INFS2200/INFS7903 Assignment 1, 2024

Total Marks	20 marks (20% of the course)
Due Date	4:00 pm, 6th September 2024
What to submit	Report file and SQL script file
Where to submit	Electronic submission via Blackboard
Assignment Version	1.2 (updated 27th August)

The goal of the project assignments is to gain practical experience in applying several database management concepts and techniques using the PostgreSQL DBMS. In particular, this assignment mainly focuses on improving database efficiency with views, indexing, and query planning.

Your main task is to first populate your database with appropriate data, then design, implement, and test the appropriate queries to perform the tasks explained in the next sections.

You must work on this project **individually.** Academic integrity policies apply. Please refer to *3.60.04 Student Integrity and Misconduct* of the University Policy for more information.

This document is in three sections: Section 1 describes the database schema for the assignment and provides instructions on downloading the script file needed to create and populate the database. Section 2 describes the tasks to be completed for this assignment. The final section describes the submission guidelines and marking scheme.

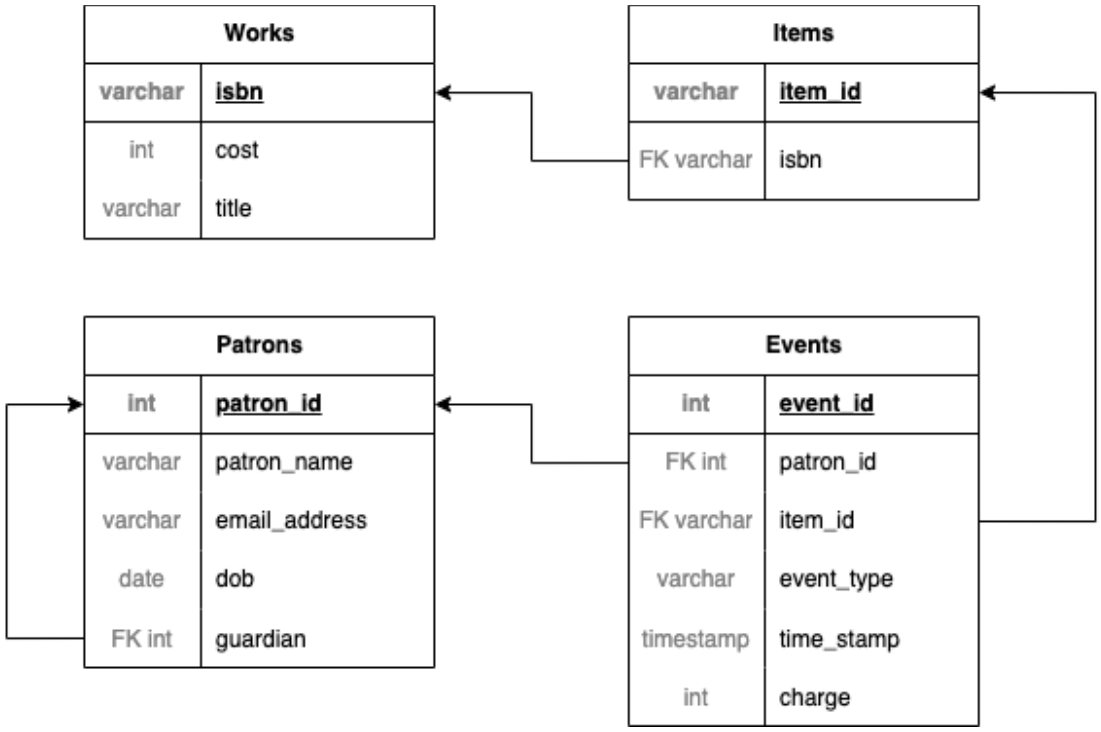
Enjoy the project!

Section 1 - The **LibraryDB** Database

You are a developer working on LibraryDB.

The LibraryDB database records information about items in the library, its patrons and circulation of those items. It has four tables:

- `Works` - records metadata shared between each item such as *title* and *author*;
- `Items` - records each individual item (copy of a Work) available for loan;
- `Patrons`- records information about each patron of the library;
- `Events` - records loans, returns, holds and losses of Items by/to Patrons.



Entity-Relationship diagram

Setup

Before starting work on this assignment, please create a new database in `psql`, connect to it with the `\c` command, and run the setup file `a1-2024-librarydb.sql` with the `\i` command (it may take a minute). This ensures that your work on this assignment is separate from your other work.

The setup file is available from Blackboard and on Manta storage:

```
$ mget -O /infs2200/stor/data/assignments/1/a1-2024-librarydb.sql
```

Section 2 - Assignment 1 Tasks

Q1 CHECK Constraint (1 mark)

Most of the column-level constraints for the schema are simple `NOT NULL`, primary-key or foreign-key constraints which are best expressed as part of the table definition. However, sometimes it is useful to *name* a constraint, which means it can be defined separately to the table.

There are four permitted values of **`event_type`:** *`Loan`, `Return`, `Hold`* and *`Loss`*.

Write a constraint (named **`CK_EVENT_TYPE`**) to enforce this requirement.

Q2 Constraints with triggers

`CHECK` constraints should only be applied to data with immutable conditions. For example, it is bad practice to formulate a check based on the current time. Instead, triggers may be used.

Libraries have underage patrons, but these patrons may not have their own contact address and are likely not legally responsible for their losses. Rather, their guardian is.

Q2.1 (2 marks)

> Write a trigger named `BI_GUARDIAN` and accompanying user-defined function named `UDF_BI_GUARDIAN` to ensure that all new patrons who are children (under 18 years old) have a guardian who is an existing patron.

Q2.2 (2 marks)

> Write a trigger named `BI_EMAIL_ADDR` and accompanying user-defined function named `UDF_BI_EMAIL_ADDR` to ensure that all new patrons who are adults (18 years and older) have an email address.

Q3 Generating keys with sequences and triggers

A simple way to achieve a unique value is with an incrementing integer sequence. In modern PostgreSQL, where an integer column type is appropriate, this behaviour can be achieved by specifying the column type as `serial` rather than `integer`. This has been done for the `Patrons` and `Events` tables.

However, the primary key for the Items will also be printed as its barcode. The Library staff would like the initials of the library (**`UQ`**) to be at the start of the barcode, followed by a ten-digit number, followed by a one-digit checksum.

The checksum is the sum (mod 10) of the other digits.

Q3.1 Sequence creation (1 mark)

> Create an integer sequence named `ITEM_ID_SEQ` for the Items table. The minimum value is `1000000000`, the maximum is `9999999999` and the increment is `1`.

Q3.2 Sequences and Triggers (2 marks)

> Create a trigger named `BI_ITEM_ID` and accompanying user-defined function named `UDF_BI_ITEM_ID` to populate the primary keys for new Items as described above.

Q3.3 Sequence identification (1 mark)

Write a query to list the Postgres-internal sequence(s) created via the use of `serial` on the `Patrons` and `Events` tables.

Q4 Business logic with triggers

Q4.1 Losses (2 marks)

Sometimes an item is permanently lost by a patron. When this happens, they need to be charged the cost of its replacement, which is stored in the `cost` field of the relevant row in the `Works` table.

Write a trigger `BI_LOSS_CHARGE` and accompanying user-defined function `UDF_BI_LOSS_CHARGE` to populate the `charge` field when a new `Loss` is inserted into `Events`.

Q4.2 Missing Returns (4 marks)

Sometimes, an item may be improperly returned (such that the `Return` event is not recorded in the database) and then re-borrowed by another patron. In this case, the library system needs to credit the previous borrower with its return.

Write a trigger `AI_MISSING_RETURN` (and accompanying user-defined function `UDF_AI_MISSING_RETURN`) that detects this situation on a new `Loan` event and subsequently inserts a `Return` for the previous borrower, timestamped to one hour earlier.

(3 marks for correctly inserting returns across three different circulation-history scenarios; 1 mark for not inserting spurious returns.)

Q4.3 Holds (5 marks)

Patrons may request a *hold* on an item. For this event type, the `time_stamp` should represent the *expiry* of the hold, rather than the time the hold was initially placed.

A hold may *only* be placed on an item if it is *[a]* not already held by any patron AND either (*[b]* available for lending, OR *[c]* currently on loan to a *different* patron). Otherwise, the hold must be rejected. (3 marks).

If a held item is currently on loan, the expiry of the hold is 42 days after its loan timestamp. Otherwise, the expiry of the hold is 14 days from the current time (2 marks).

Write a trigger `BI_HOLDS` (and accompanying user-defined function `UDF_BI_HOLDS`) to implement this functionality by rejecting the insert (if appropriate) or by setting the value of `time_stamp`.

Note: *the current time for the purposes of Q4.3 is given as the initial value of the* `time_stamp` *field. Do not use* `NOW()` *or similar. This is to allow for more reliable testing.*

Section 3 - Deliverables and Marking

There are 20 marks for functionality as listed in the task description.

Your assignment code will be tested with insertions and selections when relevant.

Triggers will be disabled when not being tested.

When your code must prevent an insertion, you should expect testing for false positives and false negatives.

Submission

Submit **two** documents to Blackboard by the due date (4pm, 6th September 2024).

Late submissions will be penalised unless you are approved for an extension (refer to Section 5.3 of the ECP).

Submit a **script** file titled **`4xxxxxxxx.sql`** (plain text with **`.sql`** extension) containing all SQL code you wrote for this assignment. Please do **not** include any `CREATE DATABASE`, `CREATE TABLE` or `psql` commands, as these will interfere with the testing procedure.

Submit a **PDF** file titled **`4xxxxxxxx.pdf`** containing a screenshot of your query/queries for Q3.3 and all results. Also, as a backup to the script file, include a screenshot of each constraint, trigger, function and sequence being created successfully.

Replace **4xxxxxxxx** with your student number for both files.

Your script file must be executable with the `\i` command in `psql`.

PSA: in Blackboard, remember to actually submit and not just save as a draft!

END OF ASSIGNMENT 1