MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

slides online at http://web.mit.edu/6.033/www/assignments/lec18.[ppt|pdf]

Isolation

6.033 Spring 2010

Lecture 18

Sam Madden

Key concepts:
Serial equivalence
Two-phase locking
Deadlock detection

Recover with cell-storage, log, cache, optimized cell-storage updates

```
recover(log):
    doneset = { }
    for each record r in log[len-1] ... log[0]:  //UNDO
        if r.type == commit
            doneset = doneset U r.TID
        if r.type == update and r.TID not in doneset:
            write(cell-storage, r.var, r.before)

    for each record r in log[0]...log[len-1]:  //REDO
        if r.type == update and r.TID in doneset:
          if (cell-storage does not reflect r)
            write(cell-storage, r.var, r.after)
```

Conflicting Operations

Given two transactions T1 and T2, and two operations o1 in T1, o2 in T2

o1 *conflicts* with o2 if either is a write and both are to the same object

E.g. T1 RA, T2 WA or T1 WB, T2 WB

Testing for Serial Equivalence (Conflict Serializability)

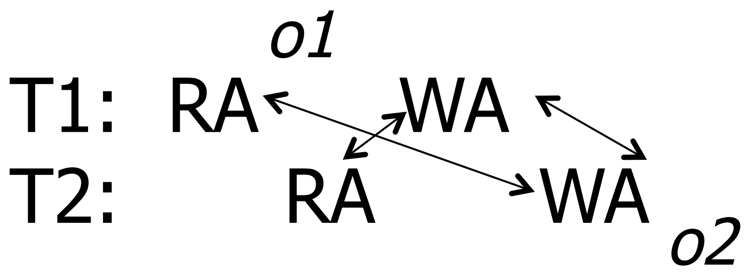
A schedule is *serial equivalent* if

- for all pairs of transactions T1 and T2,
- all conflicting pairs of ops o1 in T1 and o2 in T2 are ordered the same way

E.g., o1 always before o2 or

o2 always before o1

Not serial equivalent!



T1 precedes T2
T1 precedes T2
T2 precedes T1

Locking Protocol

Before reading/writing an object, get lock on it

(If lock isn't available, block)

When to release locks?

Locking Protocol w/ Release

Before reading/writing an object, system acquires lock on it

(If lock isn't available, block)

Release locks **after** transaction commit

Two Phase Locking (Allows locks to be released before end of transaction)

Phase 1 – system acquires lock before reading or writing an object, up to lock point

Phase 2 – releases locks on objects, after done with them and after lock point

(Never acquire locks in phase 2)

Two-phase locking with shared and exclusive locks

Phase 1:

Before reading an object, system acquires an S lock on it

Blocks if any other xaction has X lock on object

Before writing an object, system acquires an X lock on it

Blocks if any other xaction has X or S lock on object

Phase 2: Release locks on objects, after done with them and after lock point

Transaction Schedule → Log

Schedule | Log

T1	T2	BEGIN T1
lock A		BEGIN T2
RA		
	lock A (block)	
WA		UPDATE T1,A
lock B <--- lock point		
release A		
	RA	
	WA	UPDATE T2, A
	lock B (block)	
RB		
WB		UPDATE T1, B
release B		
	RB	UPDATE T2, B
	WB	COMMIT T1; COMMIT T2

Avoiding cascading aborts

Phase 1:

Before reading an object, system acquires an S lock on it

- Blocks if any other xaction has X lock on object

Before writing an object, system acquires an X lock on it

- Blocks if any other xaction has X or S lock on object

Phase 2: Release S locks on objects anytime after done with them and after lock point

Only release X locks after end of transaction