MODULE-IV

Contents:

Transaction Processing and Concurrency Control: Transaction is a logical unit of work that represents real-world events of any organisation or an enterprise whereas concurrency control is the management of concurrent transaction execution. Transaction processing systems execute database transactions with large databases and hundreds of concurrent users, for example, railway and air reservations systems, banking system, credit card processing, stock market monitoring, super market inventory and checkouts and so on.

Transaction: A **transaction** is a logical unit of work of database processing that includes one or more database access operations. A transaction can be defined as an action or series of actions that is carried out by a single user or application program to perform operations for accessing the contents of the database. The operations can include retrieval, (Read), insertion (Write), deletion and modification. A transaction must be either completed or aborted.

A transaction can include the following basic database access operations:

Operations	**Descriptions**
Retrieve	To retrieve data stored in a database.
Insert	To store new data in database.
Delete	To delete existing data from database.
Update	To modify existing data in database.
Commit	To save the work done permanently.
Rollback	To undo the work done.

Let's take an example of a simple transaction. Suppose a bank employee transfers Rs 500 from A's account to B's account. This very simple and small transaction involves several low-level tasks.

A's Account

```
Open_Account(A)
Old_Balance = A.balance
New_Balance = Old_Balance - 500
A.balance = New_Balance
Close_Account(A)
```

B's Account

```
Open_Account(B)
Old_Balance = B.balance
New_Balance = Old_Balance + 500
B.balance = New_Balance
Close_Account(B)
```

ACID Properties: A transaction is a very small unit of a program and it may contain several low level tasks. A transaction in a database system must maintain **A**tomicity, **C**onsistency, **I**solation, and **D**urability − commonly known as ACID properties − in order to ensure accuracy, completeness, and data integrity.

- **Atomicity** − This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.

  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	**T2**
Read(A)	Read(B)
A:= A-100	B:= B+100
Write(A)	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 database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.

  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.

  For example: The total amount must be maintained before or after the transaction.

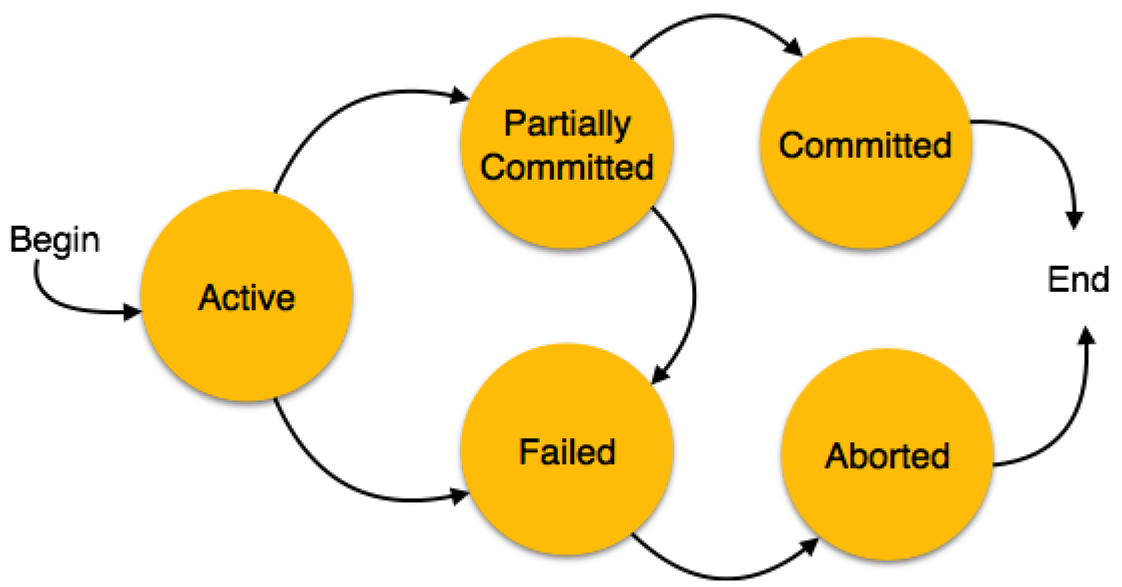
  1. Total before T occurs = 600+300=900
  2. Total after T occurs= 500+400=900

  Therefore, the database is consistent. In the case when T1 is completed but T2 fails, then inconsistency will occur.

- **Durability** − The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.

- **Isolation** − In a database system where more than one transaction is being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

States of transactions: A transaction in a database can be in one of the following states



- **Active** − In this state, the transaction is being executed. This is the initial state of every transaction.
- **Partially Committed** − When a transaction executes its final operation, it is said to be in a partially committed state.
- **Failed** − A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.
- **Aborted** − If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called aborted. The database recovery module can select one of the two operations after a transaction aborts −
    - Re-start the transaction
    - Kill the transaction
- **Committed** − If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.

SERIALIZABILITY: If multiple transactions are being executed in a multi-programming environment, then it is obvious there are possibilities that one transaction can be interleaved with the other. The execution sequence of instruction of a transaction cannot be changed but can be executed in a random fashion.

- Schedule: A sequential execution of a transaction with respect to time is called a schedule which can have many transactions in it, each compromising several tasks in it.
- Serial schedule: Transactions are executed in a serial way, which are ordered one after the other. It is the schedule where transactions are aligned such that when the first transaction finishes its cycle, second starts. When second finishes third starts and so on.

If two transactions are on a different segment of data, then executing their instructions in a random fashion will cause no problem. But if on the same data, then the result will vary and dissatisfies the property of consistency. So the parallel execution of a transaction schedule is allowed in order to resolve this problem based on the condition that transactions are either serializable or have equivalence relation among them.

Equivalence schedules can be of three types as follows

1. **Result equivalence** – After an execution, if and only if two schedules have the same result is it said to be result equivalence. This is not significant since it produces the same result for some set of values and different results for another set of values.
2. **Conflict equivalence** – Two schedules are said to be conflicting if they satisfy below criteria

- Both belong to separate transactions.
- Both should access the same data item.
- At least one of them is a write operation.

3. **View equivalence** – If transactions in both schedules perform same actions in a similar manner then it is considered as view equivalence.

<u>**Concurrency control**</u>: **Concurrency control** is the process of managing simultaneous execution of transactions (such as queries, updates, inserts, deletes and so on) in a multiprocessing database system without having them interfere with one another.
This property of DBMS allows many transactions to access the same database at the same time without interfering with each other.

The primary goal of concurrency is to ensure the atomicity of the execution of transactions in a multi-user database environment. Concurrency controls mechanisms attempt to interleave (parallel) READ and WRITE operations of multiple transactions so that the interleaved execution yields results that are identical to the results of a serial schedule execution.

Problems of concurrency control: Several problems can occur when concurrent transactions are executed in an uncontrolled manner. Following are the three problems in concurrency control.

1. Lost updates
2. Dirty read
3. Unrepeatable read

1. Lost updates:

- When two transactions that access the same database items contain their operations in a way that makes the value of some database item incorrect, then the lost update problem occurs.
- If two transactions T1 and T2 read a record and then update it, then the effect of updating of the first record will be overwritten by the second update.

Example:

Transaction-X	**Time**	**Transaction-Y**
——	t1	——
Read A	t2	——
——	t3	Read A
Update A	t4	——
——	t5	Update A
——	t6	——

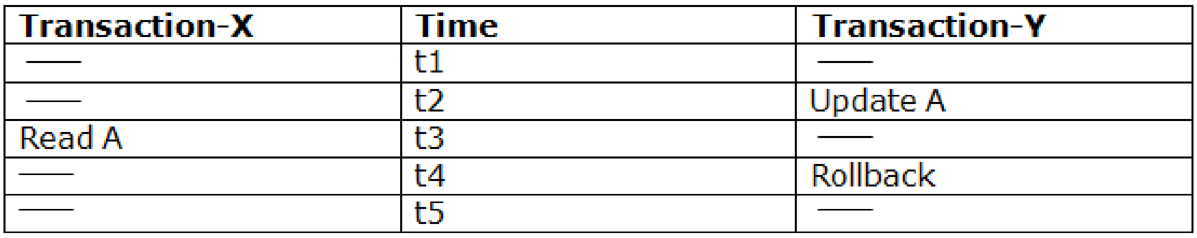
Here

- At time t2, transaction-X reads A's value.
- At time t3, Transaction-Y reads A's value.
- At time t4, Transactions-X writes A's value on the basis of the value seen at time t2.
- At time t5, Transactions-Y writes A's value on the basis of the value seen at time t3.
- So at time T5, the update of Transaction-X is lost because Transaction y overwrites it without looking at its current value.
- Such type of problem is known as Lost Update Problem as update made by one transaction is lost here.

2. Dirty Read:

- The dirty read occurs in the case when one transaction updates an item of the database, and then the transaction fails for some reason. The updated database item is accessed by another transaction before it is changed back to the original value.
- A transaction T1 updates a record which is read by T2. If T1 aborts then T2 now has values which have never formed part of the stable database.

Example:



Transaction-X	Time	Transaction-Y
——	t1	——
——	t2	Update A
Read A	t3	——
——	t4	Rollback
——	t5	——

- At time t2, transaction-Y writes A's value.
- At time t3, Transaction-X reads A's value.
- At time t4, Transactions-Y rollbacks. So, it changes A's value back to that of prior to t1.
- So, Transaction-X now contains a value which has never become part of the stable database.
- Such type of problem is known as Dirty Read Problem, as one transaction reads a dirty value which has not been committed.

3. Inconsistent Retrievals Problem (Unrepeatable read):

- Inconsistent Retrievals Problem is also known as unrepeatable read. When a transaction calculates some summary function over a set of data while the other transactions are updating the data, then the Inconsistent Retrievals Problem occurs.
- A transaction T1 reads a record and then does some other processing during which the transaction T2 updates the record. Now when the transaction T1 reads the record, then the new value will be inconsistent with the previous value.

Concurrency Control Protocol

Concurrency control protocols ensure atomicity, isolation, and serializability of concurrent transactions. The concurrency control protocol can be divided into three categories:

- Lock based protocol
- Time-stamp protocol
- Validation based protocol

Lock-Based Protocol

In this type of protocol, any transaction cannot read or write data until it acquires an appropriate lock on it. There are two types of lock:

1. Shared lock:

- It is also known as a Read-only lock. In a shared lock, the data item can only read by the transaction.
- It can be shared between the transactions because when the transaction holds a lock, then it can't update the data on the data item.

2. Exclusive lock:

- In the exclusive lock, the data item can be both reads as well as written by the transaction.
- This lock is exclusive, and in this lock, multiple transactions do not modify the same data simultaneously.

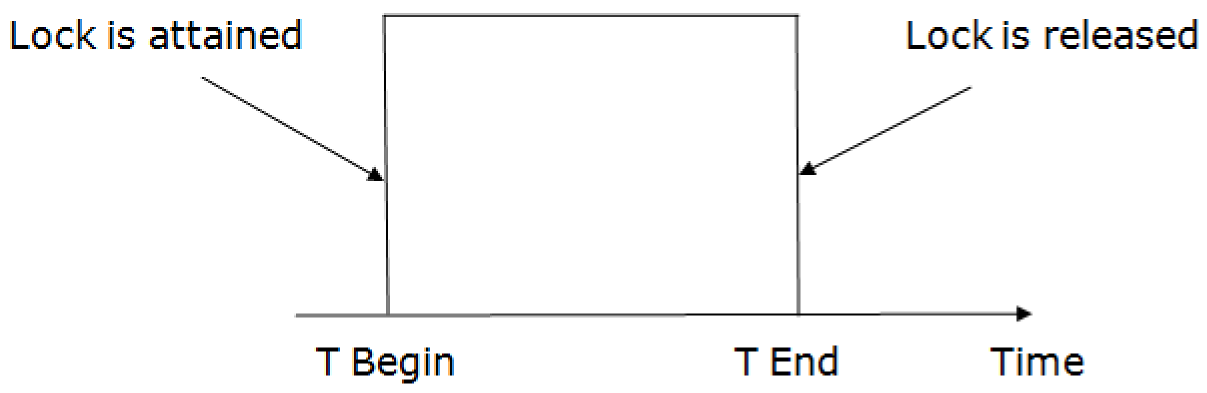
There are four types of lock protocols available:

1. Simplistic lock protocol

It is the simplest way of locking the data while transaction. Simplistic lock-based protocols allow all the transactions to get the lock on the data before insert or delete or update on it. It will unlock the data item after completing the transaction.

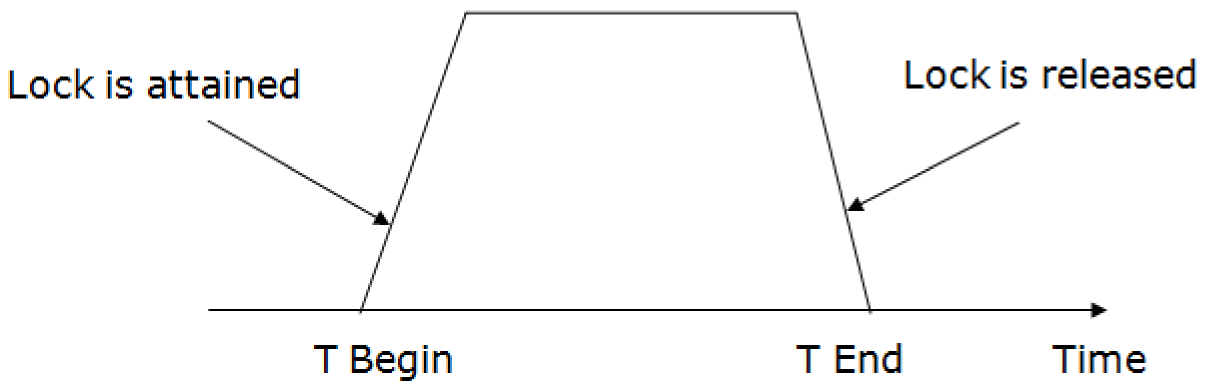
2. Pre-claiming Lock Protocol

- Pre-claiming Lock Protocols evaluate the transaction to list all the data items on which they need locks.
- Before initiating an execution of the transaction, it requests DBMS for all the lock on all those data items.
- If all the locks are granted, then this protocol allows the transaction to begin. When the transaction is completed then it releases all the lock.
- If all the locks are not granted, then this protocol allows the transaction to rolls back and waits until all the locks are granted.



3. Two-phase locking (2PL)

- The two-phase locking protocol divides the execution phase of the transaction into three parts.
- In the first part, when the execution of the transaction starts, it seeks permission for the lock it requires.
- In the second part, the transaction acquires all the locks. The third phase is started as soon as the transaction releases its first lock.
- In the third phase, the transaction cannot demand any new locks. It only releases the acquired locks.



There are two phases of 2PL:

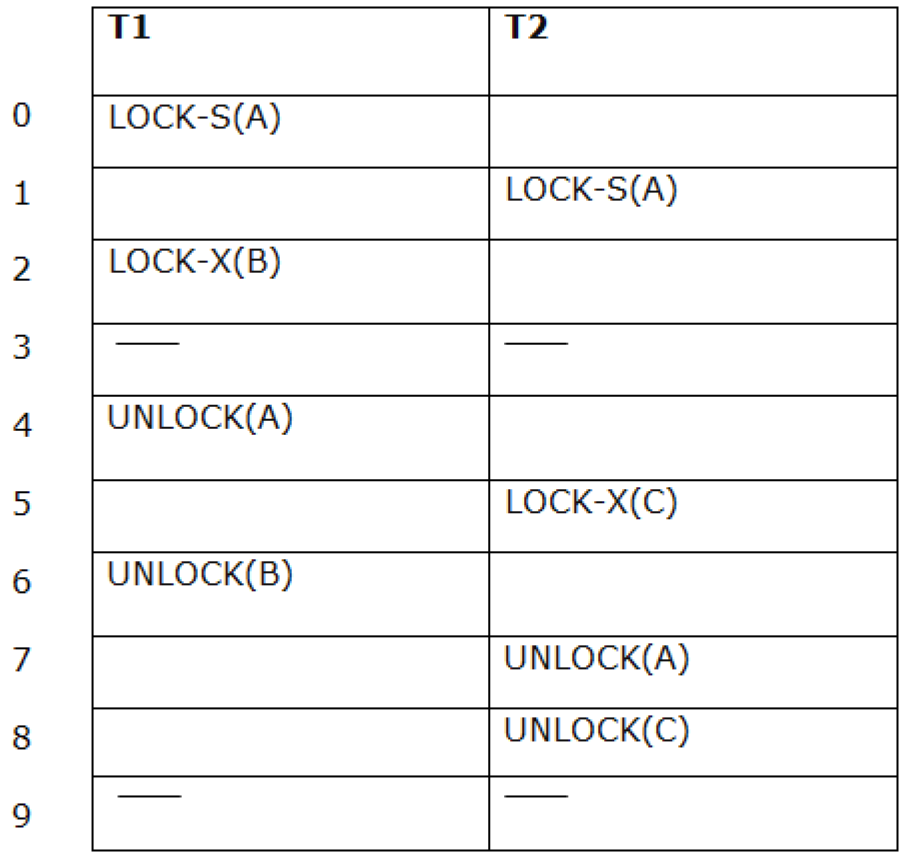
Growing phase: In the growing phase, a new lock on the data item may be acquired by the transaction, but none can be released.

Shrinking phase: In the shrinking phase, existing lock held by the transaction may be released, but no new locks can be acquired.

In the below example, if lock conversion is allowed then the following phase can happen:

1. Upgrading of lock (from S(a) to X (a)) is allowed in growing phase.
2. Downgrading of lock (from X(a) to S(a)) must be done in shrinking phase.

Example:



	T1	T2
0	LOCK-S(A)	
1		LOCK-S(A)
2	LOCK-X(B)	
3	——	——
4	UNLOCK(A)	
5		LOCK-X(C)
6	UNLOCK(B)	
7		UNLOCK(A)
8		UNLOCK(C)
9	——	——

The following way shows how unlocking and locking work with 2-PL.

Transaction T1:

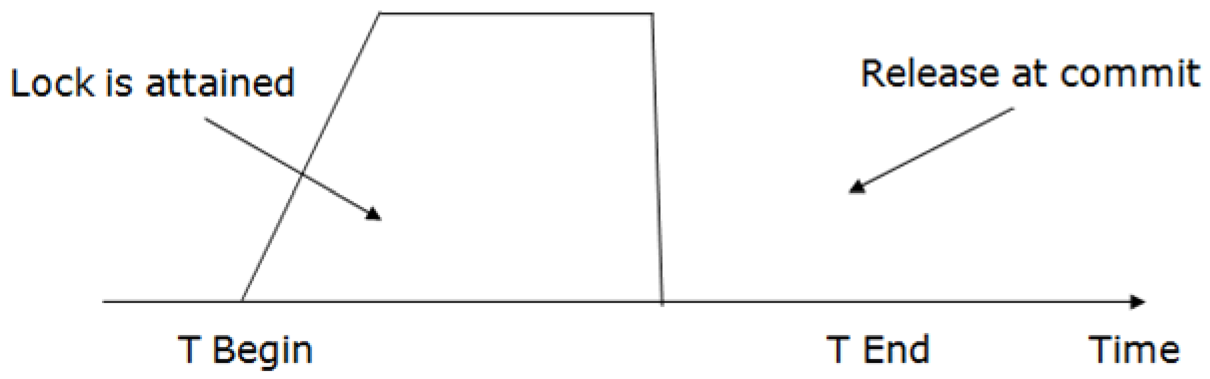
- **Growing phase:** from step 1-3
- **Shrinking phase:** from step 5-7
- **Lock point:** at 3

Transaction T2:

- **Growing phase:** from step 2-6
- **Shrinking phase:** from step 8-9
- **Lock point:** at 6

4. Strict Two-phase locking (Strict-2PL)

- The first phase of Strict-2PL is similar to 2PL. In the first phase, after acquiring all the locks, the transaction continues to execute normally.
- The only difference between 2PL and strict 2PL is that Strict-2PL does not release a lock after using it.
- Strict-2PL waits until the whole transaction to commit, and then it releases all the locks at a time.
- Strict-2PL protocol does not have shrinking phase of lock release.



It does not have cascading abort as 2PL does.

Timestamp Ordering Protocol

- The Timestamp Ordering Protocol is used to order the transactions based on their Timestamps. The order of transaction is nothing but the ascending order of the transaction creation.
- The priority of the older transaction is higher that's why it executes first. To determine the timestamp of the transaction, this protocol uses system time or logical counter.
- The lock-based protocol is used to manage the order between conflicting pairs among transactions at the execution time. But Timestamp based protocols start working as soon as a transaction is created.
- Let's assume there are two transactions T1 and T2. Suppose the transaction T1 has entered the system at 007 times and transaction T2 has entered the system at 009 times. T1 has the higher priority, so it executes first as it is entered the system first.
- The timestamp ordering protocol also maintains the timestamp of last 'read' and 'write' operation on a data.

Basic Timestamp ordering protocol works as follows:

1. Check the following condition whenever a transaction Ti issues a **Read (X)** operation:

- If W_TS(X) >TS(Ti) then the operation is rejected.
- If W_TS(X) <= TS(Ti) then the operation is executed.
- Timestamps of all the data items are updated.

2. Check the following condition whenever a transaction Ti issues a **Write(X)** operation:

- If TS(Ti) < R_TS(X) then the operation is rejected.
- If TS(Ti) < W_TS(X) then the operation is rejected and Ti is rolled back otherwise the operation is executed.

Where,

TS(Ti) denotes the timestamp of the transaction Ti.

R_TS(X) denotes the Read time-stamp of data-item X.

W_TS(X) denotes the Write time-stamp of data-item X.

Advantages and Disadvantages of TO protocol:

TO protocol ensures serializability since the precedence graph is as follows:

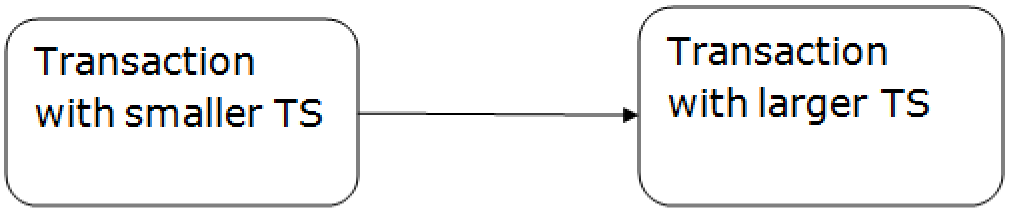


Image: Precedence Graph for TS ordering

- TS protocol ensures freedom from deadlock that means no transaction ever waits.
- But the schedule may not be recoverable and may not even be cascade- free.

Validation Based Protocol

Validation phase is also known as optimistic concurrency control technique. In the validation based protocol, the transaction is executed in the following three phases:

1. **Read phase:** In this phase, the transaction T is read and executed. It is used to read the value of various data items and stores them in temporary local variables. It can perform all the write operations on temporary variables without an update to the actual database.
2. **Validation phase:** In this phase, the temporary variable value will be validated against the actual data to see if it violates the serializability.
3. **Write phase:** If the validation of the transaction is validated, then the temporary results are written to the database or system otherwise the transaction is rolled back.

Here each phase has the following different timestamps:

Start(Ti): It contains the time when Ti started its execution.

Validation (T_i): It contains the time when Ti finishes its read phase and starts its validation phase.

Finish(Ti): It contains the time when Ti finishes its write phase.

- This protocol is used to determine the time stamp for the transaction for serialization using the time stamp of the validation phase, as it is the actual phase which determines if the transaction will commit or rollback.
- Hence TS(T) = validation(T).
- The serializability is determined during the validation process. It can't be decided in advance.
- While executing the transaction, it ensures a greater degree of concurrency and also less number of conflicts.
- Thus it contains transactions which have less number of rollbacks.

Thomas write Rule

Thomas Write Rule provides the guarantee of serializability order for the protocol. It improves the Basic Timestamp Ordering Algorithm.

The basic Thomas write rules are as follows:

- If TS(T) < R_TS(X) then transaction T is aborted and rolled back, and operation is rejected.
- If TS(T) < W_TS(X) then don't execute the W_item(X) operation of the transaction and continue processing.
- If neither condition 1 nor condition 2 occurs, then allowed to execute the WRITE operation by transaction Ti and set W_TS(X) to TS(T).

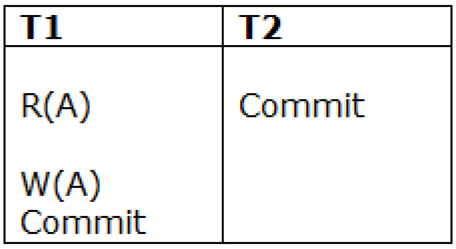
If we use the Thomas write rule then some serializable schedule can be permitted that does not conflict serializable as illustrate by the schedule in a given figure:

T1	**T2**
R(A)	
	W(A) Commit
W(A) Commit	

Figure: A Serializable Schedule that is not Conflict Serializable

In the above figure, T1's read and precedes T1's write of the same data item. This schedule does not conflict serializable.

Thomas write rule checks that T2's write is never seen by any transaction. If we delete the write operation in transaction T2, then conflict serializable schedule can be obtained which is shown in below figure.



T1	T2
R(A)	Commit
W(A)	
Commit	

Figure: A Conflict Serializable Schedule

Multiple Granularity

Let's start by understanding the meaning of granularity.

Granularity: It is the size of data item allowed to lock.

Multiple Granularity:

- It can be defined as hierarchically breaking up the database into blocks which can be locked.
- The Multiple Granularity protocol enhances concurrency and reduces lock overhead.
- It maintains the track of what to lock and how to lock.
- It makes easy to decide either to lock a data item or to unlock a data item. This type of hierarchy can be graphically represented as a tree.

Recovery with Concurrent Transaction

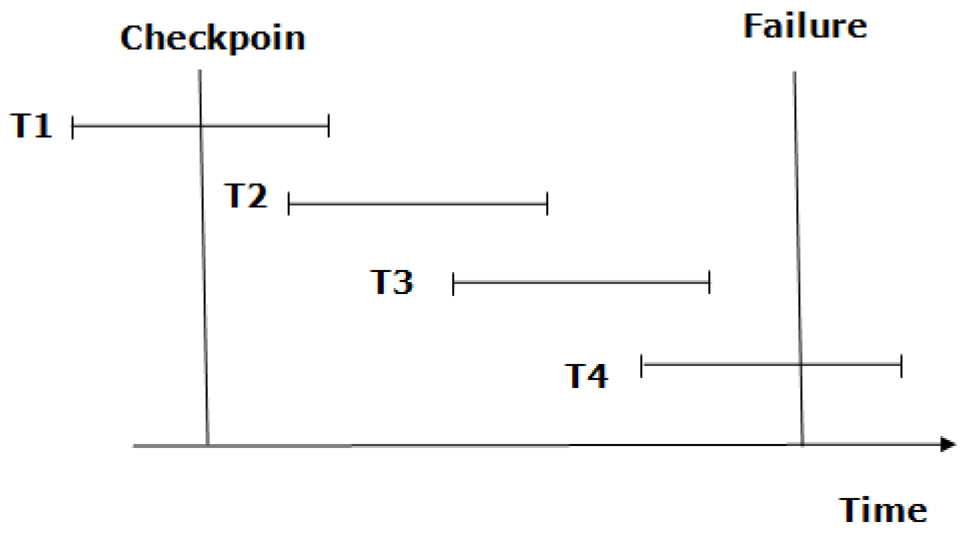
- Whenever more than one transaction is being executed, then the interleaved of logs occur. During recovery, it would become difficult for the recovery system to backtrack all logs and then start recovering.
- To ease this situation, 'checkpoint' concept is used by most DBMS.

Checkpoint

- The checkpoint is a type of mechanism where all the previous logs are removed from the system and permanently stored in the storage disk.
- The checkpoint is like a bookmark. While the execution of the transaction, such checkpoints are marked, and the transaction is executed then using the steps of the transaction, the log files will be created.
- When it reaches to the checkpoint, then the transaction will be updated into the database, and till that point, the entire log file will be removed from the file. Then the log file is updated with the new step of transaction till next checkpoint and so on.
- The checkpoint is used to declare a point before which the DBMS was in the consistent state, and all transactions were committed.

Recovery using Checkpoint

In the following manner, a recovery system recovers the database from this failure:



- The recovery system reads log files from the end to start. It reads log files from T4 to T1.
- Recovery system maintains two lists, a redo-list, and an undo-list.
- The transaction is put into redo state if the recovery system sees a log with <Tn, Start> and <Tn, Commit> or just <Tn, Commit>. In the redo-list and their previous list, all the transactions are removed and then redone before saving their logs.
- **For example:** In the log file, transaction T2 and T3 will have <Tn, Start> and <Tn, Commit>. The T1 transaction will have only <Tn, commit> in the log file. That's why the transaction is committed after the checkpoint is crossed. Hence it puts T1, T2 and T3 transaction into redo list.
- The transaction is put into undo state if the recovery system sees a log with <Tn, Start> but no commit or abort log found. In the undo-list, all the transactions are undone, and their logs are removed.
- **For example:** Transaction T4 will have <Tn, Start>. So T4 will be put into undo list since this transaction is not yet complete and failed amid.

Failure Classification

To find that where the problem has occurred, we generalize a failure into the following categories:

1. Transaction failure
2. System crash
3. Disk failure

1. Transaction failure

The transaction failure occurs when it fails to execute or when it reaches a point from where it can't go any further. If a few transaction or process is hurt, then this is called as transaction failure.

Reasons for a transaction failure could be -

1. **Logical errors:** If a transaction cannot complete due to some code error or an internal error condition, then the logical error occurs.

2. **Syntax error:** It occurs where the DBMS itself terminates an active transaction because the database system is not able to execute it. **For example,** The system aborts an active transaction, in case of deadlock or resource unavailability.

2. System Crash

- System failure can occur due to power failure or other hardware or software failure. **Example:** Operating system error.

  Fail-stop assumption: In the system crash, non-volatile storage is assumed not to be corrupted.

3. Disk Failure

- It occurs where hard-disk drives or storage drives used to fail frequently. It was a common problem in the early days of technology evolution.
- Disk failure occurs due to the formation of bad sectors, disk head crash, and unreachability to the disk or any other failure, which destroy all or part of disk storage.

Log-Based Recovery

- The log is a sequence of records. Log of each transaction is maintained in some stable storage so that if any failure occurs, then it can be recovered from there.
- If any operation is performed on the database, then it will be recorded in the log.
- But the process of storing the logs should be done before the actual transaction is applied in the database.

Let's assume there is a transaction to modify the City of a student. The following logs are written for this transaction.

- When the transaction is initiated, then it writes 'start' log.

  <Tn, Start>

- When the transaction modifies the City from 'Noida' to 'Bangalore', then another log is written to the file.

  <Tn, City, 'Noida', 'Bangalore' >

- When the transaction is finished, then it writes another log to indicate the end of the transaction.

  <Tn, Commit>

1. Deferred database modification:

- The deferred modification technique occurs if the transaction does not modify the database until it has committed.
- In this method, all the logs are created and stored in the stable storage, and the database is updated when a transaction commits.

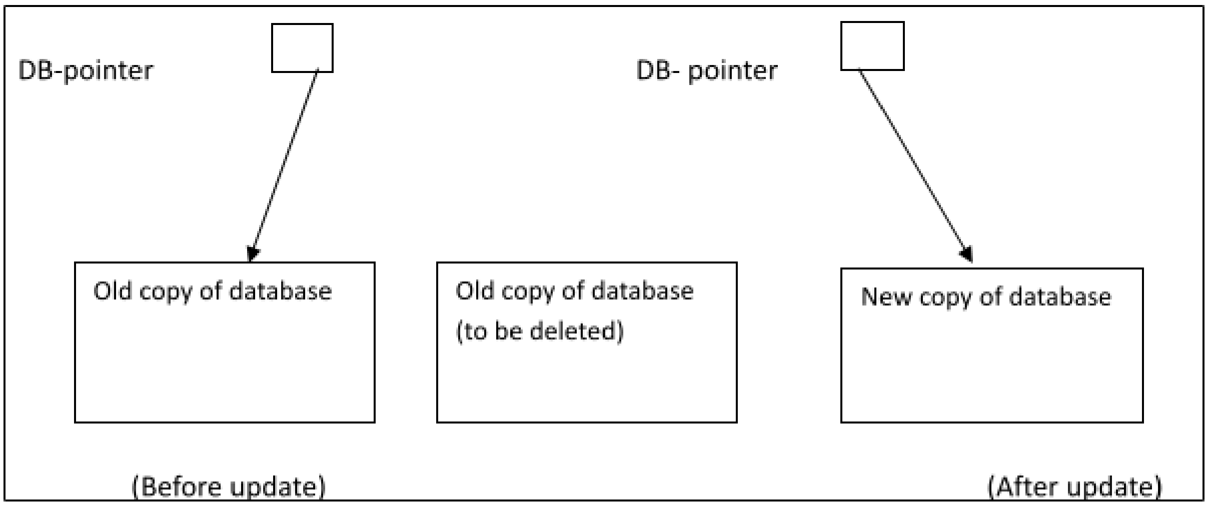
2. Immediate database modification:

- The Immediate modification technique occurs if database modification occurs while the transaction is still active.
- In this technique, the database is modified immediately after every operation. It follows an actual database modification.

3. Shadow Paging:

- This is the method where all the transactions executed in the primary memory, once all the transactions are completely executed, it will be updated to the database

  This technique is based on making copies of the database, called shadow copies.
  - In this technique, a pointer called db-pointer is maintained on disk, which points to the current copy of the database.
  - In the shadow-copy technique, a transaction that wants to update the database first creates a complete copy of the database. All updates are done on the new database copy, leaving the original copy, the shadow copy, untouched. If at any point the transaction has to be aborted, the system merely deletes the new copy. The old copy of the database has not been affected.



- If the transaction completes, the operating system is asked to make sure that all pages of the new copy of the database have been written out to disk.
- After the operating system has written all the pages to the disk, the database system updates the db-pointer to print to the new copy of the database; the new copy then becomes the current copy of the database. The old copy of the database is then deleted.
- The transaction is said to have been committed at the point where the updated db-pointer is written to disk.
- If the transaction fails at any time before db-pointer is updated, the old contents of the database are not affected. We can abort the transaction by just deleting the new copy of the database.
- Once the transaction has been committed, all the updates that is performed are in the database pointed to by db-pointer.
- Thus, either all updates of the transaction are reflected or none of the effects are reflected, regardless of transaction failure.
- The atomicity and durability property is implemented in shadow copy technique.

DEADLOCK HANDLING:-

❖ A system in a situation where there exists a set of transactions such that every transactions in the set is waiting for another transaction in the set .Thus ,we have arrived at a state where neither of these transactions can ever proceed with its normal execution .This situation is called as deadlock.

❖ For example ,there exists a set of waiting transactions { T0,T1,T2,………TN } such that T0 is waiting for a data item that T1 holds and T1 is waiting for a data item that T2 holds, and so on …Tn-1 is waiting for a data item that Tn holds ,and Tn is waiting for a data that T0 holds. None of the transactions can make progress in such a situation .

❖ There are three principles methods for dealing with the deadlock problem .These are:-

1. Deadlock prevention
2. Deadlock Detection
3. Deadlock Recovery

1. *D eadlock prevention:-*

- Deadlock prevention protocol is used to ensure that the system will never enter a deadlock state.
- There are two approaches to deadlock prevention :-
  - The first approach ensures that no cyclic wait can occur by ordering the request for locks that means each transaction locks all its data items before it begins execution .Either all are locked in one step or none are locked.

  - The second approach for preventing deadlock is to use a preemption and transaction rollbacks in preemption ,when a transaction T2 requests a lock that transaction T1 holds the lock granted to T1 may be preempted by rolling back of T1,and granting the lock to T2.To control the preemption , we assign a unique timestamp to each transaction should wait or roll back.

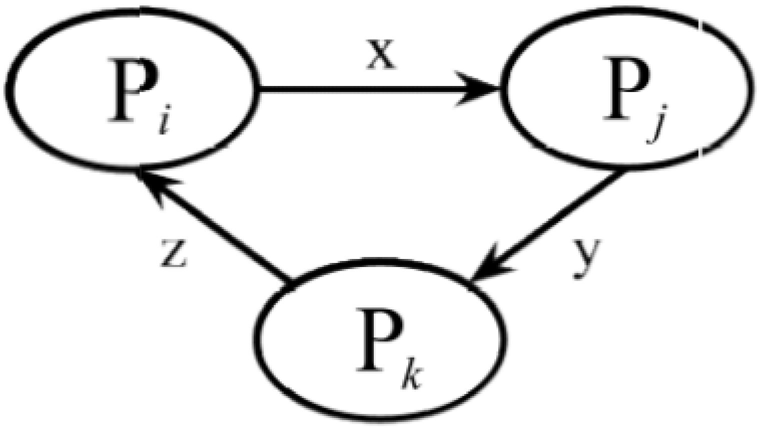
Two different deadlock prevention schemes using timestamps has been proposed:-

- The wait-die scheme is a non-preemptive technique .When a transaction Ti requests a data item currently held by Tj,Ti allowed to wait only if it has a timestamp smaller than that of Tj(that is Ti older than Tj)otherwise ,Ti is rolled back(dies).
- The wound-wait is a non preemptive technique .It is counterpart of the wait-die scheme .when a transaction Ti requests a data item currently held by Tj,Ti is allowed to wait only if it has a timestamp larger than that of Tj(that is ,Ti is younger than Tj),otherwise Tj is rolled back(dies).

2. D eadlock detection:-

- We can allow the system to enter a deadlock state, and then deadlock detection and deadlock recovery scheme. try to recover by using a

- An algorithm that examines the state of the system is invoked periodically to determine whether a deadlock has occurred .if one has ,then the system must attempt to recover from the deadlock
- Deadlocks can be described in terms of a directed graph called as wait-for-graph.
- The wait-for-graph consists of a pair G=(V,E),where V is a set of vertices and E is a set of edges .The set of vertices consists of all the transaction in the system.
- When transaction Ti requests a data item currently being held by transaction Tj, then the edge ti-->Tj is inserted in the wait-for –graph.



- A deadlock exists in the system if and only if the wait-for –graph contains a cycle. Each transaction involved deadlocked. in the cycle .Each transaction involved in the cycle is said to be

- To detect deadlocks, the system needs to maintain the wait-for-graph, and periodically to invoke an algorithm that searches for a cycle in the graph.
- In the example ,transaction T2,T3,nd T4 are all deadlocked ,because it contains the cycle
    - Transaction T2 is waiting for transaction T4
    - Transaction T4 is waiting for transaction T3
    - And ,transaction T3 is waiting for transaction T2,therefore the graph contains the cycle i.e; T2 ☾ T4 ☾ T3 ☾ T2

3. D eadlock recovery :-

- When a detection algorithm determines that a deadlock exists, the system must recover from the deadlock.
- The most common solution is to rollback one or more transactions to break the deadlock.
- Three actions need to be taken :-

Selection of a victim :-

✔ Given a set of deadlocked, we must determine which transaction transactions to roll back to break the deadlock.

✔ We should rollback those transactions that will give the minimum cost.

Rollback:-

✔ once we have decided that a particular transaction must be rolled back ,we must determine how far this transaction should be rolled back.

✔ The simplest solution is a total rollback; Abort the transaction and then restart it.

✔ Partial rollback requires the system to maintain additional information about the state of all ruining transactions.

Starvation:-

✔ In the case of a selection of victim, it may happen that the same transaction is always picked as a victim.

✔ As a result, this transaction never completes its designated task, thus this is a s tarvation.

✔ We must ensure that a transaction can be picked as a victim only a small (finite) number of times.The most common solution is to include the number of rollback in the cost factor.

Parallel Database: Parallel database system improves performance of data processing using multiple resources in parallel, like multiple CPU and disks are used parallel.

It also performs many parallelization operations like, data loading and query processing.

Goals of Parallel Databases

Improve performance:
The performance of the system can be improved by connecting multiple CPU and disks in parallel. Many small processors can also be connected in parallel.

Improve availability of data:
Data can be copied to multiple locations to improve the availability of data.
For example: if a module contains a relation (table in database) which is unavailable then it is important to make it available from another module.

Improve reliability:
Reliability of system is improved with completeness, accuracy and availability of data.

D ATA WAREHOUSE:

- Data warehouse is a repository of information gathered from multiple sources, stored under a unified schema at a single site for efficient querying and analysis.
- Data warehouse provides access to data for complex analysis, knowledge discovery and decision making.
- In comparison to traditional databases, data warehouse generally contain very large amount of data from multiple sources that may include databases from different data models .

D ATA MORT:-

- A data mort is a focused subset of a data warehouse that deals with single area of data and is organised for quick analysis.
- It can be small data warehouse itself.
- It facilitates data reporting.
- It makes data design easier and simpler.

C OMPONENTS OF DATA WAREHOUSE:-

- The data warehouse architecture shows the gathering of data, the data storage of data and the querying and the data analysis support.
- The following are the different issues to be addressed in building a warehouse:-
- When and how to gather data:-
    - In a source driven architecture for gathering the data, the data sources transmit new information either continually or periodically.
    - In a destination driven architecture, the data warehouse periodically sends requests for new data to the sources.
- What schema to use:-
    - The data sources are likely to have different schemas and may even use different data models.
    - The task of data ware house is to perform schema integration and to convert data to the integrated schema before they are stored.
- Data transformations and cleansing:-
    - The task of data correcting and pre-processing is called as data cleansing.
    - Data sources may deliver the data with numerous minor inconsistencies, which can be corrected.
    - The approximate matching of data required for data correction is called as fuzzy look.
    - Data warehouse have graphical tools to support data transformation.
    - Such graphical tools allow transformation to be specified as boxes, and edges can be created between boxes to indicate the flow of data.
- Propagation of updates :-
    - Updates on relations at the data sources must be propagated to the data ware house.
    - If the relations at the data warehouse are exactly the same as those at the data source, then the propagation is straight forward.
    - If they are not, the problem of propagating updates is basically the non-maintenance p roblem.
- What data to summarize:-
    - The raw data generated by a transaction system may be too large to store online.
    - However, we can answer many queries by maintained just summary data obtained by aggregation on a relation, rather than maintaining the entire relation.

C HARACTERISTICS OF DATA WAREHOUSES:-

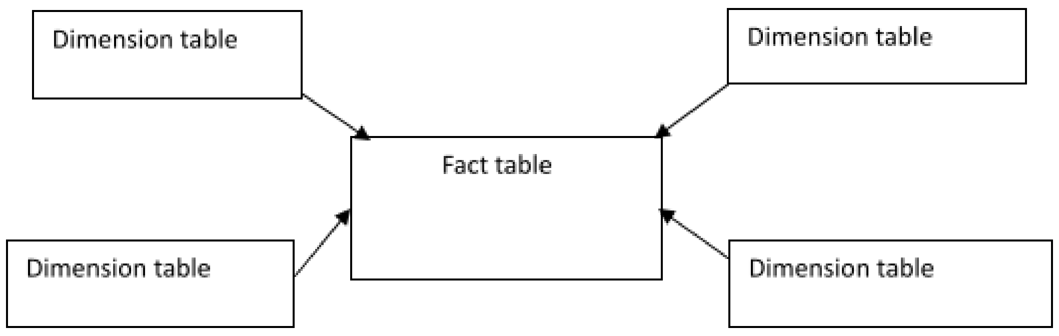
- W.H.inmow characterised data warehouse as “a subject oriented, integrated, non volatile, time variant collection of data in support of management designs “.
  - S ubject oriented: - it means all data persistent to a subject/business are a collected and stored as a single unit.
  - I ntegrated: - it means data from multiple sources are transformed and stored as a globally accepted fashion.
  - S tatic/non-volatile:-It means that once data entered into the warehouse doesn’t change frequently .It is a periodically update if required.
  - T ime-variant: - Data warehouse maintains historical data which are used to analyze the business or market trends and facilitate future productions.

E TL (Extraction, Transformation and Loading):-

- The different steps involved in getting data into a data warehouse are called as extract, transform and load or ETL tasks.
- Data need to be taken from various sources, and this process is known as data extraction.
- The data from various sources may be in different models and different schemas and they must be transformed into a common format.
- After the transformation of data, it can be loaded into the data warehouse.

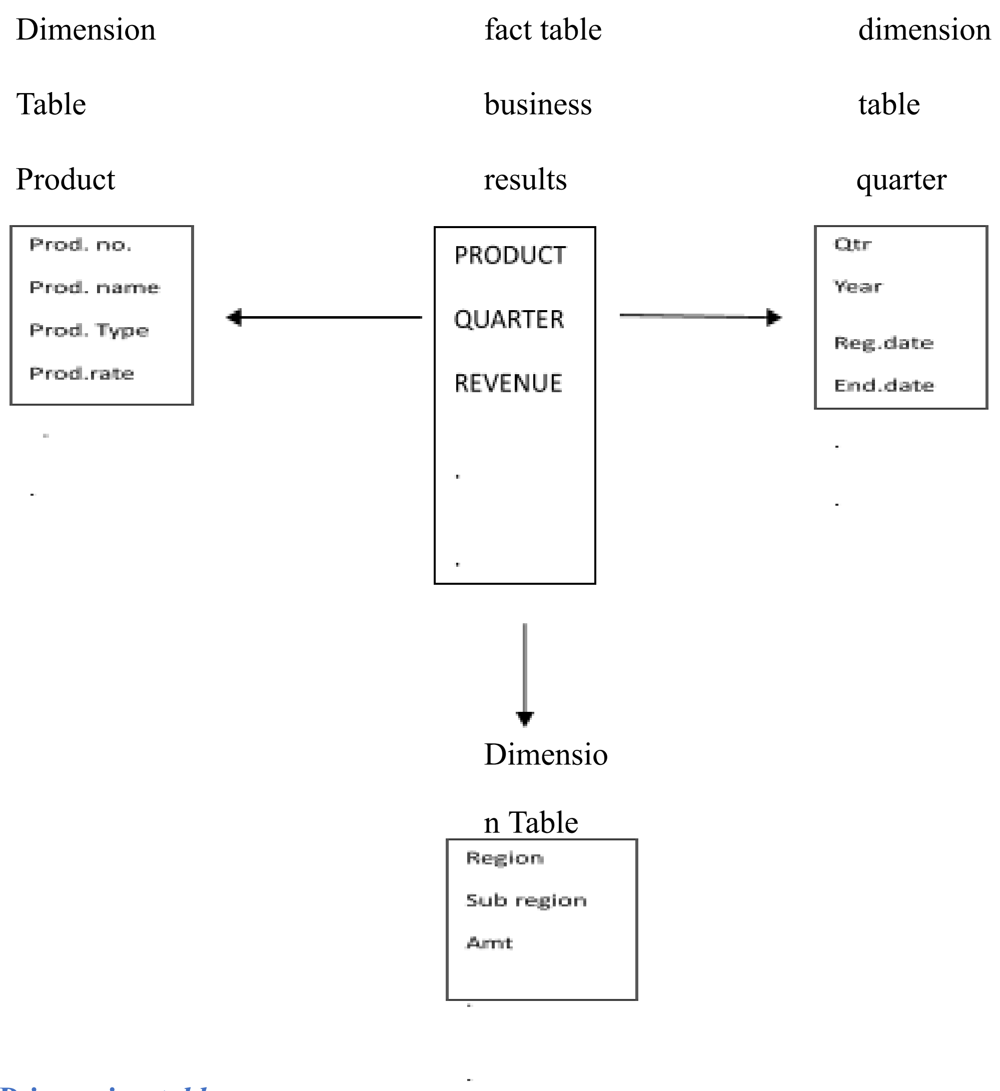
D ATA WAREHOUSE SCHEMAS :-

- **S tar schemas :-**
  - Star schema is the simplest warehouse schema.
  - It is also known as dimensional modelling /star schema there is a large central fact table with many dimension tables surrounding it.



- ***F act table:-***
  - Each data warehouse or data mart includes one or more fact tables.
  - Fact tables containing multidimensional data about an organization’s business operation and usually very large.
  - A fact table can be thought of as having rows/tuples, one per a recorded fact.

E xample :- (ex of a business results)



D imension table:-

- To minimize storage requirements, dimensions attributes of fact table are usually short identifier that are foreign keys into other table called dimension tables.
- Dimensions tables contain attributes that describe fact records in the fa t table.

D ata warehouse application:-

- Data warehouse provide a cess to data for complex analysis, knowledge discovery and decision making.
- Several types of applications –OLAP, DSS and data mining are supported.
  - ❖ O LAP (On-line Analytical Processing)is used to describe the analysis of complex data from

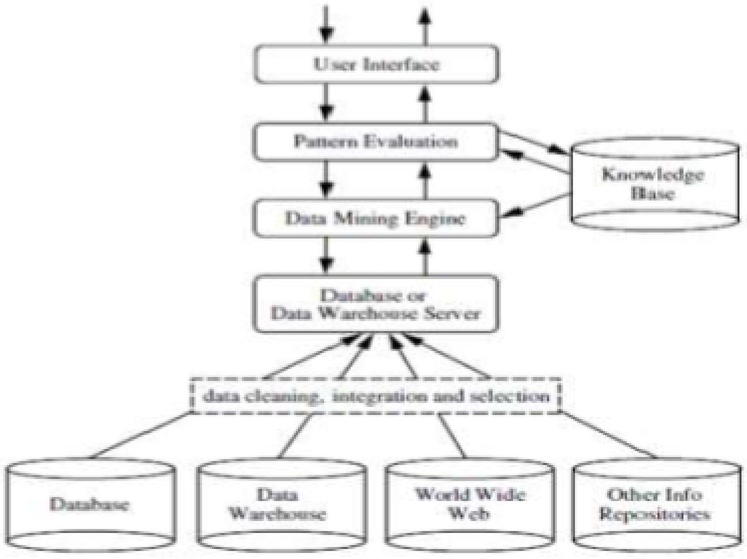
the data warehouse. In the computing capabilities for analysis that require more storage ad processing power than can be economically and efficiently located on an individual desktop.

- ❖ D SS (Decision-Support-System) support an organizations leading decision maker with higher level data for complex and important decisions.
- ❖ D ata mining refers to the mining /discovery of new information in terms of patterns or rules from a vast amount of data .data mining must be carried out efficiently on large files and databases.

D ATA MINING

- Data mining refers to the mining /discovery of new information in terms of pattern or rules from vast amount of data.
- Data mining must be carried out efficiently on large files and databases.
- Data mining means it helps in extracting meaningful new patterns of data that cannot be found necessarily by merely querying or processing data or metadata in the data warehouse.
- The main concept in data mining is the nature of the information that is discovered, the types of problems faced when trying to mine databases and the types of applications of data mining.
- Data mining can be used in conjunction with certain types of decisions.
- In the broader sense we can call data mining as "knowledge discovery process".

D ata mining architecture:-



- The first tier is the database tier where data and metadata is prepared and stored.
- The database comprises of 4 layers.
- The metadata layer is the common layer that contains the data about the data.
- The staging area layer is used for temporarily holding the data sources from various source systems. It can be held in any form e.g. flat files, tables in rdbms.
- The prepared input data layer is used as input data for data mining. This data is transformed, cleansed, consolidated and loaded into a structured schema during data preparation process.
- The data mining output can be captured in the data mining results layer so that it can be made

available to the users for visualization and analysis.

- The second tier is called data mining application where the algorithms process the data and store the results in the database.
- Data mining application has two primary components.
    a. data manager
    b. Data mining tools/algorithms.
- The data manager manages the data in the database tier and controls the data in the required format of the data mining task. Then the data manager manages the result generated by the data mining task will need to be facilitated to the target systems /front end.
- The data mining tools/algorithms is the heart of the complete architecture. The main functionality will be analyzing the data and generate the results.
- Numerous tools are available to give best possible results as output e.g. SAS, SPSS, Teradata miner and IBM Intelligent miner .these tools merely facilitate the application of algorithms on the input data.
- The third tier is the front-end layer, which facilitates the parameter settings for data mining application visualization of the results in interpretable form.
- For executing a data mining task, the user needs to provide respective input parameters. Then observe the effect on the results and change the parameter if needed based on the interpretations and understanding of the results. This facility is provided in the front end.
- The results of data mining task need formatting, conversion to user understandable form and reporting to the user.

K DD (knowledge discovery in databases):-

- Data mining as a part of knowledge discovery process.
- The knowledge discovery process comprises six phases:-
    I. Data selection
    II. Data cleansing
    III. Data enrichment

IV. Data transformation or encoding
V. Data mining
VI. Reporting and display of the discovered information

- ❖ Data selection means the selection or identification of data which we are going to mine.
- ❖ Data cleansing process is of ensuring that all values in database are consistent and are correctly recorded.
- ❖ Data enrichment typically enhances ten data with additional sources of information .for example, given the clients names and phone numbers it may give other data about age, income and append them to each record.
- ❖ Data transformation and encoding may be done to reduce the amount of data.
- ❖ Data mining refers to the mining /discovery of new information in terms of patterns or rules from vast amount of data.
- ❖ The results of data mining may be reported in a variety of formats, such as listings, graphic outputs, summary tables or visualizations.

G oals /applications of data mining or KDD:-

 The goals /applications fall into the following classes :-

a. Prediction
b. Identification
c. Classification
d. Optimization

A. P rediction:-data mining can show how certain attributes within the data will behave in the future .Examples of data mining includes the analysis of buying transactions to predict what customers will buy under certain discounts, how much sales volume a store would generate in a given period, and whether deleting a product line would yield more profits.

B. I dentification:- data patterns can be used to identify the existence of an item, an event, or an activity. For example, intruders trying to break a system may be identified by the programs executed, files accessed, and authentication is a form of identification. It ascertain whether a user is indeed a specific user or one from an authorised class and involves a comparison of parameters or images or signals against a database.

C. C lassification:-data mining can partition the data so that different classes or categories can be identified based on combinations of parameters. For example, customers in a supermarket can be categorized into discount-seeking shoppers, shoppers in a rush, loyal regular shoppers, shoppers attached to name brands, and infrequent shoppers.

D. O ptimization: - one eventual goal of data mining may be to optimize the use of limited resources such as time, space, money or materials and to maximize output variables such as sales or profits under a given set of constraints.

O bject Oriented DBMS(OODBMS):-

- In database, we concern about the management, and sharing of a large amount of reliable and persistent data.

- The relational system is suitable for a query expressed in a query language such as SQL. However, such query languages are not suitable for application development.
- Object oriented database, with the ability to treat everything as objects, including programs and data is therefore a promising concept for application development.
- In a DBMS, the relationship between two record types may be statically established or based on the context.
- Relationships exist between classes due the hierarchical structure and inheritance between subclass and superclass. In the object model, relationships may exist at object level via objects that know about each other and communicate via messages.
- In a database system, all record instances share the same set of operations which are implemented in the DBMS. In the object model, each object has its own set of operations and can be tailored to the object.
- Database records instances are accessed based on the contents. In the object model, the contents of the object are encapsulated and not accessible; therefore, the identifiers are the only means of externally identifying an object instance.

A dvantages of OODBMS: -

The following are the advantages of the OODBMS: -

- The object approach allows modification that is localized to a given level of an object hierarchy.
- The message to which an object responds is encapsulated along with the properties of the object.
- This allows constraints of the various complex forms to be easily enforced. Since the operations allowed on an object are encapsulated, its interactions with other objects are known and hence predictable. This allows ease in extension of the system.
- The inheritance mechanism allows compact codes and the overriding featured allow localization of changes.

D isadvantages of OODBMS: -

On the negative side are the following drawbacks of the OODBMS:

- Unlike the relational approach, which is started out with a formal theory and a framework for a query language, there is no formal or accepted framework of OODBMS.
- Since each object is a self-contained unit, there is no means of showing relationships among a number of objects. Inter-object references are used to show such an association indirectly.
- Performance will likely be a problem. Techniques such as associative access and architecture featured such as tagged architecture have to be investigated.
- In traditional database systems the user must know what the schema contains, such as names of relations and attributes, and pose queries and design programs using the knowledge. In the OODBMS the user must know what each object class is as well as its methods, messages and responses.

D istributed Databases :-

- A distributed database in a database that is under the control of a central database management system in which storage devices are not all attached to a common CPU.
- Distributed database system consists of loosely coupled sites where the data reside in several locations.
- Collections of data can be distributed across multiple physical locations.
- We can define a distributed database (DDB) as a collection of multiple logically interrelated databases distributed over a computer network, and a distributed database management system (DDBMS) as a software system that manages a distributed database while making the distributed transparent to the user.

T ypes of distributed database: -

Two types of distributed database are there: -

- ***H omogeneous distributed database: -***
  - In homogenous distributed database ,all sites have identical database management system software ,are aware of one another and agree to cooperate in processing user's request.
- ***H eterogeneous distributed database: -***
  - In heterogeneous distributed database, different sites may use different schemas and different database management system software. The sites may not be aware of one another, and they may provide only limited facilities for cooperation in transaction processing.

D istributes Data Storage: -

- Consider in relation 'R' that is to be stored in the database there are two approaches to store the relation in the distributed database: -

R eplication: -

- The system maintains several identical replicas(copies) of the relation, and stores each replica at a different site. The alternative to replication is to store only one copy of relation 'R'.

F ragmentation: -

- The system partitions the relation into several fragments and stores each fragment at a different site.

Fragmentation and replication can be combined: A relation can be partitioned into several fragments and they may be several replicas of each fragment.

ADVANTAGE OF DISTRIBUTED DATABASE: -

Increased Reliability and Availability: -

Reliability is defined as the probability that a system is running at a certain time point, whereas availability is the probability that the system is continuously available during a time interval.

When the data and DBMS software are distributed are several sites one site may fail while other sites continue to operate. This improves both reliability and availability.

Improved performance: -

A distributed DBMS fragments the database by keeping the data closer to where it is needed most. When a large database is distributed over multiple sites, smaller database exist at each site. As a result local queries and transactions accessing data at a single site have better performance because of the smaller local databases.

Distributed/network transparency: -

This refers to freedom for the user from the operational details of the network. It may be divide into location transparency and naming transparency.

Replication transparency: -

Copies of data may be stored at multiple sites for better availability, performance, and reliability replication transparency makes the user unaware of the existence of copies.

Fragmentation transparency: -

A global query by the user must be transformed into several fragment queries. Fragmentation transparency makes the user unaware of the existence of fragments.

Modularity: -

Systems can be modified, added and removed from the distributed database without affecting the other modules/systems.

Disadvantages of distributes database: -

Complexity: -

Extra work must be done by the DBAs to ensure that the distributed nature of the system is transparent.

Extra work must also be done to maintain multiple disparate systems, instead of one big one.

Economics: -

Increased complexity and more extensive infrastructure means extra labour cost.

Security: -

Remote database fragments must be secured, and they are not centralised so that the remote sites must be secured as well. The infrastructure must also be second.

Inexperience: -

Distributed database are difficult to work with, and as a young field there is not much readily available experience on proper practice.

Lack of standards: -

There are no tools or methodologies yet to help users convert a centralised database management system into a distributed DBMS.

Database design more complex: -

Besides of the normal difficulties, the design of a distributed database has to consider fragmentation of data, allocation of fragments to specific sites and data replication.

Data replication: -

- Data replication means the system maintains several identical replicas/copies of the relation and stores each replica at different sites.
- Fully replicated distributed database means that the replication of the whole database is present at every site.
- No replication means each fragment is stored at exactly one site. This is also called as non-redundant allocation.
- Partial replication of the data means some fragments of the database may be replicated where as others may not.
- Data distribution / data allocation means each fragment or each copy of a fragment must be assigned to a particular site in the distributed system.

Data fragmentation: -

- The system partitions the relation into several fragments and stores each fragment at a different site.
- Two types of fragmentation are possible.
- Horizontal fragmentation distributes a relation into sets of tuples/rows.
- Vertical fragmentation distributes a relation into sub-relations where each sub-relation is defined by a subset of the columns of the original relation.

M ultimedia Database:

- Multimedia database provide features that allow users to store and query different types of multimedia information, which includes images (such as photos or drawing),video clips(such as movies ,newsreels home videos), audio clips (such as songs ,speeches or phone messages), and documents (such a books or articles).

- The main types of database queries that are needed involve location multimedia sources that contain certain object of interest .For example; one may want to locate the video clips where a goal is scored in a cocker game by a certain player or team.
- The above types of queries are called as context-based retrieval, because the multimedia source is being retrieval based on its containing certain object or activities.
- A multimedia database must use some model to organise and index the multimedia sources based on their contents.
- Identifying the contents of multimedia sources is a different and time-consuming task. There are two main approached are there: -
    - Automatic analysis.
    - Manual identification

A utomatic analysis

- Automatic analysis of the multimedia sources is to identify certain mathematical characteristics of their contents.
- This approach uses different techniques depending on the type of multimedia sources.

M ultimedia identification: -

- This approach depends on manual identification of multimedia source and on using this information to index the sources.
- This approach can be ap[plied to all the different multimedia sources, but it requires a manual pre- processing phase where a person has to scan each multimedia source to identify and catalogue the objects and activities it contains so that they can be used to index these sources.