CS 344 *Database Management Systems*

Lecture Summary : 16th Nov., 2011

Review Session

(Transaction Management, ACID Properties, Concurrency Control)

Prepared by :

Jasvindar Singh 09010118

Vibhuti Kumar 09010157

Transaction:

Transaction, a series of actions (reads and writes) of database objects (pages, records etc.):

- To read a database object, it is first brought into main memory from disk, and then its value is copied into a program variable.
- To write a database object, an in-memory copy of the object is first modified and then written to disk.

ACID properties:

Atomicity

Users should be able to regard the execution of each transaction as atomic: either all actions are carried out or none are. Users should not have to worry about the effect of incomplete transactions.

Consistency

Each transaction, run by itself with no concurrent execution of other transaction, must preserve the consistency of the database.

Isolation

Transactions are isolated, or protected from the effects of concurrently scheduling other transactions.

Durability

Once the DBMS informs the user that a transaction has been successfully completed, its effects should persist even if the system crashes before all its changes are reflected on disk.

Schedule:

A schedule is a list of actions (read, write, abort, or commit) from a set of transactions. It represents an actual execution sequence.

T1	T2
R(A)	
W(A)	
	R(B)
	W(B)
R(C)	
W(C)	

T : Transaction
R(O) : Read object O
W(O): Write object O

Fig. : A schedule involving two transactions

Concurrent Execution of Transactions

Serializability :

A serializable schedule over a set S of committed transactions is a schedule whose effect on any consistent database instance is guaranteed to be identical to that of some complete serial schedule over S.

Anomalies situations :

Two actions on the same data object conflict it at least one of them is a write.

1. Write-read (WR) conflict : Reading uncommitted data

2. Read-write (RW) conflict : Unrepeatable reads

3. Write-write (WW) conflict : Overwriting uncommitted data

Lock-based concurrency control

Locking protocol :

A set of rules to be followed by each transaction in order to ensure that even though actions of several transactions might be interleaved, the net effect is identical to executing all transactions in some serial order.

Strict Two-Phase Locking:

Strict Two-Phase Locking or Strict 2PL has two rules.

1. If a transaction T wants to read (respectfully, modify) an object, it first requests a shared (respectfully, exclusive) lock on the object.

2. All the locks held by a transaction are released when the transaction is completed.

Some definitions

- A *recoverable schedule* is one in which transactions commit only after (and if!) all transactions whose changes they read commit.
- If transactions read only the changes of committed transactions, aborting a transaction can be accomplished without cascading the abort to other transactions. Such a schedule is said to *avoid cascading aborts*.
- Two schedules are said to be *conflict equivalent* if they involve the (same set of) actions of the same transactions and they order every pair of conflicting actions of two committed transactions in the same way.
- A schedule is *conflict serializable* if it is conflict equivalent to some serial schedule.
- A schedule is said to be *strict* if a value written by a transaction T is not read or overwritten by other transactions until T either aborts or commits.