CPSC437 S24 - LAB 2

DUE: February 27, 2024, 11:59PM Eastern

SUBMISSION NOTES: Your submission will include the requested output files to Gradescope.

1 Prerequisites

This lab assumes you have a working installation of Python and PostgreSQL as shown in Lab 0. Create a new database for the purpose of this lab called `library`.

Task 1: Creating a Schema File

In `task 1`, you are tasked with translating an Entity-Relationship (ER) diagram into a PostgreSQL schema file. The ER diagram provided in Figure 1 outlines the structure of a library database, including entities such as books, authors, publishers, and students, along with their relationships. Your objective is to define the PostgreSQL tables in a file named `schema.sql`, reflecting the structure and relationships depicted in the ER diagram.

To handle the multi-valued attribute `phone_number` in `student`, create a table `phone_number` linking a `student_id` with a `phone_number`. To handle the composite value `address` in `student`, simply flatten them.

Deliverable: Completed `schema.sql` and `task_1.py`.

Task 2: Writing a Seed Script

`Task 2` involves writing a Python script to seed the database tables you have defined in Task 1 with initial data. Your script should populate the database in a way that reflects a realistic initial state of a library system. Specifically, you will:

1. Insert data for 3 publishers.
2. Add 20 authors.
3. Create 200 book entries.
4. Tie each book to an edition.
5. Register 100 students.
6. Record 500 borrow transactions, associating books with students.

Implement the `seed` function within the in the file `task2_seed.py`. You may find the `execute_insert` method of `DatabaseActions` helpful. Ensure that your seeding script generates data that adheres to the constraints and relationships defined in your schema.

Deliverable: Completed `task2_seed.py`.

Task 3: Advanced SQL

In `task 3`, you will create a database trigger and implement some advanced SQL queries.

Queries

Complete the below queries in `task3_sql.py`.

1. List the top 5 most borrowed books in the library. The result should include the books title and number of times borrowed, ranked from most borrowed to least borrowed.

2. For each month, calculate the total number of books borrowed and the average duration (in days) of a borrow. Display the months in a year-month format (YYYY-MM) and order by the month ascending. Some notes:

   - If a book has not been returned yet, it should not be included in the average duration.
   - If a book was borrowed in month X and returned in month Y, it should be included in the month it was checked out.

3. Identify publishers that frequently collaborate with specific authors, where "frequent collaboration" means publishers that have published more than three books by the same author.

Triggers

Next, we will create a trigger that automatically populates the fines table when a book is returned via the following steps:

1. Append to the `schema.sql` file the `fines` table:

```
CREATE TABLE IF NOT EXISTS fines (
    fine_id SERIAL PRIMARY KEY,
    student_id INT NOT NULL,
    book_id INT NOT NULL,
    days_overdue INT NOT NULL,
    fine_amount DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (student_id) REFERENCES student(student_id),
    FOREIGN KEY (book_id) REFERENCES book(book_id)
);
```

2. Implement the function `calculate_fine` in `trigger.sql`, which will be executed when the trigger is hit. Specifically, the function should insert into the fines table the student that is being fined, the book they are being fined for, the number of days the book is overdue, and the fine that is due. An outline of the function has been provided for you.

3. Add a trigger named `trigger_calculate_fine` which triggers after update of `return_date` on the borrows relation when the new rows return date is not null and the new rows return date is greater than the due date. The trigger should execute the function `calculate_fine`.

When the above items are complete, you can load the trigger and sql function into your database by executing:

```
psql -U yourusername -d library < trigger.sql
```

or by using the `execute_file` function on an instance of the `DatabaseActions` class. from your terminal or Power Shell in the same directory as the file.

Deliverable: Completed queries in `task3_sql.py` and completed SQL function + Trigger in `trigger.sql`.

Submission

Collaboration Policy

Assignments in this course are designed to assess your individual understanding and skills. While you are encouraged to engage in discussions with the instructor, teaching fellows, undergraduate learning assistants, or classmates to deepen your understanding of course-related material, it is imperative that the work you submit is entirely your own creation. This means that any code, solutions, or written responses must be developed and written by you without copying from others.

Collaboration on assignments, unless explicitly permitted, is not allowed. This includes not only direct copying but also closely mimicking the logic or structure of another person's work. If you find yourself benefiting substantially from a discussion or an external resource, it is your responsibility to clearly acknowledge this in your submission, detailing the nature and extent of the assistance received. However, remember that your final work must still be predominantly a product of your own efforts.

As a reminder, plagiarism in any form is a serious violation of academic integrity. This includes the unauthorized use of AI tools like ChatGPT, online forums, code repositories, or any other sources that provide direct solutions. Seeking general advice and debugging help is permissible, but the line between assistance and infringement of academic integrity should not be crossed. If in doubt, consult the instructor or teaching staff for guidance on acceptable use of external resources.

Ultimately, the goal is for you to develop and demonstrate your own skills and knowledge. Adhering to these guidelines ensures a fair and meaningful educational experience for everyone involved.

Assignment Submission

Submit the following files to Gradescope:

- `schema.sql` with the table definitions drawn from the ER diagram and the `fines` table given in `Task 3`.
- `task1_create.py` (implemented for you) which is used by the autograder.
- `task2_seed.py` with your seeding implementation.
- `task3_sql.py` which implements the requested SQL queries.
- `trigger.sql` which implements the SQL function `calculate_fine` and declares the trigger.

Grading

The assignment is worth **30 points**. The grade will be broken down as:

Code Quality and Organization (8 Points) The clarity and organization of your code are crucial for this assignment. A well-commented solution that succinctly explains the use of variables, functions, and any complex approaches is required. The code itself should be intuitively structured and easy to understand, ensuring that graders can follow your logic and methodology without difficulty.

Code and Output Files (22 points):

1. Task 1 (6 points): The schema will be graded for correctness against the ER diagram.
2. Task 2 (6 points): Any sensible seeding script that has at least as many entries as requested in each table will be given full points.
3. Task 3 (10 points):
    - The queries are worth 3 points (1 point per query).
    - The SQL function is worth 3 points.
    - The trigger is worth 4 points.

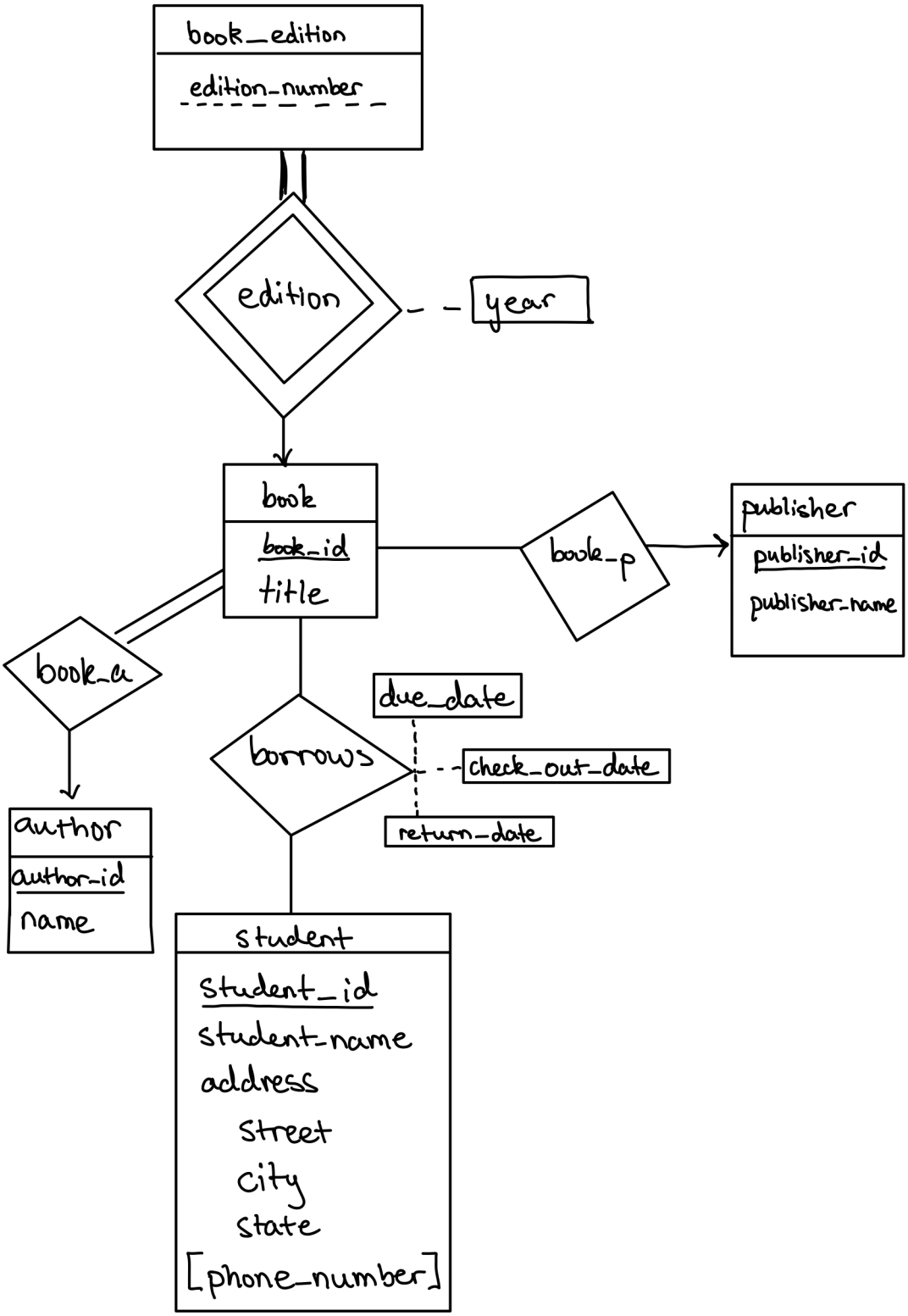

Figure 1: Library ER Diagram