# ST.XAVIER’S COLLEGE

# MAITIGHAR, KATHMANDU

****

**ASSIGNMENT ON**

**Database Management System**

**Submitted By:**

Shaurabh Chapagain

013BSCCSIT037

**Submitted To:**

Er. Sanjay Kumar Yadav

Lecturer

Department of Computer Science

St. Xavier’s College

**Data recovery**

Data recovery is the process of restoring data that has been lost, accidentally deleted, corrupted or made inaccessible for any reason.

**Purpose of Data recovery**

The purpose of a database strategy is to protect the database against data loss and reconstruct the database after data loss, which can be listed as follow:

* Planning and testing responses to different kinds of failures
* Configuring the database environment for backup and recovery
* Setting up a backup schedule
* Monitoring the backup and recovery environment
* Troubleshooting backup problems
* Recovering from data loss if the need arises

**Types of Failure**

System failure can be classified into following types:

**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 which is called transaction failure. Reasons for a transaction failure could be as follow:

* **Logical error**s: 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**

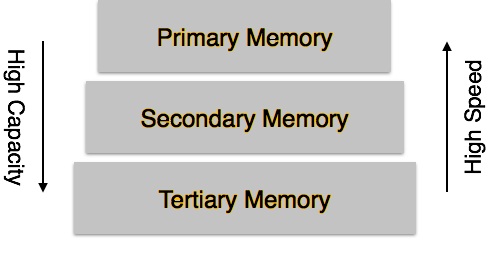
In early days of technology evolution, it was a common problem where hard-disk drives or storage drives used to fail frequently.

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.

**The Storage Hierarchy**

Databases are stored in file formats, which contain records. At physical level, the actual data is stored in electromagnetic format on some device. These storage devices can be broadly categorized into three types:

* **Primary Storage** − Memory storage that is directly accessible to the CPU comes under this category. CPU's internal memory (registers), fast memory (cache), and main memory (RAM) are directly accessible to the CPU, as they are all placed on the motherboard or CPU chipset. This storage is typically very small, ultra-fast, and volatile. Primary storage requires continuous power supply in order to maintain its state. In case of a power failure, all its data is lost.
* **Secondary Storage** − Secondary storage devices are used to store data for future use or as backup. Secondary storage includes memory devices that are not a part of the CPU chipset or motherboard, for example, magnetic disks, optical disks (DVD, CD, etc.), hard disks, flash drives, and magnetic tapes.
* **Tertiary Storage** − Tertiary storage is used to store huge volumes of data. Since such storage devices are external to the computer system, they are the slowest in speed. These storage devices are mostly used to take the back up of an entire system. Optical disks and magnetic tapes are widely used as tertiary storage.



* **Volatile storage** − 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.
* **Non-volatile storage** − 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, and non-volatile (battery backed up) RAM.

**Buffer Management**

Buffer management is a key component in achieving this efficiency. The buffer management component consists of two mechanisms: the buffer manager to access and update database pages, and the buffer cache (also called the buffer pool), to reduce database file I/O.

**Transaction log**

Transaction log backups can be used to restore a database to a specific point in time. A transaction log backup alone is not sufficient to restore a database. A backup of the data file is required as well. The data file backups are applied first. It is important that the logs are written prior to the actual modification and stored on a stable storage media, which is failsafe.

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.

**Data updates**

The database can be modified/updated 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.

**Data caching**

Caching is just the practice of storing data in and retrieving data from a high-performance store (usually memory) either explicitly or implicitly. Memory is faster to access than a file, a remote URL (usually), a database or any other external store of information you like. So if the act of using one of those external resources is **significant**then you may benefit from caching to increase performance.

**Transaction Roll-back (Undo) and Roll-forward**

**Rollback:** The Rollback transaction is a transaction which rolls back the transaction to the beginning of the transaction. The transaction can be rolled back completely by specifying the transaction name in the Rollback statement or to cancel any changes to a database during current transaction. It is permissible to use before Commit transaction.

**Roll forward:** Recovering a database by applying different transactions that recorded in the database log files. It is nothing but re-doing the changes made by a transaction i.e. after the committed transaction and to over write the changed value again to ensure consistency.

**Check pointing, shadow paging**

**Check pointing**: 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.

**Shadow paging:**

It is inconvenient to maintain logs of all transactions from the purposes of recovery. An alternative is to use a system of shadow paging. This is where the database is divided into pages that may be stored in any order on the disk. During the life of a transaction two page tables are maintained, one called a shadow page table and current page table. When a transaction begins both of these page tables point to the same locations (are identical).

During the lifetime of a transaction the shadow page table doesn't change at all. However during the lifetime of a transaction changes may be made update values etc. So whenever we update a page in the database we always write the updated page to a new location. This means that when we then update our current page table to reflect the changes that have been made.

**Disadvantages of the shadow paging system**

* Data will suffer from fragmentation as the data is divided up into pages that may or not be in linear order for large sets of related data.
* Garbage will accumulate in the pages on the disk as data is updated and pages lose any references. For example if i have a page that contains a data item X that is replaced with a new value then a new page will be created (remember we always create a new page to update data in a new location). Once the shadow page table is updated nothing will reference the old value of X.

**Recovery schemes (WAL: Write Ahead Logging protocol)**

WAL protocol Write-ahead logging (WAL) is a family of techniques for providing atomicity and durability (two of the ACID properties) in database systems.  
In a system using WAL, all modifications are written to a log before they are applied. Usually both redo and undo information is stored in the log. Write-ahead logging is employed to flush log records to the persistent log file before data pages are written or at commit time

**Failure with loss of Non-volatile storage (General concept)**

Non-volatile are those storage which survives system crashes. Examples: disk, tape, flash memory, non-volatile (battery backed up) RAM

* So far we assumed no loss of non-volatile storage
* Technique similar to check-pointing used to deal with loss of non-volatile storage
* Periodically dump the entire content of the database to stable storage
* No transaction may be active during the dump procedure; a procedure similar to check-pointing must take place
* Output all log records currently residing in main memory on to stable storage
* Output all buffer blocks onto the disk.
* Copy the contents of the database to stable storage.
* Output a record <dump> to log on stable storage.

**Recovery in multi-database system**