Assignment: Advanced Querying

COMPSCI 2DB3: Databases–Winter 2025

Deadline: March 14, 2025

Department of Computing and Software
McMaster University

Please read the **Course Outline** for the general policies related to assignments.

Plagiarism is a *serious academic offense* and will be handled accordingly.
All suspicions will be reported to the *Office of Academic Integrity*
(in accordance with the Academic Integrity Policy).

This assignment is an **individual** assignment: do not submit work of others. All parts of your submission **must** be your own work and be based on your own ideas and conclusions. Only **discuss or share** any parts of your submissions with your TA or instructor. You are **responsible for protecting** your work: you are strongly advised to password-protect and lock your electronic devices (e.g., laptop) and to not share your logins with partners or friends!

If you **submit** work, then you are certifying that you are aware of the **Plagiarism and Academic Dishonesty** policy of this course outlined in this section, that you are aware of the Academic Integrity Policy, and that you have completed the submitted work entirely yourself. Furthermore, by submitting work, you agree to automated and manual plagiarism checking of all submitted work.

Late submission policy. We allow for a five-hour grace period on submission. Hence, late submissions that are up-to five hour late are accepted. Late submissions beyond this grace period are not accepted. In case of technical issues while submitting, contact the instructor **before** the deadline.

Description

A small library holds information on their customers, on the books they have, and on any books that are loaned out to customers. The relational schema for this library consists of the following relations (SQL tables):

- **Customer**(*id*, *name*, *city*)
  Each customer has a unique identifier, a name, and a city in which they live.
- **Book**(*id*, *publisher*, *title*)
  Each book has a unique identifier, a publisher, and a title.
- **BookCopy**(*id*, *book_id*, *date*)
  Each physical copy of a book has a unique identifier, refers to a book in **Book** (via *book_id*), and has a date of purchase.
- **Loan**(*copy_id*, *customer_id*, *since*, *returned*)
  Each physical copy of a book (identified by *copy_id*, which refers to **BookCopy**) can be loaned by a customer of the library (identified by *customer_id*, which refers to **Customer**) since some date **since**. A customer can loan a book multiple times, e.g., when previous copies are already returned (in which case *returned* is true).
- **Review**(*book_id*, *customer_id*, *when*, *score*)
  Each customer (identified by *customer_id*, which refers to **Customer**) can review books (identified by **book_id**, which refers to **Book**) by assigning a score (*score*) to the book.

Part 1: The requested SQL queries

P.1 **Summary of books**.

QUERY: Write a query that returns a summary of each book: return, per book with book identifier *id* (from **Book**), a row (*id*, *ncopies*, *nloans*, *newest*) with *ncopies* the total number of physical copies, *nloans* the total number times a physical copy of the book has been loaned out, and *newest* the date at which the newest physical copy was bought. You may assume that each book has a physical copy. For books that have not yet been loaned out, *nloans* must be zero.

P.2 **Good advice**.

QUERY: A customer C might be interested in reading a book B if

- customer C did not read book B;
- there exists another customer D that read all books read by customer C; and
- customer D did read B.

Write a query that returns pairs (*customer_id*, *book_id*) such that the customer with identifier *customer_id* might be interested in the book with identifier *book_id*.

P.3 **Well-received books**.

QUERY: Consider a review $(b, c, w, s) \in$ **Review**. We say that this review is **positive** if the score s is strictly higher than the average score of all reviews by customer c **before** this review (hence, with a timestamp before w). The first review of a customer is considered **neutral**.

Write a query that returns the identifiers of all books (*id* from **Book**) that have **more** positive reviews than non-positive reviews.

P.4 **Book-age bias**?

QUERY: The library wants to know whether **older copies** of books receive lower review scores than newer copies of the same book. Unfortunately, the reviews maintained by the library do not store which **copy** of the book is stored. Hence, the library simply assumes that the review $(b, c, w, s) \in$ **Review** is a review of the last copy the customer with identifier c loaned **before** timestamp w (reviews for which no such last copy can be found are to be discarded in this analysis). With this assumption, we can define the **average review score per book copy**. Write a query that returns the identifiers of those books for which the **average review score per book copy** decrease due to the age of the books. Hence, return identifiers of those books (*id* from **Book**) such that each copy has a higher average score than **all** preceding copies. In this analysis, you can ignore book copies without reviews and you can ignore books of which at-most one copy has reviews.

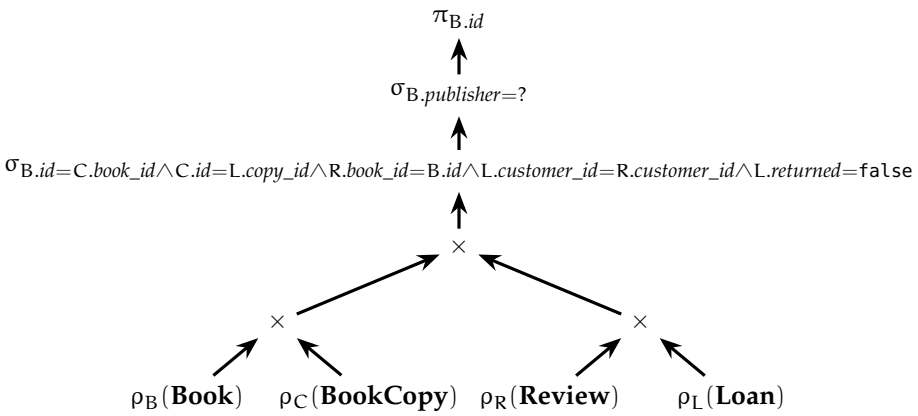
Part 2: Efficiency of queries

Assume there are 4000 distinct books, that each book has 5 copies on average, that each customer is currently loaning 4 books on average, that there are a total of 1000 customers, and that the average customer has reviewed 20 books.

Consider the following query that returns the books of a given publisher (parameter ?) of which copies are currently loaned by customers that have already written reviews of the books:

$$\pi_{\mathrm{B}.id}(\sigma_{\mathrm{B}.id=\mathrm{C}.book\_id \wedge \mathrm{C}.id=\mathrm{L}.copy\_id \wedge \mathrm{R}.book\_id=\mathrm{B}.id \wedge \mathrm{L}.customer\_id=\mathrm{R}.customer\_id \wedge \mathrm{L}.returned=\texttt{false} \wedge \mathrm{B}.publisher=?}(\\ \rho_{\mathrm{B}}(\textbf{Book}) \times \rho_{\mathrm{C}}(\textbf{BookCopy}) \times \rho_{\mathrm{R}}(\textbf{Review}) \times \rho_{\mathrm{L}}(\textbf{Loan}))).$$

Assume that this query will be evaluated via the following query execution plan:



Proceed in the following steps:

P.5 Estimate the size of the output of all intermediate steps in the above query execution plan.

P.6 Provide a **good** query execution plan for the above relational algebra query. Explain why your plan is good.

P.7 Estimate the size of the output of all intermediate steps in your **good** query execution plan. Explain each step in your estimation.

P.8 Finally, also provide an SQL query equivalent to the original relational algebra query.

Next, consider the following SQL query that returns the score of reviews written by customers (parameter $?_1$) of books that have the same title as a book of a specified publisher (parameter $?_2$).

```
SELECT R.score
FROM customer C, review R, book B
WHERE C.id = ?1 AND
      C.id = R.customer_id AND
      R.book_id = B.id AND
      B.title IN (SELECT title
                  FROM book
                  WHERE publisher = ?2);
```

Proceed in the following steps:

P.9 Translate this SQL query into a relational algebra query. Simplify the query if you see ways to do so.

P.10 Provide a **good** query execution plan for your relational algebra query. Explain why your plan is good.

P.11 Estimate the size of the output of all intermediate steps in your **good** query execution plan. Explain each step in your estimation.

Assignment

Part 1 Write the above queries. Hand in a single plain-text file (`txt`) with each of the requested queries in the following format:

```
-- Query [number]
[the query]
```

```
-- Clarifications: [any description].


[next query]
```

In the above, `[number]` is the query number (1, 2, 3, 4). Your submission:

1. must include your student number and MacID at the top of the file as a comment, e.g.:
   `-- Student number: XXXXXXXX, MacID: YYYYYY;`
2. must **only** use the constructs presented on the slides or in the book (we do **not** allow any SQL constructs that are not on the slides or in the book);
3. must work on the provided example dataset when using IBM Db2 (on `cs2db3.cas.mcmaster.ca`): test this yourself!

Part 2 Write a `PDF` file in which you answer the above 7 items. Hand in a single `PDF` file in which each answer is clearly demarcated. Your submission must include your student number and MacID.

Submissions that do not follow the above requirements will get a grade of zero.

Grading

Part 1 counts for 60%, Part 2 counts for 40%. Each of the four queries in Part 1 count equally toward the final grade for this assignment. We grade the queries in Part 1 in the same was as we did for the SQL queries of Assignment 2. Each of the seven problems in Part 2 count equally toward the final grade for this assignment. You get the full marks on a problem if it solves the described problem exactly.