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**

A database may become inconsistent because of a

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

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

I.e. atomicity and durability, despite failures

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.

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
   1. **Immediate Update**: As soon as a data item is modified in cache, the disk copy is updated.
   2. **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.
   3. **Shadow update**: The modified version of a data item does not overwrite its disk copy but is written at a separate disk location.
   4. **In-place update**: The disk version of the data item is overwritten by the cache version.
2. DATA CACHING:

The Database Management System (DBMS) is a memory buffer which stores copies of portions of the database that the DBMS is currently using. Reading from memory is much faster than reading from the disk. The DBMS therefore returns a record more quickly if it is already stored in cache. As long as the required data is stored in cache, the data is immediately available. When the required data is not stored in cache, it must be copied from the disk and then stored in cache.

The DBMS cache is transparent to the user. For example, when a user requests data, the data is automatically copied into the cache and stored there. If the data is modified, it is automatically copied back to the physical disk. These data transfers take place automatically. The user does not need to know about the cache.

The DBMS cache always contains the most recently used data. The cache is continually updated with the relevant data from the database.

1. TRANSACTION ROLL BACK(UNDO) AND ROLL FORWARD:-

Roll forward: The Rollforward is redoing the changes made by a transaction that is after the committed transaction and to over-write the changed value once again to ensure the consistency.

Roll back: The Rollback transaction is a transaction which rolls back the transaction to the beginning of the transaction (Rollback Transaction\_name). It is possible to use before Commit transaction.

1. 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:

* Alternative to log-based recovery
* 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, and the update is performed on the copy.
* To commit a transaction:
* 1. Flush all modified pages in main memory to disk
* 2. Output current page table to disk
* 3. Make the current page the new shadow page table – keep a pointer to the shadow page table at a fixed (known) location on disk.

1. RECOVERY SCHEMES (WAL: WRITE AHEAD LOGGING PROTOCOL)

In computer science, **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.

The purpose of this can be illustrated by an example. Imagine a program that is in the middle of performing some operation when the machine it is running on loses power. Upon restart, that program might well need to know whether the operation it was performing succeeded, half-succeeded, or failed. If a write-ahead log is used, the program can check this log and compare what it was supposed to be doing when it unexpectedly lost power to what was actually done. On the basis of this comparison, the program could decide to undo what it had started, complete what it had started, or keep things as they are.

WAL allows updates of a database to be done in-place. Another way to implement atomic updates is with shadow paging, which is not in-place. The main advantage of doing updates in-place is that it reduces the need to modify indexes and block lists.

1. FAILURE WITH LOSS OF NON- VOLATILE STORAGE [GENERAL CONCEPT]
2. RECOVERY IN MULTIDATABASE SYSTEM