Users and DB Programs

• End users don’t see the DB directly
  – are only vaguely aware of its design
  – may be acutely aware of part of its contents
• End users interact with DB programs
  – In "Centralized" model, all of DB and all of program is on the mainframe
    • user has "dumb terminal"
  – In "Client/Server" model, UI is local (PC) and DB is remote (mainframe)
  – In "distributed" model, UI is local, some of the DB is local, and some is elsewhere

Definition of "Transaction"

Definition: A transaction is the execution of a DB program.
• Transactions must be "atomic"
  – Their affect is all or none
  – DB must be consistent before and after the transaction executes (not necessarily during!)
  – EITHER a transaction executes fully and "commits" to all the changes it makes to the DB OR it must be as though that transaction never executed at all

A Typical Transaction

• Transfer money from savings to checking
  – Read savings; verify balance is adequate *, update savings balance and rewrite **; read checking; update checking balance and rewrite ***.
  * DB still consistent
  ** DB inconsistent
  *** DB consistent again

"Commit" and "Abort"

• A transactions which only READs expects DB to be consistent, and cannot cause it to become otherwise.
• When a transaction which does any WRITE finishes, it must either
  – COMMIT: “I'm done and the DB is consistent again” OR
  – ABORT: “I'm done but I goofed: my changes must be undone.”

Complications

• A DB may have many simultaneous users
  – explains why mainframes are still important
  – simultaneous users implies simultaneous transactions implies simultaneous DB access
    • multiprogramming/multiprocessing
• Things can go wrong
  – transactions can conflict with one another
  – programs may crash, OS may crash, disk may crash
But DB Mustn’t Crash

- Can’t be allowed to become inconsistent
  – A DB that’s 1% inaccurate is 100% unusable.
- Can’t lose data
- Can’t become unavailable

* A matter of life or death! *

Transaction Manager

- May be part of OS, a layer of middleware, or part of the DBMS
- Starts transactions
  – ensure timely, fair scheduling
- Logs their activities
  – especially start/stop, writes, commits, aborts
- Detects or avoids conflicts
- Takes recovery actions

The Log File

- Transaction starts/stops
- DB writes: "before" and "after" images
  – befores can be used to rollback an aborted transaction
  – afters can be used to redo a transaction
    (recovery from catastrophe)
- COMMITs and ABORTs

* The log itself is as critical as the DB! *

The Big TP Issues

- Concurrency Control
  – Making sure simultaneous transactions don’t interfere with one another
- Recovery
  – Taking action to restore the DB to a consistent state

The ACID Test

- Atomicity
- Consistency Preservation
- Isolation
- Durability