RDBMS Day2

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.

To BEGIN transaction
BEGIN TRANSACTION;

To INSERT a new record into the 'orders' table
INSERT INTO orders (orderid, customerid, productid, quantity, orderdate)
VALUES (nextval('ordersorderidseq'), 'customeridvalue', 'productidvalue', 'quantityvalue', 'orderdatevalue');

To COMMIT the transaction
COMMIT;

To UPDATE the 'products' table
UPDATE products
SET quantityinstock = quantityinstock - 'quantityvalue'
WHERE productid = 'productidvalue';

To ROLLBACK the transaction (in case of any errors)
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

To BEGIN transaction
BEGIN TRANSACTION;

To INSERT first record into 'orders' table
INSERT INTO orders (orderid, customerid, productid, quantity, orderdate)
VALUES (nextval('ordersorderidseq'), 'customeridvalue1', 'productidvalue1', 'quantityvalue1', 'orderdatevalue1');

To Set SAVEPOINT after first INSERT
SAVEPOINT savepoint1;

To INSERT second record into 'orders' table
INSERT INTO orders (orderid, customerid, productid, quantity, orderdate)
VALUES (nextval('ordersorderidseq'), 'customeridvalue2', 'productidvalue2', 'quantityvalue2', 'orderdatevalue2');

To Set SAVEPOINT after second INSERT
SAVEPOINT savepoint2;

To INSERT third record into 'orders' table
INSERT INTO orders (orderid, customerid, productid, quantity, orderdate)
VALUES (nextval('ordersorderidseq'), 'customeridvalue3', 'productidvalue3', 'quantityvalue3', 'orderdatevalue3');

To Rollback to the second SAVEPOINT
ROLLBACK TO SAVEPOINT savepoint2;

To COMMIT the overall transaction
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 play an important role in ensuring data integrity and facilitating recovery from unexpected events such as system failures, crashes, or outages These logs document changes made to the database throughout the sequence during transactions, including inserts, updates, and deletions. Here is how transaction logs help with data recovery:

Point-in-time recovery: Transaction logs enable point-in-time recovery, allowing database administrators to restore data at a specific point before it fails by rolling back transactions recorded in the logs when it is desired to restore the database to a consistent state .

Rollback and rollforward operations: Transaction logs facilitate rollback and rollforward operations in case of a crash or system failure. Rollback involves deleting incomplete transactions, ensuring that the database remains consistent. In rollforward, commit transactions from the log can be used to restore the database to a consistent state.

Redo and Undo Information: Transaction logs contain redo and undo information. Retax statements record changes that must be reapplied to the database for updates, while undo statements provide a means of reversing changes caused by unplanned transactions or rolling back transactions

Replication and High Availability: Transaction logging is used in database replication and high availability solutions. By replicating connection information to standby servers or secondary data centers, organizations can maintain an accurate copy of the database and reduce downtime in the event of a primary server failure

Considerations:

Imagine a scenario in which an online retail company is designing an online store with a database of customer orders, listings, and transaction history. One day, the main database server can suddenly shut down due to a hardware failure, destroying the database.

In this context, transaction logs prove helpful for data recovery. The database administrator initiates the recovery process by analyzing transaction information to determine the last consistent state of the database before failure The time-zone is detected before a power outage.

When a transaction log is used, the administrator performs a roll forward action, using the committed services from the mark to restore the state of the database within the specified time Additionally, the undo statements a from the records is used to remove incomplete or unplanned transactions that are occurring during failure

By carefully analyzing and processing connection information, the database administrator is able to successfully recover the database, ensuring minimal data loss and restoring normal functionality of the online store This condition applies highlighting the particular role of transaction records in maintaining data recovery performance in unexpected cases.