National University of Computer & Emerging Sciences, Karachi
Fall-2024 School of Computing (BSCS, BSSE, BSCY, BSAI)

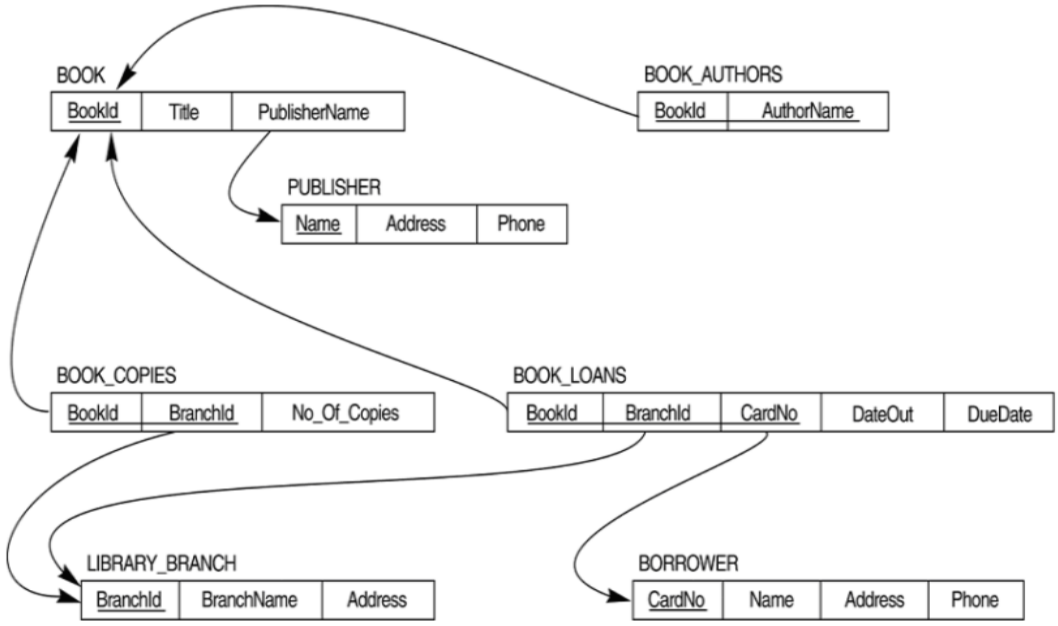
Assignment # 1

Subject: Database Systems -CS2005	**Post Date: 7/9/2024**
Total Marks:	**Due Date:** 17/9/2024
Course Instructors: Dr. Zulfiqar, Dr. Anam Qureshi, Omer Qureshi, Basit Jasani, Abeer Gauher, Atiya Jokiyo, Fizza Aqeel, Javeria Farooq, Zain Noreen	

Instructions to be strictly followed.

- For all questions involving SQL Queries:
  - ***Submit the SQL Scripts in a .txt file.***
- It should be obvious that submitting your work after the due date will result in zero points being awarded.
- Plagiarism (copying/cheating) and late submissions result in a zero mark.

Question #1: Consider the schema given below and answer the following parts.



a) How many entities (tables) are there in the schema? List their names.

Book
Publisher
Book_Authors
Book_Copies
Book_Loans
Library_Branch
Borrower

b) List the primary keys for all the tables.

Book: BookId
Publisher: PublisherName
Book_Authors: (BookId, AuthorName) (composite primary key)
Book_Copies: (BookId, BranchId) (composite primary key)
Book_Loans: (BookId, BranchId, CardNo) (composite primary key)
Library_Branch: BranchId
Borrower: CardNo

c) Identify which foreign keys exist and explain their purpose.

Book Table:

- PublisherName is a foreign key referencing Publisher(PublisherName).
  Purpose: To link each book to its publisher.

Publisher Table:

No foreign key.

Book_Authors Table:

- BookId is a foreign key referencing Book(BookId).
  Purpose: To link each author to a specific book.
  AuthorName is part of the composite primary key but is not a foreign key unless there is a separate "Author" table.

Book_Copies Table:

- BookId is a foreign key referencing Book(BookId).
  Purpose: To link the book copies to the specific book they represent.
- BranchId is a foreign key referencing Library_Branch(BranchId).
  Purpose: To specify the library branch that holds the book copies.

Book_Loans Table:

- BookId is a foreign key referencing Book(BookId).
  Purpose: To indicate which book is loaned out.
- BranchId is a foreign key referencing Library_Branch(BranchId).
  Purpose: To indicate which branch the loan transaction is associated with.
- CardNo is a foreign key referencing Borrower(CardNo).
  Purpose: To specify the borrower who has taken the book on loan.

Library_Branch Table:

- No foreign key.

Borrower Table:

- No foreign key.

d) What should be the domain of each attribute for each table?

Book Table:
BookId: Integer (unique identifier for books)
Title: String (title of the book)
PublisherName: String (must match an existing PublisherName in the Publisher table)

Publisher Table:
PublisherName: String (unique identifier, name of the publisher)
Address: String (address of the publisher)
Phone: String (phone number of the publisher, format may depend on the locale)

Book_Authors Table:
BookId: Integer (must match an existing BookId from the Book table)
AuthorName: String (name of the author)

Book_Copies Table:
BookId: Integer (must match an existing BookId from the Book table)
BranchId: Integer (must match an existing BranchId from the Library_Branch table)
No_of_copies: Integer (non-negative number representing the count of book copies)

Book_Loans Table:
BookId: Integer (must match an existing BookId from the Book table)

BranchId: Integer (must match an existing BranchId from the Library_Branch table)
CardNo: Integer (must match an existing CardNo from the Borrower table)
DateOut: Date (date the book was loaned out)
DueDate: Date (date the book is due for return)

Library_Branch Table:
BranchId: Integer (unique identifier for library branches)
BranchName: String (name of the library branch)
Address: String (address of the library branch)

Borrower Table:
CardNo: Integer (unique identifier for borrowers)
Name: String (name of the borrower)
Address: String (address of the borrower)
Phone: String (phone number of the borrower, format may depend on the locale)

e) Describe the relationship between Book and Book_Author. How would you represent an author who has written multiple books in this schema?

- One book (BookId) can have multiple authors (AuthorName).
- One author (AuthorName) can write multiple books (BookId).

To represent an author who has written multiple books in this schema:
The Book_Authors table will have multiple entries (rows) for the same AuthorName with different BookId values.

For example, if "Author A" has written "Book 1" and "Book 2", there will be two rows in the Book_Authors table:

- Row 1: BookId = 1, AuthorName = 'Author A'
- Row 2: BookId = 2, AuthorName = 'Author A'

Question #2: Consider the schema given below and answer the following parts.

Hospital (h_id, h_name, location)
Patient(p_num,p_name,age,h_id)
Doctor(d_id,d_name,h_id)
Operation_Room(o_name, time, room, d_id)
Surgery_Details (p_num, o_name)

a) Retrieve the names of the doctors operating in room 'R-11'.

```
SELECT DISTINCT d.d_name
FROM Doctor d
JOIN Operation_Room o ON d.d_id = o.d_id
WHERE o.room = 'R-11';
```

b) Identify the names of patients who are over 15 years old and are admitted to AKU (Karachi).

```
SELECT p.p_name
FROM Patient p
JOIN Hospital h ON p.h_id = h.h_id
WHERE p.age > 15 AND h.h_name = 'AKU' AND h.location = 'Karachi';
```

c) Determine the total number of patients admitted to AKU.

```
SELECT COUNT(*) AS total_patients
FROM Patient p
JOIN Hospital h ON p.h_id = h.h_id
WHERE h.h_name = 'AKU';
```

d) List the names of doctors who have operated in more than one operation room.

```
SELECT d.d_name
FROM Doctor d
JOIN Operation_Room o ON d.d_id = o.d_id
GROUP BY d.d_id, d.d_name
HAVING COUNT(DISTINCT o.room) > 1;
```

e) Provide the names of hospitals and the corresponding number of patients admitted to each, sorted in descending order by the number of patients.

```
SELECT h.h_name, COUNT(p.p_num) AS number_of_patients
FROM Hospital h
LEFT JOIN Patient p ON h.h_id = p.h_id
GROUP BY h.h_name
ORDER BY number_of_patients DESC;
```

f) Find the names of patients who are not admitted to AKU.

```
SELECT p.p_name
FROM Patient p
JOIN Hospital h ON p.h_id = h.h_id
```

```
WHERE h.h_name != 'AKU';
```

g) List the names of patients admitted exclusively to Liaquat (Lahore).

```
SELECT p.p_name
FROM Patient p
JOIN Hospital h ON p.h_id = h.h_id
WHERE h.h_name = 'Liaquat' AND h.location = 'Lahore';
```

h) Identify the patient numbers and names of those who have been operated on only by Dr. Muhammad Rafi.

```
SELECT p.p_num, p.p_name FROM Patient p JOIN Surgery_Details sd ON p.p_num = sd.p_num JOIN Operation_Room o ON sd.o_name = o.o_name JOIN Doctor d ON o.d_id = d.d_id WHERE d.d_name = 'Dr. Muhammad Rafi';
```

i) List the names of doctors who are not operating in room 'R-109'.

```
SELECT DISTINCT d.d_name
FROM Doctor d
LEFT JOIN Operation_Room o ON d.d_id = o.d_id
WHERE o.room != 'R-109' OR o.room IS NULL;
```

j) Identify the name of the patient who has undergone the highest number of surgeries.

```
SELECT p.p_name
FROM Patient p
JOIN Surgery_Details sd ON p.p_num = sd.p_num
GROUP BY p.p_num, p.p_name
ORDER BY COUNT(sd.o_name) DESC
LIMIT 1;
```

Question #3: Consider the schema given below and write each of the following queries in SQL.

Books(book_id: integer, title: string, author_id: integer, genre: string, price: real)
Authors(author_id: integer, author_name: string, nationality: string)
Orders(order_id: integer, customer_id: integer, order_date: date, total_amount: real)
Customers(customer_id: integer, customer_name: string, email: string, join_date: date)
OrderDetails(order_id: integer, book_id: integer, quantity: integer)
Reviews(review_id: integer, book_id: integer, customer_id: integer, rating: integer, comment: string)
Inventory(book_id: integer, stock_quantity: integer, reorder_level: integer)

a) Find the titles of books that have received a rating of 5 from all customers who reviewed them.

SELECT B.title FROM Books B JOIN Reviews R ON B.book_id = R.book_id GROUP BY B.book_id, B.title HAVING MIN(R.rating) = 5;

b) List the names of authors who have written more than three books in the "Science Fiction" genre.

SELECT A.author_name FROM Authors A JOIN Books B ON A.author_id = B.author_id WHERE B.genre = 'Science Fiction' GROUP BY A.author_id, A.author_name HAVING COUNT(B.book_id) > 3;

c) Identify the customers who have placed orders totaling over $500 in a single month.

SELECT C.customer_name FROM Customers C JOIN Orders O ON C.customer_id = O.customer_id GROUP BY C.customer_id, C.customer_name, YEAR(O.order_date), MONTH(O.order_date) HAVING SUM(O.total_amount) > 500;

d) Find the books whose price is less than the average price of all books by the same author.

SELECT * FROM BOOKS B1 WHERE PRICE < (SELECT AVG(PRICE) FROM BOOKS B2 WHERE B1.AUTHOR_ID = B2.AUTHOR_ID);

e) Retrieve the titles of books that have never been ordered by any customer.

SELECT TITLE FROM BOOKS WHERE BOOK_ID NOT IN (SELECT BOOK_ID FROM ORDER_DETAILS);

f) List the customer names who have given at least one review with a rating below 3.

select customer_name from customers where customer_id=(select customer_id from reviews group by customer_id having min(rating)<3);

g) For each author, display their name and the total number of books they have written.

select author_name,(select count(*) from books b group by author_id having a.author_id=b.author_id) from authors a;

h) Find the names of customers who joined in the same month and year as the customer who placed the highest-value order.

SELECT customer_name FROM Customers WHERE DATE_FORMAT(join_date, '%Y-%m') = (SELECT DATE_FORMAT(join_date, '%Y-%m') FROM Customers WHERE customer_id = (SELECT customer_id FROM Orders ORDER BY total_amount DESC));

i) List the genres that have at least one book currently below its reorder level in the inventory.

SELECT DISTINCT genre FROM Books WHERE book_id IN (SELECT book_id FROM Inventory WHERE stock_quantity < reorder_level);

j) Identify the authors who have at least one book with no reviews.

SELECT author_name FROM Authors WHERE author_id IN (SELECT author_id FROM Books WHERE book_id NOT IN (SELECT book_id FROM Reviews));

Q4) 1) A) What is the referential integrity constraint?

Referential integrity ensures that a foreign key value in one table corresponds to a valid primary key in another table. It prevents records in the child table from referencing non-existent records in the parent table.

B) Under what conditions can the foreign key be NULL. Give two examples and explain with the help of a diagram

A foreign key can be NULL when:

The relationship is optional, meaning the child record may not always have a corresponding parent.
When the foreign key references a column that allows NULL values in the parent table.

Example:
In a "Students" table, advisor_id (foreign key) references the "Advisors" table.
Some students may not have an advisor, so advisor_id can be NULL.

C) When the referential integrity is violated, what are the different actions SQL can take?

CASCADE: Deletes or updates the corresponding rows in the child table when the parent row is deleted/updated.
SET NULL: Sets the foreign key in the child table to NULL when the parent row is deleted/updated.
SET DEFAULT: Sets the foreign key to a default value if the parent row is deleted/updated.
RESTRICT/NO ACTION: Prevents the deletion or update of a parent row if there are matching child rows.

2) Discuss and differentiate inherent model based constraints, schema based constraints, and application based constraints

Inherent Model-Based Constraints:
Derived from the data model itself, e.g., entity integrity, which ensures that primary keys are unique and non-null.

Example: Every table must have a unique identifier.

Schema-Based Constraints:
Explicit constraints declared in the schema, like primary key, foreign key, or check constraints.
Example: A foreign key constraint ensuring referential integrity.

Application-Based Constraints:
Constraints that are enforced by the application code, not directly by the database schema.
Example: Business rules such as limiting the number of items a customer can order based on their account type.

In general, Model-based constraints are implicit to the database model, schema-based constraints are explicitly declared in the database schema, and application-based constraints are managed at the application level.

3) A key is a superkey but not vice versa. Explain this statement with an example

A superkey is any set of columns in a table that can be used to uniquely identify a row. A key (or candidate key) is the smallest possible superkey, meaning that if you remove any column from it, it will no longer be unique.

Example: Imagine a table called "Students" with these columns: student_id, email, phone_number.

- Superkey: {student_id, email} (this combination uniquely identifies a student)
- Key: {student_id} (it can identify a student on its own)

In this example, every key is a superkey because it uniquely identifies rows, but not every superkey is a key because some superkeys have extra, unnecessary columns.