DBMS

UNIT – IV

1. Transaction management & concurrency control

Transaction is a sequence of operations that are performed on the data in the database.

A transaction is any action that reads from and/or writes to a database.

A transaction may consists of

- A simple select statement to generate a list of table contents.
- A series of related update statement to change the values of attribute in various tables.
- A series of Insert statement to add rows to one or more tables.
- A combination of Select, Update and Insert statements.

"A transaction is a logical unit of work that must be completed or entirely aborted. No intermediate states are acceptable."

- A successful transaction changes the database from one consistent state to another.
- A consistent database state is one in which all data integrity constraints are satisfied.

➔Transaction properties:

Every transaction satisfies the properties automatically, consistency, isolation and durability. These are also called as ACID properties.

1. Atomicity:

All the operations (SQL request) of transaction be completed, otherwise the transaction is aborted.

2. Consistency:

Every transaction maintains database consistency. A transaction takes a database from one consistency state to another consistency state.

3. Isolation:

Isolation means that the data used during the execution of a transaction can't be used by a second transaction until the first one is completed.

4. Durability:

The changes made by a transaction can't be lost even in the event of system failure, those changes are saved until the transaction is performed on that data.

5. Serialisability:

Serialization is the schedule for the concurrent execution of the transaction that provides consistent rules. If only a single transaction is executed serializability is not on issue.

➔Transaction states :

A transaction may be completed or aborted. Completed transaction means all the operations in a transaction are completed successfully. Aborted transaction has been rolled back the database to its previous state.

A transaction has the following states.

1. Active:

It is the initial state, the transaction stays in this state while it is executing.

2. Partially committed:

A transaction enters into this state after the final statement has been executed.

3. Failed:

If any errors occur during the execution of transaction, it can no longer proceed, then the transaction is in failed state.

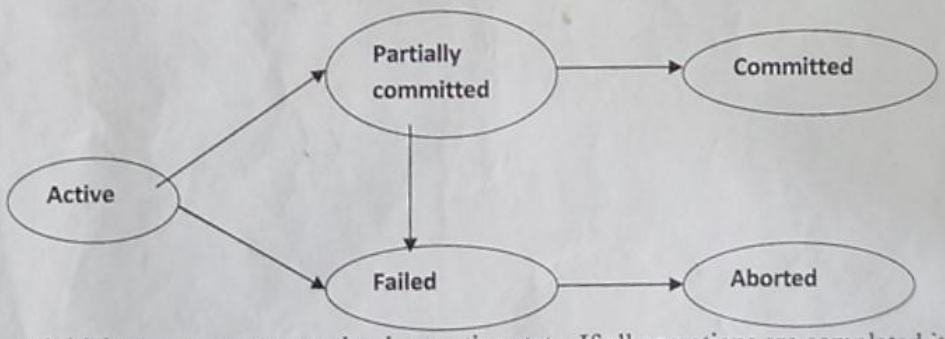
4. Aborted:

After the transaction has been rolled back and the database has been restored to its previous state.

5. Committed:

A transaction that completes its execution successfully, then it is in committed state.

A state diagram among all states of a transaction is as follows.



At initial state, every transaction is at active state. If all operations are completed it enters into committed state otherwise it enters into failed state because of any errors and it is rolled back from the database. That means this is aborted.

In the failed state, there are two options for a transaction

- It can request the transaction.
- It can kill the transaction.

➔Serializability :-

The database system must control concurrent execution for transactions to ensure database consistency. To maintain database consistency for concurrent execution of transactions, we follow a technique called "**Serializability**".

Let T1, T2 are two transactions on the accounts A and b, consider the serialization schedule of two transactions.

T1	T2
Read(A)	
Write(A)	
	Read(A)
	Write(A)
Read(B)	
Write(B)	
	Read(B)
	Write(B)

There are two different forms of schedule equivalence

- Conflict serializability.
- View serializability.

(a) Conflict Serializability:- Let us consider a schedule s in which there are two consecutive instructions I1 and I2 of the transactions T1 and T2 on the same data item 'Q'. There are four possible cases of serializability.

(1) I_1=read (Q), I_2=read (Q)

Since the same value of Q is read of T1 and T2 order of instructions is not considerable.

(2) I_1=read (Q), I_2=write (Q)

T1 does not read the value that is written by transaction T2. So order of execution is considerable and these are conflict instruction.

(3) I_1=write (Q), I_2=read (Q)

Without reading the data to be written, these are also conflict instructions.

(4) I_1=write (Q), I_2=read (Q)

Without reading the data, these are blind writes. So these are conflict.

"So, we say that I1 and I2 are conflict if they are operated by different transaction on the same data item, and at least one of these instructions is a write operation".

The conflict serializability for the above schedule is

T1	T2
Read (A)	
Write (A)	
Read (B)	
Write (B)	
	Read (A)
	Write (A)

Read (B)

Write (B)

(b) View serialisability:- The two schedules are equivalent when the data is read by T2,that is produced T1 (or) vice versa in both schedules.

A schedule is a view serializability if it is view equivalent to a serial schedule.

A view-serializable schedule is

T1	T2	T3
Read(Q)		
	Write(Q)	
Write(Q)		Write(Q)

→ Transaction management with SQL :-

The American Nation standards institute (ANSI) has defined standards for the SQL database transactions.

Transaction support is provided by two SQL statements. COMMIT and ROLLBACK back.

- A COMMIT statement is used to record(save) all changes permanently within the database.
- A ROLL BACK statement is used to abort the all changes from the database and transaction is rolled backed to its previous consistent state.

A transaction begins implicitly when the first SQL statement is encountered. Not all SQL implementations follow the ANSI standard. Some, such as SQL server use transaction management such as

BEGIN TRNSACTION; is used to indicate the beginning of a new transaction, where as oracle. RDBMS uses the SET TRANSACTION, to declare a new transaction start and its properties.

→ Concurrency control

The co-ordination of the simultaneous execution of transaction in a multi-user database system is known as concurrency control.The objective of concurrency control is to ensure serialisability of transaction in a multi user database environment. Concurrency control is important because the simultaneous execution of transaction over a shared database can create several data integrity and consistency problems.

The three main problems are

- Lost updates
- Uncommitted data
- Inconsistency retrievals

(a) Lost update:-

The last updates problems occurs when two consistency transactions T1 and T2 are updating the same data element and one of the updates is lost cover written by other transaction.

Consider the example of two transactions T1 and T2 performing on same data item 'A'. Let A=100.

T1	T2
Read(A)	
	Read(A)
A=A+100	
	A=A-50
Write(A)	
	Write(A)

A = 150

Now the rule value A=50 in the database. The update value (A+100=200) by the transaction T1 is over written by the update value (A-50=50) by the transaction T2. That means update value by transaction T1 is lost.

(b) Uncommitted data :-

The phenomena of uncommitted data occur when two transactions T1 and T2 are executed concurrently and first transaction (T1) is rolled back after the second transaction (T2) has already accessed the uncommitted data. Thus violating the isolation properly of transactions.

Consider the example

T1	T2
Read(A)	
A=A+100	
Write(A)	
	Read(A)
Rolled Back	
	A=A-50;
	Write(A)

In the above example, transaction T2 reads the uncommitted data that is produced by cancelled transaction T1.

(c) Inconsistent retrievals :-

Inconsistent retrievals occur when a transaction access data before and after another transactions finish working with such data.

Consider the example

T1	T2
	Read(A)
Read(A)	
	A=A+100
	Write(A)
Read(A)	

In the above example T1 retrieves inconsistent values of data item 'A' before and performing transaction T2 on data item A.

→ Explain the:- Scheduler ?

The scheduler is a special DBMS process that establishes the order in which the operations within concurrent transactions are executed. The scheduler helps the isolation of transactions.

To determine the appropriate order, the scheduler bases its actions on concurrency control algorithms such as locking (or) time stamping methods.

- It is important to understand that not all transactions are serializable.
- The scheduler main job is to create a serializable schedule of transactions operations.
- A serializable schedule is a schedule of a transactions operations in which the inter leaved execution of the transactions, produces the same results.
- Additionally, the scheduler facilitate data isolation to ensure that two transactions do not update the same data element at the same time.
- Data base operations require READ and /or WRITE actions that produces conflicts.
- Two operations are in conflicts when they access the same data and atleast one of them is a WRITE operations.

Consider the possible conflicts scenarios.

Transactions		RESULT
T1	T2	

Read	Read	No conflict
Read	Write	Conflict
Write	Read	Conflict
Write	Write	Conflict

→Concurrency control with locking methods

A lock guarantees exclusive use of data item to a current transaction. In other words, transaction T2 does not have access to a data item is currently being used by transaction T1.

Therefore locks are required to prevent another transaction from reading inconsistent data. Most multi-user DBM'S automatically initiate and enforce locking producers. All lock information is managed by a lock manager.

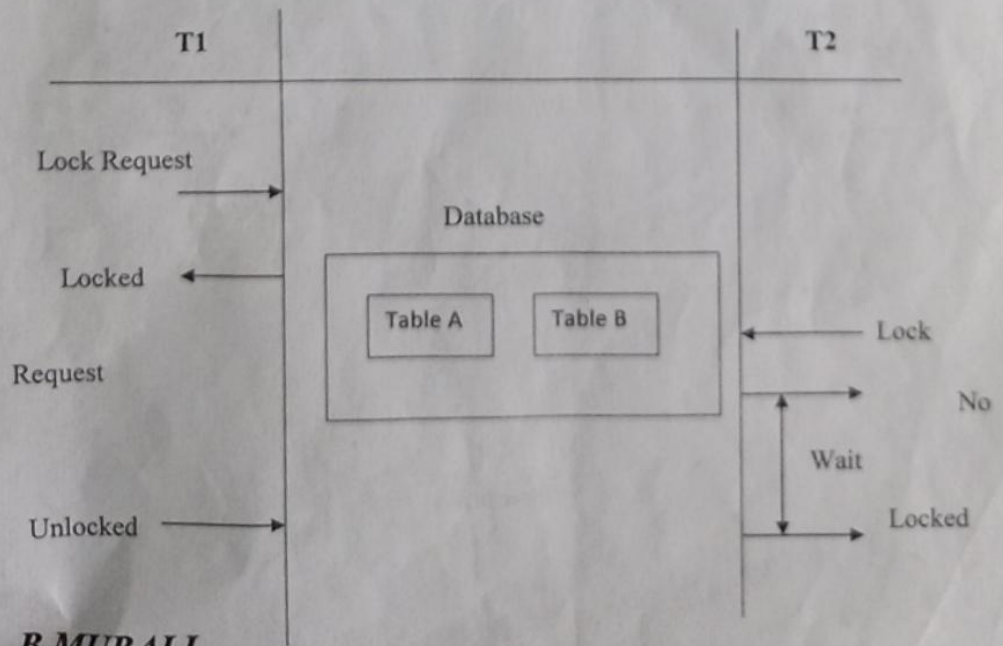
(1) Lock Granularity :-

Lock granularity indicates the level of lock use. Locking can take place at the following levels.

- Database level
- Table level
- Page level
- Row level
- Field level

(a) Database level :-

In a database level lock, the entire database is locked, thus preventing the use of any table in the database by transaction T2 while transaction T1 is being executed. This level is good for batch process, but it is unsuitable for multi-user DBMS's.



(b) Table level:-

In a table-level lock, the entire table is locked, preventing access to any row by transaction T2 whole the transaction T1 is using the table.

However, two transactions can access the same database as long as they access different tables.

Table level locks are less restrictive than database level locks. Table-level locks cause traffic jams when many transactions are waiting to access the same table.

(c) Page-level:-

In a page level lock, the DBMS will lock an entire disk page. A disc page (or) page is equivalent of a disc block.

A page has fixed size as 4k, 8k (or) 16k. Page-level lock is the most frequently used multi-user DBMS locking method.

(d) Row level:-

A row-level lock is much less restrictive than other locks. The DBMS allows concurrent transactions to access different rows of same table even the rows are located on the same page.

(e) Field level:-

The field-level lock allows concurrent transactions to access the same row as long as they require the use of different fields (attributes) within that row.

(2) Lock types:-

Regardless of the level of locking, the DBMS may use different lock types.

- Binary locks
- Shared locks
- exclusive locks

(a) Binary locks:-

A binary lock has only two states. Locked (1) (or) unlocked (0). If an object that is, a database table, page (or) row is locked by a transaction and no other transaction can use that object. As a rule, a transaction must unlock the object after its termination. Such operations are automatically managed and scheduled by the DBMS.

Consider the example, two transactions T1 and T2 on the table EMP.

T1	T2
Lock Emp	
Read Sal	
Sal=Sal+100	
Write Sal	
Unlock Emp	
	Lock Emp

Read Sal

Sal=Sal+500

Write Sal

Unlock Emp

(b) Shared locks:-

A shared lock is issued when a transaction wants to read data from the database. Shared locks allow several READ transactions to read the same data item concurrently. For example, if transaction T1 has a shared lock on data item X and transaction T2 wants to read data item X, T2 may also detain a shared lock on data item X. So several shared locks are issued to same data item at a same time.

(c) Exclusive locks:-

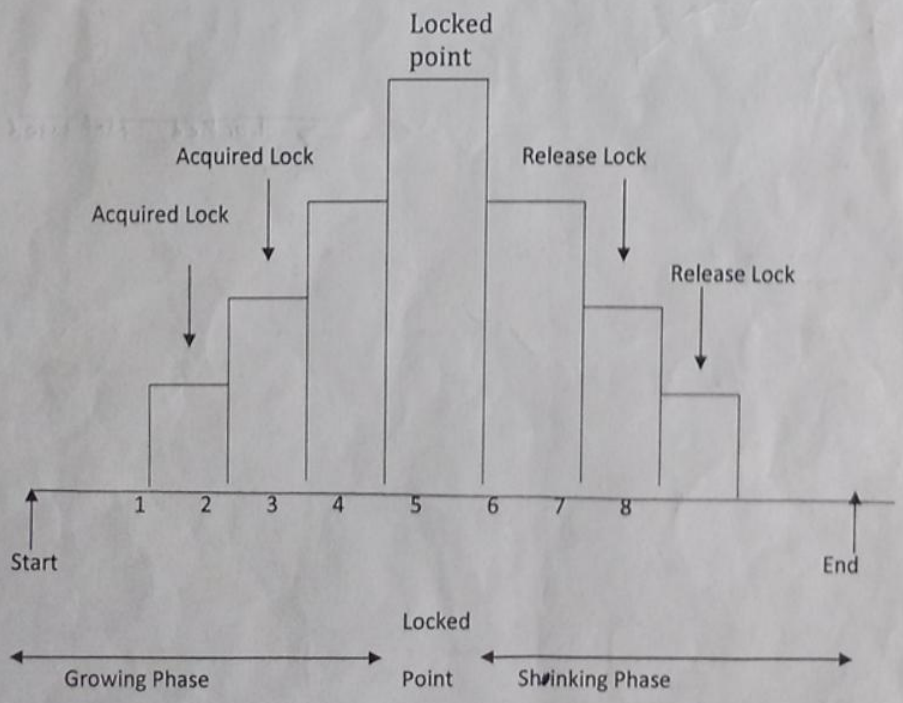
An exclusive lock is issued when a transaction wants to update (write) a data item and no locks are currently held on that data item by any other transaction. Therefore, if a shared (or) exclusive lock is already held on data item X by transaction T1, an exclusive lock cannot be granted by transaction T2 and T2 must wait to begin until T1 commits. Although locks prevent serious data inconsistencies, they can lead to two major problems.

- The resulting transaction schedule might not be serializable.
- The schedule might create dead locks.

➔Two phases locking:-

Two phases locking guarantees serializability but it does not prevent dead locks. (1) A growing phase in which a transaction acquire all required locks without unlocking any data. After acquiring all locks the transaction is in its locked point perform operations.

(2) A phase, in which a transaction releases all locks and cannot obtain any new lock.



In the above example, the transaction holds all locks until it reaches the locking point. When the locked point is reached the data are modified to the transaction requirements. Finally the transaction is completed all of its operations and realise all of its locks. Two phase locking protocol followed the rules.

1. Two transactions can not have conflicting locks.
2. No unlock operations can processed a lock operation the same transaction.
3. No data are affected until all locks are obtained.
4. All data must be modified at the locking point.
5. The Two-phase locking increase the transaction process cast and its decrease the possibilities of creation of dead locks.

→Dead locks:-

A dead lock occurs when two transactions wait indefinitely for each other to unlock data.

Dead lock system occurs if there exists a set of transactions such that every transaction in the set is waiting for another transaction in the set.

For example, a data lock occurs when two transactions T1 and T2.

T1=access data items X and Y.

T2=access data items Y and X.

If T1 has locked the data item X and wants data item Y, then it wants to unlock data item by transaction T2.

Consequently T2 has locked the data item Y and wants data item X, then it waits to unlock data item by transaction T1, then the dead lock occurs, such a dead lock is also known as deadly embrace.

Transaction	Reply	Data Items	
		Data X	Data Y
T1: LOCK(X)	OK	Locked	Not used
T2: LOCK(Y)	OK	Locked	Not used
T1: LOCK(Y)	WAIT	Locked	Locked
T2:LOCK(X)	WAIT	Locked	Locked
T1: LOCK(Y)	WAIT	Locked	Locked
T2: LOCK(X)	WAIT	Locked	Locked

Note that dead locks are possible only when one of the transactions wants to obtain an exclusive lock on a data item, no dead lock can exist among shared locks. The three basic techniques to control dead locks. They are

(1) Dead-lock prevention:- A transaction requesting a new lock is aborted when there is a possibility that a dead lock can occur. If the transaction is aborted, all changes made by this transaction are rolled back and all locks obtained by the transaction are released. The transaction is then rescheduled for execution.

(2) Dead-lock detection:- The DBMS takes the database for dead-locks. If a dead-lock is found, one of the transactions is aborted and other transaction continues.

(3) Dead-lock avoidance:- The transaction must obtain all the locks it needs before it can be executed. This technique avoids the roll back of conflict transactions.

The choice of the best deadlock control method to use depends on the database environment.

- If the probability of deadlocks is low, dead lock detection is recommended.
- If the probability of deadlock is high, dead lock prevention is recommended.
- If response time is not high on the system's priority list, Dead lock avoidance might be completed

→ Concurrency Control with time stamping :-

According to the time stamping approach for concurrent transaction, a global, unique time stamps assigns to each transaction.

The DBMS executes conflict operations in time stamp order to ensure serializability of transactions.

If two transactions are conflict, one is stopped, rollback or rescheduled and assigned a new time stamp value and other operation is continued. There are 2 schemes used to decide which transaction is rolled back and which continues execution.

- Wait / Die Scheme.
- Wound / Wait Scheme.

Wait /Die Scheme :- Using the Wait/Die scheme

If the transaction requesting the lock is older of the 2 transaction, it will wait until the other transaction are completed and the locks are released.

If the transaction requesting the lock is younger of the 2 transactions, it will die(rolled back) and rescheduled with same time stamps.

Wound/Wait Scheme :- In the Wound/Wait Scheme

If the transaction requesting the lock is older of the two transactions, it will prompt (wound) younger transaction [by rolled back] and the younger transaction is rescheduled with same time stamp.

If the transaction requesting the lock is younger of the two transactions, it will wait until the other transaction is completed and the locks are released.

Consider the scenario of Wait/Die and Wound/Wait schemes.

Requesting Lock	Owning Lock	Wait/Die	Wound/Wait
T1(314) older	T2(1398) younger	T1 Waits until T2 unlocks data	T1 Wounds T2 by rolled back T2
T2(1398) younger	T1(314) older	T2 dies (Rolled back) and reschedule with same time stamp	T2 Waits until T1 is completed.

→ Concurrency Control With Time – optimistic method

The concurrency control with optimistic method assumes that the majority of the data base transactions do not conflict. That the transactions are executed concurrently using the private copy of database objects or views of the database. By using optimistic approach, each transaction move through two or three phases. The optimistic methods are divided into 3phases.They are

*Read phase

*Validation phase

*Write phase

Read phase:

During the read phase, the transaction reads the database, executes and makes the updates to a private copy of the database values. All the update operations of the transactions are recorded in a temporary file system, which is not accessed by the remaining transactions.

Validation phase:

The transaction is validated to ensure that the changes will not effected the integrity and consistency of the database .If the validation text is positive, the transaction does not goes to the write phase .If the transition test is negative, the transaction is restarted and it goes to read phase .

Write phase:

During the write phase the changes are permanently apply to the database or it will be apply to the database.

→ Explain the DBMS facilities that are required for back up and recovery?

The database recovery is the responsible for DBA.Database are damaged for many reasons such as system problem, hard ware failures, invalid data, human errors ,program errors, net work failures and soon……The database management system must provide mechanism for restoring the database quickly and accurate of the damaging in the normal database.

Recovery Facilities:

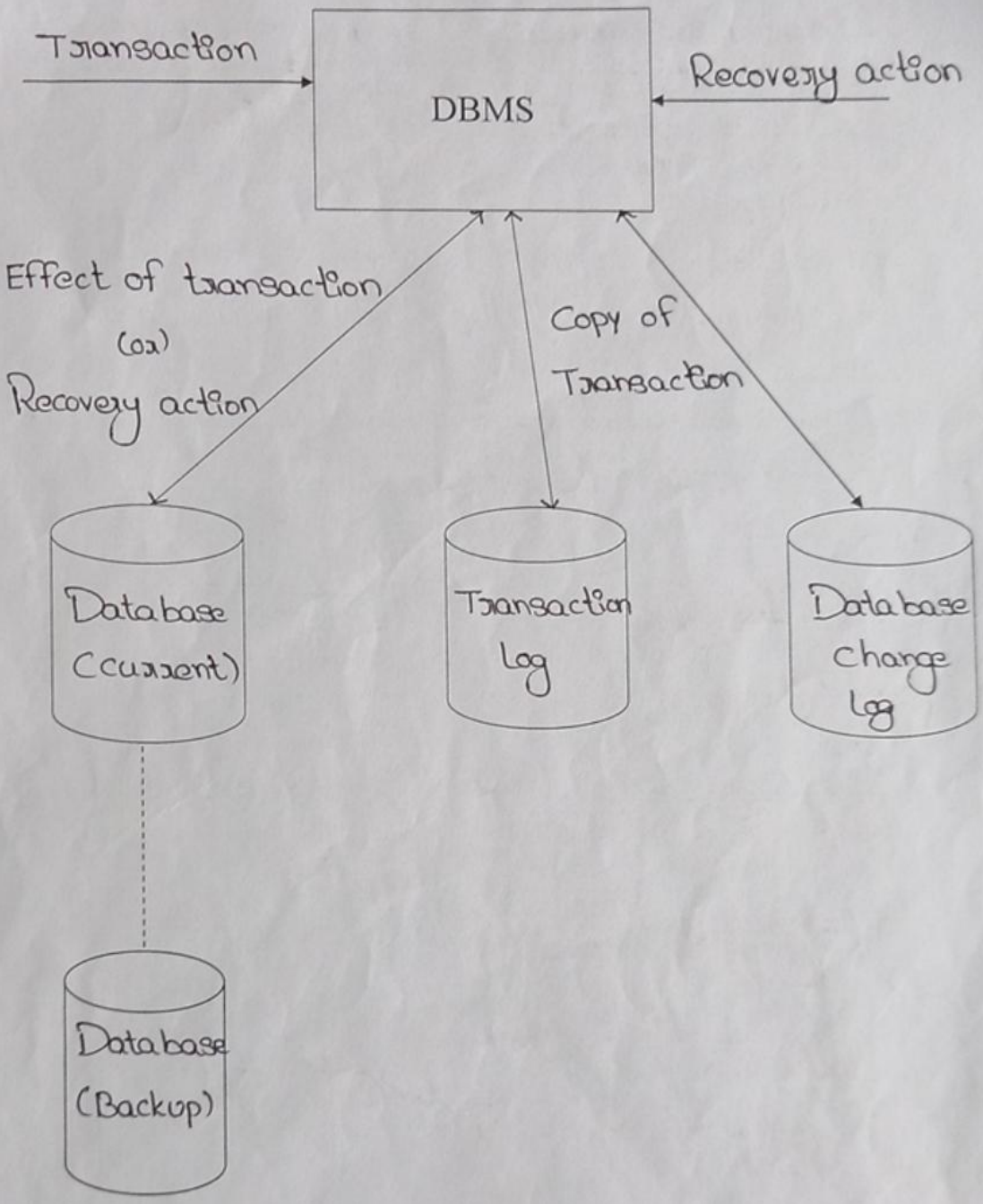
The database management system should provide 4 basic facilities were backup and recovery of a database.

Backup Facility:

The DBMS should provide back facility that produces backup copies of the entire database. The backup copy is taking the organization rules. some organization is taking everyday backup or week wise backup or month wise backup and soon…..This backup copy is storing in CD'S or DVD's or HARD Disk's.

Generalizing Facility:

The DBMS must provide generalizing facility to produce an audit trailer of transactions and data base changes.



Checkpoint Facility:

A checkpoint facility in a DBMS provides accepting any new transaction. This facility is processing are transactions are completed and verify the normal database.

Recovery Manager:

The recovery manager is a module of the DBMS which restore the database to a correct condition when a failure occurs. The recovery manager uses the transaction to restore the database.