Transaction Processing

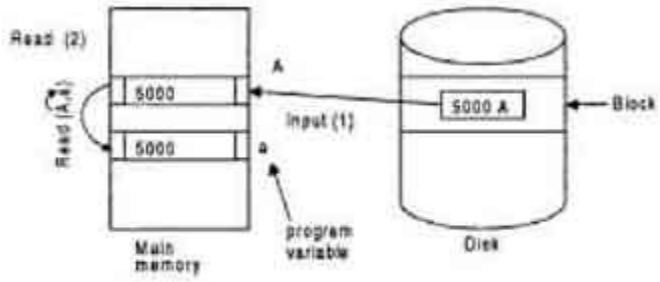
A transaction is a set of changes that must all be made together. It is a unit of program execution that access and possibly updates various data items.Moreover, it is a program unit whose execution mayor may not change the contents of a database.If the database was in consistent state before a transaction, then after execution of the transaction also, the database must be in a consistate. For example, a transfer of money from one bank account to another requires two changes to the database both must succeed or fail together.

Process of Transaction

The transaction is executed as a series of reads and writes of database objects, which are explained below:

Read Operation

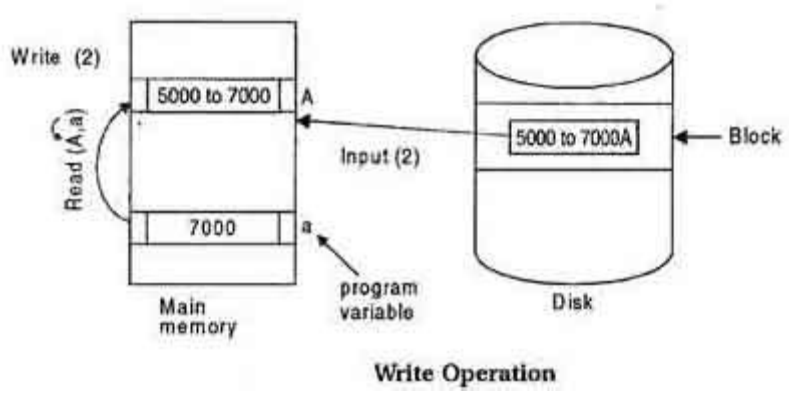
To read a database object, it is first brought into main memory from disk, and then its value is copied into a program variable as shown in figure.



Read Operation

Write Operation

To write a database object, an in-memory copy of the object is first modified and then written to disk.



Write Operation

Transaction Properties

There are four important properties of transaction that a DBMS must ensure to maintain data in the case of concurrent access and system failures. These are:

Atomicity: (all or nothing)

A transaction is said to be atomic if a transaction always executes all its actions in one step or not executes any actions at all It means either all or none of the transactions operations are performed.

Consistency: (No violation of integrity constraints)

A transaction must preserve the *consistency* of a database after the execution. The DBMS assumes that this property holds for each transaction. Ensuring this property of a transaction is the responsibility of the user.

Isolation: (concurrent changes invisibles)

The transactions must behave as if they are executed in isolation. It means that if several transactions are executed concurrently the results must be same as if they were executed serially in some order. The data used during the execution of a transaction cannot be used by a second transaction until the first one is completed.

Durability: (committed update persist)

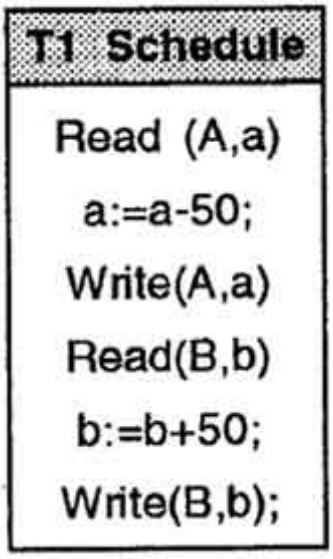
The effect of completed or committed transactions should persist even after a crash. It means once a transaction commits, the system must guarantee that the result of its operations will never be lost, in spite of subsequent failures.

The acronym ACID is sometimes used to refer above four properties of transaction that we have presented here: Atomicity, Consistency, Isolation, and Durability.

Example

In order to understand above properties consider the following example:

Let, Ti is a transaction that transfers Rs 50 from account A to account B. This transaction can be defined as:



Atomicity

Suppose that, just prior to execution of transaction Ti the values of account A and B are Rs.I000 and Rs.2000.

Now, suppose that during the execution of Ti, a power failure has occurred that prevented the Ti to complete successfully. The point of failure may be after the completion Write (A,a) and before Write(B,b). It means that the changes in A are performed but not in B. Thus the values of account A and Bare Rs.950 and Rs.2000 respectively. We have lost Rs.50 as a result 'of this failure.

Now, our database is in inconsistent state.

The reason for this inconsistent state is that our transaction is completed partially and we save the changes of uncommitted transaction. So, in order to get the consistent state, database must be restored to its original values i.e. A to Rs.I000 and B to Rs.2000, this leads to the concept of atomicity of transaction. It means that in order to maintain the consistency of database, either all or none of transaction's operations are performed.

In order to maintain atomicity of transaction, the database system keeps track of the old values of any write and if the transaction does not complete its execution, the old values are restored to make it appear as the transaction never executed.

Consistency

The consistency requirement here is that the sum of A and B must be unchanged by the execution of the transaction. Without the consistency requirement, money could be created or destroyed by the transaction. It can. be verified easily that, if the database is consistent before an execution of the transaction, the database remains consistent after the execution of the transaction.

Ensuring consistency for an individual transaction is the responsibility of the application programmer who codes the transaction.

Isolation

If several transactions are executed concurrently (or in parallel), then each transaction must behave as if it was executed in isolation. It means that concurrent execution does not result an inconsistent state.For example, consider another transaction T2, which has to display the sum of account A and B. Then, its result should be Rs.3000.

Let's suppose that both Tl and T2 perform concurrently, their schedule is shown below:

T1	T2	Status
Read (A,a)		value of *A* i.e.1000 is copied to local variable a
a:=a-50		local variable *a*=950
Write(A,a)		value of local variable *a* 950 is copied to database item *A*
	Read(A,a)	value of database item *A* 950 is copied to *a*
	Read(B,b)	value of database item *B* 2000 is copied to *b*
	display(a+b)	2950 is displayed as sum of accounts *A* and *B*
Read(B,b)		value of data base item *B* 2000 is copied to local variable *b*
b:=b+50		local variable *b*=2050
Write(B,b)		value of local variable *b* 2050 is copied to database item *B*

The above schedule results inconsistency of database and it shows Rs.2950 as sum of accounts A and B instead of Rs.3000. The problem occurs because second concurrently running transaction T2, reads A and B at intermediate point and computes its sum, which results inconsistent value. Isolation property demands that the data used during the execution of a transaction cannot be used by a second transaction until the first one is completed.

A solution to the problem of concurrently executing transaction is to execute each transaction serially 'that is one after the other. However, concurrent execution of transaction provides significant performance benefits, so other solutions are developed they allow multiple transactions to execute concurrently.

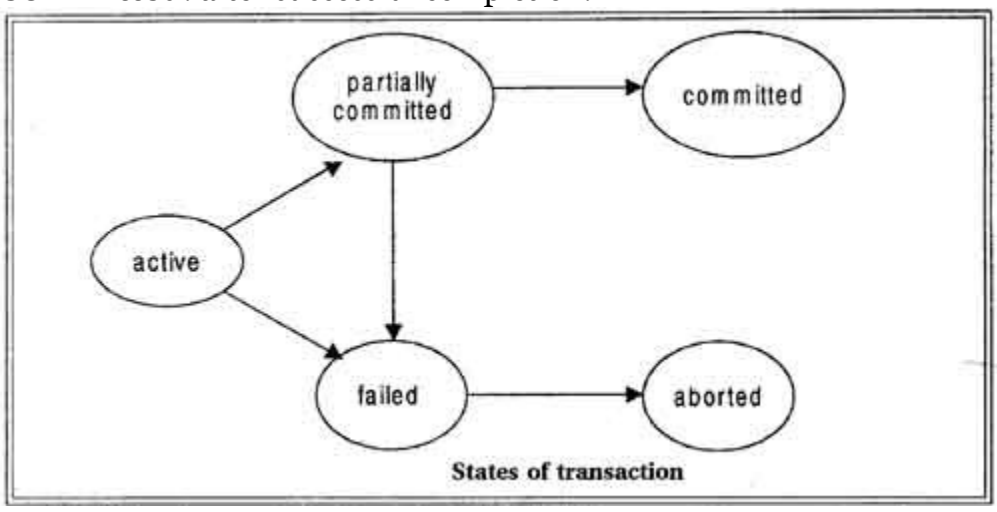
Durability

Once the execution of the transaction completes successfully, and the user who initiated the transaction has been notified that the transfer of funds has taken place, it must be the case that no system failure will result in a loss of data corresponding to this transfer of funds.

The durability property guarantees that, once a transaction completes successfully all the updates that it carried out on the database persist, even if there is a system failure after the transaction completes execution. Ensuring durability is the responsibility of a component of the database system called the recovery-management component.

States of Transaction

A transaction must be in one of the following states:

- **Active**: the initial state, the transaction stays in this state while it is executing.
- **Partially committed**: after the final statement has been executed.
- **Failed**: when the normal execution can no longer proceed.
- **Aborted**: after the transaction has been rolled back and the database has been restored to its state prior to the start of the transaction.
- **Committed**: after successful completion.
- 

States of transaction