Chapter 9 : Crash Recovery

Database Recovery Management: It is the process of restoring the database to the most recent consistent state that existed just before the failure.

3 States of database recovery

1. Pre Condition - in a consistent state
2. Condition - occurs some kind of system failure
3. Post Condition - Restore the database to the consistent state that existed before the failure

Failure Classification

1) Transaction Failure

a) Logical errors: The transaction cannot continue because of internal conditions like data mismatch, overflow etc.

b) System errors: The transaction cannot continue as the system enter into an undesirable state like deadlock.

2) System crash

a) It is the failure caused due to hardware malfunctioning or software bugs that do not affect data of nonvolatile storage. (OS fault, RAM failure)
b) It is called fail stop assumption.

3) Disk failure

a. It is the failure caused due to head crash or data transfer failure.
b. Multiple backups are used to recover from this failure.

Log based recovery (Database Recovery)

- Log is the most commonly used structure for recording database modification/ activities in Database.
- Log is sequence of log records, recording all the update
- Update log has the following fields:
    a. Transaction Identifier

b. Data Item Identifier
c. Old Value (Prior to write)
d. New Value(After write)

Log Records
<T1 Start> ——-> This is Transaction identifier
<T1, X2, 10, 15> —>Old Value is 10, New value is 15: X2 = New value
<T1 Commit> —-> If committed X2 = 15
<T2 Abort> ——> If aborted X2 = old value; results in roll back
Note: We do not mention read in the log since it does not make any changes in the database.

Eg:

Transaction (a = b = 1000)	**Log**

To: read(a)
a = a-50
write(a)
read(b)
b = b + 50
write(b)
	< To Start>
	< To, a, 1000, 950>
	<To, b, 1000, 1050>
	<To commit>

- When-ever a transaction performs a write, it is essential that the log record for that work be created before database is modified.
- Once a log record exists, we can do the modification to DB. Also we have ability to undo the modification that has already been updated in DB.

Two approaches using Logs
1. Deferred Database Modification (Modification in commit state)
2. Immediate Database Modification (Modification in active state) : before commit state

Types of Data Backup

While the idea of data backup may be simple, implementing an effective and efficient strategy can be difficult. Backup software applications are developed to reduce the complexity associated with performing backup and recovery operations.

Remember that backup is not the goal. It is merely a means to accomplish your goal, which is protecting your data. The most common backup types used are as follows:

1. Full Backup: It is a basic and complete backup operation that makes a copy of all your data to another media set such as a disk, tape, or CD. Therefore, a complete copy of all your data is made available in a single media set. It takes longer to perform and requires a lot of storage space hence it is typically used in combination with either a differential or incremental backup.

2. Incremental Backup: This operation results in the copying of only that data that has changed since your last backup operation. A backup application will record and keep track of the time and date that all backup operations occur. This operation is faster and requires less storage media.

3. Differential Backup: Similar to an incremental type, this backup will copy all changed data from a previous episode but every time it runs, it continues to copy all the data changed since the stated previous full backup.

Immediate		**Deffered**	
Log	Database	Log	Database
<To Start>		<To Start>	
<To, A, 1000, 950>		<To, A, 950>	
<To, B, 2000, 2050>		<To, B, 2050>	

	A = 950
	B = 2050
		<To Commit>	

<To Commit>		{To End}	A = 950
			B = 2050

<T1 Start>		<T1 starts>	
<T1, C, 700, 600>		<T1, C, 600>	
	C = 600	<T1 Commit>	
<T1 Commit>		{T1 End}	C = 600

Shadow Paging
- It's an alternative to Log-based Recovery
- Each entry contains a pointer to page on disk
- the key idea is to maintain two page tables during the life of transaction
    - Current Page Table
    - Shadow Page Table

Maintain two page Tables during the lifecycle of Transaction. When Transaction starts both Page Tables are identical. Shadow Page Table is never changed over duration of Transaction. Current Page Table may changed during write operator.
All i/p and o/p operations use the current Page Table to locate db pages on disk
Store Shadow Page Table in Non-Volatile storage.
When transaction starts both tables are identical
Shadow page table is never change. The current page table makes all the changes.

When Transaction commits system writes current page table to non-volatile storage. The current page table then becomes new shadow page table.

Advantages of Shadow Paging over Log-Based Techniques

i. Log-record overhead is removed
ii. Faster 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.

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 –

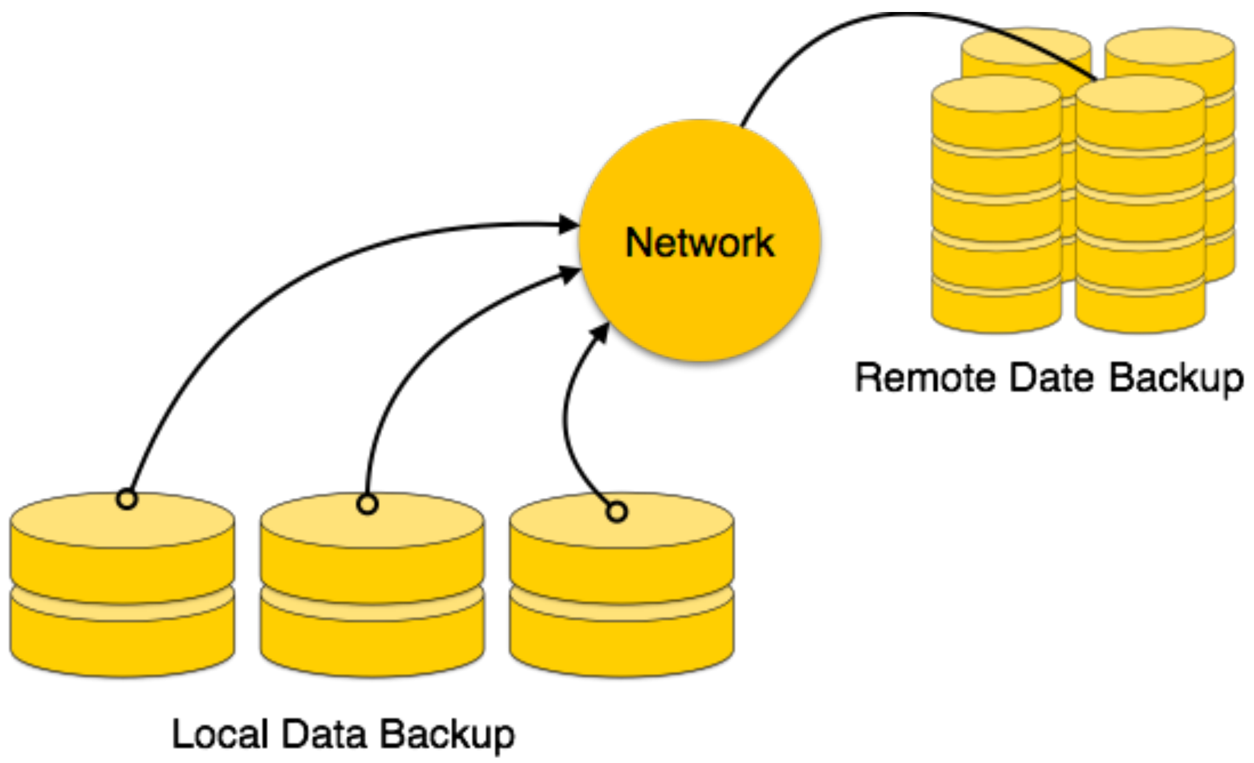
- Remote backup: Here a backup copy of the database is stored at a remote location from where it can be restored in case of a catastrophe.

- 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.

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.

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.