Database Recovery and Security

Unit 3

Database Recovery

- Database recovery is the process of restoring the database to the most recent consistent state that existed before the failure.
- If a transaction does not complete normally and terminates abnormally then all the changes made by it should be discarded.
- An abnormal termination of a transaction may be due to several reasons which includes :
- User may decide to abort the transaction issued by him/her
- There might be a deadlock in the system
- There might be a system failure
- Database recovery is important in managing a database system.
- It brings the database into the last consistent state, which existed prior to the failure.

- It preserves transaction properties: Atomicity, Consistency, Isolation, Durability.
- Recovery protects the database and associated users from unnecessary problems and avoid or reduce the possibility of having to duplicate work manually.
- It aims to allow database operations to be resumed after a failure with the minimum loss of information and at an economically justifiable cost.

Kinds of Failures :

- Kinds of failures that a transaction program during its execution can encounter are :

1. Software failures : In such cases, a software error abruptly stops the execution of the concurrent transaction, thus leading to losing the state of program execution and the state/ contents of the buffers. A buffer is the portion of RAM that stores the partial contents of database that is currently needed by the transaction . Software failures can be classified as :
   - Statement or application program failure
   - Failure due to viruses

- DBMS software failure
- Operating system failure

▪ A statement of program may cause abnormal termination if it does not execute completely. This happens if during the execution of a statement, an integrity constraint gets violated. This leads to abnormal termination of the transaction due to which any prior updates made by the transaction may still get reflected in the database leaving it in an inconsistent state.

▪ A failure of transaction can occur if some code in a transaction program lead to its abnormal termination. For example, a transaction can go into an infinite loop. In such a case the only way to break the loop is to abort the program. Similarly, the failure can be traced to the operating system or DBMS and transactions are aborted abruptly. Thus part of the transaction that was executed before abort may cause some updates in database, and hence the database is updated only partially which leads to an inconsistent state of database.

2. Hardware failure : Hardware failures are those failures when some hardware chip or disk fails. This may result in loss of data. One such problem can be that a disk gets damaged and cannot be read any more. This may be due to many reasons. For example, a voltage fluctuation in the power supply to the computer makes it off or some bad sectors may come on disk or there is a disk crash. In all the above cases, the database gets into inconsistent state.

3. External failure : A failure can also result due to an external cause, such as fire, earthquakes, floods etc. The database must be duly backed up to avoid such problems occurring due to such failures. Recovery from software failures is much quicker.

- The basic unit of recovery is a transaction.

Handling of transactions :

- Consider that some transactions are deadlocked, then at least one of these transactions has to be restarted to break the deadlock and thus the partial updates made by such restarted program in the database need to be undone do that the database does not go to an inconsistent state. So the transaction may have to be rolled back which makes sure that the transaction does not bring the database to an inconsistent state. This is one form of recovery. Let us consider a case when a transaction has committed but the changes made by the transaction have not been communicated to permanently stored physical database. A software failure now occurs and the contents of the CPU/RAM are lost. This leaves the database in an inconsistent state. Such failure requires that on restarting the system the database be brought to a consistent state using redo operation. The redo operation makes the changes made by the transaction again to bring the system to a consistent state. The database system then can be made available to the users.

- Failure Controlling Method :

Failures can be handled using different recovery techniques. We need recovery techniques as a failure control mechanism. The recovery techniques are expensive both in terms of time and in memory space for small systems. In such a case its more beneficial to better avoid the failure by some checks instead of deploying recovery technique to make database consistent. Also recovery from failure involves manpower that can be used in some other productive work if failure can be avoided.

Some precautions to control failures are :

- Having a regulated power supply.
- Having a better secondary storage system like RAID.
- Taking periodic backup of database states and keeping track of transactions after each recorded state.
- Properly testing the transaction program prior to use.
- Setting important integrity checks in the databases as well as user interfaces etc..

Database errors :

- An error is said to have occurred if the execution of a command to manipulate the database cannot be successfully completed either due to inconsistent data or due to state of a program.

- Broadly errors are classified into the following categories :

1. User error : This includes error in the program (e.g. logical error) as well as errors made by online users of database. This category of errors can be avoided by applying some check conditions in the programs or by limiting the access rights of online users e.g. read only. So only updating or insertion operation require appropriate check routines that perform appropriate checks on the data being entered or modified. In case of an error, some prompts can be passed to user to enable him/her to correct the error.

2. Consistency error : These errors occur due to inconsistent state of database caused may be due to wrong execution of commands or in case of a transaction abort. To overcome these errors the database system should include routines that check for the consistency of data entered in the database.

3. System error : These include errors in database system or the OS, e.g., deadlocks. Such errors are fairly hard to detect and require reprogramming the erroneous components of the system software.

- Database errors can result from failure or can cause failure and thus will require recovery.
- However, one of the main tasks of database system designers is to minimise the number of errors.

Recovery Techniques

- Several recovery techniques have been proposed for database systems.
- As we have seen the types of failures , so now we will discuss about how to recover from those types of failures.
- Recovery can be done using/restoring the previous consistent state (backward recovery) or by moving forward to the next consistent state as per the committed transactions (forward recovery) recovery.
- A system can recover from software and hardware failures using the forward and backward recovery only if the system log is intact.

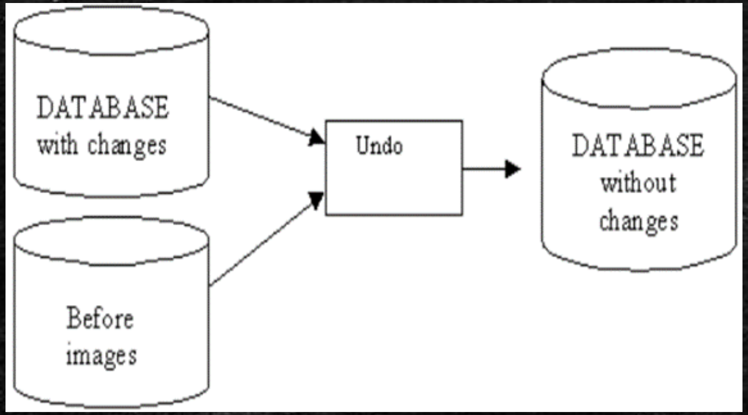
- Backward Recovery (UNDO)
- Its also known as **Roll back.** The goal of backward recovery is to bring the system from an erroneous state back to a prior correct state. It's the most widely used recovering scheme.In this scheme the uncommitted changes made by a transaction to a database are undone. Instead the system is reset to the previous consistent state of a database that is free from any errors. To do so, the systems state is recorded from time to time (checkpointed). Checkpointing can be very expensive. Despite this, its implemented more often because logging is considered as a type of checkpointing. It can be used as a general recovery mechanism. Example : lost message retransmission. This scheme is used when nature of the errors and damages caused by the faults cannot be completely and accurately accessed.

<u>Advantages :</u>

- Its simpler than forward recovery.
- Its independent of an arbitrary fault.

Disadvantages :

- The overhead to restore a process or system state to prior state can be quite high.
- There is no guarantee that faults will not occur again when the processing begins.
- Some component of system or process may not be recoverable.



- Forward Recovery (Redo)

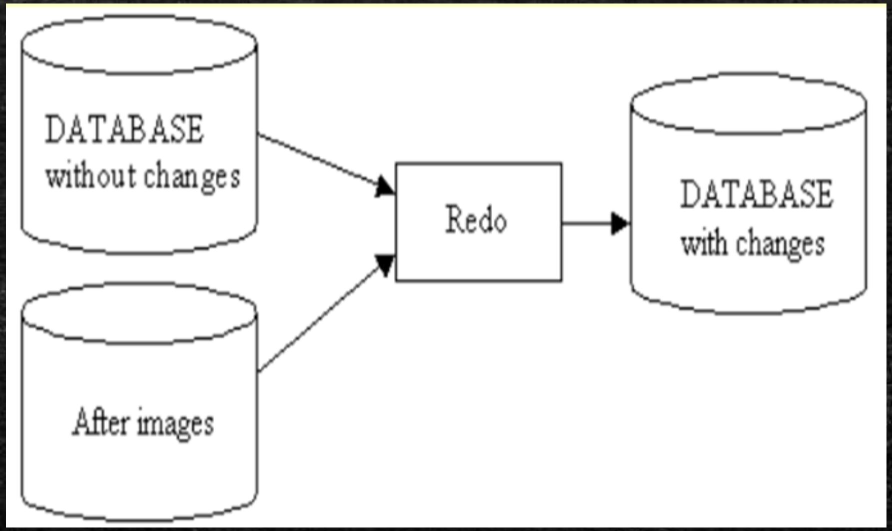
Its also known as roll forward. The goal of forward recovery is to bring a system from an erroneous state to a correct new state (not previous state) .In this scheme the committed changes made by a transaction are reapplied to an earlier copy of the database. In this scheme, we must know in advance which errors may occur. Example : error correction by special encoding of messages. This scheme is used when nature of the errors and damages caused by the faults can be completely and accurately accessed.

Advantages :

- Less overhead
- Its much faster than backward recovery.

Disadvantages :

- Limited use , used only when impact of faults understood.
- Cannot be used as a general mechanism.
- Designed specifically for a particular system.



- The Undo and Redo operations must be idempotent, i.e., executing them several times must be equivalent to executing them once. This characteristic guarantees correct behaviour of database even if a failure occurs during the recovery process.

- Depending upon the above discussed recovery scheme, several types of recovery methods have been used.

Log based recovery :

- A transaction log is a record in DBMS that keeps track of all the transactions of a database system that update any data values in the database.

- A log contains the following information about a transaction :

- A transaction begin marker

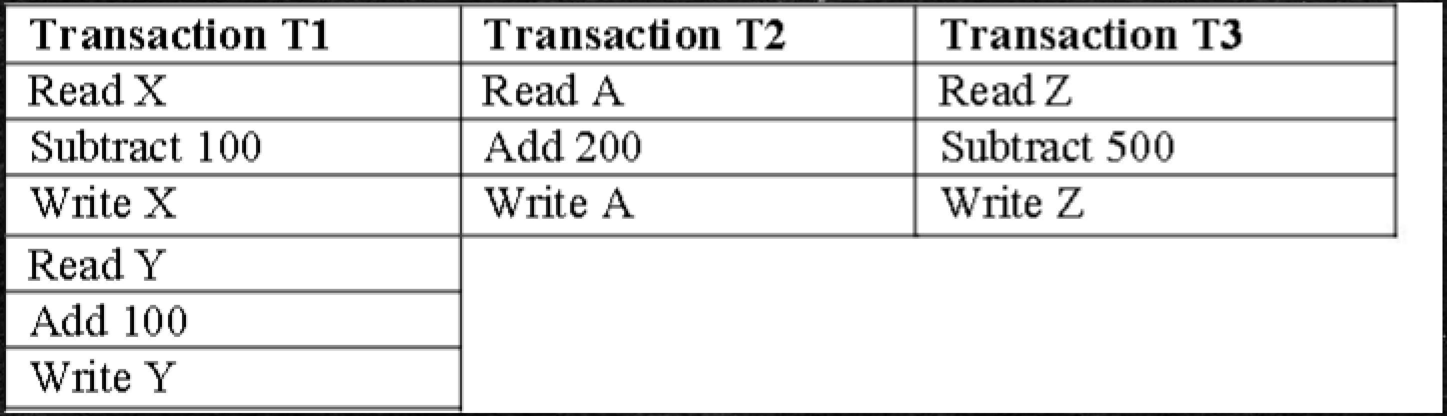
- The transaction identification: the transaction id, terminal id or user id etc..

- The operations being performed by the transaction such as update, delete, insert.
- The data items or objects that are affected by the transaction including name of the table, row number and column number.
- The before or previous values(also called UNDO values) and after or changed values (also called REDO values) of the data items that have been updated.
- A pointer to the next transaction log record, if needed.
- The COMMIT marker of the transaction.

▪ In a database system several transactions run concurrently. When a transaction commits, the data buffers used by it need not be written back to the physical database stored on the secondary storage as these buffers may be used by several other transactions that have not yet committed. On the other hand, some of the data buffers that may have updates by several uncommitted transactions might be forced back to the physical database, as they are no longer being used by the database. So the transaction log helps in remembering which transaction did which changes. Thus the system knows exactly how to separate the changes made by the transactions that have already committed from those changes that are made by the transactions that did not yet commit. Any operation such as begin transaction, insert/delete/update and end transaction (commit), adds information to the log containing the transaction identifier and enough information to undo or redo the changes.

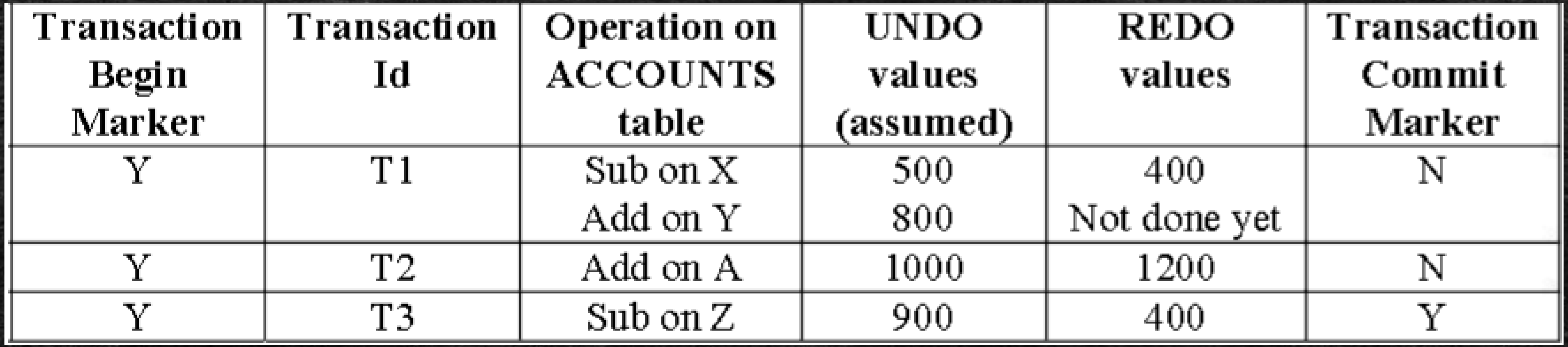
How do we recover using log :

Let us take an example of 3 concurrent transactions that are active on ACCOUNTS table to explain this.



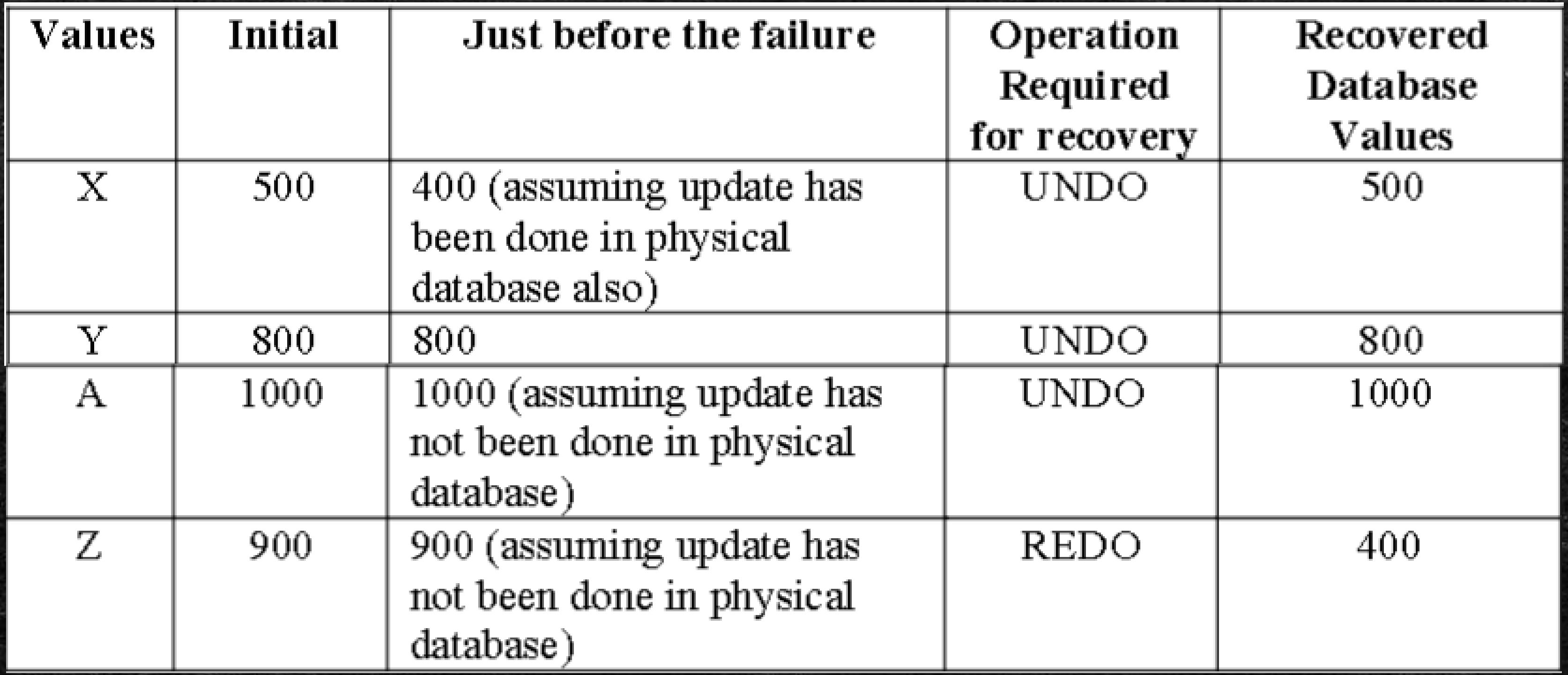
Transaction T1	Transaction T2	Transaction T3
Read X	Read A	Read Z
Subtract 100	Add 200	Subtract 500
Write X	Write A	Write Z
Read Y		
Add 100		
Write Y		

Log file of these transactions at a point is :



Transaction Begin Marker	Transaction Id	Operation on ACCOUNTS table	UNDO values (assumed)	REDO values	Transaction Commit Marker
Y	T1	Sub on X Add on Y	500 800	400 Not done yet	N
Y	T2	Add on A	1000	1200	N
Y	T3	Sub on Z	900	400	Y

Assume that at this point a failure occurs, then how the recovery of the database will be done at restart :



Values	Initial	Just before the failure	Operation Required for recovery	Recovered Database Values
X	500	400 (assuming update has been done in physical database also)	UNDO	500
Y	800	800	UNDO	800
A	1000	1000 (assuming update has not been done in physical database)	UNDO	1000
Z	900	900 (assuming update has not been done in physical database)	REDO	400

- The selection of the Undo or Redo scheme for a transaction for the recovery is done on the basis of the state of the transaction. This state is determined in two steps :

1. Look into the transaction and find all the transactions that have started. For example, in figure 3 transactions T1, T2 AND T3 candidates for recovery.

2. Find those transactions that have committed. REDO these transactions. All other transactions that have not committed so they should be rolled back, So UNDO them. For example in figure 3 UNDO will be performed on T1 and T2; and REDO will be performed on T3.

- Please note that in figure 4 some of the values may not have yet been communicated to database, yet we need to perform UNDO as we are not sure what values have been written back to the database.

- Once the recovery operation has been specified, the system just takes the required REDO or UNDO values from the transaction log and changes the inconsistent state of database to a consistent state.

Let us consider several transactions with their respective start and end times as shown in figure 5 below :

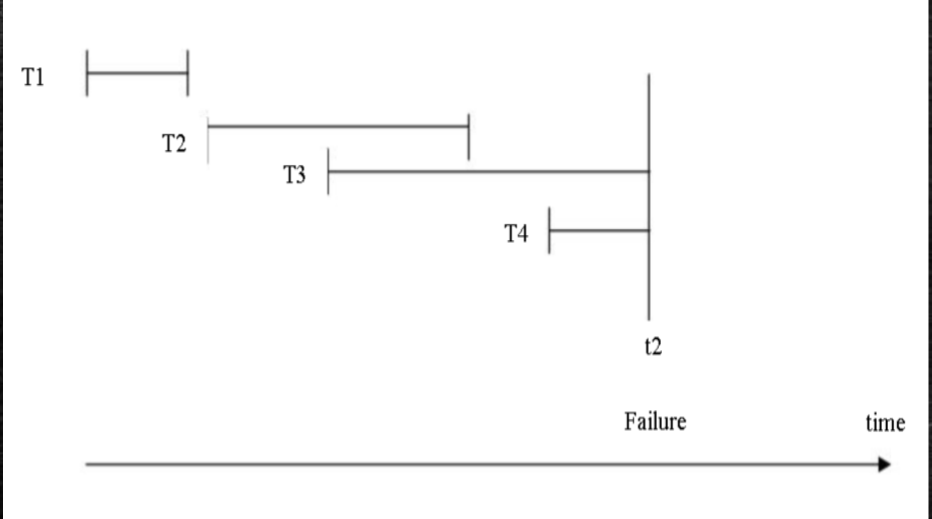


Figure 5: Execution of Concurrent Transactions

In the figure above four transactions are executing concurrently, on encountering a failure at time t2, the transactions T1 and T2 are to be done REDONE and T3 and T4 will be UNDONE. But consider a system that has thousands of parallel transactions then all those transactions that have been committed may have to be redone and all uncommitted transactions need to be undone. That is not a very good choice as it requires redoing of even those transactions that might have been committed even hours earlier. We can take checkpoints.

Checkpoint mechanism (Figure 6):

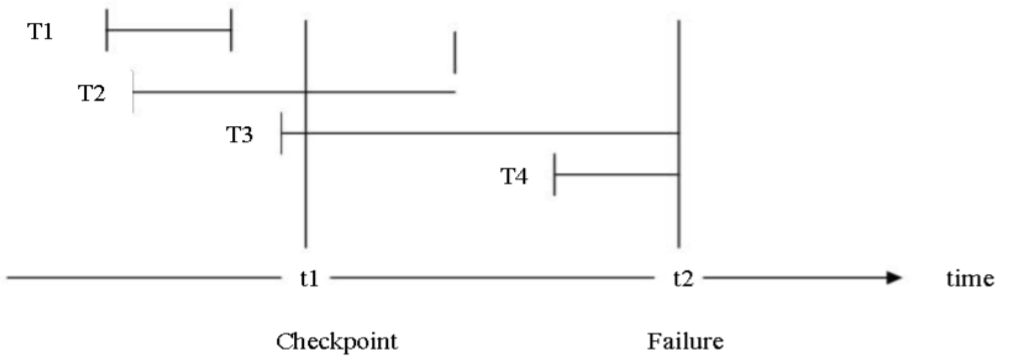


Figure 6: Checkpoint In Transaction Execution

- A checkpoint is taken at time t1 and a failure occurs at time t2. Checkpoint transfers all the committed changes to database and all the system logs to stable storage. A checkpoint is a point when all the database updates and logs are written to stable storage. At restart time after the failure the stable check pointed state is restored. Thus, we need to only REDO or UNDO those transactions that have completed or started after the checkpoint has been taken.

- Disadvantage of this scheme may be that during the time of taking the checkpoint the database would not be available and some of the uncommitted values may be put in the physical database.

- To overcome the first problem the checkpoints should be taken at times when system load is low. To overcome the second problem some systems allow some time to the ongoing transactions to complete without restarting new transactions.

- In case of figure 6, the recovery from failure at t2 will be as follows :
- The transaction T1 will not be considered for recovery as the changes made by it have already been committed and transferred to physical database at checkpoint t1.
- The transaction T2 since it has not committed till the checkpoint t1 but have committed before t2, will be REDONE.
- T3 must be UNDONE as the changes made by it before checkpoint must have been communicated to the physical database. T3 must be restarted with a new name.
- T4 started after the checkpoint, and if we strictly follow the scheme in which the buffers are written back only on the checkpoint, then nothing needs to be done except restarting the transaction T4 with a new name.
- The restart of a transaction requires the log to keep information of the new name of transaction and may be given higher priority to this newer transaction.

- During a failure we lose database information in RAM buffers, we may also lose log as it may also be stored in RAM buffers. Log ensures recovery by storing transaction log we follow a Write Ahead Log Protocol.

According to Write Ahead Log Protocol, the transaction logs are written to stable storage before any item is updated. Or more specifically it can be stated as; the undo portion of log is written to stable storage prior to any updates and redo portion of log is written to stable storage prior commit.

Log based recovery scheme can be used for any kind of failure provided you have the stored the most recent checkpoint state and most recent log as per write ahead log protocol into the stable storage. Stable storage from the viewpoint of external failure requires more than one copy of such data at more than one location.

Security And Integrity

- Information security is the protection of information against unauthorized disclosure, alteration or destruction.
- Database security is the protection of information that is maintained in a database.
- It deals with ensuring only the "right people" get the rights access to the "right data". By right people are mean to those people who have the right to access/update the data that they are requesting to access/update with the database.
- This ensures that the confidentiality of the data is maintained.
- For example, in an educational institution, information about a student's grade should be made available only to that institution, information about a student's grades should be made available to that student, whereas only the university authorities should be able to update that information. Another example can be in case of medical records of patients in a hospital, these could accessible only to health care officials.

- Specification of access rules about who has what type of access to what information is known as problem of authorization.
- These access rules are defined at the time database is defined. The person who writes access rules is called on authorizer. The process of ensuring that information & other protected object are accessed only in authorized ways is called access control.
- The term integrity is also applied to data & to the mechanism that help to ensure its correctness. Integrity refers to the avoidance of accidental loss of consistency.
- Protection of database contents from unauthorized access includes legal & ethical issues, organization policies as well as database management policies.
- Various aspects of security problem : legal, social and ethical aspects; physical controls; policy questions; operational problems; hardware control; operating system security; database administration concerns.

- To protect database several levels of security measures are maintained: -

1. **Physical :** The site or sites containing the computer system must be physically secured against illegal entry of unauthorized person.

2. **Human :** An authorisation is given to a user to reduce the chance of any information leakage and unwanted manipulation.

3. **Operating System :** Even though a foolproof security measures are taken to secure database System, weakness in the O.S. security may serve as a means of unauthorized access to the database.

4. **Network :** Since databases allow distributed or remote access through terminals or network, software level security within the network software is an important issue to be taken under consideration.

5. **Database System** : The data items in a database need a fine level of access control. For example, a user may only be allowed to read a data item and is allowed to issue queries but would not be allowed to deliberately modify the data. It's the responsibility of the database system to ensure that these access restrictions are not violated.

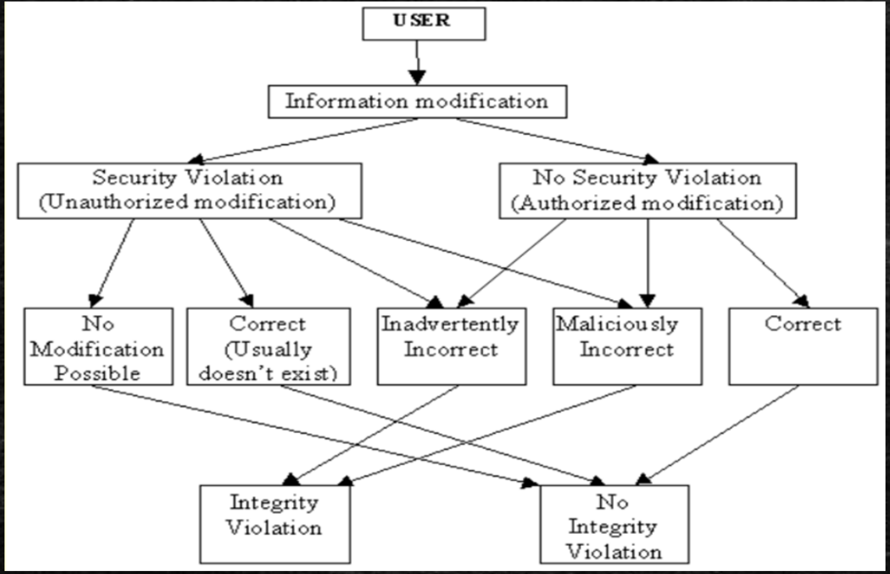
- To ensure database security requires implementation of security at all levels.
- The Database Administrator (DBA) is responsible for implementing the database security policies in a database system.
- The organization or data owners create these policies. DBA creates or cancels the user accounts assigning appropriate security rights to user accounts including power of granting and revoking certain privileges further to other users.

Difference between data security and data integrity :

- Data security is the protection of information that is maintained in database against unauthorised access, modification or destruction. Data integrity is the mechanism that is applied to ensure that data in the database is correct and consistent.

Relationship between Security and Integrity

- Database security usually refers to access, where as database integrity refers to avoidance of accidental loss of consistency. But generally, the turning point or the dividing line between security & integrity is not always clear. Fig. Below shows relationship between data security & integrity.



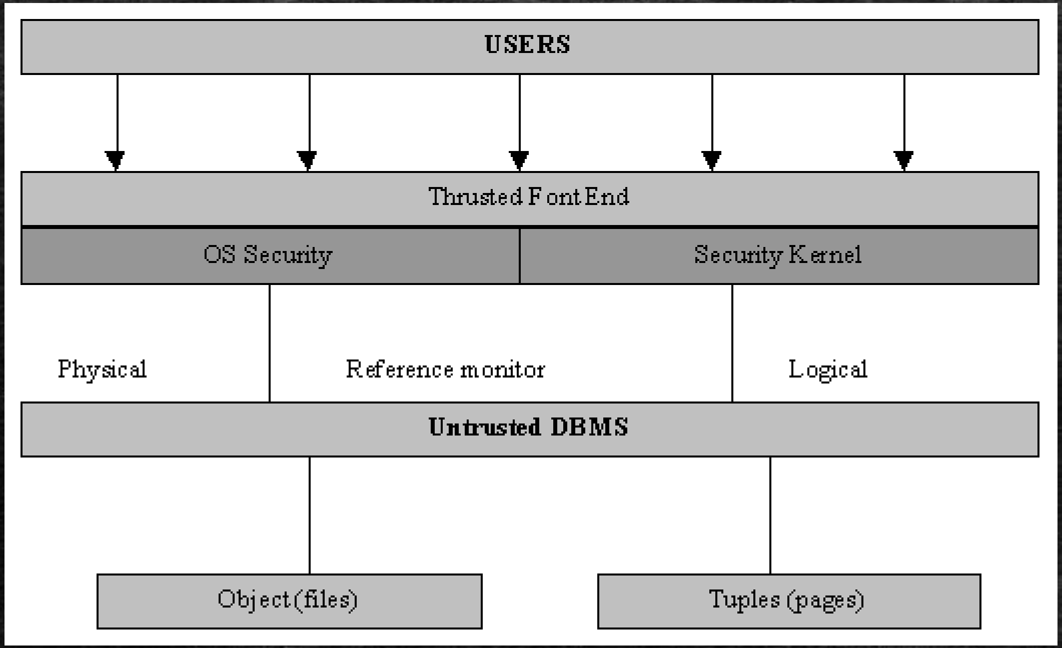
Difference Between Operating System And Database Security

- Security in operating system can be implemented at several levels ranging from password for access to system, to the isolation of concurrent executing processes with the system.

- There are few differences between security measures taken at operating system level as compared to those that of database system. These are :

- Database system protects more objects, as the data is persistent in nature. Also database security is concerned with different levels of granularity such as file, tuple, an attribute value or an index. Operating system security is primarily concerned with the management and use of resources.

- Database system objects can be complex logical structures such as views, a number of which can map to the same physical data objects. Moreover different architectural levels vis. Internal, conceptual and external levels, have different security requirements. Thus database security is concerned with the semantics- meaning of data, as well as with its physical representation. Operating system can provide security by not allowing any operation to be performed on the database unless the user is authorized for the operation concerned.



Database Security subsystem

Authorisation

- Authorisation is the culmination of the administrative policies of the organisation.
- Authorisation is the set of rules that cab be used to determine which user has what type of access to what portion of the database.
- Following forms of authorisation are permitted on database items :

1. READ : It allows reading of data object, but not modification, deletion or insertion of data object.
2. INSERT : It allows insertion of new data, but not the modification of existing data. Eg : insertion of a new tuple in a relation.
3. UPDATE : It allows modification of data, but not its deletion. But data items like primary key attributes may not be modified.
4. DELETE : It allows deletion of data only.

- A user may be assigned all, none or a combination of these types of authorisation broadly known as access authorisation.
- The above mentioned operations where manipulation operations, in addition to these a user may be granted control operations like :

1. Add : It allows adding new objects such as new relations.
2. Drop : It allows the deletion of relation in a database.
3. Alter : It allows addition of new attributes in a relation or daletion of existing attributes from the database.
4. Propagate Access Control : Its an additional right that allows a user to propagate the access control or access right which s/he has to some other. i.e., if user A has access right R over relation S, then if s/he has propagate access control, s/he can propagate her/his access right R over relation S to user B either fully or partially. In SQL we use WITH GRANT OPTION for this purpose.

- The ultimate form of authority is given to the database administrator. He is the one who authorizes new users, restructure the database and so on.

Database Access Control Model :

- It was developed by Lampson and extended by Graham and Denning.
- It has 3 components :

1. Objects : It's a set of objects where objects are those entities to which access must be controlled.
2. Subjects : It's a set of subjects where subjects are entities that request access to objects.
3. Access Rights : A set of all access rules which can be thought of as forming an access(often known as authorization matrix).

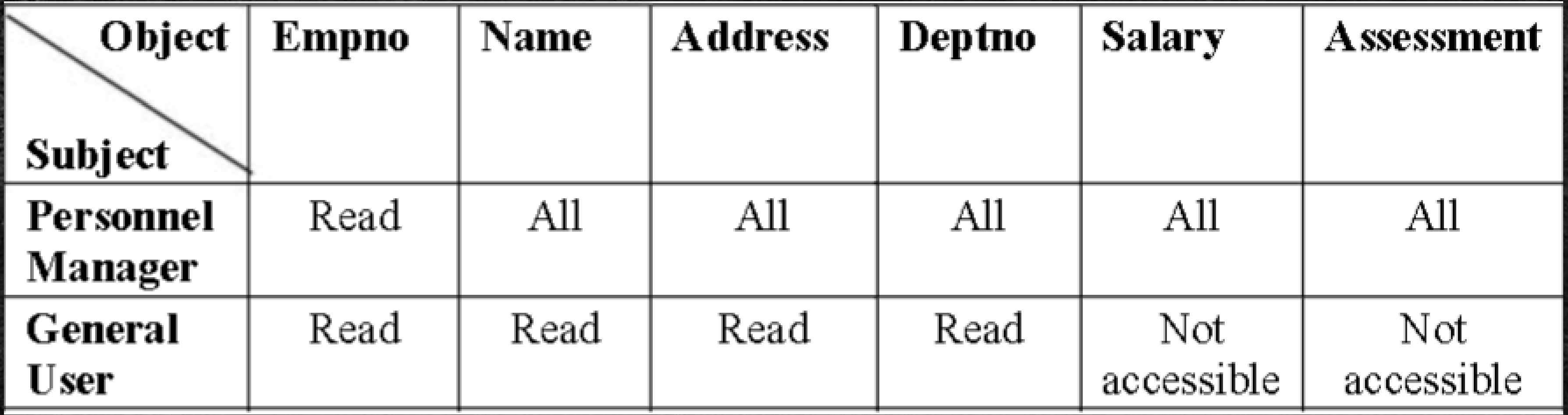
- Example :

Consider the relation : Employee (Empno, Name, Address, Deptno, Salary, Assessment)

Assume there are two users : Personnel manager and general user. What access rights must be given to each user. There are two possibilities one is unauthorised access or other is limited access.

$$\begin{tikzcd} {} & \boxed{\text{Object}} \arrow[l, "\text{Unconstrained access}"'] \arrow[r, "\text{Strictly limited access}"] & {} \end{tikzcd}$$

Sample authorization matrix for above example :



Object / Subject	Empno	Name	Address	Deptno	Salary	Assessment
Personnel Manager	Read	All	All	All	All	All
General User	Read	Read	Read	Read	Not accessible	Not accessible

In the above matrix, Personnel Manager and general user are the two subjects. Objects of the database are Empno, Name, Address, Deptno, Salary and Assessment. According to the above authorization matrix, personnel manager can perform any operation on the database of an employee except for updating Empno that may be self-generated(once generated cannot be changed). General user can only read the data but can not update, delete or insert the data into the database. Information regarding salary and assessment are not accessible to the general user.