**ST.XAVIER’S COLLEGE**

MAITIGHAR, KATHMANDU



Database Management System

Assignment #

Submitted By:

Aabhash Dhakal

013BSCCSIT001

2nd year/ 4th semester

Submitted to:

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

# Transaction Management

* 1. **Introduction**

The term transaction refers to a collection of operations that form a single logical unit of work. For instance, transfer of money from one account to another is a transaction consisting of two updates, one to each account. The transaction-management component of a database system allows application developers to focus on the implementation of individual transactions, ignoring the issues of concurrency and fault tolerance.

* 1. **Transactions**

Collections of operations that form a single logical unit of work are called transactions. A database system must ensure proper execution of transactions despite failures—either the entire transaction executes, or none of it does. Furthermore, it must manage concurrent execution of transactions in a way that avoids the introduction of inconsistency. All database reads/writes are performed as a transaction.

It begins with BEGIN TRANSACTION

It ends with COMMIT or ROLLBACK

The transaction manager is sometimes known as the TP Monitor (transaction processing monitor)

Transactions have ACID properties, which can briefly be described as:

* + Atomicity - all or nothing
  + Consistency - preserves database integrity
  + Isolation - execute as if they were run alone
  + Durability - results aren’t lost by a failure

Recovery subsystems guarantee A & D i.e. Atomicity and Durability, similarly, Concurrency control guarantees Isolation, while application programs guarantee Consistency.

* 1. **Transaction Recovery**

Recovery means to restore the database to a correct state after some failure has rendered the current state incorrect or suspect. Recovery is based on redundancy. To recover a database, the source for the recovery must be information that has been stored redundantly somewhere else. Physical redundancy is desirable, whereas, logical redundancy is not.

In large multiprocessing environments, transactions can “steal” buffer space from their predecessors, which can cause early disk writing. Similarly, the DBMS can use a “no force” policy, meaning that writing to disk is held until additional transactions complete.

* 1. **System Recovery**

The system takes checkpoints automatically. Upon system restart after a crash, transactions that finished successfully prior to the crash are redone, and those that were not complete prior to the crash are undone.

REDO and UNDO logs

ARIES: Algorithms for Recovery and Isolation Exploiting Semantics – recovery by repeating history – REDO first, then UNDO.

* 1. **Media Recovery**

Disk failure can corrupt the persistent database. The database must be restored from backup. The transaction logs can be used to roll forward from the backup point, to recover as much of the recent transaction history as possible.

* 1. **Two-Phase Commit**

Two-Phase Commit is required for distributed or heterogeneous environments, so that correctness is maintained in case of failure during a multi-part COMMIT. Prepare phase has all local resource managers force logs to a persistent log, local managers reply ok or not.

Commit phase – if all replies are ok, the coordinator commits, and orders the local managers to complete the process; otherwise all are ordered to ROLLBACK.

* 1. **SQL facilities**

START TRANSACTION

< option commalist > ;

The option commalist specifies an access point, an isolation level, or both. Access mode can be READ ONLY or READ WRITE. Isolation level sets isolation from other transactions. SAVEPOINT establishes a point within a transaction to which you can ROLLBACK