ST. XAVIER’S COLLEGE

**Maitighar,Kathmandu**

**(Affiliated to Tribhuvan University)**



**Database Management System**

**Lab Assignment #9**

**Submitted By**

Ajita Khatiwada

B.Sc. CSIT

Year II/IV Semester

013BSCIT004

**Submitted To**

Er. Sanjay Kumar Yadav

Lecturer,

Department of Computer Science

St. Xavier’s College

Maitighar, Kathmandu

**Submitted On**

2nd October ,2015.

1. **Databse Recovery**

Retrieving deleted/inaccessible data from electronic storage media (hard drives, removable media, optical devices, etc...)

Typical causes of loss include:

* 1. Electro-mechanical Failure
  2. Natural Disaster
  3. Computer Virus
  4. Data Corruption
  5. Computer Crime

1. **Purpose of data recovery**

A database may become inconsistent because of a

* 1. transaction failure (abort)
  2. database system failure (possibly caused by OS crash)
  3. media crash (disk-resident data is corrupted)

The recovery system ensures the database contains exactly those updates produced by committed transactions .

1. **Types of failure**

**1.Transaction failures**

* 1. **Logical errors**: transaction cannot complete due to some internal error condition
  2. **System errors**: the database system must terminate an active transaction due to an error condition (e.g., deadlock)

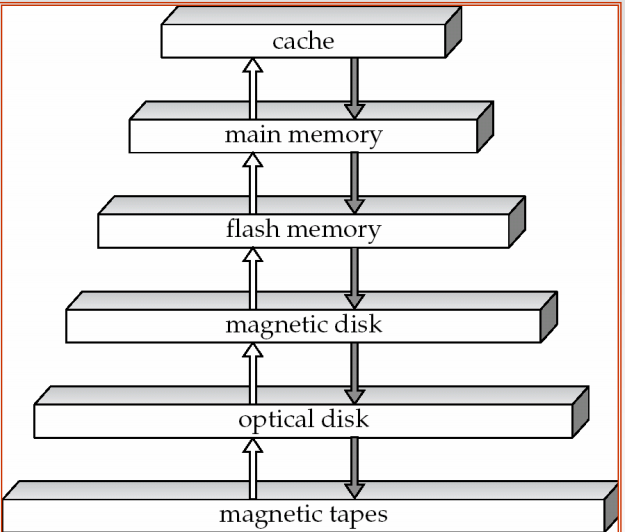
**2. System crash:** A power failure or other hardware or software failure causes the system to crash.Fail-stop assumption: non-volatile storage contents are assumed to not be corrupted by system crash. Database systems have numerous integrity checks to prevent corruption of disk data

**3** .**Disk failure**: a head crash or similar disk failure destroys all or part of disk storage

**4. Media failures**

* problems with disk head, unreadable media surface
* (parts of ) information on secondary storage may be lost
* lose a disk or a disk controller fails
* a head crash
* file corruption
* the overwriting or deletion of a datafile
* Any of the multiplexed control files are deleted or lost
* any datafile belonging to the system or the undo tablespace is deleted or lost.
* entire redo log group is lost.[1]

1. **The storage hierarchy**



* primary storage: Fastest media but volatile (cache, main memory). 
* secondary storage: next level in hierarchy, non-volatile, moderately fast access time z also called on-line storage z E.g. flash memory, magnetic disks 
* tertiary storage: lowest level in hierarchy, non-volatile, slow access time z also called off-line storage z E.g. magnetic tape, optical storage[2][3]

1. **Buffer management**

Data must be in RAM for DBMS to operate on it. Buffer Mgr hides the fact that not all data is in RAM

* When a Page is Requested
  + Buffer pool information table contains:
  + <frame#, pageid, pin\_count, dirty>
* If requested page is not in pool:
  + Choose a frame for replacement (only un-pinned pages are candidates)
  + If frame is “dirty”, write it to disk
  + Read requested page into chosen frame
  + Pin the page and return its address.
* Once we choose a page to remove
  + A page is dirty, if its contents have been changed after writing
  + Buffer Manager keeps a dirty bit
  + Say we choose to evict, if P is dirty, we write it to disk [4]

1. **Transaction log**

A **transaction** is an action, or a series of actions, carried out by a single user or an application program, which reads or updates the contents of a database.

* The transaction log records the details of all transactions
  + Any changes the transaction makes to the database
  + How to undo these changes
  + When transactions complete and how
  + The log is stored on disk, not in memory
  + If the system crashes it is preserved
* Write ahead log rule:

The entry in the log must be made before COMMIT processing can complete

1. **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.[6]

1. **Data caching**
   1. 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.
   2. The flushing is controlled by **Modified** and **Pin-Unpin** bits.
      1. **Pin-Unpin**: Instructs the operating system not to flush the data item.

**ii.Modified**: Indicates the AFIM of the data item[7]

1. **Transaction roll back(undo)&roll forward**

**forward recovery(roll forward)** - Assume that there is a disaster causing us to restore the DBMS to the state is was in two days ago, as that is the date of our last reliable backup. Forward recovery would then allow us to process the journal of transactions to bring the DBMS back to the present. Obviously, this requires that journals be kept around for some time to make this possible.

**backward recovery** - also called **rollback**, this is the process of backing away from an error condition by restoring the system to an earlier state. You can do this by simply performing a restore, or by undoing the transactions in a journal in reverse order.[5]

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.[8]

1. **Check pointing,Shadow paging**

A checkpoint is a procedure to limit the amount of work for Restart.

**Shadow paging** is an alternative to log-based recovery; this scheme is useful if transactions execute serially

Idea: maintain *two* page tables during the lifetime of a transaction –the **current page table**, and the **shadow page table**

Store the shadow page table in nonvolatile storage, such that state of the database prior to transaction execution may be recovered.

* + Shadow page table is never modified during execution

To start with, both the page tables are identical. Only current page table is used for data item accesses during execution of the transaction.

Whenever any page is about to be written for the first time

* + A copy of this page is made onto an unused page.
  + The current page table is then made to point to the copy

The update is performed on the copy

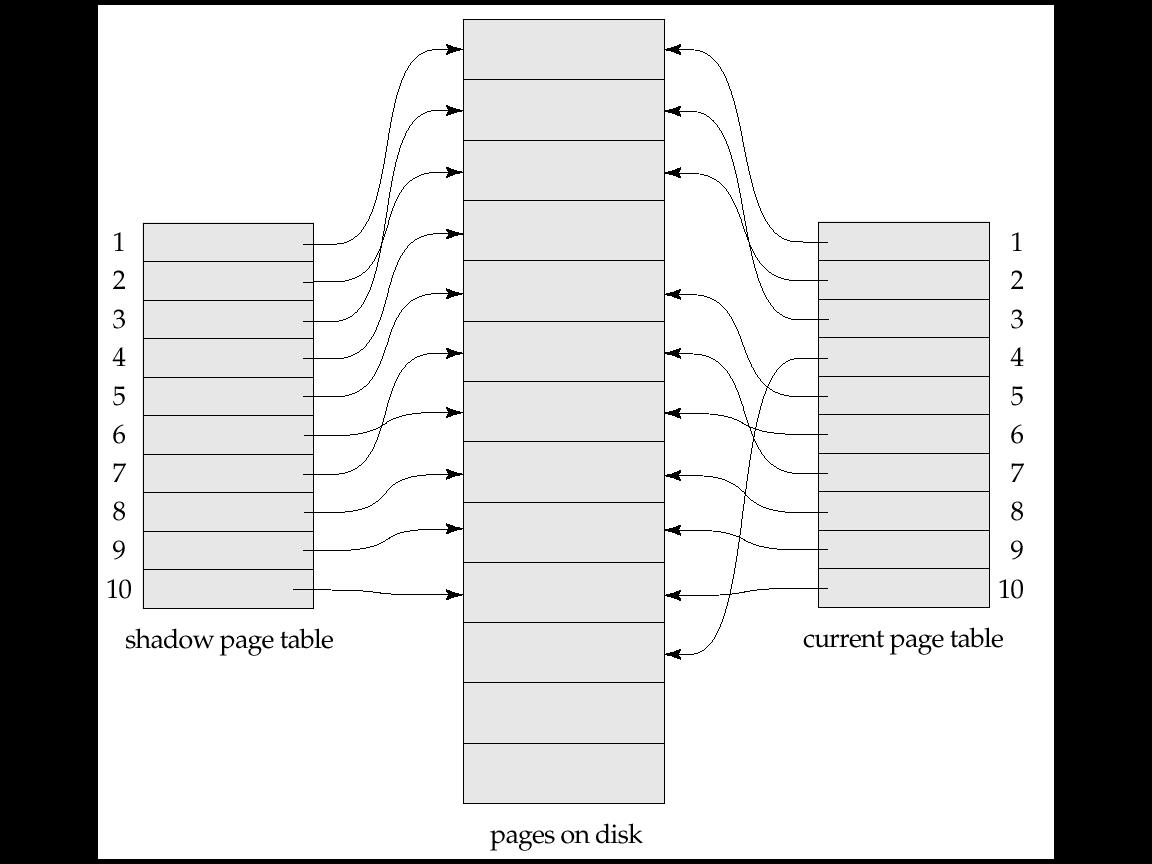


Fig :page table

1. **Recovery schemes(WAL)**

**Write-Ahead Logging**

When in-place update (immediate or deferred) is used then log is necessary for recovery and it must be available to recovery manager. This is achieved by Write-Ahead Logging (WAL) protocol. WAL states that

* + For Undo: Before a data item’s AFIM is flushed to the database disk (overwriting the BFIM) its BFIM must be written to the log and the log must be saved on a stable store (log disk).
  + For Redo: Before a transaction executes its commit operation, all its AFIMs must be written to the log and the log must be saved on a stable store.[9]

1. **Failure with loss of non volatile storage (general concepts** )

Technique similar to checkpointing used to deal with loss of non-volatile storage

* 1. Periodically **dump** the entire content of the database to stable storage
  2. No transaction may be active during the dump procedure; a procedure similar to checkpointing must take place
     1. Output all log records currently residing in main memory onto stable storage.
     2. Output all buffer blocks onto the disk.
     3. Copy the contents of the database to stable storage.
     4. Output a record <**dump**> to log on stable storage.

1. **Recovery in Multidatabase systems**

A multi database transaction require access to multiple databases.

– The DBs may even be stored on different types of DBMS.

• Some DBMS may be relational, whereas others are object oriented, etc.

– Each DBMS involved in the multi database transaction may have its own recovery technique and transaction manager separate from those of the other DBMSs.

• Use a two-level recovery mechanism to maintain the atomicity of a multi database transaction.

– A global recovery manager, or coordinator.

– The local recovery managers.

The coordinator usually follows a two-phase commit protocol.

– Phase 1

– When all participating databases signal the coordinator that the part of the multi database transaction has concluded, the coordinator sends a message «prepare to commit» to each participant to get ready for committing the transaction.

– Each participating database receiving that message will force-write all log records and needed information for local recovery to disk and then send a «ready to commit» -or OK- signal to the coordinator or «cannot commit» -or not OK- if it fails for some other reasons.

– If the coordinator does not receive a reply from a database

– Phase 2

• If all the participants DB reply «OK» and also the coordinator, the transaction is successful and the coordinator sends a «commit» signal for the transaction to the participant databases.

– Each participant database completes transaction commit by writing a [commit] entry for the transaction in the log and permanently updating the database if needed.

• If one or more participating DBs or the coordinator sends «not OK» message, the transaction fails and the coordinator sends a message to «rollback» -or UNDO- the local effect of the transaction to each participating database.

– The UNDO of the local effect is done by using the log at each participating database

**Reference :**

[1]” <http://webdocs.cs.ualberta.ca/~zaiane/courses/cmput391/slides/L9-391-04.pdf>”, October 1, 2015.

[2]” [http://codex.cs.yale.edu/avi/db-book/db5/slide-dir/ch11.pdf”,October](http://codex.cs.yale.edu/avi/db-book/db5/slide-dir/ch11.pdf) 2,2015

[3]” [http://codeidol.com/community/sql/the-storage-hierarchy/3437/”October](http://codeidol.com/community/sql/the-storage-hierarchy/3437/) 2 2015

[4]” <http://www.inf.uni-konstanz.de/dbis/teaching/ss04/architektur-von-dbms/download/local/buffermanager.pdf>” Oct-1 2015

[5]” [http://stevevincent.info/CIS331\_7.htm”,October](http://stevevincent.info/CIS331_7.htm) 2 2015

[6]” http://faculty.ksu.edu.sa/benchikhm/IS335/Chap6-Recovery-techniques.pdf”, Oct-2 2015

[7][8][9]” https://www.cs.purdue.edu/homes/ake/cs348/Chapter19.ppt”,October 2 2015