SUSTECH CS307 Fall 2024 - Project Part I

> **Contributors**: Liu Tengfei, Li Zhaohan, Liu Jinrun, Zhou Xingyu, Gong Mingdao, Wang Zhong-Qiu, Wang Weiyu
>
> The final interpretation of this project belongs to the course group.

General Requirements

- A group project with only **2 teammates** who are better in the **same lab session**. Each group should finish the project independently and submit **only one report**. If a group submits multiple reports by different members, we will only choose one randomly. If a group submits multiple reports by only one member, we will only score the latest one.
- The teammate(s) you choose for Project I will also be your teammate(s) for Project II. It is not allowed to change teammates once paired. The teammates in one group will share the same score. In other word, there is **no contribution ratio**. Be very careful when choosing your teammate. You don't want to have a teamate who is always **coasting** and couting on you to finish the project.
- Submit the report by the deadline. **Late submissions will receive a score of zero**. You'd better submit at least one version before the deadline.
- **Do NOT copy ANY sentences and figures** from the Internet and your classmates. Plagiarism is strictly prohibited and will have severe consequences.
- The number of pages for your report should be **between 8 and 20**. Reports shorter than 8 pages will receive a penalty in score. Nonetheless, reports longer than 20 pages will NOT earn you a higher score.

Database management systems (DBMS) are extremely useful when we need to deal with massive data. It can help us manage them in a convenient manner and improve the efficiency of both retrieval and modification. Hence, your work in Project I is composed of the following parts:

1. Find the inherent relationships of the provided data and then **design an E-R diagram** based on the relationships you found.
2. **Design a relational database** using PostgreSQL according to the provided data file and your E-R diagram.
3. **Import all data** into the database as fast, automatically and efficiently as possible.
4. Compare the performance of data operations (including insert, delete, update and query) between database and raw file I/O in a programming language. **The ONLY allowed programming language is *Java***.

Section 1 - Background

This project is about the structure of a publication management system named Scholarly Universe System to Check (SUSTC), which is similar to Google Scholar. The provided data contains basic information about articles, and it is complete (that is, all the cited articles of an article can be found in the provided data). The data is an excerpt of the full NIH NLM PubMed 2024 Baseline. For those who are curious about the pre-processing step which leads to this dataset, see scripts here.

The data is given as a `.ndjson` file, in which *each line* is a valid JSON object *but the whole file isn't*. You cannot parse the file directly into a JSON array.

Be sure to parse the `.ndjson` file as-is. When we say something is "an array", it means the `.ndjson` gives this field as an JSON array, and you should parse and store it *plurally*; When we say something "may not exist", it means if you access it from its parent object, it may fail because the key and the value may do not exist *at all*.

Data Description

- **`id`** : **unique identifier**, is a 1 to 8-digit accession number with no leading zeros.
- **`title`** : the entire title of the article.
- **`pub_model`** : use to identify the medium/media in which the cited article is published. There are five possible values for pub_model : 'Print' , 'Print-Electronic' , 'Electronic' , 'Electronic-Print' , 'Electronic-eCollection'.
- **`date_created`** : year-month-day.
- **`date_completed`** : *may not exist*. Format: year-month-day.
- **`keywords`** : *an array that may not exist*. Describe the content of the article.
- **`journal`** : contains `id`, `country`, `title` (the full journal title), `issn` (just like ISBN, ISSN is an eight-digit serial number used to uniquely identify a serial publication) and `journal_issue` (including `volume` and `issue`, *any of them may not exist, and the whole `journal_issue` field may not exist as well*).
- **`authors`** : each author is represented *either* as a `collective_name` *or* (`fore_name`, `last_name` , `initials`, `affiliation`), in the latter case, the entry contains at least `last_name`, but other fields may not exist. Note that `affiliation` *is an array*.

> **Duplication handling**
>
> - If an author *only* has `collective_name`, it is considered duplicated with another author that has the same `collective_name`.
> - If author A *only* has `last_name`, it is considered duplicated with another author B that *only* has `last_name` AND that `last_name` is the same.
> - If an author has `(fore_name, last_name, initials)`, it is considered duplicated with another author that has the same triplet.
>
> Note that `affiliation` is not considered during duplication handling.
>
> Hint:
>
> - There is a notoriously counterintuitive design of SQL, where two `null` are always considered *not equal*, which leads to `UNIQUE` constraints accepting two same `(null, 'Alan Turing', null)` records. You may need to store empty values as `''` (empty string) instead of `null` to circumvent this. As a safeguard, **make sure your constraints work as intended *before* actually importing data!**
> - Submissions with `collective_name` stored into `last_name` are *accepted* with the condition that a new column should be added to indicate the stored `last_name` is actually a `collective_name`.

- `publication_types` : *an array that may not exist*. Each contains `id` and `name`. for example `{"publication_types":[{"id":"D016428","name":"Journal Article"}, {"id":"D013487","name":"Research Support, U.S. Gov't, P.H.S."}]}`
- `grant`: *an array that may not exist.* Each grant contains `grant_id`, `acronym`(2-letter grant acronym), `country` and `agency`, in which *only* `agency` *is guaranteed to exist*.
- `article_ids` : *an array that may not exist*. different types of record ID attached to one article, contains `type` and `id`. `PMID` is one record type which named `pubmed` here. For example, one `article_ids` may be `[{'ty':'pubmed','id':3}, {'ty':doi,'id':'10.1016/0006-291x(75)90498-2'}, ...]` in which there are two different types, namely `pubmed` and `doi`, listed.
- `references` : *an array that may not exist*. include references to the article, each in the form of `id`.

Section 2 - Task and Report Requirements

2.1 Group Information

You need to write down the **membership information**, name and SID.

2.2 Task 1: E-R Diagram (15%)

Make an E-R Diagram of your database design with any diagram software. Hand-drawn diagrams will NOT be accepted. Please follow the standards of E-R diagrams (e.g., graphic representation, mapping cardinality, ...). In the report, you must provide a snapshot or an embedded vector graphics of the E-R diagram. Also, please specify the name of the software you used for drawing the diagram.

2.3 Task 2: Database Design (25%)

Design the tables and columns based on the background supplied above. First, you shall generate the database diagram via the "Show Visualization" feature of DataGrip and embed a snapshot or a vector graphics into your report. Then, briefly describe the design of the tables and columns including (but not limited to) the meanings of the tables and columns.

In addition, submit an SQL file as an attachment that contains the DDLs (i.e., statements for creating tables) for all the tables you created. Make it into a separate file but do not copy and paste the statements into the report.

Some notes:

1. Design a database by **PostgreSQL** allowing us to manage all the information mentioned above.
2. Your design needs to follow the requirements of the **three normal forms**.
3. Use **primary key** and **foreign key** to indicate important attributes and relationships about your data.
4. Every row on each table should be uniquely identified by its primary key (you may use simple or composite primary key).
5. Every table should be included via one link or multiple links. **No isolated tables** can be included. (每个表要有外键，或者有其他表的外键指向)
6. Your design should contain **no circular links**. (对于表之间的外键方向，不能有环)

7. Each table should always have at least one mandatory (“Not Null”) column (including the primary key but not the id column).
8. Use appropriate types for different attributes.
9. Your design is as easy to expand as possible.

2.4 Task 3: Data Import (28% + 3%)

In this task, you should **write a script** to import the data into the database you designed. After importing the data, you should also make sure **all the data is successfully imported**.

In the report, you are required to accomplish the **basic requirements** (18% out of 28%):

1. Write a script to import the data file.
2. Describe in your report how your script imports data. Clearly state the steps, necessary prerequisites, and cautions to run the script and import data correctly. Provide the number of records in each Entity table.

You may also finish the following **advanced tasks** to get the remaining points (10% out of 28%):

1. Find more than one way to import data and provide a comparative analysis of the computational efficiencies between these ways.
2. Optimize your script. Describe how you optimize it and analyze how fast it is compared with your original script.

For the advanced tasks, please make sure to describe your test environment, procedures, and actual time costs. You are required to write a paragraph or two to analyze the experiment results. You may refer to the requirements for reporting experimental results in Task 4 for details.

If you do in-depth exploration outside of the Lab in data import and get exceptionally good efficiency, we will give you **bonus points** (1%~3%).

2.5 Task 4: Compare DBMS with File I/O (30% + 7%)

In the report, you are required to finish the following **basic requirements** (20% out of 30%):

1. A description of your test environment, including (but not limited to):

   a. Hardware specification, including the CPU model, size of memory.

   b. Software specification, including the version of your DBMS and operating system, the programming language you choose, and the development environment (the version of the language, the specific version of the compilers and libraries, etc.).

   c. When reporting on the environment, think about the following question: if someone else is going to replicate your experiments, what necessary information should be provided for him/her?
2. A specification of how you organize the test data in the DBMS and the data file, including how you generate the test SQL statements and what data format / structure of the files are.
3. A description of your test SQL script and the source code of your program. DO NOT copy and paste the entire script and the program in the report. Instead, please submit source codes as attachments.

4. A comparative study of the running time for the corresponding statements / operations. You are encouraged to use data visualization to present the results. Be sure to use consistent style for graphics, tables, etc. Aside from a list / figure of the running times, you are required to describe the major differences with respect to running performance, what you find interesting in the results, what insights you may show to other people in the experiments, etc.

In addition to the basic requirements, you can also think about some of the following **advanced tasks** (but not limited to the following ones) to challenge yourself and get the remaining points. (10% out of 30%)

1. Can your database deal with high concurrency? You may try to achieve the above benchmarks in a higher order of magnitude, such as hundreds of thousands of selections.
2. Can you compare the performance with different database software (e.g., MySQL, MariaDB, SQLite), file systems, disk performance and type, programming languages, libraries, or operating systems?

If you do in-depth exploration outside of the Lab in manipulating database, and get exceptionally good efficiency, you can earn the **bonus** (1%~7%).

Section 3 - Submission (2%)

Submit a report named “Report_sid1_sid2.pdf” in **PDF format** and **a .zip archive** of necessary attachments (such as SQL scripts and source code files) on the Blackboard website before **23:59 on November 10th 2024**, Beijing Time (UTC+8). For the report, replace the sid1/sid2 with your members’ student ID. For attachments, please put them into separate directories based on the task and compress them into a .zip archive.

Section 4 - Disclaimer

The characters, businesses, and events in the background of this project are purely fictional. The items in the files are randomly generated fake data. Any resemblance to actual events, entities or persons is entirely coincidental and should not be interpreted as views or implications of the teaching group of CS307.