Transaction management in DBMS

Krishna Agrawal, 19111031

February 18, 2022

1 Introduction to Transactions

Transactions are a set of operations used to perform a logical set of work. A transaction usually means that the data in the database has changed. One of the major uses of DBMS is to protect the user's data from system failures. It is done by ensuring that all the data is restored to a consistent state when the computer is restarted after a crash. The transaction is any one execution of the user program in a DBMS. Executing the same program multiple times will generate multiple transactions.

2 Uses of Transaction Management

- The DBMS is used to schedule the access of data concurrently. It means that the user can access multiple data from the database without being interfered with each other. Transactions are used to manage concurrency.
- It is also used to satisfy ACID properties.
- It is used to solve Read/Write Conflict.
- It is used to implement Recoverability, Serializability, and Cascading.
- Transaction Management is also used for Concurrency Control Protocols and Locking of data.

3 Operations of Transaction

Following are the main operations of transaction:

1. Read(X): Read operation is used to read the value of X from the database and stores it in a buffer in main memory.
2. Write(X): Write operation is used to write the value back to the database from the buffer.

   Let's take an example to debit transaction from an account which consists of following operations:

   1. R(X); 2. X = X - 500; 3. W(X);

   Let's assume the value of X before starting of the transaction is 4000.

- The first operation reads X's value from database and stores it in a buffer.
- The second operation will decrease the value of X by 500. So buffer will contain 3500.
- The third operation will write the buffer's value to the database. So X's final value will be 3500.

But it may be possible that because of the failure of hardware, software or power, etc. that transaction may fail before finished all the operations in the set.

For example: If in the above transaction, the debit transaction fails after executing operation 2 then X's value will remain 4000 in the database which is not acceptable by the bank.

To solve this problem, we have two important operations:

3. Commit: It is used to save the work done permanently.
4. Rollback: It is used to undo the work done.

4 Properties of Transaction

There are properties that all transactions should follow and possess. The four basic are in combination termed as ACID properties. ACID properties and its concepts of a transaction are put forwarded by Haerder and Reuter in the year 1983. The ACID has a full form and is as follows:

- Atomicity: The 'all or nothing' property. A transaction is an indivisible entity that is either performed in its entirety or will not get performed at all. This is the responsibility or duty of the recovery subsystem of the DBMS to ensure atomicity.
- Consistency: A transaction must alter the database from one steady-state to another steady state. This is the responsibility of both the DBMS and the application developers to make certain consistency. The DBMS can ensure consistency by putting into effect all the constraints that have been mainly on the database schema such as integrity and enterprise constraints.
- Isolation: Transactions that are executing independently of one another is the primary concept followed by isolation. In other words, the frictional effects of incomplete transactions should not be visible or come into notice to other transactions going on simultaneously. It is the responsibility of the concurrency control sub-system to ensure adapting the isolation.
- Durability: The effects of an accomplished transaction are permanently recorded in the database and must not get lost or vanished due to subsequent failure. So this becomes the responsibility of the recovery sub-system to ensure durability.

5 DBMS States of Transaction

A database, the transaction can be in one of the following states -

- Active state

  The active state is the first state of every transaction. In this state, the transaction is being executed. For example: Insertion or deletion or updating a record is done here. But all the records are still not saved to the database.

- Partially committed

  In the partially committed state, a transaction executes its final operation, but the data is still not saved to the database. In the total mark calculation example, a final display of the total marks step is executed in this state.

- Committed

  A transaction is said to be in a committed state if it executes all its operations successfully. In this state, all the effects are now permanently saved on the database system.

- Failed state

  If any of the checks made by the database recovery system fails, then the transaction is said to be in the failed state. In the example of total mark calculation, if the database is not able to fire a query to fetch the marks, then the transaction will fail to execute.

- Aborted

  If any of the checks fail and the transaction has reached a failed state then the database recovery system will make sure that the database is in its previous consistent state. If not then it will abort or roll back the transaction to bring the database into a consistent state. If the transaction fails in the middle of the transaction then before executing the transaction, all the executed transactions are rolled back to its consistent state.

  After aborting the transaction, the database recovery module will select one of the two operations:

  - Re-start the transaction
  - Kill the transaction

6 Serializability

When multiple transactions are being executed by the operating system in a multiprogramming environment, there are possibilities that instructions of one transactions are interleaved with some other transaction.

- Schedule - A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each comprising of a number of instructions/tasks.

- Serial Schedule - It is a schedule in which transactions are aligned in such a way that one transaction is executed first. When the first transaction completes its cycle, then the next transaction is executed. Transactions are ordered one after the other. This type of schedule is called a serial schedule, as transactions are executed in a serial manner.

In a multi-transaction environment, serial schedules are considered as a benchmark. The execution sequence of an instruction in a transaction cannot be changed, but two transactions can have their instructions executed in a random fashion. This execution does no harm if two transactions are mutually independent and working on different segments of data; but in case these two transactions are working on the same data, then the results may vary. This ever-varying result may bring the database to an inconsistent state.

To resolve this problem, we allow parallel execution of a transaction schedule, if its transactions are either serializable or have some equivalence relation among them.

7 Disadvantage of using a Transaction

1. It may be difficult to change the information within the transaction database by end-users.
2. We need to always roll back and start from the beginning rather than continue from the previous state.