ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**Database Management System**

**Theory Assignment#9**

**Submitted by:**

Arjun Neupane

013BSCCSIT009

**Submitted to:**

|  |  |
| --- | --- |
| Er. Sanjay K. Yadav  Lecturer, St. Xavier’s College |  |

Date of submission: October 1, 2015

**Database 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.

**1.Purpose of Data Recovery:**

In enterprise information technology (IT), data recovery typically refers to the restoration of data to a desktop, laptop, server, or external storage system from a backup. 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 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.

An organization's disaster recovery plan should make known who in the organization is responsible for recovering data, provide a strategy for how data will be recovered and document acceptable recovery point and recovery time objectives.

**2. 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.

The reasons for 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.

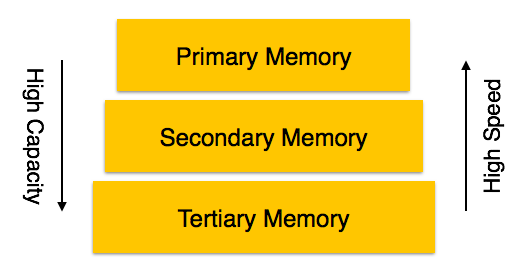
**2.3. Disk Failure:**

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.

**3. 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 –



**3.1. 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.

**3.2. 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.

**3.3. 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.

**4. Buffer Management:**

A DBMS must manage a huge amount of data, and in the course of processing the required space for the blocks of data will often be greater than the memory space available. For this there is the need to manage a memory in which to load and unload the blocks. The buffer manager is responsible primarily for managing the operations inherent saving and loading of the blocks.

In fact, the operations that provide the buffer manager are these:

•**FIX:** This command tells the operator of the buffer to load a block from disk and return the pointer to the memory where it is loaded. If the block was already in memory, the buffer manager needs only to return the pointer, otherwise he must load from disk and bring it into memory. If the buffer memory is full but it is possible to have 2 situations: or the possibility of releasing a portion of memory that is occupied by transactions already completed. In this case, before freeing the area the content is written to disk if any block of this area had been changed.

There is the possibility of free memory to be occupied because transitions still ongoing. In this case, the buffer manager can work in 2 ways:

* In the first mode (STEAL), the operator of the free buffer memory occupied by a transition already active, possibly saving your changes to disk.
* In the second mode (NOT STEAL), the transition requested block is made to wait until the free memory.

• **SET DIRTY:** invoking this command, you mark a block of memory as amended. Before introducing the last 2 commands you need to anticipate that the DMBS can operate in 2 modes: Force and NOT FORCE. When working in FORCE mode, the rescue disk is in synchronous mode with the commit of a transaction. When working mode is NOT FORCE the rescue is carried out from time to time in asynchronous manner. Typically, commercial database operating mode NOT FORCE because this allows an increase in performance: the block may undergo multiple changes in memory before being saved, then you can choose to make the saves when the system is unloading.

•**Force:** This command will cause the operator of the buffer to make the writing in synchronously with the completion (commit) the transaction

•**FLUSH:** This command will cause the operator of the buffer to perform the rescue,when in how NOT FORCE.

**5. Transaction Log:**

A transaction is a sequential group of database manipulation operations, which is performed as if it were one single work unit. In other words, a transaction will never be complete unless each individual operation within the group is successful. If any operation within the transaction fails, the entire transaction will fail.

**Properties:**

Transactions have the following four standard properties, usually referred to by the acronym ACID:

**Atomicity:** ensures that all operations within the work unit are completed successfully; otherwise, the transaction is aborted at the point of failure and previous operations are rolled back to their former state.

**Consistency:** ensures that the database properly changes states upon a successfully committed transaction.

**Isolation:** enables transactions to operate independently on and transparent to each other.

**Durability:** ensures that the result or effect of a committed transaction persists in case of a system failure.

There are following commands used to control transactions:

**COMMIT:** to save the changes.

**ROLLBACK:** to rollback the changes.

**SAVEPOINT:** creates points within groups of transactions in which to ROLLBACK

**SET TRANSACTION:** Places a name on a transaction.

**6. Data Updates:**

**7. Data Coaching:**

**8. Transaction Roll back (Undo):**

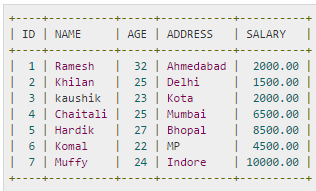
The **ROLLBACK** command is the transactional command used to undo transactions that have not already been saved to the database.

The **ROLLBACK** command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

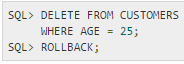
The syntax for ROLLBACK command is as follows:

****

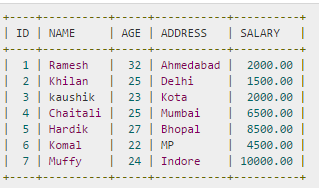
Example: consider a table CUSTOMER having following records



Following is the example, which would delete records from the table having age = 25 and then ROLLBACK the changes in the database.



As a result, delete operation would not impact the table and SELECT statement would produce the following result:

****

**9. Transaction Roll forward:**

**10. Check Pointing, Shadow paging**

**11. Recovery Schemes (WAL: Write Ahead Logging Protocol)**

**12. Failure with Loss of Non-volatile storage [General Concepts]**

**13. Recovery in Multi database System**

**Reference:**

http://www.tutorialspoint.com/dbms/dbms\_data\_recovery.htm