CS 564 Midterm Exam

Open book, open notes, open internet – closed friends and neighbors

If you find a question ambiguous or difficult, please do not ask other students in the class. Instead, please post your questions on Piazza. Please do not post answers to other students' questions – please leave that to the instructor and the TAs. Feel free to post follow-up questions if you find an answer ambiguous. Before you post a question, please check whether a similar question is already posted and perhaps even answered.

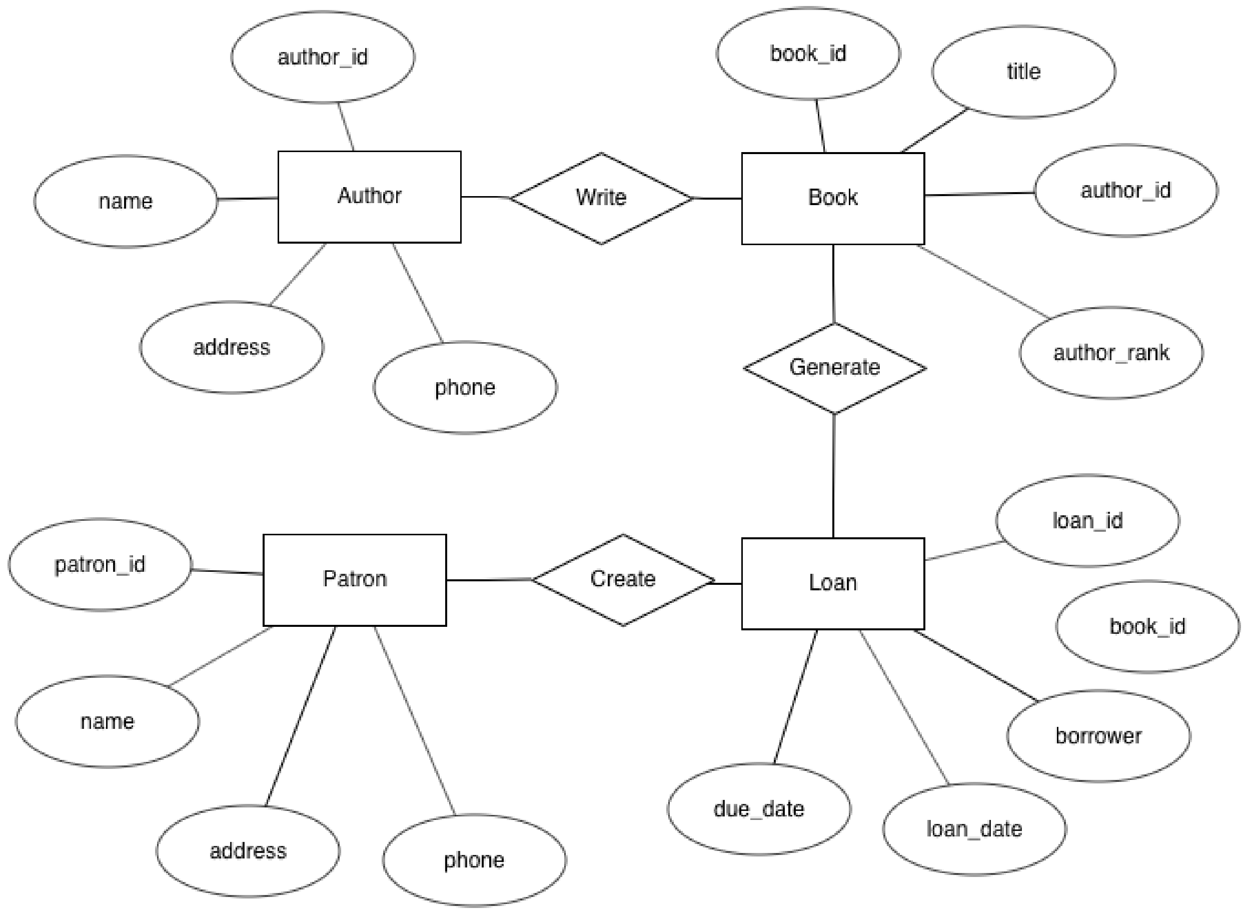
Please submit your exam as a single pdf file by **Friday, November 1, 12:00PM noon** (not midnight) on Canvas. Make sure to submit before the deadline-- late submissions will not be accepted.

Unless you have very readable handwriting, please type your answers. If we can't read it, we can't credit it. For some questions, please draw by hand and label by hand, but make sure it's all readable – then scan it or photograph it and include it in your one pdf file.
For questions 8-11, please be concise and use short, succinct sentences. 30-100 words should suffice. Do not copy from the book, from Wikipedia, or from other internet sources.

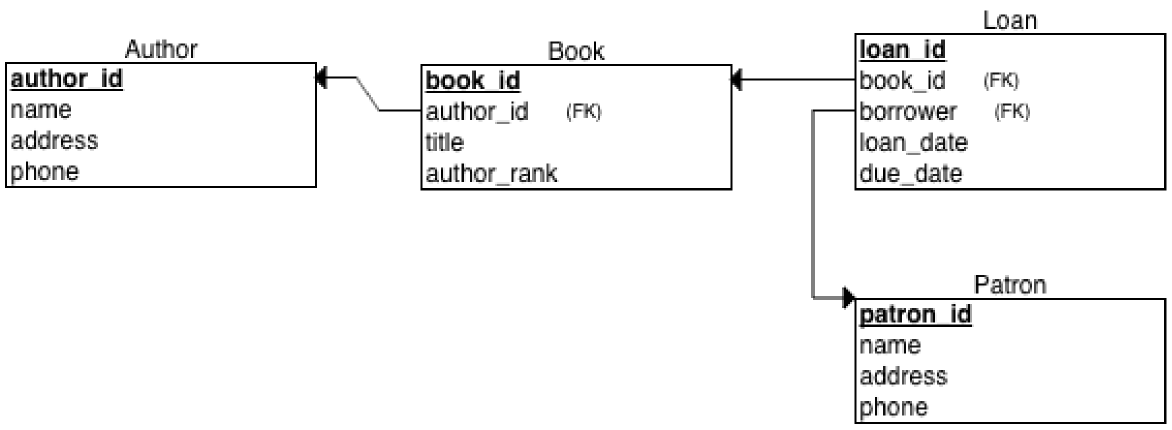
Starting on the next page, answer each of the following questions or assignments:

Name: Yiyang Lin
Student ID: 908 028 8724

1. Draw an ER diagram for a (much simplified) database for a lending library. Map the following concepts to entities, relationships, and attributes: book, title, author(s), borrower, loan date (when the book was borrowed from the library), due date, patron (a person who may borrow books), name, address, phone (some attributes apply to both the library and to each patron). You may add a few attributes but keep that to a small number. For better or worse, there may be books with the same title, but perhaps by different authors. A patron may borrow the same book multiple times.



2. Map your ER diagram to a schema (table names, column names, primary and foreign keys). You can use the TPC-H schema as an example, but make sure you clearly indicate the primary and foreign keys.



3. Write “create table” statements for these tables – include all integrity constraints that make sense. Try to include each kind of integrity constraint at least once in the database. Each table should have a primary key and each table should include a foreign key or be referenced by a foreign key.

```
CREATE TABLE Author (
                    author_id INTEGER,
                    name CHAR(20),
                    address CHAR(50),
                    phone CHAR(50),
                    PRIMARY KEY(author_id)
                  );

CREATE TABLE Book (
                    book_id INTEGER,
                    title CHAR(50),
                    author_rank INTEGER,
                    PRIMARY KEY(book_id),
                    FOREIGN KEY (author_id) REFERENCES Author (author_id)
                 );
CREATE TABLE Patron (
                    patron_id INTEGER,
                    name CHAR(20),
                    address CHAR(50),
                    phone CHAR(50),
                    PRIMARY KEY(patron_id)
                  );

CREATE TABLE Loan (
                    loan_id INTEGER,
                    loan_date datetime,
                    due_date datetime,
                    FOREIGN KEY (book_id) REFERENCES Book (book_id),
                    FOREIGN KEY (borrower) REFERENCES Patron (patron_id)
                 );
```

4. Write the following queries over your tables. Feel free to use "with" clauses where convenient; use nested queries sparingly.
   a. List all book titles in ascending order – include the first author's name for each book, and include duplicate titles.

```
SELECT
        Book.title, Author.name
ROM
        Book
JOIN
        Author
ON
        Book.author_id = Author.author_id
AND
        Book.author_rank = 1
ORDER BY
        Book.title ASC
```

   b. Count the number of books that the library owns.

```
SELECT
        COUNT ( DISTINCT book_id ) AS "Number of Books"
FROM
        Book;
```

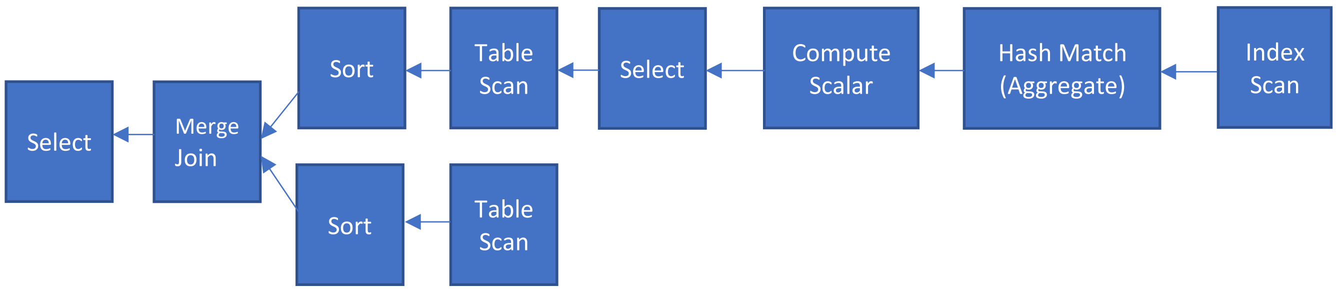
   c. For each book borrowed at least once, list the title, the number of times it has been borrowed, and the most recent due date (whether that due date is in the past or in the future)

```
WITH list
AS (
            SELECT book_id, COUNT (loan_id), MAX (due_date)
            FROM Loan
            GROUP BY book_id
  )
 SELECT
          Book.name, list.loan_id, list.due_date
 FROM
        list
 JOIN
        Book
 ON
        list.book_id = Book.book_id
```

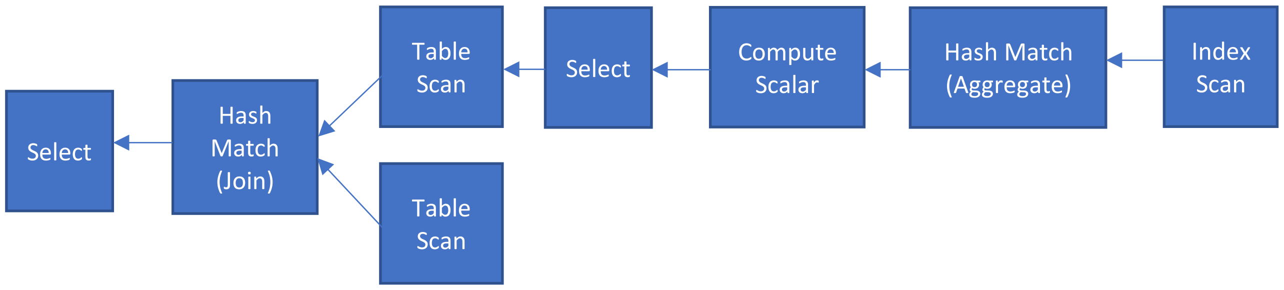
5. For the last query in question 4, draw or write an expression in relational algebra. (As we did in class, assume the algebra includes a grouping/aggregation operation in addition to the usual select, project, and join operations).

$$\pi_{Book.name,list.loan_{id},list.due_{date}}\Big(\rho_{list}\big({}_{book_{id}}F_{book_{id},count(loan_{id}),\max(due_{date})}(\text{Loan})\big) \infty_{list.book_{id}=Book.book_{id}} Book \Big)$$

6. Map your relational algebra expression to a query execution plan (draw or write).



7. Provide an alternative query execution plan (draw or write).



8. Decide which one of your two alternatives is likely more efficient – state your choice and give reasons for your choice, or state that you can't decide and give reasons why either plan might be more efficient.

If the book_id field in both tables are already sorted, then choose merge join. If the book_id is not known as sorted, then choose hash join

9. Explain (in your own words) the differences between an inner join, a (left) semi join, and a (left) outer join. Your explanation should address the sets of columns and the sets of rows in each operation's output.

INNER JOIN returns matched rows and all columns from both the first and second table.
LEFT SEMI JOIN returns matched rows and columns from the first table.
LEFT OUTER JOIN returns all the rows and columns from first table and matched rows from second table.

10. Explain (in your own words) the differences between index nested loops join, merge join, and hash join (all used to compute an inner join). Your explanation should include required properties of the join inputs and properties of the join output. For the hash join, you may assume that one join input fits in memory. For extra points, explain what happens if neither join input fits in memory.

In Nested loops join, the secondary table is "driven" by the driving table. It scans the driving table first, and the result is matched against the secondary table. It requires a small driving table and small returned results, and it supports all joining conditions.

In Hash Join, the smaller table is used as driving table and fully scanned, and a RAM hash table is thus created. This hash table is a then matched with rows in the other table. It requires an equijoin predicate. The first-time joined result is returned slower due to hashing in memory.

Merge Join requires all input data to be sorted by the join columns then the Merge Join operator.

For Hash Join, if neither join input fits in memory, the optimizer will split the fill into several files. Those cannot be fed into RAM will be stored on the disk temporarily.

11. For extra points, explain (in your own words) the differences and similarities between hash aggregation and hash join. For the hash join, you may assume that one join input fits in memory. For extra points, explain differences and similarities if neither input nor output fit in memory.

When store one row for each group in hash aggregation, the total memory requirement is actually proportional to the number and size of the output groups or rows. When store each build row in hash join, the total memory requirement is proportional to the number and size of the build rows.