ST. XAVIER’S COLLEGE

**Maitighar,Kathmandu**

**(Affiliated to Tribhuvan University)**



**Database Management System**

**Lab Assignment #**

**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**

29th October ,2015.

**Transaction management**

1. **Introduction**

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. A transaction is a ‘logical unit of work’ on a database .Each transaction does something in the database .No part of it alone achieves anything of use or interest.

* A transaction is a logical unit of work
* It begins with BEGIN TRANSACTION
* It ends with COMMIT or ROLLBACK
* The transaction manager is sometimes known as the TP Monitor (transaction processing monitor)
* Atomicity: The manager guarantees that if any part of the transaction fails, the entire transaction will be rolled back, and the database set to its state before BEGIN

**Example of transaction**

Transfer £50 from account A to account B Read(A) A = A – 50. Write(A) Read(B) B = B+50.Write(B)

Atomicity - shouldn’t take money from A without giving it to B

Consistency - money isn’t lost or gained

Isolation - other queries shouldn’t see A or B change until completion

Durability - the money does not go back to A[1].

2.**Transaction Recovery**

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

-I.e. atomicity and durability, despite failures.  
Modifying the database without ensuring that the transaction will commit may leave the database in an inconsistent state.Consider transaction *Ti* that transfers $50 from account *A* to account *B*; goal is either to perform all database modifications made by *Ti* or none at all. Several output operations may be required for *Ti* (to output *A* and *B*). A failure may occur after one of these modifications have been made but before all of them are made. To ensure atomicity despite failures, we first output information describing the modifications to stable storage without modifying the database itself.

We study two approaches:

* + **log-based recovery**, and
  + **shadow-paging** [2].
* ROLLBACK will return the database to the previous COMMIT point
* 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

**3.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

**4.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

**5.Two phase Commit**

* 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

**6.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[3].

References :

[1]”[http://www.cs.nott.ac.uk/~psznza/G51DBS/dbs15.pdf” ,October](http://www.cs.nott.ac.uk/~psznza/G51DBS/dbs15.pdf”%20,October) 29 2015.

[2]” http://labe.felk.cvut.cz/~stepan/AE3B33OSD/Lesson12-Recovery\_Architectures.pdf”, October 29 2015

[3]” europa.nvc.cs.vt.edu/~cegyhazy/cs5614/Ch15.ppt”, October 29 2015