

Overview of Transaction Management

12 January 2010
Lecture 13

January 12, 2010

ISE 322: Database Systems

3

Our Topics Today

- Two important tasks the database does:
 - **Concurrency Control**: Let multiple users use the database at once
 - **Crash Control**: Let transactions/database fail gracefully
- Failures can be:
 - Transactions that don't complete
 - Loss of power
 - File corruption
 - Disk failure
- Fundamental concept: A **transaction**

January 12, 2010

ISE 322: Database Systems

2

Outline

- ACID Properties
- Transactions and Schedules
- Concurrent Execution of Transactions
- Source: R&G Chapter 16.1-16.3.2

January 12, 2010

ISE 322: Database Systems

3

What is a transaction?

- A **transaction** is **one execution of a user program on a DBMS**
 - Multiple program executions → multiple transactions
 - Multiple users → multiple transactions
 - To the database, a transaction is a series of reads/writes
- If it finishes: **COMMIT**. Otherwise it fails: **ABORT**
- Almost all DBMS' **interleave** multiple transactions
 - So they must provide **guarantees** about behavior (concurrency)
 - They must protect the transactions from each other (concurrency)
 - If one transaction fails, the others must be protected (crash)
 - **Why interleave?**
 - To improve database transaction throughput
 - Use latencies that arise during execution of individual transactions

January 12, 2010

ISE 322: Database Systems

4

ACID

- Four properties:
 - **Atomicity**
 - **Consistency**
 - **Isolation**
 - **Durability**
- Define how DBMS' offer **concurrency** and **crash** control
- The **log** is essential here (we'll talk more about it later)

January 12, 2010

ISE 322: Database Systems

5



"As always, should you or any of your IM force be caught or killed, the Secretary will disavow any knowledge of your actions.

"Good luck, Jim. This tape will self-destruct in five seconds."

January 12, 2010

ISE 322: Database Systems

6

Atomicity

- Transactions are either executed entirely or not at all
 - Some transaction cannot complete due to:
 - Internal failure, Power failure
 - Crash, Unexpected property discovered
- Such transactions **abort**

The database's job: ensure aborted transactions "never occurred"

- Means undoing everything they did (use the log)
- Nobody else should have used data written by it (?)

Otherwise, an aborted transaction may leave the database in an incorrect state

January 12, 2010

ISE 322: Database Systems

7

Consistency

"Be kind, please rewind" – Blockbuster video

Consistency is a high level database property

- Data has some "correct" or "logical" state which ought to be preserved
 - Ex. All employees are in the Employees table
 - Ex. The salaries in the Employees table are correct
- The database **can't** enforce them – it's the user's job

- Consistency is a **contract**:
 - The database promises that transactions will start with a consistent database
 - Each transaction must promise that if it receives a consistent database, then it will return a consistent database at completion
 - What if the transaction doesn't complete?
 - How can transactions promise this?

January 12, 2010

ISE 322: Database Systems

8

Isolation

"Have you ever been alone in a crowded room" – Jack's Mannequin

Isolation means that even though many transactions may be going on at once:

- The transactions **can't tell** (think time sharing systems)
- Result of the many concurrent transactions is **equivalent to some serial execution** of them all (ideally)

- Example: T1 and T2 execute at the same time.
 - Both complete **successfully**
 - T1 **couldn't tell** that T2 ran at the same time or vice versa
 - After they complete, to an outsider **it looks like one ran after the other** (no guarantee if it's T1;T2 or T2;T1)

January 12, 2010

ISE 322: Database Systems

9

Durability

Durability means that if a transaction commits its effects are **never lost**



Conversely, an aborted transaction is eventually forgotten

- The log is used for this
 - Must be safe from crashes
 - Must reflect every committed transaction (at least)

January 12, 2010

ISE 322: Database Systems

10

ACID Together

Putting ACID together, we get:

- Transactions are atomic collections of reads/writes
- They run in simulated isolation
- If one finishes (commits), its changes are never lost
- If one doesn't finish (aborts), an outsider couldn't tell it ever ran
- To an outsider, it appears that all successful transactions could have run serially
- Transactions always receive a consistent database (as per the contract)

January 12, 2010

ISE 322: Database Systems

11

So far

- ACID Properties
- Transactions and Schedules
- Concurrent Execution of Transactions

January 12, 2010

ISE 322: Database Systems

12

Schedules and Notation

- We may show the running of several transactions in a "schedule"
 - Time is shown from top to bottom
 - Each transaction is shown in a separate column
 - A **complete schedule** shows commits/aborts for all transactions
 - A **serial schedule** has no interleaving
- If T reads O we write $R_T(O)$
- If T writes O we write $W_T(O)$
- A completed transaction sends **commit**: $Commit_T$
- If not **abort**: $Abort_T$
- We will drop the subscript if it's obvious

January 12, 2010

ISE 322: Database Systems

13

Schedule Model

Two assumptions

- Transactions communicate only via database
 - Important for isolation and durability (why?)
 - We can't help if two transactions communicate in an out of band manner (why would we?)
- A database is a fixed collection of independent object
 - Addition and deletion complicate things
 - We'll relax this later
 - Dependency also complicates (locks)
- What's an object?

January 12, 2010

ISE 322: Database Systems

14

A Simple Schedule

T_1	T_2
$R(A)$	
$W(A)$	
	$R(B)$
	$W(B)$
$R(C)$	
$W(C)$	

January 12, 2010

ISE 322: Database Systems

15

Serializability

- Essential concept: **Serializability**
 - Schedule S1 is **serializable** if we can find some (different) schedule on the transactions S2 which is **serial** and is the **same** S1
 - Serializability is a property of *schedules*

What does "the same" mean?

- For now – identical outcomes (*view serializability*):
 - The order of actions in each transaction is identical
 - If T reads value $X=x_i$ in S1, it reads $X=x_i$ in S2 (and not some other value $X=x_j$)
 - If at the end of S1, $X = x_i$, then $X = x_i$ at the end of S2

January 12, 2010

ISE 322: Database Systems

16

Example: Interleaving 1

Example: The schedule below is serializable:

T_1	T_2
$R(A)$	
$W(A)$	
	$R(A)$
	$W(A)$
$R(B)$	
$W(B)$	
	$R(B)$
	$W(B)$
Commit	Commit

It is equivalent to running $T_1;T_2$.

January 12, 2010

ISE 322: Database Systems

17

Example: Interleaving 2

Example: The following schedule is also serializable, but in a different way than the previous one:

T_1	T_2
	$R(A)$
	$W(A)$
$R(A)$	
	$R(B)$
	$W(B)$
$W(A)$	
$R(B)$	
$W(B)$	
Commit	Commit

It is equivalent to running $T_2;T_1$. □

January 12, 2010

ISE 322: Database Systems

18

In summary

- ACID Properties
- Transactions and Schedules
- Concurrent Execution of Transactions