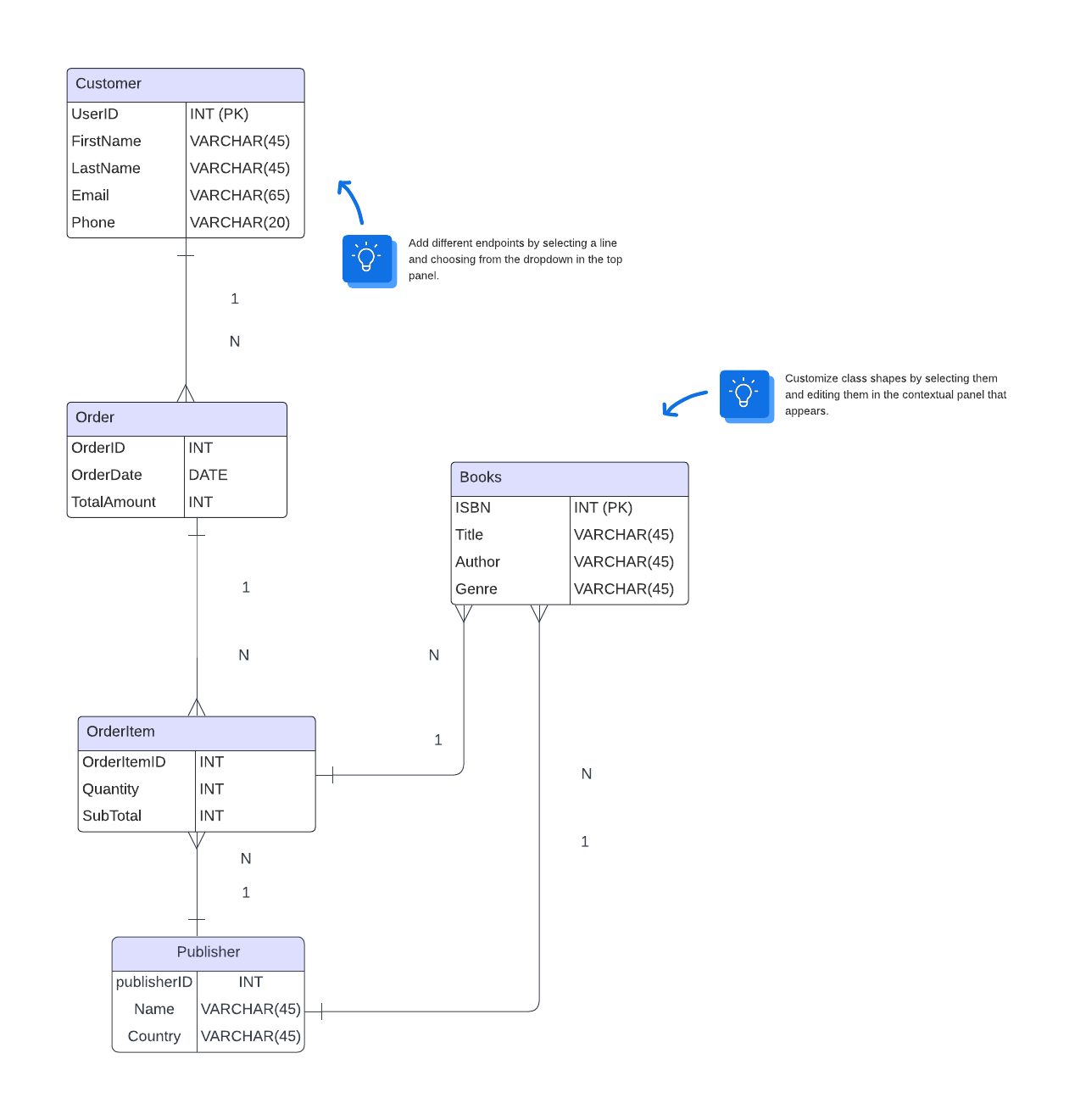
**Assignment1:**

Analyse 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.

Scenario: Online Book Store



**Entities:**

**Customer:**

Represents individuals who purchase books.

Attributes:

CustomerID (Primary Key), FirstName, LastName, Email, Phone, Address

**Book:**

Represents individual books available in the bookstore.

Attributes: ISBN (Primary Key), Title, Author, Genre, Price

**Order:**

Represents a customer’s order for one or more books.

Attributes: OrderID (Primary Key), OrderDate, TotalAmount

**OrderItem:**

Represents individual book items within an order.

Attributes: OrderItemID (Primary Key), Quantity, Subtotal

**Publisher:**

Represents publishing companies.

Attributes: PublisherID (Primary Key), PublisherName, Country

**Customer-Order Relationship:**

A customer can place multiple orders.

An order is placed by a single customer.

Cardinality:

One-to-Many (1:N)

**Order-OrderItem Relationship:**

An order can contain multiple order items (books).

An order item belongs to a single order.

Cardinality:

One-to-Many (1:N)

**Book-Publisher Relationship:**

A book is published by a single publisher.

A publisher can publish multiple books.

Cardinality:

Many-to-One (N:1) `

**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.

1. Create the Books table:

CREATE TABLE Books (

book\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255) NOT NULL,

author\_id INT,

genre\_id INT,

publication\_year INT,

ISBN VARCHAR(13),

FOREIGN KEY (author\_id) REFERENCES Authors(author\_id),

FOREIGN KEY (genre\_id) REFERENCES Genres(genre\_id)

);

2. Create the Authors table:

CREATE TABLE Authors (

author\_id INT PRIMARY KEY AUTO\_INCREMENT,

author\_name VARCHAR(255) NOT NULL

);

3. Create the Genres table:

CREATE TABLE Genres (

genre\_id INT PRIMARY KEY AUTO\_INCREMENT,

genre\_name VARCHAR(255) NOT NULL

);

**Assignment 3:**

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.

ACID stands for Atomicity, Consistency, Isolation, and Durability. These properties ensure the reliability and integrity of database transactions.

1. Atomicity: Atomicity ensures that a transaction is treated as a single, indivisible unit of work. It means that either all the changes made by the transaction are committed, or none of them are. If any part of the transaction fails, all changes are rolled back, leaving the database in its original state.

2. Consistency: Consistency ensures that a transaction brings the database from one valid state to another. It means that the database's integrity constraints, such as primary key or foreign key constraints, are not violated during the transaction. If a transaction violates any constraints, it is rolled back, and the database remains unchanged.

3. Isolation: Isolation ensures that concurrent transactions do not interfere with each other. It means that each transaction is executed as if it were the only transaction running, even if multiple transactions are executing concurrently. Isolation levels, such as Read Uncommitted, Read Committed, Repeatable Read, and Serializable, define the degree of isolation and the level of access to data that concurrent transactions have.

4. Durability: Durability guarantees that once a transaction is committed, its changes are permanent and will survive any subsequent system failures. The changes made by the transaction are stored in non-volatile memory, such as disk storage, to ensure their durability.

Suppose we have a table called "Products" with columns "product\_id" and "quantity". We want to update the quantity of a specific product while maintaining data integrity and concurrency control.

START TRANSACTION;

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

SELECT quantity FROM Products WHERE product\_id = 1;

UPDATE Products SET quantity = quantity - 10 WHERE product\_id = 1;

COMMIT;

This example shows a transaction that decreases the quantity of a product by 10. The transaction starts with the START TRANSACTION statement and sets the isolation level to READ COMMITTED.

**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.

CREATE DATABASE LibraryDatabase;

CREATE TABLE Books (

book\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255) NOT NULL,

author\_id INT,

genre\_id INT,

publication\_year INT,

ISBN VARCHAR(13),

FOREIGN KEY (author\_id) REFERENCES Authors(author\_id),

FOREIGN KEY (genre\_id) REFERENCES Genres(genre\_id)

);

CREATE TABLE Authors (

author\_id INT PRIMARY KEY AUTO\_INCREMENT,

author\_name VARCHAR(255) NOT NULL

);

CREATE TABLE Genres (

genre\_id INT PRIMARY KEY AUTO\_INCREMENT,

genre\_name VARCHAR(255) NOT NULL

);

ALTER TABLE Books ADD COLUMN language VARCHAR(50);

DROP TABLE OldAuthors;

**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.

1. Create the Books table:

CREATE TABLE Books (

book\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255) NOT NULL,

author\_id INT,

genre\_id INT,

publication\_year INT,

ISBN VARCHAR(13),

);

INSERT INTO Books (title, author\_id, genre\_id, publication\_year, ISBN) VALUES

('The Life of Ikigai', 1, 1, 2020, '1234567890123'),

('Famous Five', 2, 2, 2019, '1234567890124'),

('On Earth We are Briefly Gorgeous', 3, 3, 2018, '1234567890125');

CREATE INDEX idx\_title ON Books(title);

SELECT \* FROM Books WHERE title=’The Life of Ikigai’;

DROP INDEX idx\_title ON Books;

**Analysis:**

Creating an index on the title column significantly improves the performance of queries that search for specific titles. Dropping the index slows down such queries but can slightly improve the performance of write operations and reduce storage requirements. In a real-world scenario, the trade-off between query performance and resource usage should be carefully considered based on the specific needs of your application.

**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.

CREATE USER 'myuser'@'localhost' IDENTIFIED BY 'mypassword';  
GRANT SELECT,INSERT,UPDATE ON librarydatabase.\* TO 'myuser'@'localhost';  
REVOKE UPDATE ON librarydatabase.\* FROM 'myuser'@'localhost';  
-- SELECT user, host FROM mysql.user WHERE user='myuser';  
-- FLUSH PRIVILEGES;  
-- GRANT GRANT OPTION ON \*.\* TO 'myuser'@'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.

INSERT INTO Books (title, author\_iD, publication\_year)

VALUES ('Famous Five', 1, 1942),

('The Life of Ikigai', 2, 2020),

('On Earth We Are Briefly Gorgeous', 3, 2019);

UPDATE Books

SET publication\_year = 1950

WHERE Title = 'Famous Five';

DELETE FROM Books

WHERE title = 'On Earth We Are Briefly Gorgeous';

**DAY-2**

**Assignment 1:**

Write a SELECT query to retrieve all columns from a 'customers' table, and modify it to return only the customer name and email address for customers in a specific city.

SELECT \* FROM CUSTOMERS;

SELECT customer\_name, email\_address

FROM customers

WHERE city = 'specific\_city';

**Assignment 2:**

Craft a query using an INNER JOIN to combine 'orders' and 'customers' tables for customers in a specified region, and a LEFT JOIN to display all customers including those without orders.

SELECT orders., customers.

FROM orders

INNER JOIN customers ON orders.customer\_id = customers.customer\_id

WHERE customers.region = 'specified\_region';

SELECT customers., orders.

FROM customers

LEFT JOIN orders ON customers.customer\_id = orders.customer\_id;

**Assignment 3:**

Utilize a subquery to find customers who have placed orders above the average order value, and write a UNION query to combine two SELECT statements with the same number of columns.

SELECT customers.\*

FROM customers

WHERE customer\_id IN (

SELECT customer\_id

FROM orders

GROUP BY customer\_id

HAVING AVG(order\_value) > (

SELECT AVG(order\_value)

FROM orders

    )

);

SELECT customer\_name, email\_address

FROM customers

WHERE city = 'city\_1'

UNION

SELECT customer\_name, email\_address

FROM customers

WHERE city = 'city\_2';

**Assignment 4:**

Compose SQL statements to BEGIN a transaction, INSERT a new record into the 'orders' table, COMMIT the transaction, then UPDATE the 'products' table, and ROLLBACK the transaction.

BEGIN TRANSACTION;

INSERT INTO orders (order\_id, customer\_id, order\_date, order\_value)

VALUES (12345, 67890, '2024-05-19', 100.00);

COMMIT;

BEGIN TRANSACTION;

UPDATE products

SET stock\_quantity = stock\_quantity - 10

WHERE product\_id = 54321;

ROLLBACK;

**Assignment 5:**

Begin a transaction, perform a series of INSERTs into 'orders', setting a SAVEPOINT after each, rollback to the second SAVEPOINT, and COMMIT the overall transaction.

BEGIN TRANSACTION;

INSERT INTO orders (order\_id, customer\_id, order\_date, order\_value)

VALUES (12345, 67890, '2024-05-19', 100.00);

SAVEPOINT sp1;

INSERT INTO orders (order\_id, customer\_id, order\_date, order\_value)

VALUES (12346, 67891, '2024-05-19', 200.00);

SAVEPOINT sp2;

INSERT INTO orders (order\_id, customer\_id, order\_date, order\_value)

VALUES (12347, 67892, '2024-05-19', 300.00);

SAVEPOINT sp3;

INSERT INTO orders (order\_id, customer\_id, order\_date, order\_value)

VALUES (12348, 67893, '2024-05-19', 400.00);

SAVEPOINT sp4;

ROLLBACK TO sp2;

COMMIT;

**Assignment 6:**

Draft a brief report on the use of transaction logs for data recovery and create a hypothetical scenario where a transaction log is instrumental in data recovery after an unexpected shutdown.

Transaction logs are a crucial component of database management systems, serving as a record of all changes made to data. In the event of an unexpected shutdown or data loss, transaction logs play a vital role in data recovery. By maintaining a sequential record of transactions, logs enable the restoration of data to a consistent state, ensuring minimal data loss and downtime.

Hypothetical Scenario:

Company X's database server experiences an unexpected shutdown due to a power failure, resulting in data corruption and loss. Fortunately, the database was configured to maintain transaction logs. To recover the data, the database administrator follows these steps:

1. Identifies the last consistent checkpoint from the transaction log.

2. Applies all transactions from the log to the database, starting from the last checkpoint.

3. Rolls back incomplete transactions to ensure data consistency.

Thanks to the transaction log, Company X is able to recover their data up to the point of the shutdown, minimizing losses and downtime. The log's sequential record of transactions enabled the restoration of data to a consistent state, ensuring business continuity.

Benefits of Transaction Logs:

- Ensures data consistency and integrity

- Enables rapid data recovery in case of unexpected shutdowns or data loss

- Supports auditing and tracking of database changes

- Facilitates database replication and backup processes

In conclusion, transaction logs are an indispensable tool for data recovery and maintenance. By regularly backing up transaction logs, organizations can ensure swift recovery from unexpected events, minimizing data loss and downtime.