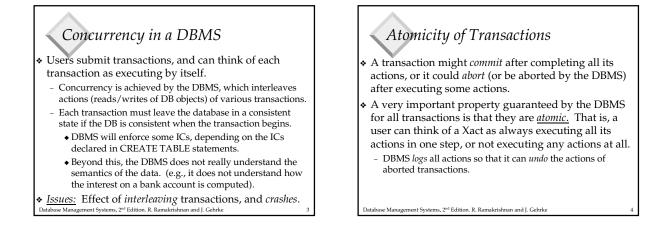
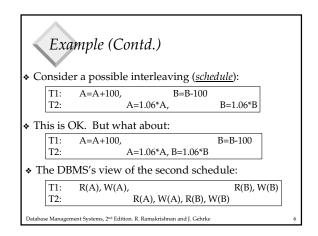




- Concurrent execution of user programs is essential for good DBMS performance.
 - Because disk accesses are frequent, and relatively slow, it is important to keep the cpu humming by working on several user programs concurrently.
- A user's program may carry out many operations on the data retrieved from the database, but the DBMS is only concerned about what data is read/written from/to the database.
- A <u>transaction</u> is the DBMS's abstract view of a user program: a sequence of reads and writes.
- Database Management Systems, 2nd Edition. R. Ramakrishnan and J. Gehrke



Consider two transactions (*Xacts*): T1: BEGIN A=A+100, B=B-100 END T2: BEGIN A=1.06*A, B=1.06*B END Intuitively, the first transaction is transferring \$100 from B's account to A's account. The second is crediting both accounts with a 6% interest payment. There is no guarantee that T1 will execute before T2 or vice-versa, if both are submitted together. However, the net effect *must* be equivalent to these two transactions running serially in some order. Database Maagement Systems, ²⁴⁴ Edition. Ramakrishnan and J. Certre

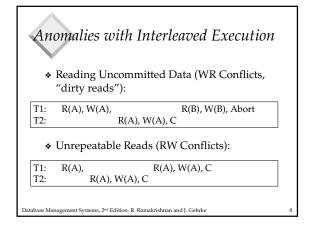


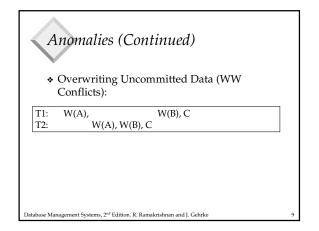
Scheduling Transactions

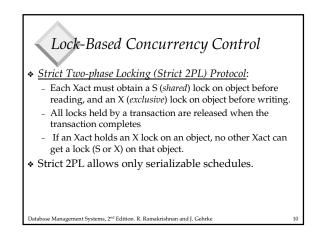
- ◆ <u>Serial schedule:</u> Schedule that does not interleave the actions of different transactions.
- ◆ <u>Equivalent schedules:</u> For any database state, the effect (on the set of objects in the database) of executing the first schedule is identical to the effect of executing the second schedule.
- <u>Serializable schedule</u>: A schedule that is equivalent to some serial execution of the transactions.

(Note: If each transaction preserves consistency, every serializable schedule preserves consistency.)

Database Management Systems, 2nd Edition. R. Ramakrishnan and J. Gehrke







Aborting a Transaction

- If a transaction *Ti* is aborted, all its actions have to be undone. Not only that, if *Tj* reads an object last written by *Ti*, *Tj* must be aborted as well!
- Most systems avoid such *cascading aborts* by releasing a transaction's locks only at commit time.
- If *Ti* writes an object, *Tj* can read this only after *Ti* commits.
 In order to *undo* the actions of an aborted transaction, the DBMS maintains a *log* in which every write is recorded. This mechanism is also used to recover from system crashes: all active Xacts at the time of the crash are aborted when the system comes back up.
 Database Management Systems, 2^{ad} Edition. Ramakrishnan and J. Cehrke 11

The Log The following actions are recorded in the log: Ti writes an object: the old value and the new value. Log record must go to disk <u>before</u> the changed page! Ti commits/aborts: a log record indicating this action. Log records are chained together by Xact id, so it's easy to undo a specific Xact. Log is often duplexed and archived on stable storage.

 All log related activities (and in fact, all CC related activities such as lock/unlock, dealing with deadlocks etc.) are handled transparently by the DBMS.

12

Database Management Systems, 2nd Edition. R. Ramakrishnan and J. Gehrke

Recovering From a Crash

- * There are 3 phases in the *Aries* recovery algorithm:
 - <u>Analysis</u>: Scan the log forward (from the most recent checkpoint) to identify all Xacts that were active, and all dirty pages in the buffer pool at the time of the crash.
 - <u>Redoes</u> Redoes all updates to dirty pages in the buffer pool, as needed, to ensure that all logged updates are in fact carried out and written to disk.
 - <u>Undo</u>: The writes of all Xacts that were active at the crash are undone (by restoring the *before value* of the update, which is in the log record for the update), working backwards in the log. (Some care must be taken to handle the case of a crash occurring during the recovery process!)

13

Database Management Systems, 2nd Edition. R. Ramakrishnan and J. Gehrke



- Concurrency control and recovery are among the most important functions provided by a DBMS.
- Users need not worry about concurrency.
 - System automatically inserts lock/unlock requests and schedules actions of different Xacts in such a way as to ensure that the resulting execution is equivalent to executing the Xacts one after the other in some order.
- Write-ahead logging (WAL) is used to undo the actions of aborted transactions and to restore the system to a consistent state after a crash.
- Consistent state: Only the effects of commited Xacts seen. Database Management Systems, 2nd Edition. R. Ramakrishnan and J. Gehrke