Transaction management in DBMS

Krishna Agrawal, 19111031

March 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.

Overview of Transaction Management:

A transaction in Oracle begins when the first executable SQL statement is encountered. An executable SQL statement is a SQL statement that generates calls to an instance, including DML and DDL statements.

When a transaction begins, Oracle assigns the transaction to an available undo tablespace to record the rollback entries for the new transaction.

A transaction ends when any of the following occurs:

- A user issues a COMMIT or ROLLBACK statement without a SAVEPOINT clause.
- A user runs a DDL statement such as CREATE, DROP, RENAME, or ALTER. If the current transaction contains any DML statements, Oracle first commits the transaction, and then runs and commits the DDL statement as a new, single statement transaction.
- A user disconnects from Oracle. The current transaction is committed.
- A user process terminates abnormally. The current transaction is rolled back.

After one transaction ends, the next executable SQL statement automatically starts the following transaction.

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: If all the operations in a transaction are completed successfully then commit those changes to the database permanently.
4. 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.

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: It states that all operations of the transaction take place at once if not, the transaction is aborted.

  There is no midway, i.e., the transaction cannot occur partially. Each transaction is treated as one unit and either run to completion or is not executed at all.

  Atomicity involves the following two operations:

  Abort: If a transaction aborts then all the changes made are not visible.

  Commit: If a transaction commits then all the changes made are visible.

  Example: Let's assume that following transaction T consisting of T1 and T2. A consists of Rs 600 and B consists of Rs 300. Transfer Rs 100 from account A to account B.

  T1

  Read(A)

  A:= A-100

  Write(A)

  T2

  Read(B)

  B:= B+100

  Write(B)

  After completion of the transaction, A consists of Rs 500 and B consists of Rs 400.

  If the transaction T fails after the completion of transaction T1 but before completion of transaction T2, then the amount will be deducted from A but not added to B. This shows the inconsistent database state. In order to ensure correctness of database state, the transaction must be executed in entirety.

- Consistency: The integrity constraints are maintained so that the database is consistent before and after the transaction.

  The execution of a transaction will leave a database in either its prior stable state or a new stable state.

  The consistent property of database states that every transaction sees a consistent database instance.

  The transaction is used to transform the database from one consistent state to another consistent state.

- Isolation: It shows that the data which is used at the time of execution of a transaction cannot be used by the second transaction until the first one is completed.

  In isolation, if the transaction T1 is being executed and using the data item X, then that data item can't be accessed by any other transaction T2 until the transaction T1 ends.

  The concurrency control subsystem of the DBMS enforced the isolation property.

- Durability: The durability property is used to indicate the performance of the database's consistent state. It states that the transaction made the permanent changes.

  They cannot be lost by the erroneous operation of a faulty transaction or by the system failure. When a transaction is completed, then the database reaches a state known as the consistent state. That consistent state cannot be lost, even in the event of a system's failure.

  The recovery subsystem of the DBMS has the responsibility of Durability property.

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.
- Non-serial Schedule - If interleaving of operations is allowed, then there will be non-serial schedule. It contains many possible orders in which the system can execute the individual operations of the transactions.
- Serializable schedule - The serializability of schedules is used to find non-serial schedules that allow the transaction to execute concurrently without interfering with one another. It identifies which schedules are correct when executions of the transaction have interleaving of their operations. A non-serial schedule will be serializable if its result is equal to the result of its transactions executed serially.

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.