**(Recovery Techniques)**

**Database Recovery Techniques | DBMS**

**Database Recovery** is a process of recovering or restoring data in the database when a data loss occurs or data gets deleted by system crash, hacking, errors in the transaction, damage occurred coincidentally, by viruses, sudden terrible failure, commands incorrect implementation, etc. Data loss or failures happen in databases like other systems but the data stored in the database should be available whenever it's required. For fast restoration or recovery of data, the database must hold tools which recover the data efficiently. It should have atomicity means either the transactions showing the consequence of successful accomplishment perpetually in the database or the transaction must have no sign of accomplishment consequence in the database.

From any failure set of circumstances, there are both voluntary and involuntary ways for both, backing up of data and recovery. So, recovery techniques which are based on deferred update and immediate update or backing up data can be used to stop loss in the database.

**Crash recovery**

**Crash recovery** is the operation through which the database is transferred back to a compatible and operational condition. In DBMS, this is performed by rolling back insufficient transactions and finishing perpetrated transactions that even now existed in memory when the crash took place.

With many transactions being implemented with each second shows that, DBMS may be a tremendously complex system. The fundamental hardware of the system manages to sustain robustness and stiffness of software which depends upon its complex design. It’s anticipated that the system would go behind with some methodology or techniques to restore lost data when it fails or crashes in between the transactions.

**Classification of failure**

The following points are the generalization of failure into various classifications, to examine the source of a problem,

1. **Transaction failure**: a transaction has to terminate when it arrives at a point from where it can’t extend any further and when it fails to implement the operation.
2. Transaction failure reasons could be,

* **Logical errors:** The errors which take place in some code or any intrinsic error situation, where a transaction cannot properly fulfill.
* **System errors:** The errors which take place when the database management system is not able to implement the active transaction or it has to terminate it because of some conditions in a system.

1. **System Crash**: There are issues which may stop the system unexpectedly from outside and may create the system condition to crash. For example, disturbance or interference in the power supply may create the system condition of fundamental hardware or software to crash or failure.
2. **Disk Failure**: Disk failures comprise bad sectors evolution in the disk, disk inaccessibility, and head crash in the disk, other failures which damage disk storage completely or its particular parts.

**Storage structure**

The storage structure can be classified into two following categories,

* **Volatile storage**: A volatile storage cannot hold on crashes in the system. These devices are located within reach of CPU. Examples of volatile storage are the main memory and cache memory.
* **Non-volatile storage**: A non-volatile storage are created to hold on crashes in the system. These devices are enormous in the magnitude of data storage, but not quick in approachability. Examples of non-volatile storage are hard-disks, magnetic tapes, flash memory, and RAM.

**Recovery and Atomicity**

To recover and also to sustain the transaction atomicity, there are two types of methodology,

* Sustaining each transaction logs and before actually improving the database put them down onto some storage which is substantial.
* Sustaining shadow paging, in which on a volatile memory the improvements are completed and afterward, the real database is reformed.

**Log-based Recovery**

The log is an order of sequence of records, which sustains the operations record accomplished by a transaction in the database. Before the specific changes and improvements survive on a storage media which is stable and failing securely, it’s essential that the logs area unit put down in storage.

Following are the workings of Log-based Recovery,

The log file is not damaged on a stable storage media.

Log-based recovery puts down a log regarding a transaction when a transaction begins to be involved in the system and starts implementation.

**Recovery with Concurrent Transactions**

The logs are interleaved, when multiple transactions are being implemented in collateral. It would be difficult for the system of recovery to make an order of sequence of all logs again, and then start recovering at the time of recovery. Most recent times Database systems use the abstraction of 'checkpoints' to make this condition uncomplicated.

**Checkpoint**

The checkpoint is an established process where all the logs which are previously used are clear out from the system and stored perpetually in a storage disk. Checkpoint mention a point before which the DBMS was in a compatible state, and all the transactions were perpetrated.

**What is Database Recovery?**

It is the method of restoring the database to its correct state in the event of a failure at the time of the transaction or after the end of a process. Earlier, you have been given the concept of database recovery as a service that should be provided by all the DBMS for ensuring that the database is dependable and remains in a consistent state in the presence of failures. In this context, dependability refers to both the flexibility of the DBMS to various kinds of failure and its ability to recover from those failures. In this chapter, you will gather a brief knowledge of how this service can be provided. To gain a better understanding of the possible problems you may encounter in providing a consistent system, you will first learn about the need for recovery and its types of failure, which usually occurs in a database environment.

**What is the Need for Recovery of data?**

The storage of data usually includes four types of media with an increasing amount of reliability: the main memory, the magnetic disk, the magnetic tape, and the optical disk. Many different forms of failure can affect database processing and/or transaction, and each of them has to be dealt with differently. Some data failures can affect the main memory only, while others involve non-volatile or secondary storage also. Among the sources of failure are:

* Due to hardware or software errors, the system crashes, which ultimately resulting in loss of main memory.
* Failures of media, such as head crashes or unreadable media that results in the loss of portions of secondary storage.
* There can be application software errors, such as logical errors that are accessing the database that can cause one or more transactions to abort or fail.
* Natural physical disasters can also occur, such as fires, floods, earthquakes, or power failures.
* Carelessness or unintentional destruction of data or directories by operators or users.
* Damage or intentional corruption or hampering of data (using malicious software or files) hardware or software facilities.

Whatever the grounds of the failure are, there are two principal things that you have to consider:

* Failure of main memory, including that database buffers.
* Failure of the disk copy of that database.

**Recovery Facilities**

Every DBMS should offer the following facilities to help out with the recovery mechanism:

* Backup mechanism makes backup copies at a specific interval for the database.
* Logging facilities keep tracing the current state of transactions and any changes made to the database.
* Checkpoint facility allows updates to the database for getting the latest patches to be made permanent and keep secure from vulnerability.
* Recovery manager allows the database system for restoring the database to a reliable and steady-state after any failure occurs.

**Database Recovery in DBMS**

In DBMS, Database recovery is the process of resorting the database to a correct state following a failure. In general, backup and recovery refer to the various procedures involved in the protection of a database against data loss. Such that no data will be lost after failure.

**Database Recovery Techniques**

There are basically two methods to recover a database which is further divided into parts as shown in the diagram below.

**Log Based Recovery**

It is the most used structure for recovering database modification. In log-based recovery, a log file is maintained for recovery purposes. A logfile is a sequence of the log record. There are four types of log record as follows:

* <Start> log record
* <Update> log record
* <Commit> log record
* <Abort> log record.

**<Start> log record –**

It contains information about the start of each transaction.

**<Update> log record**

It describes a single database write and has the following fields.

                     < Ti, Xj, V1, V2>

Here, Ti is a transaction identifier, Xj is the data item, V1 is the old value of data item and V2 is the modified value.

**<Commit> log record**

When a transaction is successfully committed or completed then <commit> log record is stored in the logfile.

**<Abort> data record**

When a transaction is aborted due to any reason <abort> log record is stored in a log file.

**Deferred Database Modification**

It ensures [transaction](https://digitalnoteshub.com/2020/10/transaction.html) atomicity by recording all modifications in the log but deferring the execution of all write operations of a transaction until then transaction until the transaction partially commits.

The following table shows the state of a log record and database.

Log Database Sequence of time

< T1, start> (buffer)

<T1, A, 900>

<T1, B, 2100>

<T1, commit>

A=900, B = 2100 we do not the value of A and b stored in a physical disk or not, because the system decides output operations.

< T2, Start>

< T2, C, 2800>

< T2, commit>

C = 2800

Examples of state crash before the completion of the transaction is shown in the below table.

< T1, start> <T1, start> <T1, start>

<T1, A, 900> <T1, A, 900> <T1, A, 900>

<T1, B, 2100> <T1, B, 2100> <T1, B, 2100>

<T1, commit> <T1, commit>

<T2, start> <T2, start>

<T2, C, 2800> <T2, C, 200>

<T2, Commit>

**Immediate Database Modifications**

The immediate database modification techniques allow database modifications to output while the transaction is still in the active state. In order to understand operations, let us consider the format of the log record.

< Ti, Xj, V(old), V(new)>

Ti is a transaction identifier, Xj is the data item, V(old) is an old value of old data item and V(new) is the modified or new value of the data item.

Log Database Sequence of time

<T1, start>

<T1, A, 1000,900>

A=900 uncommitted update. If a transaction is aborted, new values must be replaced by old values.

<T1, B, 2000, 2100>

B=2100

<T1, Commit>

<T2, Start>

<T2, C, 3000,2800>

C=2800

<T2, commit>

Examples of a system crash the completion of a transaction

<T1, start> <T1, start> <T1, start>

<T1, A,1000,900> <T1, A, 1000,900> <T1, A, 1000,900>

<T1, B,2000,2100> <T1, B, 2000,2100> <T2, B, 2000, 2100>

<T1, commit> <T1, commit>

<T2, start> <T2, start>

<T2, C, 3000,2800> <T2, C, 3000, 2800>

<T2, commit>

**Checkpoints**

Checkpoints are also a data recovery technique. In the log-based algorithm, a log file is used. But keeping and maintaining logs in real-time may fill out all the memory space available in a system. A checkpoint is a technique in which all the previous logs are removed from the system and stored permanently in a storage disk.

Checkpoints declare a point before all the transactions were committed. Checkpoint maintains two lists an undo list and redo list.

Redo List – If the transaction recovery system sees a log with < Ti, start> and <Ti, commit> or just <Ti, commit>. Then it puts the transaction in the redo-list.

Undo List – If the recovery system sees a log with < Ti, start> but no commit or abort log found, then it puts the transaction in undo list.

The following actions are performed when a system issues a checkpoint.

All the transactions which are in undo list are then undone and their logs are also removed and the transactions which are in redo list their previous logs are removed.

**Database systems**, like any other computer system, are subject to failures but the data stored in it must be available as and when required. When a database fails it must possess the facilities for fast recovery. It must also have atomicity i.e. either transactions are completed successfully and committed (the effect is recorded permanently in the database) or the transaction should have no effect on the database.

There are both automatic and non-automatic ways for both, backing up of data and recovery from any failure situations. The techniques used to recover the lost data due to system crash, transaction errors, viruses, catastrophic failure, incorrect commands execution etc. are database recovery techniques. So to prevent data loss recovery techniques based on deferred update and immediate update or backing up data can be used.

Recovery techniques are heavily dependent upon the existence of a special file known as a **system log**. It contains information about the start and end of each transaction and any updates which occur in the **transaction**. The log keeps track of all transaction operations that affect the values of database items. This information is needed to recover from transaction failure.

* The log is kept on disk start\_transaction(T): This log entry records that transaction T starts the execution.
* read\_item(T, X): This log entry records that transaction T reads the value of database item X.
* write\_item(T, X, old\_value, new\_value): This log entry records that transaction T changes the value of the database item X from old\_value to new\_value. The old value is sometimes known as a before an image of X, and the new value is known as an afterimage of X.
* commit(T): This log entry records that transaction T has completed all accesses to the database successfully and its effect can be committed (recorded permanently) to the database.
* abort(T): This records that transaction T has been aborted.
* checkpoint: 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.

A transaction T reaches its **commit** point when all its operations that access the database have been executed successfully i.e. the transaction has reached the point at which it will not **abort** (terminate without completing). Once committed, the transaction is permanently recorded in the database. Commitment always involves writing a commit entry to the log and writing the log to disk. At the time of a system crash, item is searched back in the log for all transactions T that have written a start\_transaction(T) entry into the log but have not written a commit(T) entry yet; these transactions may have to be rolled back to undo their effect on the database during the recovery process

* **Undoing –** If a transaction crashes, then the recovery manager may undo transactions i.e. reverse the operations of a transaction. This involves examining a transaction for the log entry write\_item(T, x, old\_value, new\_value) and setting the value of item x in the database to old-value.There are two major techniques for recovery from non-catastrophic transaction failures: deferred updates and immediate updates.
* **Deferred update –** This technique does not physically update the database on disk until a transaction has reached its commit point. Before reaching commit, all transaction updates are recorded in the local transaction workspace. If a transaction fails before reaching its commit point, it will not have changed the database in any way so UNDO is not needed. It may be necessary to REDO the effect of the operations that are recorded in the local transaction workspace, because their effect may not yet have been written in the database. Hence, a deferred update is also known as the **No-undo/redo algorithm**
* **Immediate update –** In the immediate update, the database may be updated by some operations of a transaction before the transaction reaches its commit point. However, these operations are recorded in a log on disk before they are applied to the database, making recovery still possible. If a transaction fails to reach its commit point, the effect of its operation must be undone i.e. the transaction must be rolled back hence we require both undo and redo. This technique is known as **undo/redo algorithm.**
* **Caching/Buffering –** In this one or more disk pages that include data items to be updated are cached into main memory buffers and then updated in memory before being written back to disk. A collection of in-memory buffers called the DBMS cache is kept under control of DBMS for holding these buffers. A directory is used to keep track of which database items are in the buffer. A dirty bit is associated with each buffer, which is 0 if the buffer is not modified else 1 if modified.
* **Shadow paging –** It provides atomicity and durability. A directory with n entries is constructed, where the ith entry points to the ith database page on the link. When a transaction began executing the current directory is copied into a shadow directory. When a page is to be modified, a shadow page is allocated in which changes are made and when it is ready to become durable, all pages that refer to original are updated to refer new replacement page.

Some of the backup techniques are as follows :

* **Full database backup –** In this full database including data and database, Meta information needed to restore the whole database, including full-text catalogs are backed up in a predefined time series.
* **Differential backup –** It stores only the data changes that have occurred since last full database backup. When same data has changed many times since last full database backup, a differential backup stores the most recent version of changed data. For this first, we need to restore a full database backup.
* **Transaction log backup –** In this, all events that have occurred in the database, like a record of every single statement executed is backed up. It is the backup of transaction log entries and contains all transaction that had happened to the database. Through this, the database can be recovered to a specific point in time. It is even possible to perform a backup from a transaction log if the data files are destroyed and not even a single committed transaction is lost.

**OTHER REFRENCES**

•      [Database Recovery Techniques in DBMS - GeeksforGeeks](https://www.geeksforgeeks.org/database-recovery-techniques-in-dbms/)

•     [DATABASE RECOVERY IN DBMS AND ITS TECHNIQUES (whatisdbms.com)](https://whatisdbms.com/database-recovery-in-dbms-and-its-techniques/)

•    [Database Recovery Techniques | DBMS (includehelp.com)](https://www.includehelp.com/dbms/database-recovery-techniques.aspx)

**SUGGESTED BOOK REFERENCES**

* C.J.Date, “An Introduction to DatabaseSystems”, Addison Wesley.
* Thomas M. Connolly, Carolyn & E.Begg,“Database Systems: A Practical Approach to Design, Implementationand Management”, 5/E, University of Paisley, Addison-Wesley.
* Rob,”Database Principal Fundamental Design, Cengage Learning.