**Serializability**

When multiple transactions are being executed by the operating system in a multiprogramming environment, there are possibilities that instructions of one transactions are interleaved with some other transaction.

* **Schedule** − A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each comprising of a number of instructions/tasks.
* **Serial Schedule** − It is a schedule in which transactions are aligned in such a way that one transaction is executed first. When the first transaction completes its cycle, then the next transaction is executed. Transactions are ordered one after the other. This type of schedule is called a serial schedule, as transactions are executed in a serial manner.

## Equivalence Schedules

An equivalence schedule can be of the following types −

### Result Equivalence

If two schedules produce the same result after execution, they are said to be result equivalent. They may yield the same result for some value and different results for another set of values. That's why this equivalence is not generally considered significant.

### View Equivalence

Two schedules would be view equivalence if the transactions in both the schedules perform similar actions in a similar manner.

For example −

* If T reads the initial data in S1, then it also reads the initial data in S2.
* If T reads the value written by J in S1, then it also reads the value written by J in S2.
* If T performs the final write on the data value in S1, then it also performs the final write on the data value in S2.

### Conflict Equivalence

Two schedules would be conflicting if they have the following properties −

* Both belong to separate transactions.
* Both accesses the same data item.
* At least one of them is "write" operation.

Two schedules having multiple transactions with conflicting operations are said to be conflict equivalent if and only if −

* Both the schedules contain the same set of Transactions.
* The order of conflicting pairs of operation is maintained in both the schedules.

**Note** − View equivalent schedules are view serializable and conflict equivalent schedules are conflict serializable. All conflict serializable schedules are view serializable too.

# Data Backup

## Loss of Volatile Storage

A volatile storage like RAM stores all the active logs, disk buffers, and related data. In addition, it stores all the transactions that are being currently executed. What happens if such a volatile storage crashes abruptly? It would obviously take away all the logs and active copies of the database. It makes recovery almost impossible, as everything that is required to recover the data is lost.

Following techniques may be adopted in case of loss of volatile storage −

* We can have **checkpoints** at multiple stages so as to save the contents of the database periodically.
* A state of active database in the volatile memory can be periodically **dumped** onto a stable storage, which may also contain logs and active transactions and buffer blocks.
* <dump> can be marked on a log file, whenever the database contents are dumped from a non-volatile memory to a stable one.

### Recovery

* When the system recovers from a failure, it can restore the latest dump.
* It can maintain a redo-list and an undo-list as checkpoints.
* It can recover the system by consulting undo-redo lists to restore the state of all transactions up to the last checkpoint.

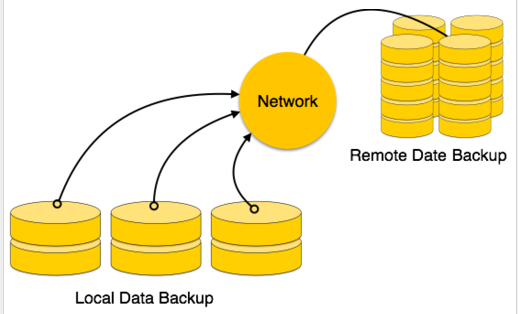
## Database Backup & Recovery from Catastrophic Failure

A catastrophic failure is one where a stable, secondary storage device gets corrupt. With the storage device, all the valuable data that is stored inside is lost. We have two different strategies to recover data from such a catastrophic failure −

* Remote backup &minu; Here a backup copy of the database is stored at a remote location from where it can be restored in case of a catastrophe.
* Alternatively, database backups can be taken on magnetic tapes and stored at a safer place. This backup can later be transferred onto a freshly installed database to bring it to the point of backup.

## Remote Backup

Remote backup provides a sense of security in case the primary location where the database is located gets destroyed. Remote backup can be offline or real-time or online. In case it is offline, it is maintained manually.



Online backup systems are more real-time and lifesavers for database administrators and investors. An online backup system is a mechanism where every bit of the real-time data is backed up simultaneously at two distant places. One of them is directly connected to the system and the other one is kept at a remote place as backup.

As soon as the primary database storage fails, the backup system senses the failure and switches the user system to the remote storage. Sometimes this is so instant that the users can’t even realize a failure.

## Crash Recovery

DBMS is a highly complex system with hundreds of transactions being executed every second. The durability and robustness of a DBMS depends on its complex architecture and its underlying hardware and system software. If it fails or crashes amid transactions, it is expected that the system would follow some sort of algorithm or techniques to recover lost data.

## Failure Classification

To see where the problem has occurred, we generalize a failure into various categories, as follows −

### Transaction failure

A transaction has to abort when it fails to execute or when it reaches a point from where it can’t go any further. This is called transaction failure where only a few transactions or processes are hurt.

Reasons for a transaction failure could be −

* **Logical errors** − Where a transaction cannot complete because it has some code error or any internal error condition.
* **System errors** − Where the database system itself terminates an active transaction because the DBMS is not able to execute it, or it has to stop because of some system condition. For example, in case of deadlock or resource unavailability, the system aborts an active transaction.

### System Crash

There are problems − external to the system − that may cause the system to stop abruptly and cause the system to crash. For example, interruptions in power supply may cause the failure of underlying hardware or software failure.

Examples may include operating system errors.

### Disk Failure

In early days of technology evolution, it was a common problem where hard-disk drives or storage drives used to fail frequently.

Disk failures include formation of bad sectors, unreachability to the disk, disk head crash or any other failure, which destroys all or a part of disk storage