Database Transaction Management :-

A transaction is a set of logically related operations. For example, you are transferring money from your bank account to your friend's account, the set of operations would be like this:

Simple Transaction Example

1. Read your account balance

2. Deduct the amount from your balance

3. Write the remaining balance to your account

4. Read your friend's account balance

5. Add the amount to his account balance

6. Write the new updated balance to his account

This whole set of operations can be called a transaction. Although I have shown you read, write and update operations in the above example but the transaction can have operations like read, write, insert, update, delete.

In DBMS, we write the above 6 steps transaction like this:

Lets say your account is A and your friend's account is B, you are transferring 10000 from A to B, the steps of the transaction are:

1. R(A);

2. A = A - 10000;

3. W(A);

4. R(B);

5. B = B + 10000;

6. W(B);

In the above transaction R refers to the Read operation and W refers to the write operation.

Transaction failure in between the operations

Now that we understand what is transaction, we should understand what are the problems associated with it.

The main problem that can happen during a transaction is that the transaction can fail before finishing the all the operations in the set. This can happen due to power failure, system crash

Edit with WPS Office

etc. This is a serious problem that can leave database in an inconsistent state. Assume that transaction fail after third operation (see the example above) then the amount would be deducted from your account but your friend will not receive it.

To solve this problem, we have the following two operations

Commit: If all the operations in a transaction are completed successfully then commit those changes to the database permanently.

Rollback: If any of the operation fails then rollback all the changes done by previous operations.

Even though these operations can help us avoiding several issues that may arise during transaction but they are not sufficient when two transactions are running concurrently. To handle those problems we need to understand database ACID properties.

A Database Transaction is a logical unit of processing in a DBMS which entails one or more database access operation. In a nutshell, database transactions represent real-world events of any enterprise.

All types of database access operation which are held between the beginning and end transaction statements are considered as a single logical transaction in DBMS. During the transaction the database is inconsistent. Only once the database is committed the state is changed from one consistent state to another.

A transaction is a program unit whose execution may or may not change the contents of a database.

The transaction concept in DBMS is executed as a single unit.

If the database operations do not update the database but only retrieve data, this type of transaction is called a read-only transaction.

A successful transaction can change the database from one CONSISTENT STATE to another

DBMS transactions must be atomic, consistent, isolated and durable

If the database were in an inconsistent state before a transaction, it would remain in the inconsistent state after the transaction.

Why do you need concurrency in Transactions?

A database is a shared resource accessed. It is used by many users and processes concurrently. For example, the banking system, railway, and air reservations systems, stock market monitoring, supermarket inventory, and checkouts, etc.

I. Not managing concurrent access may create issues like:

Edit with WPS Office

II. Hardware failure and system crashes

III. Concurrent execution of the same transaction, deadlock, or slow performance

IV. States of Transactions

The various states of a transaction concept in DBMS are listed below:

State Transaction types:-

I. Active State :- A transaction enters into an active state when the execution process begins. During this state read or write operations can be performed.

II. Partially Committed:- A transaction goes into the partially committed state after the end of a transaction.

III. Committed State:- When the transaction is committed to state, it has already completed its execution successfully. Moreover, all of its changes are recorded to the database permanently.

IV. Failed State:- A transaction considers failed when any one of the checks fails or if the transaction is aborted while it is in the active state.

V. Terminated State:- State of transaction reaches terminated state when certain transactions which are leaving the system can't be restarted.

Once a transaction states execution, it becomes active. It can issue READ or WRITE operation.

Once the READ and WRITE operations complete, the transactions becomes partially committed state.

Next, some recovery protocols need to ensure that a system failure will not result in an inability to record changes in the transaction permanently. If this check is a success, the transaction commits and enters into the committed state.

If the check is a fail, the transaction goes to the Failed state.

If the transaction is aborted while it's in the active state, it goes to the failed state. The transaction should be rolled back to undo the effect of its write operations on the database.

The terminated state refers to the transaction leaving the system.

What are ACID Properties?

ACID Properties are used for maintaining the integrity of database during transaction processing. ACID in DBMS stands for Atomicity, Consistency, Isolation, and Durability.

Edit with WPS Office

Atomicity: A transaction is a single unit of operation. You either execute it entirely or do not execute it at all. There cannot be partial execution.

Consistency: Once the transaction is executed, it should move from one consistent state to another.

Isolation: Transaction should be executed in isolation from other transactions (no Locks). During concurrent transaction execution, intermediate transaction results from simultaneously executed transactions should not be made available to each other. (Level 0,1,2,3)

Durability: · After successful completion of a transaction, the changes in the database should persist. Even in the case of system failures.

ACID Property in DBMS with example:

Below is an example of ACID property in DBMS:

Transaction 1: Begin X=X+50, Y = Y-50 END

Transaction 2: Begin X=1.1*X, Y=1.1*Y END

Transaction 1 is transferring $50 from account X to account Y.

Transaction 2 is crediting each account with a 10% interest payment.

If both transactions are submitted together, there is no guarantee that the Transaction 1 will execute before Transaction 2 or vice versa. Irrespective of the order, the result must be as if the transactions take place serially one after the other.

Types of Transactions

1. Based on Application areas

I.Non-distributed vs. distributed

II. Compensating transactions

III. Transactions Timing

IV. On-line vs. batch

2. Based on Actions

I. Two-step

II. Restricted

III. Action model

Edit with WPS Office

3. Based on Structure

I. Flat or simple transactions: It consists of a sequence of primitive operations executed between a begin and end operations.

II. Nested transactions: A transaction that contains other transactions.

III. Workflow

What is a Schedule?

A Schedule is a process creating a single group of the multiple parallel transactions and executing them one by one. It should preserve the order in which the instructions appear in each transaction. If two transactions are executed at the same time, the result of one transaction may affect the output of other.

Example

Initial Product Quantity is 10

Transaction 1: Update Product Quantity to 50

Transaction 2: Read Product Quantity

If Transaction 2 is executed before Transaction 1, outdated information about the product quantity will be read. Hence, schedules are required.

Parallel execution in a database is inevitable. But, Parallel execution is permitted when there is an equivalence relation amongst the simultaneously executing transactions. This equivalence is of 3 Types.

RESULT EQUIVALENCE:

If two schedules display the same result after execution, it is called result equivalent schedule. They may offer the same result for some value and different results for another set of values. For example, one transaction updates the product quantity, while other updates customer details.

View Equivalence

View Equivalence occurs when the transaction in both the schedule performs a similar action. Example, one transaction inserts product details in the product table, while another transaction inserts product details in the archive table. The transaction is the same, but the tables are different.

CONFLICT Equivalence

In this case, two transactions update/view the same set of data. There is a conflict amongst

Edit with WPS Office

transaction as the order of execution will affect the output.

What is Serializability?

Serializability is the process of search for a concurrent schedule who output is equal to a serial schedule where transaction ae execute one after the other. Depending on the type of schedules, there are two types of serializability:

I. Conflict

II. View

Edit with WPS Office