**ST.XAVIER’S COLLEGE**

MAITIGHAR, KATHMANDU



Database Management System

Assignment #9

Submitted By:

Aabhash Dhakal

013BSCCSIT001

2nd year/ 4th semester

Submitted to:

|  |  |
| --- | --- |
| Er. Sanjay Kumar Yadav  Lecturer  Department of Computer Science |  |

# Database Recovery

* 1. **Purpose of Data Recovery**

Recovery techniques can be used to restore data in a system to a usable state. Such techniques are widely used in filing systems and database systems in order to cope with failures. A failure is an event at which the system does not perform according to specifications. Some failures are caused by hardware faults (e.g., a power failure or disk failure), software faults {e.g., bugs in programs or invalid data), or human errors (e.g., the operator mounts a wrong tape on a drive, or a user does something unintentional). A failure occurs when an erroneous state of the system is processed by algorithms of the system. The term error is, in this context, used for that part of the state which is "incorrect." An error is thus a piece of information which can cause a failure.

In order to cope with failures, additional components and abnormal algorithms can be added to a system. These components and algorithms attempt to ensure that occurrences of erroneous states do not result in later system failures; ideally, they remove these errors and restore them to "correct" states from which normal processing can continue.

* 1. **Types of Failure**

Failures may be

|  |  |
| --- | --- |
| Transaction | Caused by errors within the transaction processes. |
| System | Caused by failure of network or operating system or physical threats to the system as a whole. |
| Media | Failure of hard disk, out of memory errors, out of disk space errors. |

Transaction errors, system errors, system crashes, concurrency problems and local errors or exceptions are the more common causes of system failure.  The system must be able to recover from such failures without loss of data.

* 1. **The Storage Hierarchy**
  2. **Buffer Management**
  3. **Transaction Log**

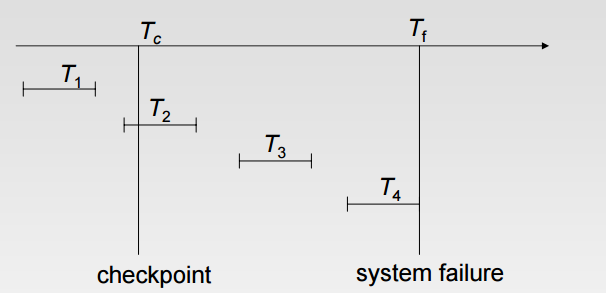
A DBMS uses a transaction log to keep track of all transactions that update the database. The information stored in this log is used by the DBMS for a recovery requirement triggered by a ROLLBACK statement, a program’s abnormal termination, or a system failure such as a network discrepancy or a disk 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 (updates, 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.
  1. **Data Updates**
  2. **Data Caching**
  3. **Transaction Rollback and Roll-forward**
  4. **Check Pointing, shadow paging**

We can streamline the recovery procedure by periodically performing check pointing. All log records correctly residing in main memory are output onto stable storage. Similarly, we output all modified buffer blocks to the disk. Finally, we write a log record <checkpoint> onto stable storage.

During recovery, we need to consider only the most recent transaction Ti that started before the checkpoint and transactions that started after Ti.

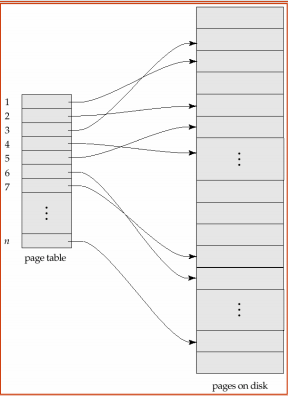
* Scan backwards from end of log to find the most recent <checkpoint> record.
* Continue scanning backwards till a record <Ti start> is found.
* Need only consider the part of log following above start record. Earlier part of log can be ignored during recovery, and can be erased whenever desired.
* For all transactions (starting from Ti or later) with no <Ti commit>, execute undo (Ti).
* Scanning forward in the log, for all transactions starting from Ti or later with a < Ti commit>, execute redo (Ti).

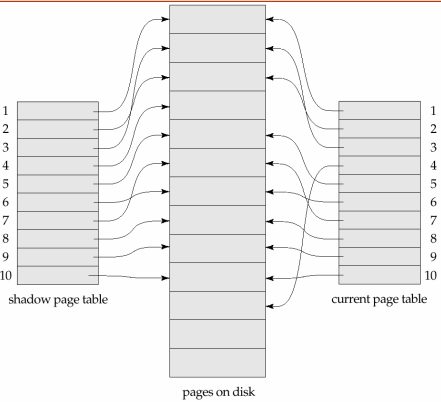


* T1 can be ignored.
* T2 and T3 re-done.
* T4. Undone.

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. z 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





* 1. **Recovery Schemes (WAL : Write Ahead Logging Protocol)**
  2. **Failure with Loss of Non-Volatile Storage (General Concepts)**
  3. **Recovery in Multi-database system**