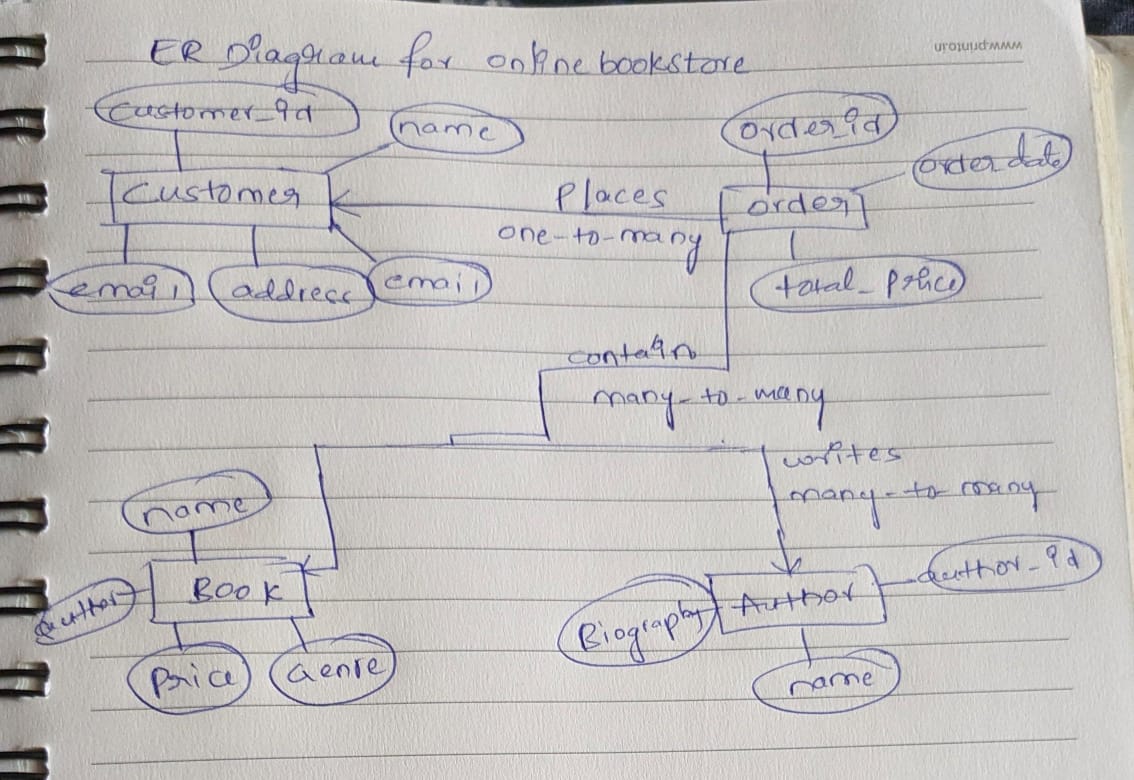
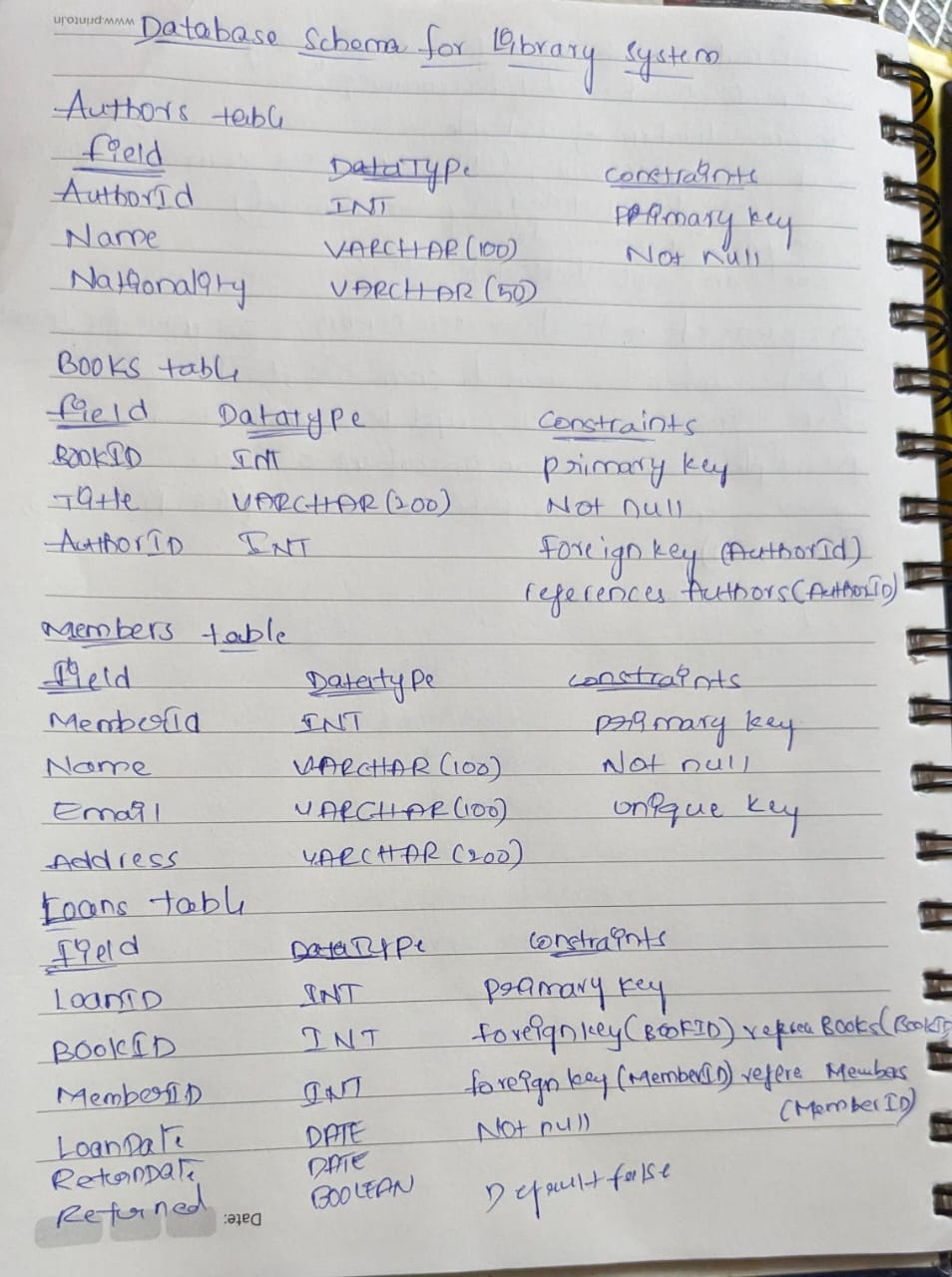
**MySQL**

**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.**

****

**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.**

****

**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 stands for:**

Atomicity: This property ensures that a transaction is treated as a single unit of work, meaning either all of its operations are successfully completed, or none are. If any part of the transaction fails, the entire transaction is rolled back, ensuring data integrity.

Consistency: This property ensures that the database remains in a consistent state before and after the transaction. In other words, the integrity constraints, such as foreign key constraints, check constraints, and triggers, are enforced, and the data meets all validation rules.

Isolation: This property ensures that transactions are executed independently of each other, as if they were the only transactions running on the system. This prevents interference between transactions and ensures that one transaction's intermediate state is not visible to other transactions until it is committed.

Durability: This property ensures that once a transaction is committed, its effects are permanently stored in the database, even in the event of a system failure. This is typically achieved through mechanisms like write-ahead logging or database backups.

EXAMPLE:

-- Begin transaction

START TRANSACTION;

-- Deduct amount from sender's account

UPDATE accounts

SET balance = balance - 100

WHERE account\_id = 123;

-- Add amount to receiver's account

UPDATE accounts

SET balance = balance + 100

WHERE account\_id = 456;

-- Commit transaction

COMMIT;

**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.**

To create a new database we will use

Create Database DatabaseName;

ALTER table Library ADD Book\_Name varchar(10) Not Null;

ALTER table Library DELETE Column Book\_No;

**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.**

Whenever we enter any primary key constaint or Unique key constaints into table automatically index values will get created. As they uses these indexes, the query execution will get speed up incase of SELECT and WHERE clauses, but incase of DML commands the processing speed will get decreases. In overall these is related to the system nothing user have to do with this .

Index is one kind of table having 2 colums named as row\_id and the coloum name for which it is creating index.

**Creating index:**

Syntax:create index index\_name on table\_name(col);

**To drop a index:**

Drop index index\_name;

When we will use index for searching the data in the the table the process execution is more faster. If we did not use any index then the process speed will become a bit slow.So it is good to have index.

**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.**

Creating a databese user:

Create ‘user1’ identified by ‘user@1’;

GRANT SELECT, INSERT, UPDATE, DELETE ON testdb.\* TO 'user1';

REVOKE UPDATE, DELETE ON testdb.\* FROM 'user1';

DROP USER 'user1';

**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.**

Inserting new records into the Libfrary table

**INSERT**:

Insert into Library values(1011,’Atomic habits’,’James clear’,600);

Insert into Library values(1891,’Be your own sunshine’,’Mary’,300);

**UPDATE:**

Update table Library where book\_num=1891;

**DELETE:**

Delete from Library where book\_Name=’Atomic Habits’;

**Inserting BULK records into the table:**

Insert into Library values((1088,’Rich dad poor dad’, ’Williams’, 450),(8988, ’The woman in me’ , ’Tracy martin’, 780));

**Day-10 Assignment**

**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 from customer where city=’Anantapur’;

**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.**

**For Specified region:**

SELECT c.customer\_id, c.customer\_name, c.region, o.order\_id, o.order\_date, o.amount FROM customers c INNER JOIN orders o ON c.customer\_id = o.customer\_id WHERE c.region = 'North America';

**All customers without orders:**

SELECT c.customer\_id, c.customer\_name, c.region, o.order\_id, o.order\_date, o.amount FROM

customers c LEFT JOIN orders o ON c.customer\_id = o.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 c.CustomerID, c.CustomerName, o.OrderID, o.OrderValue FROM Customers c INNER JOIN

Orders o ON c.CustomerID = o.CustomerID WHERE o.OrderValue > (SELECT AVG(OrderValue) FROM Orders);

**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.**

Start transaction;

Insert into orders values(‘Mango juice’,2,’Pushpa’,80);

Commit;

Update table orders set Customer\_name=’Pushpa Royal’ where price=80;

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.**

Start transaction;

Insert into Orders values(‘Gobi Rice’,1,’Siree’,150);

Savepoint s1;

Insert into Orders values(‘Mushroom fried rice’,2,’Arun’,180);

Savepoint s2;

Insert into Orders values(‘ fried rice’,3,’ShuShi’,1000);

Savepoint s3;

Rollback to s2;

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.**

**Report on the Use of Transaction Logs for Data Recovery**

**Introduction**

Transaction logs are critical components of database management systems (DBMS), designed to ensure data integrity and facilitate recovery in case of failures such as unexpected shutdowns, hardware malfunctions, or software errors. This report discusses the role of transaction logs in data recovery and presents a hypothetical scenario illustrating their importance.

What are Transaction Logs?

Transaction logs, also known as write-ahead logs, are detailed records of all transactions performed on a database. They capture every change made to the database, including inserts, updates, deletes, and schema modifications. Each entry in the transaction log typically contains:

Transaction ID

Timestamp

Operation type (INSERT, UPDATE, DELETE)

Affected table and row(s)

Old and new values (for updates)

Commit or rollback status

**How Transaction Logs Facilitate Data Recovery**

Point-in-Time Recovery: Transaction logs enable the restoration of the database to a specific point in time before an error or failure occurred. This is crucial for recovering data from logical errors such as accidental deletions or erroneous updates.

Crash Recovery: In the event of an unexpected shutdown or crash, the transaction log ensures that the database can be brought back to a consistent state. The recovery process involves two main steps:

Redo: Committed transactions that were not yet written to the data files are reapplied to ensure all committed changes are present in the database.

Undo: Uncommitted transactions are rolled back to ensure that no partial or incomplete transactions affect the database integrity.

Backup and Restore Operations: Transaction logs work in conjunction with regular database backups. A full backup captures the entire database at a specific point, while the transaction logs capture all subsequent changes. During a restore operation, the database is restored from the full backup, and the transaction logs are replayed to bring the database up to the desired state.

**Hypothetical Scenario: Data Recovery After an Unexpected Shutdown**

Scenario Description

Imagine a financial services company that manages a database for processing transactions. The database server unexpectedly shuts down due to a power failure at 3:00 PM. At the time of the shutdown, several transactions were in progress, including a critical transaction transferring funds between accounts.

Steps for Data Recovery Using Transaction Logs

Initial Assessment: The database administrator (DBA) assesses the situation once power is restored. The database was last backed up at midnight, and transaction logs are available up to the point of failure at 3:00 PM.

Restore from Backup: The DBA restores the database from the full backup taken at midnight.

Apply Transaction Logs:

The DBA applies the transaction logs to the restored database. The logs contain all transactions from midnight to 3:00 PM.

During this process, the system performs a redo operation to apply all committed transactions recorded in the logs, ensuring that all changes made between the backup and the shutdown are included.

**Crash Recovery Process:**

The recovery process identifies transactions that were in progress at the time of the crash.

An undo operation is performed for any transactions that were not committed by 3:00 PM, ensuring that the database does not reflect any partial changes.

Database Consistency Check: After applying the transaction logs, the DBA runs consistency checks to ensure that the database is in a coherent state with no corruption or data loss.

Verification and Resumption: Once consistency is confirmed, the DBA verifies that the critical transactions, including the fund transfer, have been correctly applied. The system is then brought back online for normal operations.

Conclusion

Through the use of transaction logs, the financial services company successfully recovers its database to the exact state it was in at 3:00 PM, just before the power failure. This scenario highlights the essential role of transaction logs in ensuring data integrity and facilitating robust disaster recovery procedures. By capturing all changes made to the database and enabling precise point-in-time recovery, transaction logs provide a critical safeguard against data loss and corruption in modern DBMS environments.