Backup and Recovery

Backup and Recovery

- 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.

Backup and Recovery

- Loss of Volatile Storage
  - 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.

Backup and Recovery

- Loss of Volatile Storage
  - 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.

Backup and Recovery

- 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.

Backup and Recovery

- Database Backup & Recovery from Catastrophic Failure
  - 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.

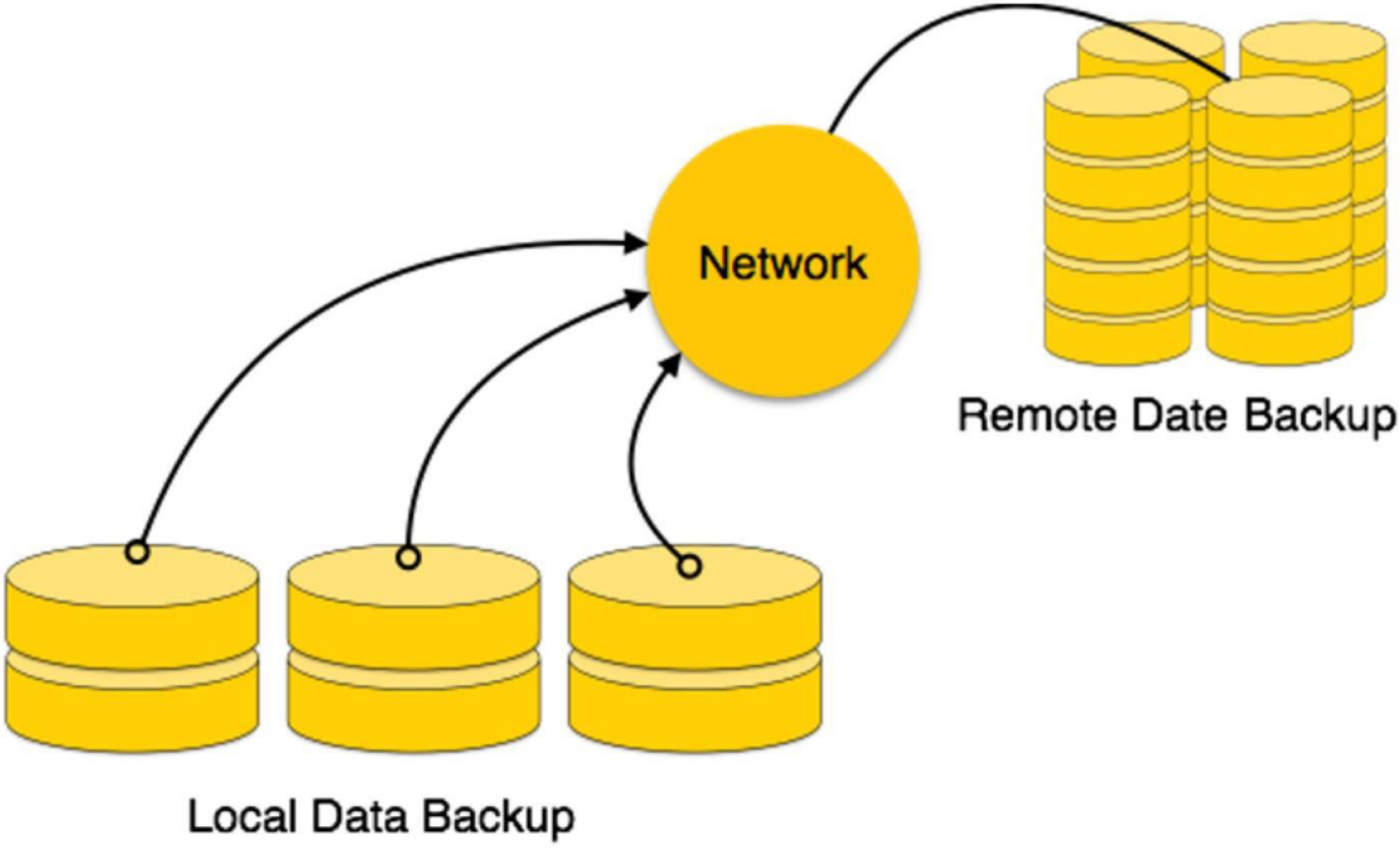
Backup and Recovery

- Database Backup & Recovery from Catastrophic Failure
  - Grown-up databases are too bulky to be frequently backed up. In such cases, we have techniques where we can restore a database just by looking at its logs.
  - So, all that we need to do here is to take a backup of all the logs at frequent intervals of time.
  - The database can be backed up once a week, and the logs being very small can be backed up every day or as frequently as possible.

Backup and Recovery

- 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
      - In this case, it is maintained manually
    - real-time
    - online

Backup and Recovery



Backup and Recovery

- Remote Backup
  - 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
    - The other one is kept at a remote place as backup.

Backup and Recovery

- 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

Backup and Recovery

- Failure Classification
  - To see where the problem has occurred, we generalize a failure into various categories
    - Transaction failure
      - **Logical errors**
      - **System errors**
    - System Crash
    - Disk Failure

Backup and Recovery

- Failure Classification
  - 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.

Backup and Recovery

- Failure Classification
  - Transaction failure
    - 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.

Backup and Recovery

- Failure Classification
  - 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.

Backup and Recovery

- Failure Classification
  - 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
      - any other failure, which destroys all or a part of disk storage

Backup and Recovery

- Storage Structure
  - We have already described the storage system. In brief, the storage structure can be divided into two categories
    - Volatile storage
    - Non-volatile storage

Backup and Recovery

- Storage Structure
  - **Volatile storage**
    - As the name suggests, a volatile storage cannot survive system crashes.
    - Volatile storage devices are placed very close to the CPU; normally they are embedded onto the chipset itself.
    - For example, main memory and cache memory are examples of volatile storage.
    - They are fast but can store only a small amount of information.

Backup and Recovery

- Storage Structure
  - **Non-volatile storage**
    - These memories are made to survive system crashes.
    - They are huge in data storage capacity, but slower in accessibility.
    - Examples may include
      - hard-disks
      - magnetic tapes
      - flash memory
      - non-volatile (battery backed up) RAM

Backup and Recovery

- Recovery and Atomicity
  - When a system crashes, it may have several transactions being executed and various files opened for them to modify the data items.
  - Transactions are made of various operations, which are atomic in nature.
  - But according to ACID properties of DBMS, atomicity of transactions as a whole must be maintained, that is, either all the operations are executed or none.

Backup and Recovery

- Recovery and Atomicity
  - When a DBMS recovers from a crash, it should maintain the following
    - It should check the states of all the transactions, which were being executed.
    - A transaction may be in the middle of some operation; the DBMS must ensure the atomicity of the transaction in this case.
    - It should check whether the transaction can be completed now or it needs to be rolled back.
    - No transactions would be allowed to leave the DBMS in an inconsistent state.

Backup and Recovery

- Recovery and Atomicity
  - There are two types of techniques, which can help a DBMS in recovering as well as maintaining the atomicity of a transaction
    - Maintaining the logs of each transaction, and writing them onto some stable storage before actually modifying the database.
    - Maintaining shadow paging, where the changes are done on a volatile memory, and later, the actual database is updated.

Backup and Recovery

- Log-based Recovery
  - Log is a sequence of records, which maintains the records of actions performed by a transaction.
  - It is important that the logs are written prior to the actual modification and stored on a stable storage media, which is failsafe.

Backup and Recovery

- Log-based Recovery
  - Log-based recovery works as follows
    - The log file is kept on a stable storage media.
    - When a transaction enters the system and starts execution, it writes a log about it
    - When the transaction modifies an item X, it write logs as follows on the next slide
    - When the transaction finishes, it logs it as follows on the next slide

Backup and Recovery

<T_n, Start>

<T_n, X, V_1, V_2>

<T_n, commit>

- A transaction, T_n, enters the system and starts execution
- It reads T_n has changed the value of X, from V_1 to V_2
- The transaction, T_n, finishes

Backup and Recovery

- Log-based Recovery
  - The database can be modified using two approaches
    - **Deferred database modification** – All logs are written on to the stable storage and the database is updated when a transaction commits.
    - **Immediate database modification** – Each log follows an actual database modification. That is, the database is modified immediately after every operation.

Backup and Recovery

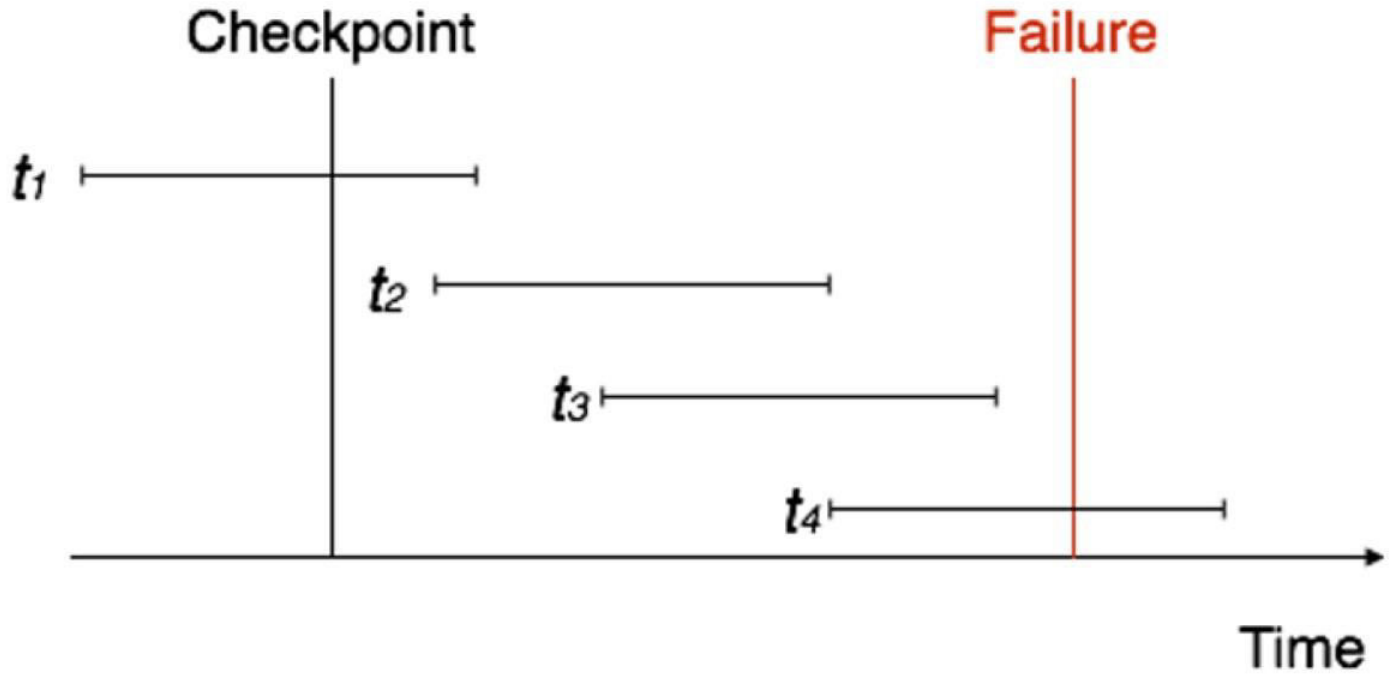
- Recovery with Concurrent Transactions
  - When more than one transaction are being executed in parallel, the logs are interleaved.
  - At the time of recovery, it would become hard for the recovery system to backtrack all logs, and then start recovering.
  - To ease this situation, most modern DBMS use the concept of 'checkpoints'.

Backup and Recovery

- Recovery with Concurrent Transactions
  - Checkpoint
    - Keeping and maintaining logs in real time and in real environment may fill out all the memory space available in the system.
    - As time passes, the log file may grow too big to be handled at all.
    - Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk.
    - Checkpoint declares a point before which the DBMS was in consistent state, and all the transactions were committed.

Backup and Recovery

- Recovery with Concurrent Transactions
  - Recovery
    - When a system with concurrent transactions crashes and recovers, it behaves in the following manner



Backup and Recovery

- Recovery with Concurrent Transactions
  - Recovery
    - When a system with concurrent transactions crashes and recovers, it behaves in the following manner
      - The recovery system reads the logs backwards from the end to the last checkpoint.
      - It maintains two lists, an undo-list and a redo-list.
      - If the recovery system sees a log with $< T_n,$ Start> and $< T_n,$ Commit>, or just a $< T_n,$ Commit>, it puts the transaction in the redo-list.
      - If the recovery system sees a log with $< T_n,$ Start> but no commit or abort log found, it puts the transaction in undo-list.

Backup and Recovery

- Recovery with Concurrent Transactions
  - Recovery
    - All the transactions in the undo-list are then undone and their logs are removed.
    - All the transactions in the redo-list and their previous logs are removed and then redone before saving their logs.