ST.XAVIER’S COLLEGE

**Maitighar, Kathmandu**



**DBMS LAB ASSIGNMENT**

**Submitted By:**

**Utsav Luitel**

013BSCCSIT046

**Submitted To:**

**Er. Sanjay Kumar Yadav** ………..…………….

**Lecturer,**

**Department of Computer Science**

**Date of Submission:- *2015 October***

Database Recovery

In computing, **data recovery** is a process of salvaging inaccessible **data** from corrupted or damaged secondary storage, removable media or files, when the **data** they store cannot be accessed in a normal way.

* 1. **Purpose of Data recovery.**

As a backup administrator, your principal duty is to devise, implement, and manage a backup and recovery strategy. In general, the purpose of a backup and recovery strategy is to protect the database against data loss and reconstruct the database after data loss. Typically, backup administration tasks include the following:

* 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

Example:

* If the system crashes before a fund transfer transaction completes its execution, then either one or both accounts may have incorrect value. Thus, the database must be restored to the state before the transaction modified any of the accounts.

**3) The storage hierarchy**

The range of memory and storage within and attached to a computer system is known as the Storage Hierarchy and to help understand this further can be categorised into 4 segments. As memory and storage devices move down the hierarchy they reduce in performance and cost/MB or GB but tend to rise in capacity through to the last category which includes removable media which in effect has no restriction on capacity a device can store.

Primary Storage is the top level and is made up of CPU registers, CPU cache and memory which are the only components that are directly accessible to the systems CPU. The CPU can continuously read data stored in these areas and execute all instructions as required quickly in a uniform manner. Secondary Storage differs from primary storage in that it is not directly accessible by the CPU. A system uses input/output (I/O) channels to connect to the secondary storage which control the data flow through a system when required and on request

Secondary storage is non-volatile so does not lose data when it is powered down so consequently modern computer systems tend to have a more secondary storage than primary storage. All secondary storage today consist of hard disk drives (HDD), usually set up in a RAID configuration, however older installations also included removable media such us magneto optical or MO

1. **Types of failure**

**User Error:**

A database administrator can do little to prevent user errors (for example, accidentally dropping a table). Usually, user error can be reduced by increased training on database and application principles. Furthermore, by planning an effective recovery scheme ahead of time, the administrator can ease the work necessary to recover from many types of user errors.

**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.

**Storage Structure**

* **Volatile storage:**

examples:

* main memory,
* cachememory
* **Nonvolatile storage:**
  + survives system crashes
  + examples: disk, tape, flash memory,
  + non-volatile (battery backed up) RAM
* **Stable storage:**
  + a mythical form of storage that survives all failures
  + approximated by maintaining multiple copies on distinct
  + nonvolatile media

**Transactions roll back and roll forward**

Database buffers in the buffer cache in the SGA are written to disk only when necessary, using a least-recently-used algorithm. Because of the way that the DBWn process uses this algorithm to write database buffers to datafiles, datafiles might contain some data blocks modified by uncommitted transactions and some data blocks missing changes from committed transactions.

Two potential problems can result if an instance failure occurs:

* Data blocks modified by a transaction might not be written to the datafiles at commit time and might only appear in the redo log. Therefore, the redo log contains changes that must be reapplied to the database during recovery.
* After the roll forward phase, the datafiles may contain changes that had not been committed at the time of the failure. These uncommitted changes must be rolled back to ensure transactional consistency. These changes were either saved to the datafiles before the failure, or introduced during the roll forward phase.

To solve this dilemma, two separate steps are generally used by Oracle for a successful recovery of a system failure: rolling forward with the redo log (cache recovery) and rolling back with the rollback segments (transaction recovery).

**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 [4].

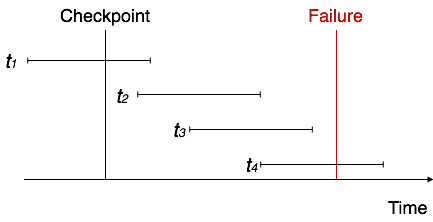
**Roll forward:** Recovering a database by applying different transactions that recorded in the database log files. It is nothing but re-doing the changes made by a transaction i.e. after the committed transaction and to over write the changed value again to ensure consistency [4].

**9. Checkpoint**

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.

Recovery

When a system with concurrent transactions crashes and recovers, it behaves in the following manner −



The recovery system reads the logs backwards from the end to the last checkpoint.

It maintains two lists, an undo-list and a redo-list.

If the recovery system sees a log with <Tn, Start> and <Tn, Commit> or just <Tn, Commit>, it puts the transaction in the redo-list.

If the recovery system sees a log with <Tn, Start> but no commit or abort log found, it puts the transaction in undo-list.

# Data Update

* + **Immediate Update**: As soon as a data item is modified in cache, the disk copy is updated.
  + **Deferred Update**: All modified data items in the cache is written either after a transaction ends its execution or after a fixed number of transactions have completed their execution.
  + **Shadow update**: The modified version of a data item does not overwrite its disk copy but is written at a separate disk location.
  + **In-place update**: The disk version of the data item is overwritten by the cache version.

# Data Caching

* + Data items to be modified are first stored into database cache by the Cache Manager (CM) and after modification they are flushed (written) to the disk.
  + The flushing is controlled by **Modified** and **Pin-Unpin** bits.
    - **Pin-Unpin**: Instructs the operating system not to flush the data item.
    - **Modified**: Indicates the AFIM of the data item.

# Transaction Roll-back (Undo) and Roll-Forward (Redo)

* + To maintain atomicity, a transaction’s operations are redone or undone.
    - **Undo**: Restore all BFIMs on to disk (Remove all AFIMs).
    - **Redo**: Restore all AFIMs on to disk.
  + Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.

# Checkpointing

* + Time to time (randomly or under some criteria) the database flushes its buffer to database disk to minimize the task of recovery. The following steps defines a checkpoint operation:
    1. Suspend execution of transactions temporarily.
    2. Force write modified buffer data to disk.
    3. Write a [checkpoint] record to the log, save the log to disk.
    4. Resume normal transaction execution.

During recovery redo or undo is required to transactions appearing after [checkpoint] record.

# Shadow Paging

* The AFIM does not overwrite its BFIM but recorded at another place on the disk. Thus, at any time a data item has AFIM and BFIM (Shadow copy of the data item) at two different places on the disk.



X and Y: Shadow copies of data items

X' and Y': Current copies of data items

* To manage access of data items by concurrent transactions two directories (current and shadow) are used.

The directory arrangement is illustrated below. Here a page is a data item.

