- SQL Alchemy documentation. Available in: https://docs.sqlalchemy.org/en/20/index.html (Consulted between: 2024 May 05th and 2024 May 28th).
- Matplotlib documentation. Available in: https://matplotlib.org/ (Consulted between: 2024 May 15th and 2024 June 1st).
- NLTK library documentation. Available in: `https://www.nltk.org/`(Consulted between: 2024 May 18th and 2024 June 1st).
- Sankeyflow library description. Available in: `https://pypi.org/project/sankeyflow/#description`(Consulted between: 2024 May 18th and 2024 June 1st).

7 Appendices