Final Project: Design and Implementation of a Database System

Introduction

This final project for the database course consists of three main parts: the design and implementation of a simple relational database, the design and implementation of an ETL (Extract, Transform, Load) process, and finally, the design and implementation of a data warehouse. Each part is described in detail below.

Definitions
Data Warehouse: A repository that collects data from various sources and files. The purpose of this collection is to use this data together to provide business analytics for the organization.
ETL (Extract, Transform, Load): A process used during the construction of a data warehouse to extract data from different sources, transform it as needed, and load it into the data warehouse. This operation can be performed by commercial tools designed for this purpose, stored procedures, functions, queries, or scripts written in common programming languages.

Relational Database Design (30 Points)

In the first part of this project, you are required to design a relational database with the following requirements. Write the queries for its creation and draw its data model, attaching it as an appendix. Note that the designed database must be in 5th Normal Form (5NF).

A library has several books. Some books may have more than one copy available. Each book has an ISBN (International Standard Book Number), title, description, edition number, publication date, and publisher's name. Each book can have one or more authors, belong to one or more genres, be written in one or more languages, and may have one or more translators.

Each library member has personal details such as name, date of birth, membership date, membership number, contact information, and address. A member can borrow one or more copies of one or more books for a specified period, during which those copies cannot be borrowed by others.

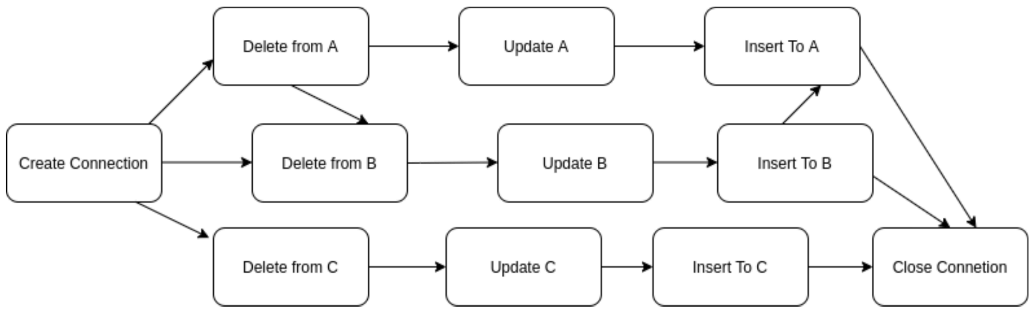
ETL Design and Implementation (40 Points)

In the second part, you will create an ETL pipeline. The ETL pipeline should connect two databases of the same type and transfer data from the source database to the destination database. Each table in the source database must have a corresponding table with the same name in the destination database, and after the transfer, they should be identical.

The data transfer in this pipeline must be performed record by record. You are not allowed to completely wipe the destination database and copy the source database over because you might want to apply transformations on the records or the data volume may not permit it. For transferring data from each table in the source database to its counterpart in the destination database, consider the following:
- Records deleted in the source database must be deleted in the destination database.
- Records updated in the source database (with the primary key remaining unchanged but one or more other columns changed) must be updated in the destination database.
- New records inserted into the source database must be inserted into the destination database.

These records must be identified in each condition. Deleting all records from the destination database and copying all records from the source database in each ETL execution is not feasible for large data volumes and is not acceptable in this project.

You must also consider table dependencies. If table A has a foreign key to table B, a record cannot be added to table A without the corresponding record in table B being added first, and a record cannot be deleted from table B without the corresponding record in table A being deleted first. This requires creating a Directed Acyclic Graph (DAG) of fine-grained tasks to perform them in order. For example, in the following database, table A depends on table B, and table C is independent of both. Proper ETL execution with this DAG is possible. (For guidance: You can use topological sort to traverse this DAG.)



In various stages of the ETL implementation, to obtain the list of tables, types of keys, dependencies, etc., do not hard code them. Use appropriate queries to the database to retrieve them.
It is recommended to implement the ETL in **Python**. However, you may also implement it in C++, C, or Java.
At the end, you must submit your source code along with documentation explaining it.

Data Warehouse Design (30 Points)

In the final part of this project, we focus on one of the main requirements of data warehouse design: the ability to travel back in time. We want to know the state of the source database at a specific point in the past (including what records it contained). Therefore, you must:

- Know the exact time each record was added to the data warehouse.
  - This can be done by adding a column to each table for this purpose.

- Maintain records in the data warehouse when they are deleted or updated in the source database, transferring them to a history table with the same name in the data warehouse.
  - This should include when they were removed from the main table and the reason (deleted or updated).

These two conditions are sufficient for time travel. Explain how you can travel back in time in the data warehouse using these two conditions. Then write the queries for their creation and draw their data model, attaching it as an appendix.

At the end, you must submit a zip file containing three folders: Database, ETL, and Data Warehouse. Each of the Database and Data Warehouse folders must include a `schema.sql` file for the related queries and a `data_model.pdf` file for the data model diagram. The Data Warehouse folder should also include an `answer.pdf` file containing your explanation for the question posed. The ETL folder must include the source code and a `pdf` file explaining the code structure.