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 in a database system record detailed information about all transactions that modify the data. These logs are crucial for data recovery, ensuring data integrity and consistency. Here, I'll provide a simplified example of what a transaction log might look like in a hypothetical database system and describe the types of entries you might find in it.

Hypothetical Scenario:
Consider a simplified banking database with the following actions:

1. John Doe deposits $500 into his account.
2. Jane Smith transfers $200 to another account.

A system shutdown occurs unexpectedly after Jane's transfer is initiated but before it is committed.
Transaction Log Entries
Initial Setup
For simplicity, assume our database contains a single table Account with fields AccountID, Name, and Balance.
Sample Transactions

1. Transaction 1: John Doe Deposits $500
2. Transaction 2: Jane Smith Transfers $200

Simplified Transaction Log

Here is a textual representation of what these transactions might look like in the transaction log:

Timestamp	TransactionID	Action	Details
2024-06-03 10:00	TXN001	BEGIN	Transaction start for John Doe's deposit
2024-06-03 10:01	TXN001	UPDATE	UPDATE Account SET Balance = Balance + 500 WHERE AccountID = 1
2024-06-03 10:02	TXN001	COMMIT	Transaction commit for John Doe's deposit
2024-06-03 10:05	TXN002	BEGIN	Transaction start for Jane Smith's transfer
2024-06-03 10:06	TXN002	UPDATE	UPDATE Account SET Balance = Balance - 200 WHERE AccountID = 2
2024-06-03 10:06	TXN002	UPDATE	UPDATE Account SET Balance = Balance + 200 WHERE AccountID = 3
2024-06-03 10:07	TXN002	COMMIT	Transaction commit for Jane Smith's transfer

Explanation of Log Entries

Transaction 1: John Doe's Deposit

BEGIN: Marks the beginning of the transaction.
UPDATE: Records the update statement that modifies John's account balance.
COMMIT: Indicates that the transaction has been successfully completed and changes are committed to the database.

Transaction 2: Jane Smith's Transfer

BEGIN: Marks the beginning of the transaction.
UPDATE: Records the update statement that deducts $200 from Jane's account.
UPDATE: Records the update statement that adds $200 to the recipient's account.
COMMIT: Indicates that the transaction has been successfully completed and changes are committed to the database.

Handling an Unexpected Shutdown
Assume an unexpected shutdown occurred immediately after Jane's first update but before the second update and commit. The log might look like this:

Timestamp	TransactionID	Action	Details
2024-06-03 10:05	TXN002	BEGIN	Transaction start for Jane Smith's transfer
2024-06-03 10:06	TXN002	UPDATE	UPDATE Account SET Balance = Balance - 200 WHERE AccountID = 2
2024-06-03 10:07	SYSTEM	SHUTDOWN	Unexpected system shutdown

Recovery Process

Upon restarting, the DBMS will check the transaction log to determine the state of the database:

Identify Uncommitted Transactions: The DBMS will detect that TXN002 was not committed due to the system shutdown.

Rollback Uncommitted Transactions: The DBMS will rollback TXN002, undoing the update made to Jane's account, ensuring her balance is accurate.

Updated Transaction Log After Recovery

Timestamp	TransactionID	Action	Details
2024-06-03 10:05	TXN002	BEGIN	Transaction start for Jane Smith's transfer
2024-06-03 10:06	TXN002	UPDATE	UPDATE Account SET Balance = Balance - 200 WHERE AccountID = 2
2024-06-03 10:07	SYSTEM	SHUTDOWN	Unexpected system shutdown
2024-06-03 10:08	TXN002	ROLLBACK	Rollback transaction due to incomplete state

Conclusion

This example illustrates the crucial role of transaction logs in maintaining data integrity and enabling data recovery. By recording all database changes, transaction logs provide a reliable mechanism to restore the database to a consistent state after unexpected failures.