



IBM Systems Group

Transaction Manager Concepts

Dr. Nick Bowen,
VP UNIX and xSeries SW Development
October 17, 2003

Yale | Oct 2003

© 2003 IBM Corporation

IBM Systems Group

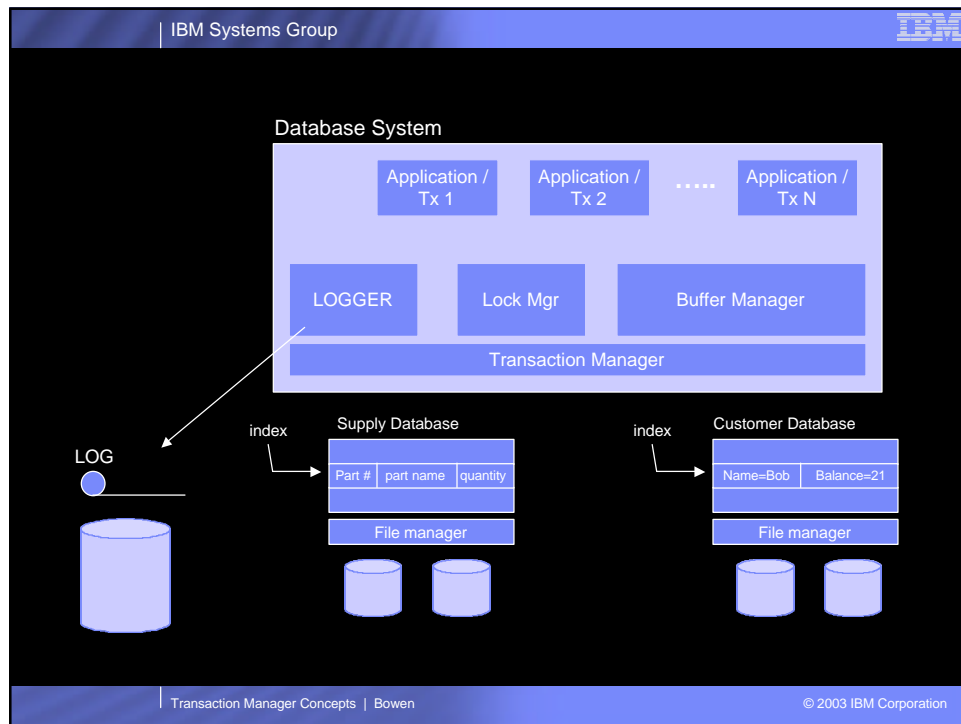


Agenda

- Database Concepts
- Transaction Manager Overview
- Logging, REDO and UNDO
- Two Phase Commit
- Distributed Transaction Processing

Transaction Manager Concepts | Bowen

© 2003 IBM Corporation



IBM Systems Group

ACID

Atomicity	<p>"All or nothing"</p> <p>undo aborted</p> <p>redo committed</p> <p>coordinate with distributed</p> <p>"A correct transformation of the source"</p>
Consistency	<p>Aborting any transactions that fail to pass R.M. consistency tests</p> <p>"Once done, it is done"</p>
Durability	<p>Forcing all log records of committed transactions to durable memory</p> <p>Redoing any recently committed work at restart</p>
Isolation	Individual transactions do not bump into one another

Transaction Manager Concepts | Bowen

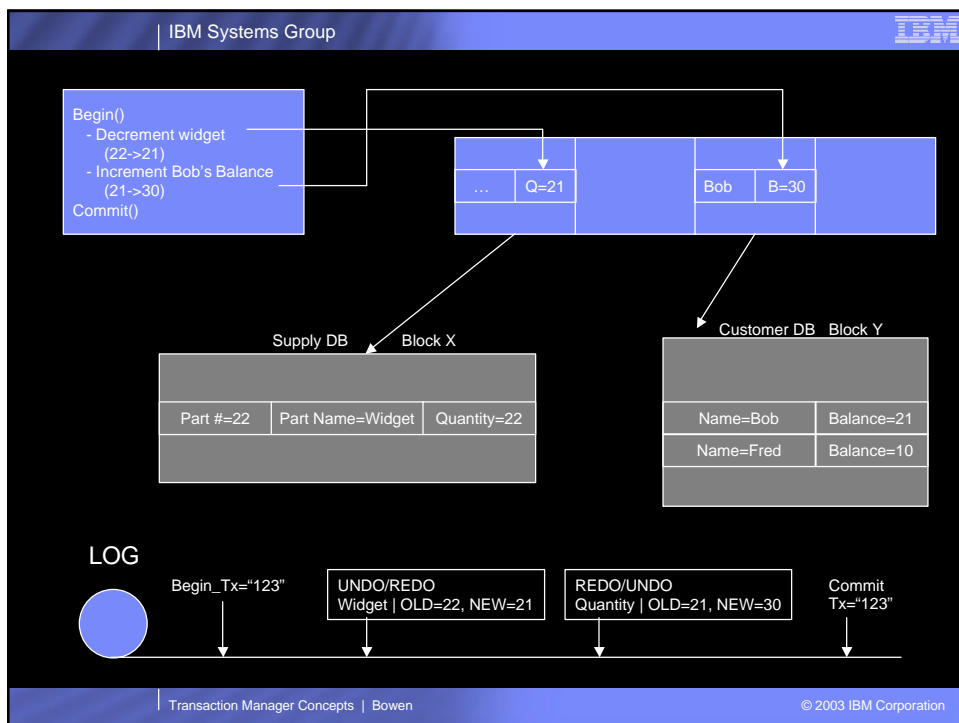
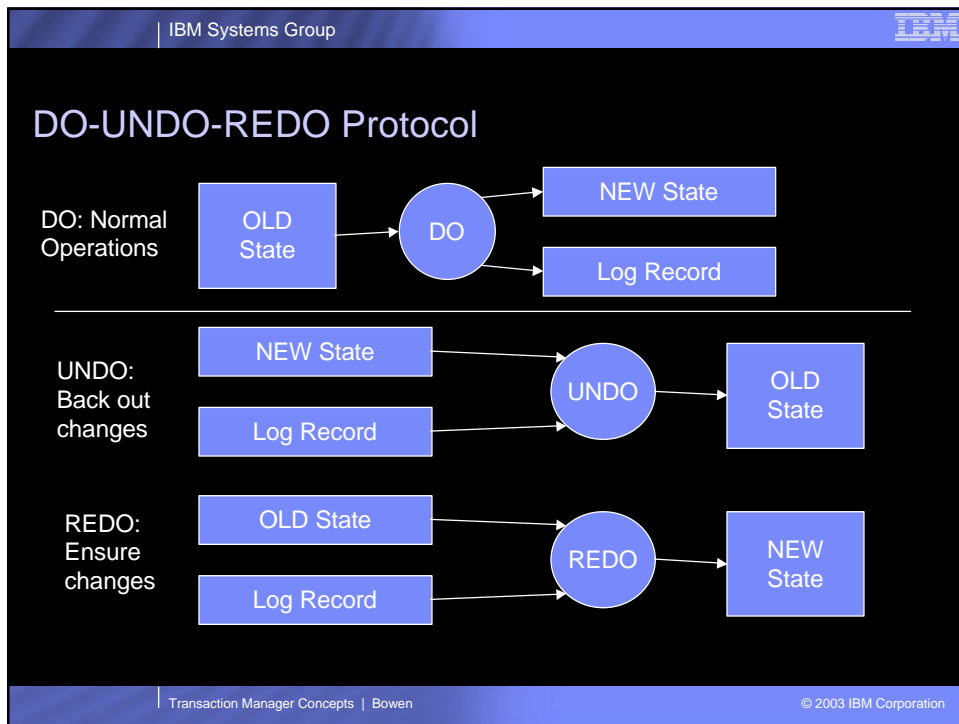
© 2003 IBM Corporation

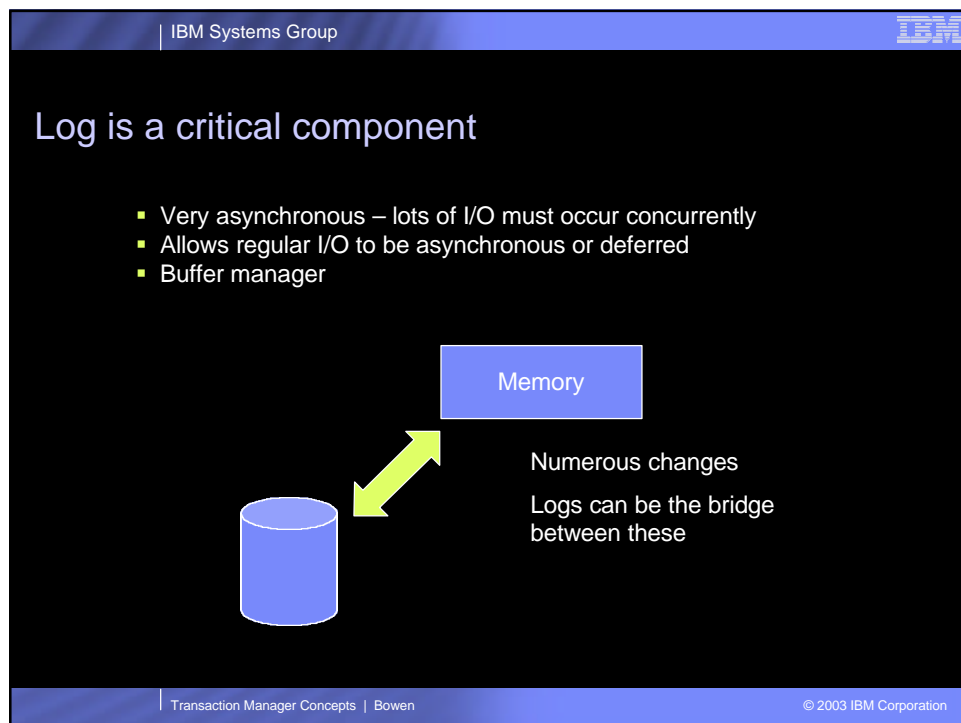
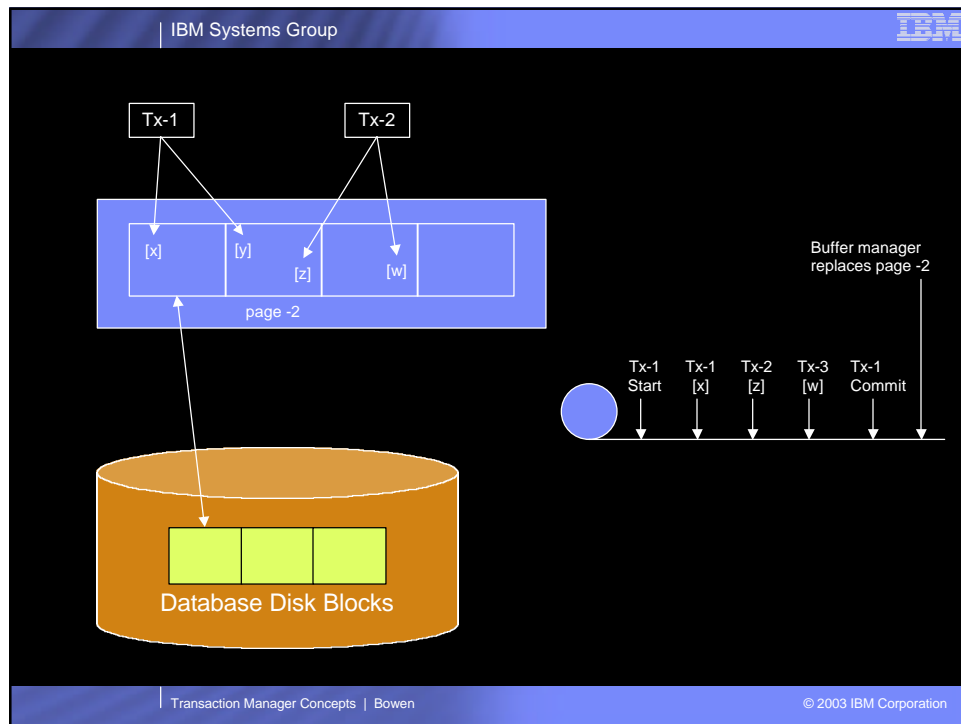
DB101

- Application: DBM – eCommerce
 - Supply DB (inventory) – decrement
 - Billing - create bill
 - Shipping - create order for factory for ship
- Failure reasons (3)
 - Normally just overhead – it all works
 - Programming style error, e.g. abort changes
 - System crashes, credit card authorized failure, DB crashes

DO-UNDO-REDO Protocol

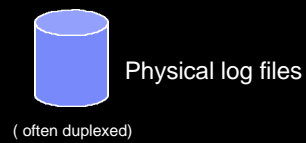
- Programming style for resource managers
- Structure each operation (“DO”) so it can be undone or redone
- Any operation should perform
 - The actual operation (“DO”)
 - A log record
 - An “UNDO” program
 - A “REDO” program



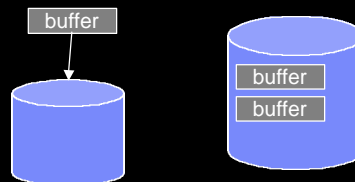


Logs (continued)

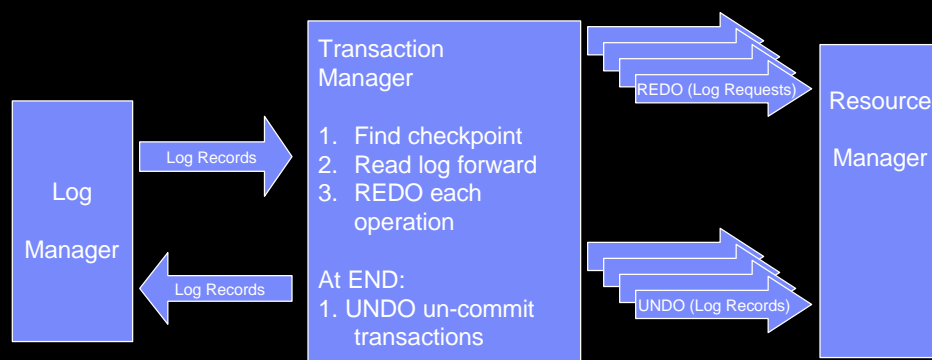
- Log force at every Tx would be a serial bottleneck
- If lots of Tx's are concurrently committing (approximately) a log force can handle lots of Tx's

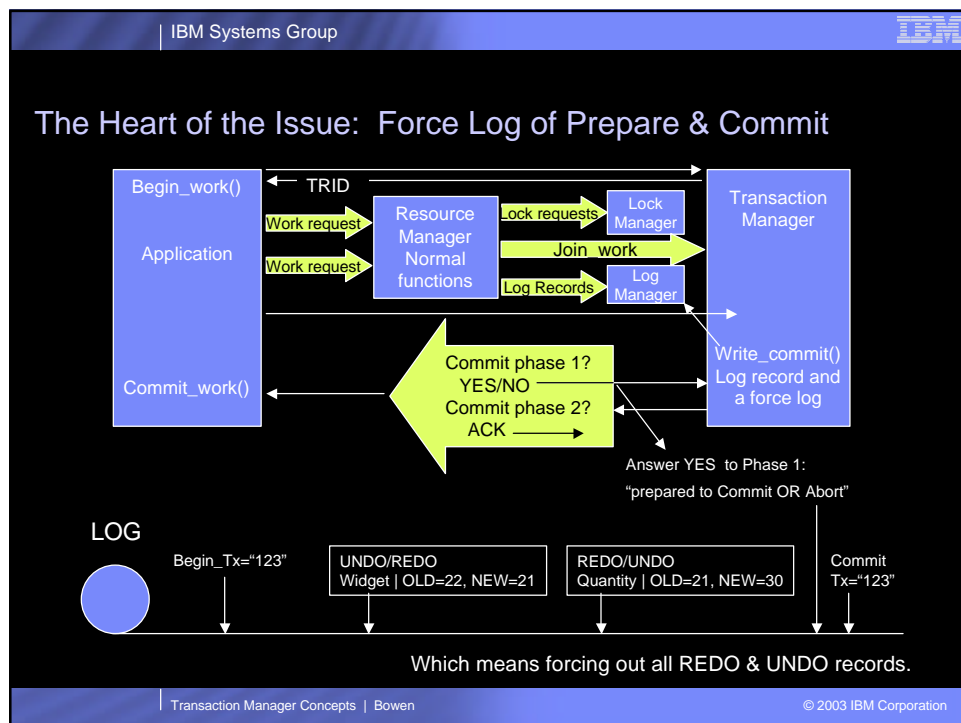
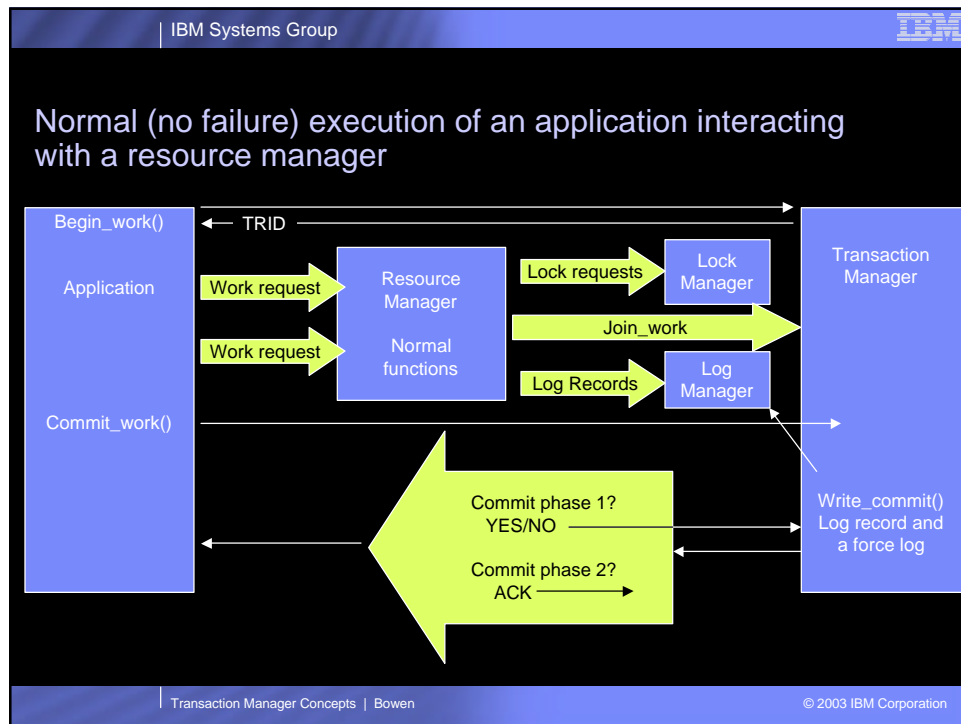


- WADS concept



Data Flow in a System Restart





Two Phase Commit

- We have shown two phase for one system, one RM
- Can be extended:
 - Multiple RM's on a single system
 - Multiple RM's on distributed systems

Two phase commit: Making computations atomic

- Tx Manager asks RM's to commit
 - Can be multiple ones
 - Can be on different systems
- Each Vote YES or NO
 - Vote YES means "Prepared to commit or Abort"
- Tx Manager proceeds to commit if all YES
 - Forces "End Phase 1" log record
 - This is the "atomic instance"

For example...

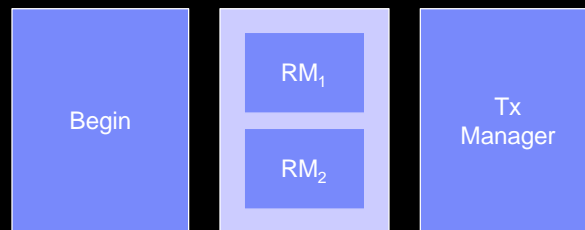
- Application wants to commit Tx TRID=555
 1. Application must begin a Tx
 2. RM's must tell Tx Manager they want to join a transaction.
 3. Major Stages:

Phase 1	Prepare	Ask each RM to vote: YES/NO
	Decide	Tx mgr logs a commit record (if all yes) to durable storage. This is a precise commit point, when the bits hit the disk.
Phase 2	Commit	Tx tells RM's about decision
	Complete	Once all RM's "ACK" commit, write a completion record

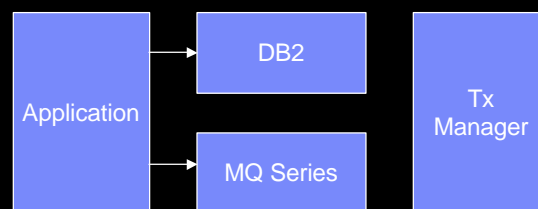
Restart Processing

- Transaction manager – finds all
 - Begin_work with no commit record => Transactions to abort
 - Begin_work with commit => RM's must be told with no complete about Tx's
 - Begin_work + commit + complete => Ignore
- RM Restart
 - Must find work either process an abort (UNDO) or ensure everything is done (REDO)

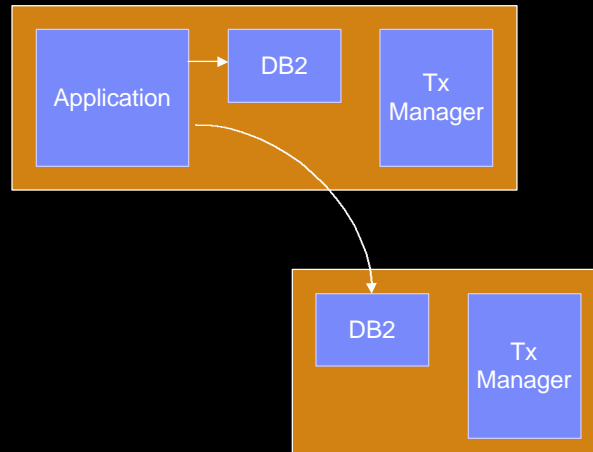
Why is this complicated?



...between two products

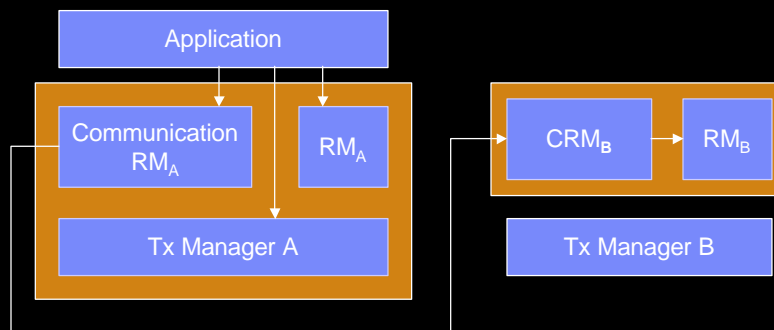


...between distributed systems

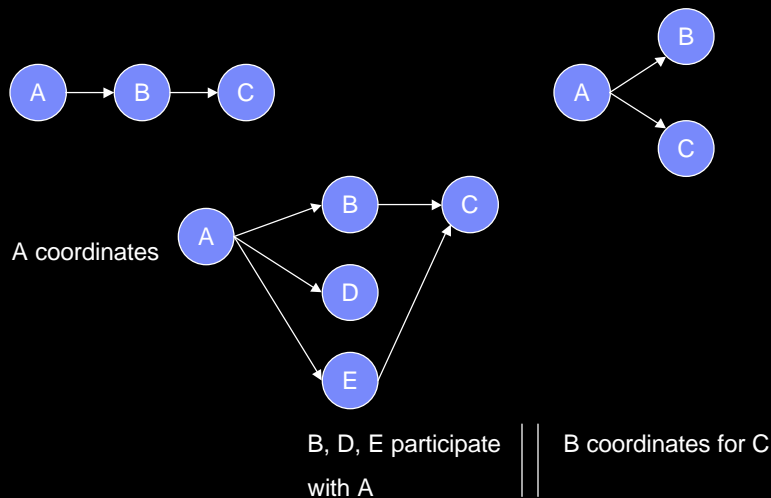


Distributed Transactions

- Root Tx's manager – where application performs original begin_work



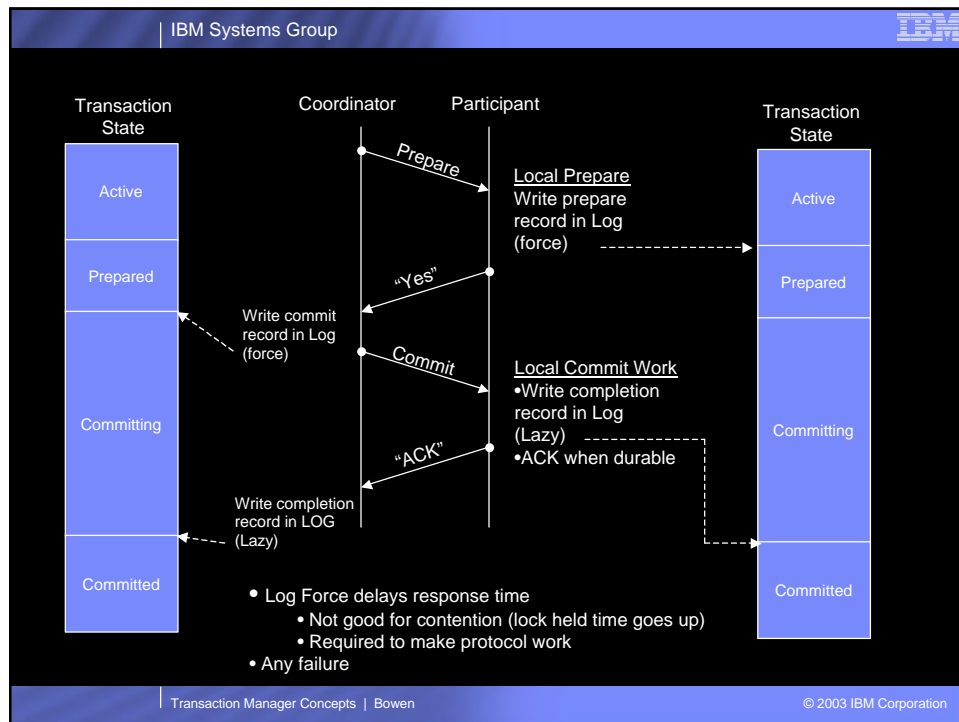
Can get complicated...



Distributed Two Phase

O_1	Local prepare – call local RM's Distributed prepare – send prepares on each outgoing session
<u>Commit!!!</u>	Decide – all local RM's vote YES & outgoing sessions vote YES
O_2	Commit – tell RM's & outgoing session Complete – same

- Prepare now has a time-out
- Commit record has names of distributed RM's who need to know about phase 2 commit
- "In doubt" of CRM – voted YES, but does not know. Must poll upstream Tx Mgr
- Presumed abort protocol (optimization: coordinator is queried on Tx that is not active, prepared, or committed, then assume aborted)
 - presumption that no record found == abort
 - participant sends ACK only when the completion state is durable
 - coordinator only sends records the completion log when all commit messages have been acknowledged



IBM Systems Group

Putting it All Together

- Basic transaction & database concepts
- Normal commit processing
- Importance of the LOG
- Two Phase Commit
- Extending for Distributed Processing
- Further Reading "Transaction Processing: Concepts and Techniques," Jim Gray and Andreas Reuter.
 - Chapter 10 "Transaction Manager Concepts" pages 529-582
 - Note - - 1000 other pages!!!!

Transaction Manager Concepts | Bowen

© 2003 IBM Corporation