THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

INFS2200/INFS7903 Assignment 2, 2024

Total Marks	20 marks (20% of the course)
Due Date	4:00 pm, 25th October 2024
What to submit	Report file and SQL script file
Where to submit	Electronic submission via Blackboard
Assignment version	1.0 (last updated 19th September 2024)

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: This section describes the database schema for the assignment and provides instructions on downloading the script file needed to create and populate the database. The middle section describes the tasks to be completed for this assignment. The final section describes the submission guidelines and marking scheme.

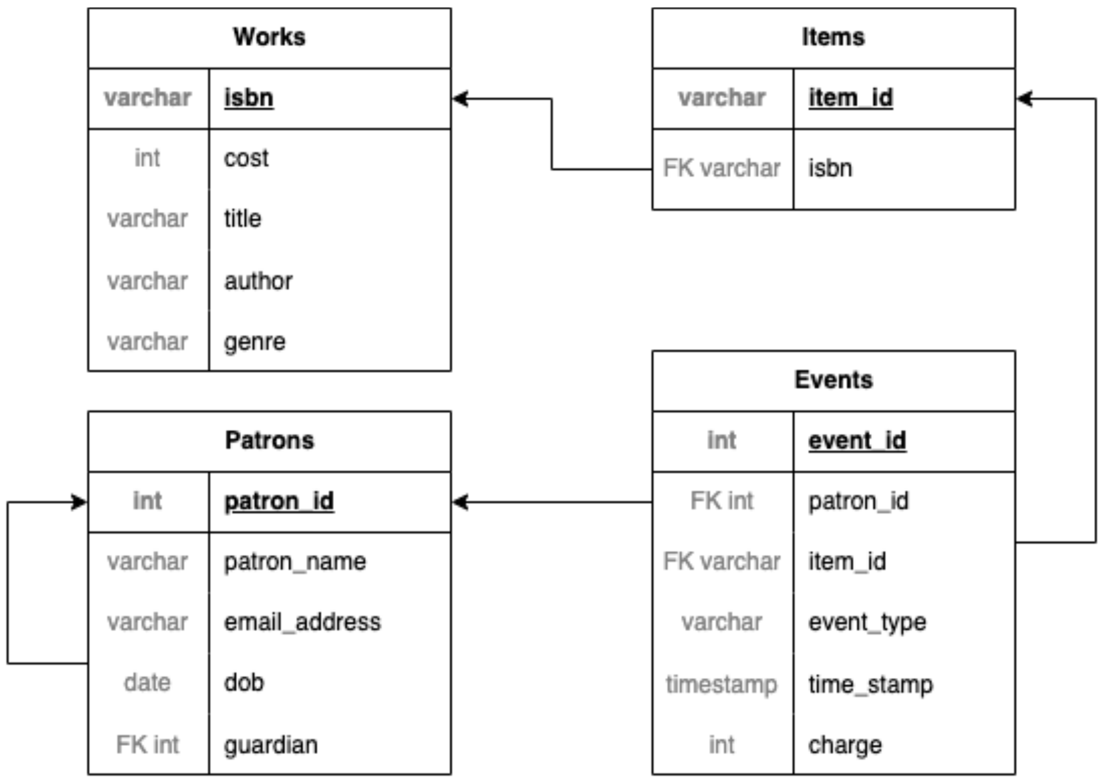
Enjoy the project!

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-Relation diagram

There are two schema additions from Assignment 1: `author` and `genre` fields have been added to the `Works` table.

Further, the trigger `BI_LOANS` and associated user-defined function `udf_bi_loans` have been added to the Events table. (No triggers implemented in Assignment 1 are present.)

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 `a2-2024-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 with `mget` as follows:

```
mget -O /infs2200/stor/data/assignments/2/a2-2024-librarydb.sql
```

Note: as of this version of the assignment, the setup script version is `2.0`

Section 2 - **Assignment Tasks**

Q1 Views (5 marks total)

Q1.1 View A (1 mark)

The librarians would like to confirm the most popular genres.

Create a view *V_POPULAR_GENRES* to determine the five most popular genres, as determined by number of times an item has been borrowed or held.

Query the view (***SELECT*** * ***FROM*** *V_POPULAR_GENRES;*) and report the results.

Report and explain the query plan. You should refer to the reported times, costs, the number of rows fetched, your knowledge of query planning (and anything else you feel is relevant).

Q1.2 View B (2 marks)

When a child loses an item, the cost is charged to their account but their guardian is responsible for paying it.

Create a view `V_COSTS_INCURRED` to identify the five patrons who are responsible for paying the most charges incurred from January through June 2024.

Hint: consider using Common Table Expressions to build up your query in a semantically meaningful way.

Q1.3 Materialised view (2 marks)

1. Create a materialised view using the same query as `V_COSTS_INCURRED`, called `MV_COSTS_INCURRED`. (0 marks)
2. Report, compare and explain the two following queries, which select the contents of the view and materialised view, respectively. (2 marks)

Query A:

```
SELECT * FROM V_COSTS_INCURRED;
```

Query B:

```
SELECT * FROM MV_COSTS_INCURRED;
```

You should refer to the reported times, costs, the number of rows fetched, your knowledge of query planning (and anything else you feel is relevant).

Q2 Indexes and performance (6 marks total)

Q2.1 Basic index (2 marks)

(1) Create an index (*IDX_EVENT_ITEM*) on the `event_type` and `item_id` fields of the `Events` table. (0.5 marks)

(2) Re-run your query from Q1.1 again (***SELECT*** ** **FROM** V_POPULAR_GENRES;*). Report the query plan. Compare this query plan to those from Q1.1 and explain the differences, if any. (1.5 marks)

Q2.2 Function-based index (4 marks)

Due to a previous decision in the database design, author's names are stored as a single field. However, for library purposes the *surname* of the author is relevant for deriving the *call number*. For this purpose, the surname is defined as the last word (space-delimited) in the `author` field.

(1) Create an expression to identify each author's surname and present the results. You may use string processing functions, including the regex functions. (1 mark)

(2) Report and explain the query plan. You should refer to the reported times, costs, the number of rows fetched, your knowledge of query planning (and anything else you feel is relevant). (1 mark)

(3) Create a function-based index to potentially speed up queries on this expression. (0.5 marks)

(4) Report the query plan with the function-based index in place. Explain the differences to the previous plan, if any. You should refer to the reported times, costs, the number of rows fetched, your knowledge of query planning (and anything else you feel is relevant). (1.5 marks)

Bonus content*:* in real libraries, books are assigned a call number derived from the subject matter and authors' surname. Books are stored on the shelves in call number order. This makes call numbers an example of a physical indexing scheme!

Q3 Indexes and query planning (4 marks)

PostgreSQL allows you to influence the query planner with runtime configuration parameters in `psql`.

https://www.postgresql.org/docs/16/runtime-config-query.html

In this section we will suppress the use of several different scan types to compare queries.

Report and explain the execution plan for the following two queries (A & B):

(1) with both index and sequential scans enabled;
(2) with index scans enabled and sequential scans suppressed;
(3) with index scans suppressed and sequential scans enabled.

Explain why the query planner might choose one method over another.

You should refer to the reported times and costs from the execution plan, the number of rows fetched, your knowledge of query planning (and anything else you feel is relevant).

Note: *For this question, bitmap and index-only scans count as index scans too.*

Query A:

```
SELECT * FROM Events WHERE event_id < 100;
```

Query B:

```
SELECT * FROM Events WHERE event_id >= 100;
```

(2 marks for explaining up to 6 possible query plans; 2 marks for explaining the query planner's choices.)

Be sure to reset both index and sequential scans back to on before proceeding to another question.

Q4 Transactions (4 marks)

In PostgreSQL, a transaction is set up by surrounding the SQL commands of the transaction with *BEGIN* and *COMMIT* commands. For example, a banking transaction would look like:

```
BEGIN;
UPDATE accounts SET balance = balance - 100.00
   WHERE name = 'Alice';
-- some more statements etc etc
COMMIT;
```

https://www.postgresql.org/docs/current/tutorial-transactions.html

In this question we will examine what happens when transactions occur concurrently.

(1) In a `psql` connection, begin a transaction and then perform a query to identify an item which is currently returned. (1.5 marks)
(2) In a *second* `psql` connection, begin *another* transaction and attempt to record a hold on that item for some patron 14 days in the future, then commit. (0.5 marks)
(3) In your first connection, attempt to record a loan on that item for a different patron (at the present time), then commit. (0.5 marks)

Explain what happened both in database terms and in Library terms. You may include a schedule or precedence graph of the transaction events if it aids your explanation. (1.5 marks)

Hint: *which event(s) were recorded for the item?*

Include screenshots of both transactions.

You can use the `now()` function to provide a time stamp in steps 2 and 3.

Section 3 - **Deliverables and Marking**

There are 19 marks for functionality, understanding and explanations, and 1 mark for presentation.

Submission

Submit two documents to Blackboard by the due date (4pm, 25th October 2024).

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

(1) A report file titled `4xxxxxxxx.pdf` in PDF format containing relevant code, screenshots and answers to any questions.

(2) A script file titled `4xxxxxxxx.sql` (plain text with `.sql` extension) containing all SQL code (and any `psql` commands) you wrote for this assignment.

Replace 4xxxxxxxx with your student number.

You will be marked primarily on the contents of your report file. However, the teaching team must be able to reproduce the screenshots in your report file by first running *a2-2024-librarydb.sql* then the relevant parts of your script file.

If code is not provided for any question such that your screenshots cannot be reproduced, your marks will be halved for that question.

Report File & Presentation (1 mark)

Your report file should include the following content:

- Answers to all the questions in the Assignment Tasks section.
- If you are asked to write SQL statements, you need to include those statements in your report.
- After you execute a SQL statement, if PostgreSQL produces any output (e.g. query result, query execution time, query plan, etc), you should also include a screenshot of the output as well.

The presentation mark is given as follows:

- Ordinary query results do not exceed a page (any queries returning more than 20 rows may show just the first and last 10)
- No tuples are broken across lines
- All query plans are shown in full (if it takes more than one page, break it at an appropriate spot)

END OF ASSIGNMENT 2