

Concurrency Control

Chapter 17

3/31/09 EECS 484: Database Management Systems, Kristen LeFevre 1

Agenda

- Last time: transactions, schedules, lock basics
- How can we use locking to guarantee schedules with desirable properties?
 - Serializable, recoverable, avoid cascading aborts (ACA)
- Lock Manager Implementation
- Deadlock
 - Definition, detection, and prevention

3/31/09 EECS 484: Database Management Systems, Kristen LeFevre 2

Two-Phase Locking (2PL)

- **2PL:**
 - If T wants to read (modify) an object, first obtains a shared (exclusive) lock
 - If T releases any lock, it can acquire *no new locks!*
 - Guarantees serializability!
- **Strict 2PL:**
 - Hold all locks until end of Xact
 - Guarantees serializability and ACA too!
 - Note ACA schedules are always recoverable

3/31/09 EECS 484: Database Management Systems, Kristen LeFevre 3

Example

- Two accounts, A and B
- Two transactions, T1 and T2
- T1: Transfer \$100 from A to B
- T2: Add 10% interest to A and B

3/31/09

EECS 484: Database Management Systems, Kristen LeFevre

4

Example - Strict 2PL

- T1 acquires S lock on A
- T1 acquires X lock on A
- T1 acquires S lock on B
- T1 acquires X lock on B
- T1 releases all locks
- T2 acquires S lock on A

...

T1: Transfer \$100 from A to B	T2: Add 10% interest to A & B
begin	
	begin
R(A); A -= 100	
W(A)	
R(B); B += 100	Wait
W(B)	
commit	
	R(A); A *= 1.1
	W(A)
	R(B); B *= 1.1
	W(B)
	commit

3/31/09

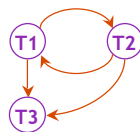
EECS 484: Database Management Systems, Kristen LeFevre

5

Precedence Graph

- Precedence (or serializability) graph for schedule S
 - A node for each committed transaction in S
 - An arc from T_i to T_j if some action in T_i precedes and conflicts with some action in T_j

T1	T2	T3
R(A)		
	W(A)	
	Commit	
W(A)		
Commit		
		W(A)
		Commit



3/31/09

EECS 484: Database Management Systems, Kristen LeFevre

6

Terminology

- Two schedules are *conflict equivalent* if
 - involve the same transactions
 - order each pair of conflicting transactions in the same way
- A schedule is *conflict serializable* if it is conflict equivalent to a serial schedule
- All conflict serializable schedules are also serializable

3/31/09

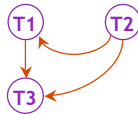
EECS 484: Database Management Systems, Kristen LeFevre

7

Conflict Serializability & Graphs

- Theorem:** A schedule is conflict serializable if and only if its precedence graph is acyclic
 - Equivalent serial schedule is given by any topological sort over graph

T1	T2	T3
	W(A)	
	Commit	
R(A)		
W(A)		
Commit		
		W(A)
		Commit



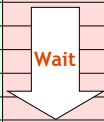
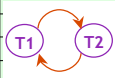
3/31/09

EECS 484: Database Management Systems, Kristen LeFevre

8

Example

T1: Transfer \$100 from A to B	T2: Add 10% interest to A & B	T1: Transfer \$100 from A to B	T2: Add 10% interest to A & B
begin		begin	
R(A); A -= 100		R(A); A -= 100	
W(A)		W(A)	
	R(A); A *= 1.1		
	W(A)		
	R(B); B *= 1.1		
	W(B)		
	commit		
R(B); B += 100			
W(B)			
commit			



3/31/09

EECS 484: Database Management Systems, Kristen LeFevre

9

2PL & Serializability

- 2PL guarantees acyclic precedence graph
 - Guarantees a conflict serializable schedule
 - Intuitively, equivalent serial schedule given by order in which transactions enter shrinking phase
- Why Strict 2PL?
 - Guarantees ACA (read only committed values)
 - How? Hold X locks until end of transaction; no WW or WR conflicts

3/31/09

EECS 484: Database Management Systems, Kristen LeFevre

10

Exercise

- Is this schedule allowed by 2PL? Strict 2PL?
- Is it serializable?
 - If so, what is the corresponding serial schedule?
- Is the schedule recoverable? ACA?

T1: Transfer \$100 from A to B	T2: Add 10% interest to A, B & C
begin	
	begin
	R(C); C *= 1.1
	W(C)
R(A); A -= 100	
W(A)	
R(B); B += 100	
W(B)	
commit	
	R(A); A *= 1.1
	W(A)
	R(B); B *= 1.1
	W(B)
	commit

3/31/09

EECS 484: Database Management Systems, Kristen LeFevre

11

Lock Manager Implementation

- Main data structure: Lock Table
 - Hash table with Object ID (OID) as key
 - For each object:
 - List of transactions holding the lock
 - Lock Mode (Shared or Exclusive)
 - Pointer to queue of waiting requests
 - On lock release (OID, XID)
 - Update list of transactions
 - Grant request to first transaction of queue
 - (Optionally, prioritize upgrades)

3/31/09

EECS 484: Database Management Systems, Kristen LeFevre

12

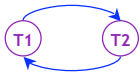
What about deadlocks?

T1 gets S, then X lock on A

T2 gets S, then X lock on B

T1 waits for lock on B

T2 waits for lock on A



“Waits-for” Graph

T1: Transfer \$100 from A to B	T2: Add 10% interest to A & B
begin	
	begin
R(A); A -= 100	
W(A)	
	R(B); B *= 1.1
	W(B)
R(B); B += 100	
W(B)	
commit	
	R(A); A *= 1.1
	W(A)
	commit

3/31/09

EECS 484: Database Management Systems, Kristen LeFevre

13

Deadlock Detection

- Observation:
 - Deadlocks are rare
 - Often involve only a few transactions
 - Detect rather than prevent
- Lock Mgr maintains waits-for graph
- Periodically check graph for cycles. Abort **some** Xact to break the cycle.
- Simpler hack: *time-outs*. Abort if no progress made for a while.

3/31/09

EECS 484: Database Management Systems, Kristen LeFevre

14

Deadlock Prevention

- Assign priorities to transactions
 - Use timestamp to assign priorities
- Ti requests a lock, held by Tj (in a conflicting mode)
 - **Wait-Die**: If Ti has higher priority, it waits; else Ti aborts
 - **Wound-Wait**: If Ti has higher priority, abort Tj; else Ti waits
 - After abort, restart with original timestamp
 - Guarantees the transaction eventually runs!

3/31/09

EECS 484: Database Management Systems, Kristen LeFevre

15

Non-Locking CC Protocols

- Locking most common technique for guaranteeing transaction serializability
 - Used in most commercial systems
- Other approaches exist (beyond scope of class):
 - Optimistic CC
 - In locking protocols, avoid conflict by blocking (waiting)
 - In optimistic CC, instead undo transactions if there has been a conflict
 - Anticipates that conflicts rarely occur
 - Multiversion CC
 - Make sure transactions never have to wait to read an object
 - Maintain multiple versions of each object, each with a timestamp
 - Used by Oracle

3/31/09

EECS 484: Database Management Systems, Kristen LeFevre

16

Summary

- Locking commonly used by DBMS for concurrency control
- 2PL guarantees a serializable schedule
- Strict 2PL guarantees a serializable, recoverable, ACA schedule
- Lock manager handles lock requests from transactions
- Deadlock rare, but must be detected or prevented

3/31/09

EECS 484: Database Management Systems, Kristen LeFevre

17

Announcements

- Book Exercises: 16.1, 16.3, 17.5
- Additional Review Exercise: Recall the three kinds of conflicts from last class (RW, WR, WW). For each, think of an example where the conflict occurs. How does 2PL prevent these conflicts?

3/31/09

EECS 484: Database Management Systems, Kristen LeFevre

18
