Assignment 1: Analyze a given business scenario and create an ER diagram that includes entities, relationships, attributes, and cardinality. Ensure that the diagram reflects proper normalization up to the third normal form.

Er Diagram:

➔ **Business scenario:**

A business of Survey which use to collect feedback from customer for product services.

➔ **Define Attributes for Each Entity:**

- **Customer**: CustomerID (PK), Name, Email, Phone, Address
- **Survey**: SurveyID (PK), SurveyDate, CustomerID (FK)
- **Question**: QuestionID (PK), QuestionText, QuestionType
- **Response**: ResponseID (PK), SurveyID (FK), QuestionID (FK), AnswerText, AdditionalComments
- **FeedbackCategory**: CategoryID (PK), CategoryName, SurveyID (FK)

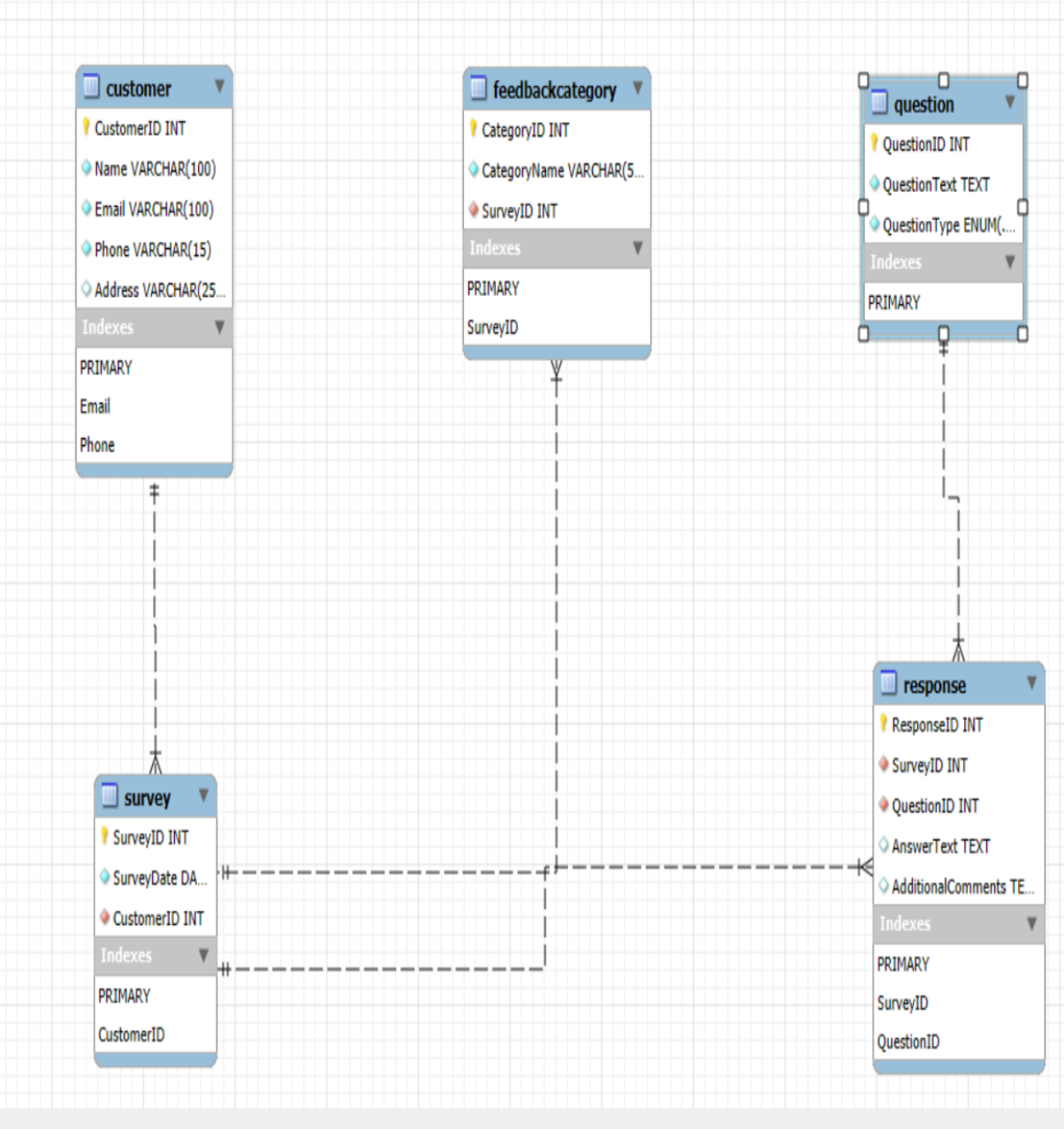
➔ **Relationship and cardinality:**

- **Customer and Survey: 1: N or one to many relation**
- **Survey and Response**: **1: N or one to many relation**
- **Question and Response**: **1: N or one to many relation**
- **Survey and FeedbackCategory**: **1: N or one to many relation**

➔ **Normalization 3NF:**

- Each cell should have only one value and column should not have multiple options.
- Remove partial dependency like ***SurveyID*** *and* ***QuestionID*** *are separated as Question depends on Survey*
- Remove transitive dependencies so that no attributes depend indirectly on a non-key attribute.

ER Diagram:

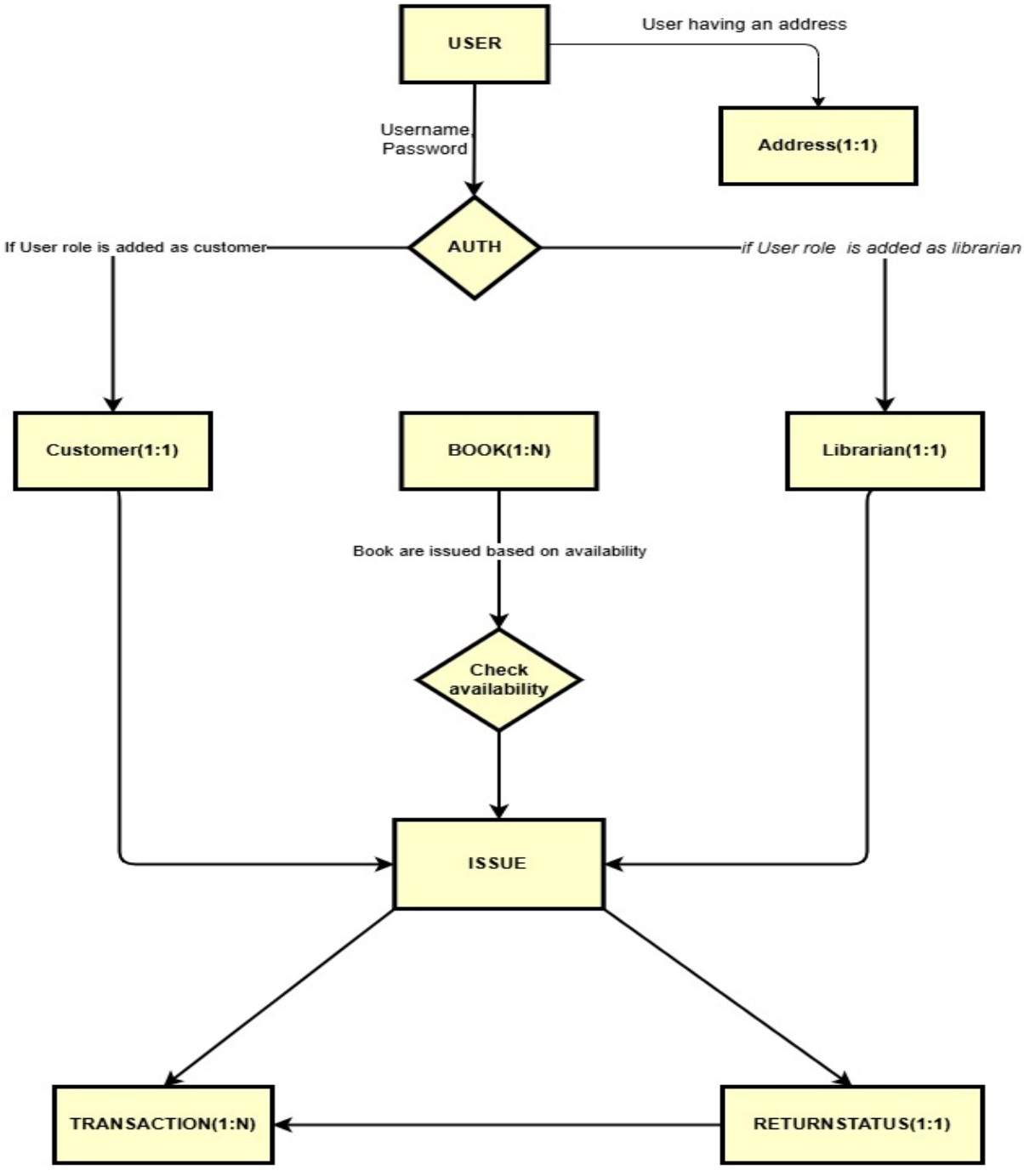


Ref:

MYSQL Reverse engineering

Assignment 2: Design a database schema for a library system, including tables, fields, and constraints like NOT NULL, UNIQUE, and CHECK. Include primary and foreign keys to establish relationships between tables.

Library Management Flow (REF: Draw.io tool to design):



QUERIES (TOOL:VS code and mysql):

- Create DB and use:

```
Create Database Library_management_db;
use Library_management_db;
```

- Users Table:

```
CREATE TABLE Users (
    UserID INT AUTO_INCREMENT PRIMARY KEY,
    Username VARCHAR(50) NOT NULL UNIQUE,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(100) NOT NULL UNIQUE,
    Password VARCHAR(255) NOT NULL,
    Phone VARCHAR(15) NOT NULL UNIQUE,
    Role ENUM('Librarian', 'Customer','Other') NOT NULL,
    created_at bigint,
    updated_at BIGInt
);
```

- Address Table:

```
Create Table Address(
    UserId INT NOT NULL unique,
    Address_line1 VARCHAR(255),
    Address_line2 VARCHAR(255),
    City VARCHAR(100),
    State VARCHAR(100),
    Zip VARCHAR(20),
    Country VARCHAR(100)
);
```

- Librarian Table:

```
CREATE TABLE Librarian (
    LibrarianID INT AUTO_INCREMENT PRIMARY KEY,
    HireDate DATE NOT NULL,
    UserID INT NOT NULL UNIQUE,
    FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE ON UPDATE CASCADE
);
```

- Customer Table:

```
CREATE TABLE Customer (
   CustomerID INT AUTO_INCREMENT PRIMARY KEY,
   RegistrationDate DATE NOT NULL,
   UserID INT NOT NULL UNIQUE,
   FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE ON UPDATE CASCADE
);
```

- Books Table:

```
CREATE TABLE Books (
   BookID INT AUTO_INCREMENT PRIMARY KEY,
   Title VARCHAR(200) NOT NULL,
   Author VARCHAR(100) NOT NULL,
   ISBN VARCHAR(13) NOT NULL UNIQUE,
   Publisher VARCHAR(100),
   PublishedYear YEAR NOT NULL,
   TotalCopies INT NOT NULL CHECK (TotalCopies >= 0),
   AvailableCopies INT NOT NULL CHECK (AvailableCopies >= 0),
   Genre VARCHAR(50)
);
```

- IssueStatus Table:

```
CREATE TABLE IssueStatus (
   IssueID INT AUTO_INCREMENT PRIMARY KEY,
   CustomerID INT NOT NULL,
   BookID INT NOT NULL,
   IssueDate DATE NOT NULL,
   DueDate DATE NOT NULL,
   LibrarianID INT NOT NULL,  -- issued by
   FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
   FOREIGN KEY (BookID) REFERENCES Books(BookID),
   FOREIGN KEY (LibrarianID) REFERENCES Librarian(LibrarianID)
);
```

- ReturnStatus Table:

```
CREATE TABLE ReturnStatus (
   ReturnID INT AUTO_INCREMENT PRIMARY KEY,
   IssueID INT NOT NULL UNIQUE,
   Return_date DATE NOT NULL,
   Fine_amount DECIMAL(10, 2) DEFAULT 0 CHECK (Fine_amount >= 0),
   LibrarianID INT NOT NULL, -- Receiver
   FOREIGN KEY (IssueID) REFERENCES IssueStatus(IssueID),
   FOREIGN KEY (LibrarianID) REFERENCES Librarian(LibrarianID)
);
```

- Transactions table:

```
CREATE TABLE Transactions (
   TransactionID INT AUTO_INCREMENT PRIMARY KEY,
   IssueID INT,
```

```
    Amount DECIMAL(10, 2) NOT NULL CHECK (Amount > 0),
    Transaction_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Transaction_type ENUM('Fine', 'book','Membership', 'Other') NOT NULL,
    FOREIGN KEY (IssueID) REFERENCES IssueStatus(IssueID)
);
```

- ❖ Audit tables to get details of operations:

```
CREATE TABLE AuditLogs (
    LogID INT AUTO_INCREMENT PRIMARY KEY,
    Action ENUM('ADD', 'ISSUE', 'RETURN', 'ZERO_COPIES','Other' ) NOT NULL Default 'Other',
    BookID INT,
    CustomerID INT,
    LibrarianID INT,
    Timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Details TEXT,
    FOREIGN KEY (BookID) REFERENCES Books(BookID) ON DELETE CASCADE,
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID) ON DELETE SET NULL,
    FOREIGN KEY (LibrarianID) REFERENCES Librarian(LibrarianID) ON DELETE SET NULL
);
```

- ❖ CONSTRAINTS:

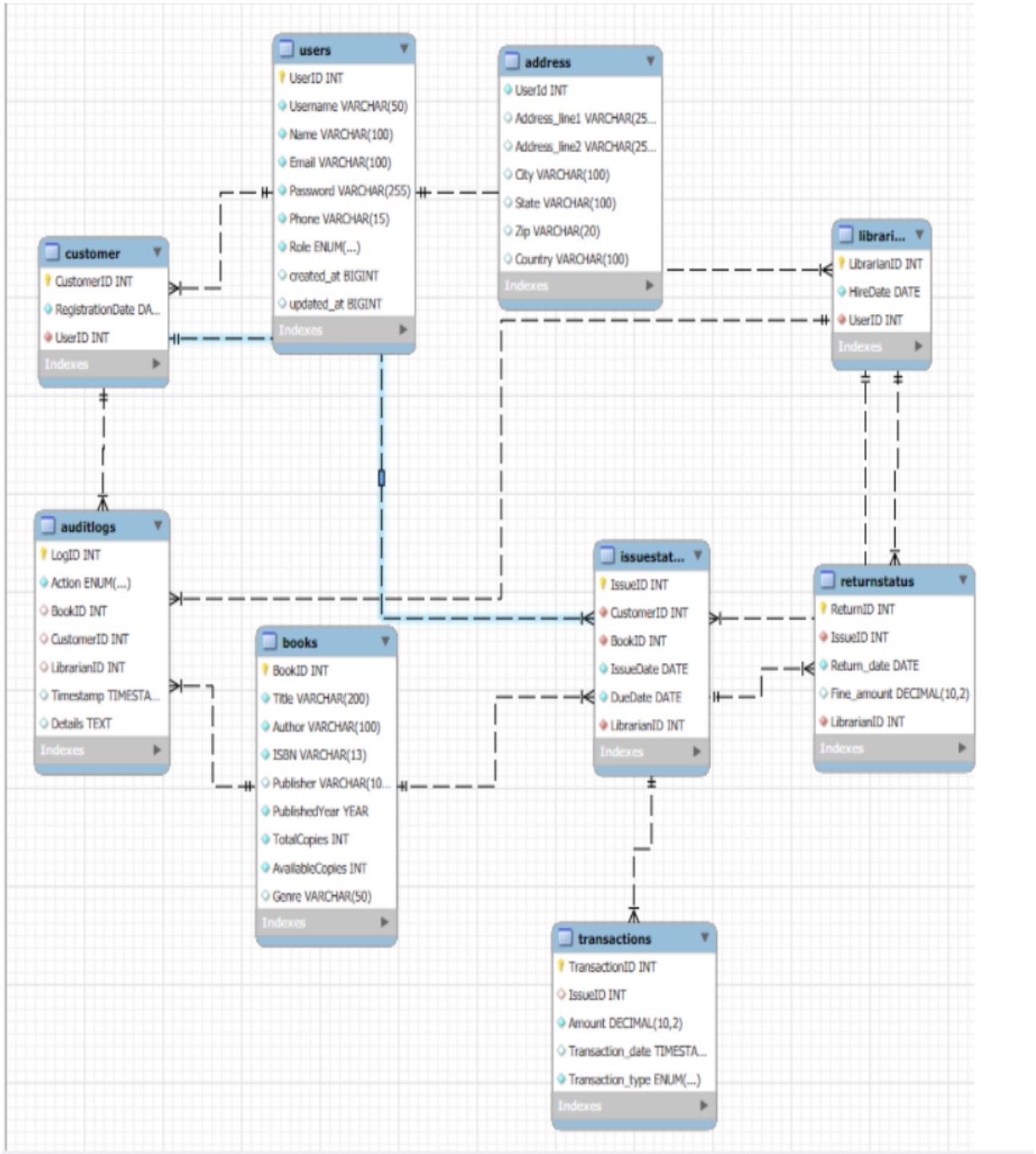
  - **Primary key (PK)**: It is used to uniquely identify particular columns. For ex. In user table User Id uniquely defines a user. It is unique in a table and cannot be null.

  - **NOT Null**: this constraint ensures that a particular field can't be null, a value must be provided. e.g. A customer must be a user hence in customer table for each customer a userID should be provided it can't be NULL.

  - **Unique**: To make a field or record such that it can't hold duplicates. Eg. An email or username must be unique for every user.

  - **Check**: There are some conditions needed to be checked before doing a transaction like book total copies or available copies can be a positive value or zero.

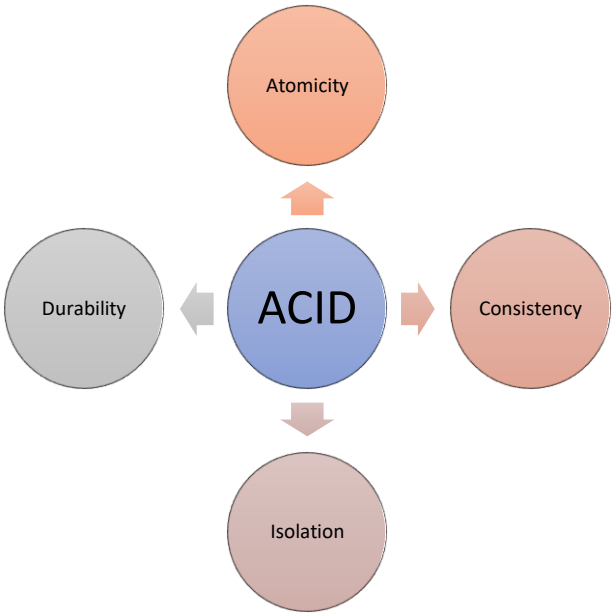
ER DIAGRAM (TOOL: **MYSQL** REVERSE ENGINEERING):



Assignment 3: Explain the ACID properties of a transaction in your own words. Write SQL statements to simulate a transaction that includes locking and demonstrate different isolation levels to show concurrency control.

ACID PROPERTIES:

- **TOOL: MS Word SmartArt**



- **DETAILS:**
  - **The ACID PROPERTIES** are fundamental guarantees provided to ensure the database transactions that should occur in possibly best manner so that better reliability and consistencies of data can be managed irrespective of a very small and a very large data.

  - Ther Acid properties are basically 4 following concepts:

    - **Atomicity:** Atomicity ensures that an entire transaction should occur at once or should not occur at all i.e., **all-or-nothing process**.
      Example: In our **Library Management System**, when a user signs up by providing details like email, username, and password, the transaction must ensure that all the details are added to the database together. If any issue occurs during this process such as a database error while saving the password the entire transaction should be aborted, and no data should be saved. This ensures the user can retry the signup process without issues like partial data insertion that might prevent future login or re-registration attempts.

    - **Consistency:** Consistency ensures that a **dB moves from one valid state to another valid state after a transaction**. Every transaction must maintain the defined rules and constraints, or even relationships in the database to ensure data integrity set by DBA.

Example: In the **Library Management System**, consistency ensures that every transaction such as adding or issuing books maintains the validity of data by following rules. Like, when a book is issued, the Available Copies should always decrease while respecting constraints like Available Copies less than Total Copies. If this rule is violated (e.g., Available Copies isn't decremented), the database state becomes inconsistent.

- **Isolation:** Isolation defined as **every transaction should occurs without any interference**, so that multiple transaction can be occur without affecting any other transaction.
  Example: In the **Library Management System**, isolation ensures that transactions like adding books, issuing books to customers, and imposing fines are executed independently. In our system there can be multiple users, multiple librarians and multiple customers, when a librarian is adding books to the system, the transaction must be isolated from other transactions such as issuing books or imposing fines, so that intermediate states (like an increase in Available Copies or a fine) are not seen by other transactions until the transaction is complete. This ensures that no transaction will be affected by changes in another.

- **Durability**: It Guarantees that once a transaction is committed, its **changes are permanent, even if the system brakes**.
  Example: In the **Library Management System**, once a book is successfully issued and the transaction is committed, the updates to Available Copies and the Transactions table should remain intact even if the system crashes right after commit changes. This ensures that the issued book's details and the fine applied to the customer are preserved so that it maintains the integrity of the data.

Assignment 4: Write SQL statements to CREATE a new database and tables that reflect the library schema you designed earlier. Use ALTER statements to modify the table structures and DROP statements to remove a redundant table.

Modifying Library Schema:

- Db Queries Are Already Demonstrated in Assignment 2 At Queries Section.

➔ **ALTER Statement:**

- Add a column for rating of book in books table:

```
ALTER TABLE Books
ADD COLUMN rating int;
```

- Modify genre column in books for added length:

```
ALTER TABLE Books
Modify COLUMN genre varchar(255);
```

- Rename column rating of book in books table to book_rating:

```
ALTER TABLE Books
Rename COLUMN rating to book_rating;
```

- Drop column book_rating from books table:

```
ALTER TABLE Books
Drop column book_rating;
```

- Drop column book_rating from books table:

```
ALTER TABLE Address
Add constraint address_userid_pk userID Primary Key;
```

➔ **DROP Statement:**

- Dropping of Address table:

```
DROP TABLE Address;
```

Assignment 5: Demonstrate the creation of an index on a table and discuss how it improves query performance. Use a DROP INDEX statement to remove the index and analyze the impact on query execution.

Indexing in SQL:

- Indexes In SQL is used to get **faster retrieval** of data from tables.
- **Primary and unique key are auto index** in database.

Create Index:

- **Syntax:**

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

- **Example:** In Our Library management database for book table:
  - When There is no index other than PK or Unique key:all 4 rows have been search and retrieved.

```
mysql> Explain select * from books;
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
|  1 | SIMPLE      | books | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    4 |   100.00 | NULL  |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql> Explain select * from books where AvailableCopies>=6;
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | books | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    4 |    33.33 | Using where |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

  - **Create index** in AvailableCopies:

```
Create index idx_copies on books(AvailableCopies);
```

- The below image clearly shows that as we create index on available copies now query searches and retrieve for a way less rows. This helps in getting data faster.

```
mysql> Create index idx_copies On books(AvailableCopies);
Query OK, 0 rows affected (0.47 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> Explain select * from books where AvailableCopies>=6;
+----+-------------+-------+------------+-------+---------------+------------+---------+------+------+----------+-----------------------+
| id | select_type | table | partitions | type  | possible_keys | key        | key_len | ref  | rows | filtered | Extra                 |
+----+-------------+-------+------------+-------+---------------+------------+---------+------+------+----------+-----------------------+
|  1 | SIMPLE      | books | NULL       | range | idx_copies    | idx_copies | 4       | NULL |    2 |   100.00 | Using index condition |
+----+-------------+-------+------------+-------+---------------+------------+---------+------+------+----------+-----------------------+
1 row in set, 1 warning (0.00 sec)

mysql> Explain select * from books;
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
|  1 | SIMPLE      | books | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    4 |   100.00 | NULL  |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

- **Dropping Index**:

```
Drop index idx_copies on books;
```

  - Again, if retrieve data with same query:

```
mysql> Drop index idx_copies on books;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> Explain select * from books where AvailableCopies>=6;
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | books | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    4 |    33.33 | Using where |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql>
```

Assignment 6: Create a new database user with specific privileges using the CREATE USER and GRANT commands. Then, write a script to REVOKE certain privileges and DROP the user.

User and privileges (DCL- Data control language):

- Creating Database user:

```
Create user user_name@host identified by 'password';
```

- Grant Permission to user:

```
Grant insert, update, delete on database.table to 'user_name';
```

- Apply Privileges Immediately:

```
Flush privileges;
```

❖ A Practical Example:

```
mysql> Create user sanjog_new identified by '123456';
Query OK, 0 rows affected (0.21 sec)

mysql> grant insert,update,delete on library_management.books to 'sanjog_new';
Query OK, 0 rows affected (0.02 sec)

mysql> Flush privileges;
Query OK, 0 rows affected (0.03 sec)
```

- Connecting to user and showing tables and using INSERT permission (select permission is not given hence given SELECT command denied to user 'sanjog_new'@'localhost' for table 'books';

```
mysql> system mysql -u sanjog_new -p
Enter password: ******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 59
Server version: 8.0.40 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| library_management |
| performance_schema |
+--------------------+
3 rows in set (0.03 sec)

mysql> use library_management;
Database changed
mysql> show tables;
+-------------------------------+
| Tables_in_library_management |
+-------------------------------+
| books                         |
+-------------------------------+
1 row in set (0.02 sec)
```

```
mysql> select * from books;
ERROR 1142 (42000): SELECT command denied to user 'sanjog_new'@'localhost' for table 'books'
mysql> desc books;
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra          |
+---------------+-------------+------+-----+---------+----------------+
| BookID        | int         | NO   | PRI | NULL    | auto_increment |
| Title         | varchar(200)| NO   |     | NULL    |                |
| Author        | varchar(100)| NO   |     | NULL    |                |
| ISBN          | varchar(13) | NO   | UNI | NULL    |                |
| Publisher     | varchar(100)| YES  |     | NULL    |                |
| PublishedYear | year        | NO   |     | NULL    |                |
| TotalCopies   | int         | NO   |     | NULL    |                |
| AvailableCopies | int       | NO   |     | NULL    |                |
| Genre         | varchar(50) | YES  |     | NULL    |                |
+---------------+-------------+------+-----+---------+----------------+
9 rows in set (0.04 sec)

mysql> INSERT INTO Books VALUES('Book 1', 'Author 1', '1234561470123', 'Publisher 1', 2010, 11, 1, 'Horror');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> INSERT INTO Books VALUES(6,'Book 1', 'Author 1', '1234561470123', 'Publisher 1', 2010, 11, 1, 'Horror');
Query OK, 1 row affected (0.02 sec)
```

- Revoke Permissions:

```
REVOKE ALL PRIVILEGES FROM 'sanjog_new'@'localhost';
```

- Drop User:

```
DROP USER 'sanjog_new'@'localhost';
```

Assignment 7: Prepare a series of SQL statements to INSERT new records into the library tables, UPDATE existing records with new information, and DELETE records based on specific criteria. Include BULK INSERT operations to load data from an external source.

TRANSACTIONS IN SQL:

- Insert record (single insertion);

```
mysql> INSERT INTO Users (Username, Name, Email, Password, Phone, Role, created_at, updated_at)
    -> VALUES ('john_doe', 'John Doe', 'john.doe@example.com', 'hashed_password', '1234567890', 'Customer', UNIX_TIMESTAMP(), UNIX_TIMESTAMP());
Query OK, 1 row affected (0.05 sec)

mysql>
mysql> INSERT INTO Books (Title, Author, ISBN, Publisher, PublishedYear, TotalCopies, AvailableCopies, Genre)
    -> VALUES ('The Great Gatsby', 'F. Scott Fitzgerald', '9780743273565', 'Scribner', 1925, 10, 10, 'Classic');
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> INSERT INTO Librarian (HireDate, UserID)
    -> VALUES ('2023-05-01', 1);
Query OK, 1 row affected (0.03 sec)

mysql>
```

- Insert Multiple records:

```
mysql> INSERT INTO Books (Title, Author, ISBN, Publisher, PublishedYear, TotalCopies, AvailableCopies, Genre)
    -> VALUES
    -> ('1984', 'George Orwell', '9780451524935', 'Plume', 1949, 15, 15, 'Dystopian'),
    -> ('To Kill a Mockingbird', 'Harper Lee', '9780060935467', 'J.B. Lippincott & Co.', 1960, 20, 20, 'Fiction');
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

- Update records:

```
mysql> UPDATE Books SET AvailableCopies = 9 WHERE BookID = 1;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE Users SET Email = 'new@example.com', Phone = '0987654321', updated_at = UNIX_TIMESTAMP() WHERE UserID=^C
mysql> ^C
mysql> ^C
mysql> select * from users;
+--------+----------+----------+----------------------+-----------------+------------+----------+------------+------------+
| UserID | Username | Name     | Email                | Password        | Phone      | Role     | created_at | updated_at |
+--------+----------+----------+----------------------+-----------------+------------+----------+------------+------------+
|      1 | john_doe | John Doe | john.doe@example.com | hashed_password | 1234567890 | Customer | 1737319880 | 1737319880 |
+--------+----------+----------+----------------------+-----------------+------------+----------+------------+------------+
1 row in set (0.00 sec)

mysql> UPDATE Users SET Email = 'new@example.com', Phone = '0987654321', updated_at = UNIX_TIMESTAMP() WHERE UserID=1
    -> ;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from users;
+--------+----------+----------+-----------------+-----------------+------------+----------+------------+------------+
| UserID | Username | Name     | Email           | Password        | Phone      | Role     | created_at | updated_at |
+--------+----------+----------+-----------------+-----------------+------------+----------+------------+------------+
|      1 | john_doe | John Doe | new@example.com | hashed_password | 0987654321 | Customer | 1737319880 | 1737320280 |
+--------+----------+----------+-----------------+-----------------+------------+----------+------------+------------+
1 row in set (0.00 sec)

mysql>
```

- Delete records:

```
mysql> DELETE FROM Books WHERE title = 'Book 1';
Query OK, 1 row affected (0.09 sec)

mysql> Delete from customer where userID=1;
Query OK, 1 row affected (0.02 sec)
```

- Bulk Import:
- File: an external SQL file with cmd's

F: > _wipro_711 > NMS > Assigns > mysql > books.sql

```
INSERT INTO Books (Title, Author, ISBN, Publisher, PublishedYear, TotalCopies, AvailableCopies, Genre)
VALUES
('Pride and Prejudice', 'Jane Austen', '9780141439518', 'Modern Library', 1813, 10, 10, 'Classic'),
('Moby Dick', 'Herman Melville', '9781503280786', 'Penguin Classics', 1851, 8, 8, 'Adventure'),
('Great' 'Expectations','Charles Dickens',9780141439563,'Penguin Classics',1861,10,10,'Classic'),
('War and Peace','Leo Tolstoy',9781400079988,'Vintage',1869,12,12,'Historical Fiction'),
('Anna Karenina','Leo Tolstoy',9780143035008,'Penguin Classics',1877,10,9,'Classic'),
('Wuthering Heights','Emily Bronte',9780141439556,'Penguin Classics',1847,8,8,'Classic');
```

- Using **SOURCE** cmd:

```
mysql> source F:/_wipro_711/NMS/Assigns/mysql/books.sql;
Query OK, 6 rows affected (0.02 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from books;
```

BookID	Title	Author	ISBN	Publisher	PublishedYear	TotalCopies	AvailableCopies	Genre
1	Book A	Author X	1234567890123	Publisher A	2020	10	9	Fiction
2	Book B	Author Y	2234567890123	Publisher B	2021	8	3	Non-Fiction
3	Book C	Author Z	3234567890123	Publisher C	2022	12	7	Fiction
4	Book D	Author X	4234567890123	Publisher D	2019	15	10	Fantasy
7	The Great Gatsby	F. Scott Fitzgerald	9780743273565	Scribner	1925	10	10	Classic
8	1984	George Orwell	9780451524935	Plume	1949	15	15	Dystopian
9	To Kill a Mockingbird	Harper Lee	9780060935467	J.B. Lippincott & Co.	1960	20	20	Fiction
10	Pride and Prejudice	Jane Austen	9780141439518	Modern Library	1993	10	10	Classic
11	Moby Dick	Herman Melville	9781503280786	Penguin Classics	1951	8	8	Adventure
12	GreatExpectations	Charles Dickens	9780141439563	Penguin Classics	1961	10	10	Classic
13	War and Peace	Leo Tolstoy	9781400079988	Vintage	1969	12	12	Historical Fiction
14	Anna Karenina	Leo Tolstoy	9780143035008	Penguin Classics	1977	10	9	Classic
15	Wuthering Heights	Emily Bront½	9780141439556	Penguin Classics	1947	8	8	Classic

```
13 rows in set (0.00 sec)

mysql>
```