Fall 2022 CS307 Project Part I

Contributors:

Leader and Overall Design: SUN Kebin

Data Preparation and Documentation: ZHANG Liyu, SUN Kebin, LENG Ziyang

Other Contributors: WANG Lishuang, Li Boyan, Lu Diyun

Review: WANG Weiyu

This project description is extended from the one of Spring 2022 CS307.

General Requirement

- This is a group project with only 2 ~ 3 teammates who are in the same lab session. Each group should finish the project independently and submit only one report.
  - The teammate you select for Project I will also be your teammate for Project II. It is not allowed to change teammates once paired.
- You should submit the report before the deadline. All late submissions will receive a score of zero.
- DO NOT copy ANY sentences and figures from the Internet and your classmates. Plagiarism is strictly prohibited.
- The number of pages for your report should be between 8 and 20. Reports less than 8 pages will receive a penalty in score, however, ones with more than 20 pages will NOT earn you more score.

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

4. Compare the performances of data retrieval and modification between database and raw file I/O in a programming language. The ONLY allowed programming language is Java.[1]

[1] It is not unacceptable to finish THIS project in other common programming language. Yet, since the project II of this course will involve automatic benchmarking, you will have to translate your code to Java by then.

Section 1 Background

This project is based on the requirements for a fictional international shipping organization named Shipments across Urbans and Seas Through Containers (SUSTC). In this project, you need to design a database for the Management Department of SUSTC. It will be used for storing the organized structure of the staffs and the details of the deliveries logged in SUSTC.

As a reference in your design process, we provide the structure of the staffs and companies who will log in SUSTC, the objects which may be managed by SUSTC, as well as data samples that will be stored in the database.

1.1 Targets That SUSTC Manages

The courier service companies who want to access to the interior information of SUSTC will join in SUSTC. However, since SUSTC is an organization oversea, the companies without international services will not be accepted. Therefore, there will not be many logged companies.

The couriers, on the contrary, can log themselves in SUSTC as a staff if they work for one of the companies logged in SUSTC. Part-time couriers are not considered. Hence, one courier can only have one servicing company.

The ships can be logged in SUSTC as devices by their owning courier services companies.

The containers, instead, do not belong to any company and are only used on ships to pack the items to same size. Yet, they are also tracked to make sure no loss or damage is made during the deliveries.

1.2 Data Description

The given `.csv` file includes 500,000 records. Here is the explanation of the columns of this `csv` file:

- Item Name: the unique name of the shipped item.
- Item Class: the type (or class) name of the corresponding item.
- Item Price: the actual price of the corresponding item.
- Item Retrieval City: the delivery start city that the retrieval courier picks up the corresponding item.
- Retrieval Start Time: the delivery start time.
- Retrieval Courier: the retrieval courier's name who picks up the item.
- Retrieval Courier Gender: the retrieval courier's gender.
- Retrieval Courier Age: the retrieval courier's age.
- Retrieval Courier Phone Number: the retrieval courier's phone number.
- Delivery Finished Time: the finishing time of the whole delivery (from the start city to the target city). Can be EMPTY if the delivery is not finished.
- Delivery City: the target city's name of the delivery.
- Delivery Courier: the name of the courier who delivers the item. Can

be EMPTY if the item has not arrived at the target city.

- Delivery Courier Gender: the gender of the courier who delivers the item. Will be EMPTY if "Delivery Courier" is EMPTY.
- Delivery Courier Age: the age of the courier who delivers the item. Will be EMPTY if "Delivery Courier" is EMPTY.
- Delivery Courier Phone Number: the phone number of the courier who delivers the item. Will be EMPTY if "Delivery Courier" is EMPTY.
- Item Export City: the export city's name of the item of this record.
- Item Import City: the import city's name of the item of this record.
- Item Export Tax: the export tax of the item of this record.
- Item Import Tax: the import tax of the item of this record.
- Item Export Time: the export time of the item of this record. Can be EMPTY if it has not been exported.
- Item Import Time: the import time of the item of this record. Can be EMPTY if it has not been imported.
- Container Code: the unique code to identify the container that is used to store the item. Will be EMPTY if "Item Export Time" is EMPTY.
- Container Type: the type of the container that is used to store the item. Will be EMPTY if "Item Export Time" is EMPTY.
- Ship Name: the unique name to identify the ship that is used to ship the item. Will be EMPTY if "Item Export Time" is EMPTY.
- Company Name: the name of the company that manages this shipment. The retrieval and delivery couriers and the ship are all part of it.
- Log Time: the last update time of this record.

1.3 Notices

The general route of a delivered item is: Retrieval City → Export City → Packing and Shipping → Import City → Target City.

Some of the columns can have EMPTY values and they are all connected, please do NOT ignore the relations between them. In the meantime, we assume that one courier has only one working cities for simplicity.

It is not mandatory that you design a table to manage the item specifically, since a delivery table if often enough. Also, you may treat inland and port cities equally in your E-R diagram (you may split them into different tables in your database design). Similarly, you are not required to express the container-ship relation in your design since they are not bounded. Whether your design contains them or not, there shall be $1-n$ and $m-n$ relations in the database at least. Please try your best.

Even if you do not intend to finish the advanced tasks in Task 4, it is NOT okay to ignore the ships, containers, or cities in your database design. Otherwise, there will be a penalty in score of Task 1 & 2.

Section 2 Requirements

Notice at the beginning: Some tasks consist of basic and advanced requirements. You may not get full points if you only meet the basic ones.

2.1 Basic Information of Group and Workloads

1. Names, student IDs, and the lab session of the group members.
2. The contributions and the percentages of contributions for each group member. Please clearly state which task(s)/part of the task(s) is/are done by which member in the group.
3. If you failed to link a task or its sub-tasks to one of the group members, we will NOT count the score for the part you miss (since we do not know who accomplished this task).

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

Make an E-R Diagram of your database design with any diagram software. Hand-drawn results will NOT be accepted. Please follow the standard of E-R diagrams. In the report, you are required to provide a snapshot or an embedded vector graphics of the E-R diagram (15% out of 15%). Also, please specify the name of the software/online service you used for drawing the diagram.

2.3 Task 2: Database Design (25%)

2.3.1 General Requirements

Design the tables and columns based on the background provided 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 tables and columns. (25% out of 25% for both combined)

In addition, please submit an SQL file as an attachment that contains the DDLs (`create table` statements) for all the tables you created. Please make it into a separate file but not copy and paste the statements into the report.

2.3.2 Detailed Notices

1. All data items should base on the file `shipment_records.csv`.
2. Your design needs to follow the requirements of the three normal forms.
3. Use a primary key and foreign key(s) to indicate important attributes and relationships about your data.
4. Every row in each table should be uniquely identified by its primary key. (You may use a simple or a composite primary key.)
5. Every table should be involved in a foreign key. Isolated table is NOT allowed.
6. Your design shall NOT contain circular foreign key links. （表之间的外

键方向不能成环。例如：A 表有外键关联 B 表，B 表有外健关联 C 表，C 表有外健关联 A 表，这是不允许的。）

7. Each table should contain at least one mandatory (“Not Null”) column (including the primary key but not the ID column).
8. Other than the system-generated self-increment ID column, there should be at least one column with the “unique” constraint.
9. You should use appropriate data types for different fields.
10. Your design should be easy to expand if requirements changed.

2.4 Task 3: Data Import (25%)

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

In the report, you are required to accomplish the basic requirements (15% out of 25%):

1. Write a script to import the data file.
2. Write a description in your report of how your script import data. You should clearly state the steps, necessary prerequisites, and cautions to run it and import data correctly.

You may also finish the following advanced requirements to get the remaining points (10% out of 25%):

1. Find more than one way to import data and provide a comparative analysis of the computational efficiencies between these ways.
2. Try to optimize your script. Describe how you optimized 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.

2.5 Task 4: Compare DBMS with File I/O (35%)

2.5.1 General Requirements

In this task, you are required to compare the performance of data retrieval and manipulation between database APIs and file APIs in a programming language. Please conduct the comparative analysis according to the following steps:

1. Benchmarking with database APIs: Based on the database you created, you are required to write a program in Java that accesses the database via database APIs and contains a series of `INSERT`, `DELETE`, `UPDATE`, and `SELECT` statements. You may specify the number of statements in each type on your own and decide which data to be modified and read. However, the operations of each statement type cannot be too small to

fail illustrating the strength and weakness of database APIs over file I/O (depending on the programming language, tens of thousands of records each would usually be enough). Finally, you need to record the running time of each statement type or each statement. Here, we provide some typical test descriptions in database you can refer to:

a) `INSERT`: First, randomly drop out some rows of the `csv` file and import it into your database. Then evaluate the time cost of importing rest of the rows of the `csv` file.
b) `DELETE`: First, import all data of the `csv` file into your database. Then, evaluate the time cost of deleting arbitrarily chosen rows of your database's delivery record table.
c) `UPDATE`: First, import all data of the `csv` file into your database. Then, evaluate the time cost of update all the EMPTY values of your database's delivery record table to arbitrary values.
d) `SELECT`: First, import all data of the `csv` file into your database. Then, evaluate the time cost of finding all delivery records that have not finished or finding all the delivery records that had been packed by an arbitrary container, etc. You may pay more attention on `SELECT` statement tests since it is used more frequently than other ones in many real-world scenarios.

2. Benchmarking with file APIs: This step is designed to replicate all your operations in the first step, but via a generic programming language's standard file APIs. First, create file(s) that store(s) the same data as you have in the tables in the DBMS. Then, write a program to insert, delete, update, and find the data items as you did in the SQL statements and queries. Be sure that the file operations (and the number of operations) are identical to the SQL operations (and the number of the statements). Finally, record the running time of each operation (type) as in database API benchmarks.
3. Comparative analysis: Compare the recorded running time of the same operation/statement from the DBMS and the file, respectively. You may conduct comparisons from multiple levels, such as comparing statements with corresponding operations (statement-level) or comparing the total time of all statements in a specific type with the corresponding operation type (type-level).

2.5.2 Detailed Basic Requirements

In the report, you are required to finish the following basic requirements (20% out of 35%):

1. A description of your test environment, including (but not limited to):
   a. Hardware specification, including the CPU model, size of memory, whether you are using a solid-state disk (SSD) or hard disk drive (HDD), how fast your disk is (sequential and random).

   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 the environment, you can think about this question: if someone else is going to replicate your experiment, 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 do 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. Besides a list/figure of the running time, 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.

Some notes on how to finish this task in a better way:

1. You can perform the above benchmarks with different orders of magnitude of statements/operations (e.g., from hundreds to thousands to ten(s) of thousand(s)).
2. You can choose or design any format you want to store data in the file, such as plain text formats (CSV, JSON, XML, etc.) or a self-defined binary format.
3. Please only stick to standard file APIs, i.e., the **`java.io`** in Java. The only exception is that if you choose to use JSON and XML, you may utilize third-party JSON/XML libraries if standard library does not provide it, e.g., **`Gson`**.[2]
4. We acknowledge that there are numerous libraries that can facilitate the data manipulation works or even speed up the performance of insertions and selections significantly (e.g., **`pandas`** in Python). You are encouraged to also compare the performance of these libraries with DBMS. However, you should conduct the analysis of DBMS vs. standard file APIs beforehand.
5. Some useful resources:
   a. Advantage of database management system over file system

[2] If you insist on using another language, this rule still holds. For example, you shall stick to **`iostream`** and **`fstream`** in C/C++, the **`file`** object in Python, or the **`System.IO`** in C#. For JSON and XML, you may utilize **`json`** package for Python for instance. As for languages like C# whose standard library provides **`System.Text.Json`** and **`System.Xml`**, you shall not use replacements.

b. Advantages of Database Management System
c. Characteristics and benefits of a database

2.5.3 Detailed Advanced Requirements

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. (15% out of 35%)

1. If SUSTC wishes to figure out which containers have serviced (the time that the container being on a ship) for several years (depending on type) to inform the ports to perform maintenance, can your database deal with such problem? How about your file I/O program? If so, which will perform well? If not, why?
2. Can you identify the retrieval and delivery couriers who collect/send the greatest number of items for each company in each city through your database? How about your file I/O program? If so, which will perform well? If not, why?
3. If a company have different type of items (for example, half of the "Item Class" columns' distinct values) that will be exported, which cities it shall choose respectively to reach the minimal overall export cost? Can your database deal with such problem? How about your file I/O program? If so, which will perform well?
4. Can your database deal with high concurrency? You may try to perform the above benchmarks in a higher order of magnitude, such as hundreds of thousands of selections.
5. 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?

Section 3 How to Submit

Submit the report in PDF format with necessary attachments (such as SQL scripts and source code files) on the Sakai website before 23:00 on 30th October 2022, Beijing Time (UTC+8). 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.