ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**Database Management System**

**Theory Lab Assignment**

**SUBMITTED BY:**

Sneha Prasai

013BSCCSIT040

**SUBMITTED TO**

|  |  |
| --- | --- |
| **Er. Sanjay Kr. Yadav**  **( Lecturer )** |  |
| **Department of Computer Science** | |

DATABASE RECOVERY:

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

1. PURPOSE OF DATA REOVERY

The data recovery process may vary, depending on the circumstances of the data loss, the data recovery software used to create the backup, and the backup target media. For example, many desktop and laptop backup software platforms allow end users to restore lost files themselves, while restoration of a corrupted database from a[tape backup](http://searchstorage.techtarget.com/definition/tape-backup) is a more complicated process that requires IT intervention. Data recovery can also be provided as service. Such services are typically used to retrieve important files that were not backed up and accidentally deleted from a computer's file system but still remain on disk in fragments.

1. TYPES OF FAILURE

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

2.1 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.

2.2 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.

2.3 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.

1. 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** − The 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.

1. BUFFER MANAGEMENT

The **buffer** is the part of Memory Manager available for storage of **copies** of disk blocks. The subsystem responsible for the allocation of buffer space is called the **buffer manager**. The buffer manager handles all requests for blocks of the database. If the block is already in Memory Manager, the address in Memory Manager is given to the requestor. If not, the buffer manager must read the block in from disk (possibly displacing some other block if the buffer is full) and then pass the address in Memory Manager to the requestor.

The buffer manager must use some sophisticated techniques in order to provide good service:

* 1. **Replacement Strategy** - When there is no room left in the buffer, some block must be removed to make way for the new one. Typical operating system memory management schemes use a ``least recently used'' (**LRU**) method. (Simply remove the block least recently referenced.) This can be improved upon for database applications.
  2. **Pinned Blocks** - For the database to be able to recover from crashes, we need to restrict times when a block maybe written back to disk. A block not allowed to be written is said to be **pinned**. Many operating systems do not provide support for pinned blocks, and such a feature is essential if a database is to be ``crash resistant''.
  3. **Forced Output of Blocks** - Sometimes it is necessary to write a block back to disk even though its buffer space is not needed. (Called the **forced output** of a block.) This is due to the fact that MM contents (and thus the buffer) are lost in a crash, while disk data usually survives.
  4. Buffer manager should free the space occupied by a borrow block as

1. TRANSACTION LOG

A DBMS uses a transaction log to keep track of all transactions that update the database. The information stored in this log is used by the DBMS for a recovery requirement triggered by a ROLLBACK statement, a program’s abnormal termination, or a system failure such as a network discrepancy or a disk crash. Some RDBMSs use the transaction log to recover a database forward to a currently consistent state. After a server failure, for example, Oracle automatically rolls back uncommitted transactions and rolls forward transactions that were committed but not yet written to the physical database.

While the DBMS executes transactions that modify the database, it also automatically updates the transaction log.

The transaction log stores:

* A record for the beginning of the transaction.
* For each transaction component (SQL statement):
  + The type of operation being performed (update, delete, insert).
  + The names of the objects affected by the transaction (the name of the table).
  + The “before” and “after” values for the fields being updated.
  + Pointers to the previous and next transaction log entries for the same transaction.
* The ending (COMMIT) of the transaction.

1. DATA UPDATES
2. DATA CACHING
3. TRANSACTION ROLL BACK(UNDO) AND ROLL FORWARD
4. CHECK POINTING, SHADOW PAGING
5. RECOVERY SCHEMES (WAL: WRITE AHEAD LOGGING PROTOCOL)
6. FAILURE WITH LOSS OF NON- VOLATILE STORAGE [GENERAL CONCEPT]
7. RECOVERY IN MULTIDATABASE SYSTEM