# Data & Information Management

*Project Specification 2022-2023*

VRIJE UNIVERSITEIT BRUSSEL

**Professor**

Bas Ketsman (bas.ketsman@vub.be)

**Teaching Assistants**

Tim Baccaert (tim.baccaert@vub.be)
Samuel Ngugi Ndung'u (samuel.ngugi@vub.be)

## Overview

In this group project you will solve a set of query problems over the *DBLP computer science bibliography*[1], in groups of four people. Each group will get to choose one relational and one non-relational database management system with which to solve these problems.

The project consists of three major segments:

1. **Data Preparation:** clean and ingest the raw DBLP data into both of your systems.
2. **Querying:** solve the given query problems using both systems.
3. **Benchmarking:** properly measure how long each query takes on each system.

Lastly, you will bundle these findings in a report.

## Formalities

First, you will have to create a group with three other students. Sit together and create the following two rankings:

- a top 3 ranking of your three preferred **relational database systems**, and
- a top 3 ranking for your three favorite **non-relational database systems**

If you are lacking some inspiration, check out the Database of Databases[2] for a quite comprehensive selection of databases to choose from. Notice that you are free to choose any non-relational database system (including key-value stores, document stores, ...) which often have limited expressive capabilities. If you encounter a problem that cannot be solved using your database system you have to manually write a program that extracts the relevant data and computes the query problem.

One member of your group should submit all members (name and student number) and choices via Canvas before **Friday, 14th of October** at **23:59**. If you do not submit a group you will be randomly assigned to a group and a pair of databases.

## Part 1: Data Preparation

You may download the data for this project from the following link:
`https://dblp.org/xml/dblp.xml.gz`

For this section of the project, you will parse, clean and load the DBLP dataset into your database systems of choice. You are free to use any programming language, as long we can verify the steps you took to load your data (by for instance, sharing the script you used).

[1] `https://dblp.org`
[2] `https://dbdb.io`

The data itself is one large XML file containing entries that can be of a certain publication type such as `article`, `inproceedings`, and `proceedings` which respectively store journal articles, conference articles and proceedings (which are bundled conference articles appearing within a single conference)[3]. For more information about the structure of this file, you can look at DBLP's documentation[4].

For the problems we ask you to solve below, we are only interested in a certain subset of the data. Hence, you should only extract what is relevant. We now describe the data entities and attributes that we expect you to store:

- **Conference articles** with their DBLP key, authors, title, pages, year of publication, and booktitle (booktitles are used for the name of the conference);
- **Conference proceedings** with their DBLP key, editors, title, booktitle, publisher, volume, and year; and
- **Journal articles** with their DBLP key, author, title, journal, volume, number, and year.

The DBLP dataset is imperfect, make sure to take the following things into account. The DBLP file only contains ASCII, and therefore translates UTF-8 characters to their HTML-encoded variant. Some XML parsers may crash on this, so make sure to take this into account when parsing. You can always cross-reference by using the search functionality on the DBLP website, subsequently using the the "download record" functionality to output the XML for a specific entry. Futhermore, the DBLP key given to each entry in the file is supposed to be unique, make sure to remove any duplicate entries before querying.

## Part 2: Querying

Next, you will implement the queries listed in the table below for **both** of your database systems. The queries are ranked by difficulty, the more difficult a query, the more points it is worth.

| Difficulty | Query Problem |
|---|---|
| Easy | **E1:** Who is the publisher of the PODS conference proceedings? |
| Easy | **E2:** What are the titles of the articles that Martin Gröhe wrote in the Theory of Computing Systems journal? (Sort in alphabetic order) |
| Medium | **M1:** How many articles were published in the SIGMOD conference proceedings this year? |
| Medium | **M2:** How many articles were published in the oldest journal, and what is its title? |
| Medium | **M3:** What was the median amount of articles published for each year of the CIDR conference. |
| Medium | **M4:** In which year did the SIGMOD conference have the most papers with over 10 authors? |
| Medium | **M5:** Who were the most frequent editors for the PODS conference? How many times were they an editor? |
| Medium | **M6:** For the researcher(s) with the most overall (conference & journal) publications: to how many different conferences did they publish? |
| Hard | **H1:** For each researcher that published to the ICDT conference in 2020: Who was their most frequently occurring co-author (conference & journal)? How many times did they collaborate? |
| Hard | **H2:** Compute the Erdős number (Erdös in DBLP) of Dan Suciu (c.f., explanation below). |
| Bonus | **B1:** Invent an interesting query that incorporates a cyclic join. |
| Bonus | **B2:** Create your own recursive query that illustrates something useful. |

Note that you will only get points for the bonus problems **B1** and **B2** if you have completed all

[3] This terminology is inspired by the BiBTeX format http://www.bibtex.org/Format/.
[4] https://dblp.uni-trier.de/xml/docu/dblpxml.pdf

non-bonus problems. **Make sure to add comments to your queries where possible, and pay attention to code quality, this will have an impact on your grade.**

- For problem **M2** the dataset uses journals with the year marked as 0 to denote unknown or missing journals, these are not supposed to be included as answers to this query.
- For problem **H2**, the Erdős number is the minimal collaborative distance between the mathematician Paul Erdős and some author. Notice that some authors may have never collaborated with a co-author that has a finite Erdős number. For these authors, the query will likely run a long time before concluding there is no path. For more information, check out the wikipedia page on the Erdős number[5].

### Intermediary Feedback (Optional)

We hold an optional feedback session on **Wednesday, 16th of November**. Here, you can ask the assistants for feedback with regards to your data preparation methods (Part 1) and the implementation of your queries (Part 2).

## Part 3: Benchmarking

For this part, you will benchmark the latency each of your implemented queries for each system. That is, you store a timestamp `start` before the query is sent and store a timestamp `stop` when the answer is fully returned. The latency is then `stop - start`. Notice however, that we are interested in the latency of an **online** system in the sense that you should assume data can be updated frequently. Therefore, if you choose to materialize all queries, you should include the time required to materialize the query in your benchmark. Make sure to show the amount of time spent materializing clearly, as this is latency you pay for every time materialized data is updated. You will have to take into account a number of things:

- Utilize the facilities a database system gives you to speed it up. Make sure that you are using index structures and constraints to speed up queries where you can.
- Make sure to execute and time each query on the same computer, your benchmark results will only make sense in the context of the same machine.
- Execute and time the same query *at least* twenty times, and interleave the executions of each query with each other. For example, it is better to benchmark: **E1**, **E2**, **M1**, ... for twenty times in a row compared to **E1** twenty times followed by **E2** twenty times.
- When using database systems (such as BigQuery, Hive, HBase, Neo4J, ...) implented in a VM-based language (such as Java, Scala, C#, ...), add 10 extra executions to your benchmark and throw away the first ten. This is because VM warmup may negatively impact latency in the beginning.
- When plotting your benchmark data, make sure you show the distribution of your benchmark. For instance by using a boxplot or by explicitly showing a histogram for all of your query executions.
- You can always do the benchmarks on multiple computer systems available to your group, this may identify potential problems with your benchmark.

You will discuss these benchmarks in the report. The goal is to optimise for **the best you can do within the system at hand,** and not the best possible performance irrespective of the database system. In short, how well do your queries perform **in the context of your systems**?

[5] https://en.wikipedia.org/wiki/Erd%C5%91s_number

## Report

Make use of the following ACM LaTeX-template for your report:
https://www.acm.org/publications/proceedings-template

Use the `acmart` document class with the options `nonacm`, `sigconf`, and `screen`. The report is limited to a maximum of 8 pages excluding code listings, figures, references and an appendix (for your raw benchmark figures, if desired).

We expect a decently written and structured report worthy of a Master student. Your report should mention the following things:

- Which database systems did you use? What are their main characteristics? What makes them unique or especially useful for a certain scenario?
- What was the division of work within your group?
- Discuss how you approached or solved the assigned query problems. Are there any particular implementation difficulties you ran into (for a certain system)?
- Do a comparative study of your benchmark results for both database systems. Is there one database system that excells where another one falls short? Why would this be the case? Relate this to the characteristics of your database systems (look at query plans, did a given table have an index?, was it able to use cached data?, etc.). Explain to us why you think one system might be more suitable than another.

## Submission

Below is a short summary of all the deliverables that we expect for this project. You can submit it to our course's Canvas space as a single archive file `DaIM-group-<group_nr>.zip`. It is important that **a single member of your group submits the project**, agree beforehand who is responsible for this. The archive should contain the following files:

- **Data Preparation**: A folder called `data/` containing the scripts or source code of the programs used to clean and load the dataset into your database systems.
- **Queries**: Two files containing your implemented queries for both database systems.
- **Report**: A pdf document with the filename `report-group-<group_nr>.pdf`.

The deadline for this project is **Friday, 23rd of December** at **23:59**.

## Evaluation Policy

This project is **40%** of your final grade, late submissions amount to **0/20** for the project.

We would like to emphasize that **plagiarism is not tolerated under any circumstances**, suspected cases will be reported to the dean of the faculty immediately (c.f., Article 118 of the Teaching and Exam Regulations[6]). Note that plagiarism includes copying queries from other students, using queries from the web without attribution, utilizing an AI-based paraphrasing tool to mask your plagiarism (trust me, it shows), and anything else that can be considered plagiarism under §2 of article 118. Sanctions for plagiarism (see §5) can include 0 for the course, exclusion from the examination period, expulsion from the university, and prohibition from re-enrolling.

[6] https://www.vub.be/sites/default/files/2022-09/2022_Reglementen_OER_2022-2023_ENG.pdf