CIS 451/551

week 8: Transaction Processing
transaction requirements

• a database transaction is a unit of work
• starts with an SQL BEGIN statement, or START TRANSACTION
• finishes with a COMMIT (or ABORT or ROLLBACK)

• expect to be compliant with ACID properties:
  • Atomicity
  • Consistency
  • Isolation
  • Durability
ACID

• Atomicity
  • a xact is a single unit of work, must do all or none of it
  • if ABORT then must UNDO work

• Consistency
  • assumption that if db is in consistent state, then xact will leave db in consistent state

• Isolation
  • xacts should not interfere with one another
  • need to introduce locks (usually)

• Durability
  • once xact is finished (COMMIT), its changes are permanent
  • may need to REDO upon system crash
isolation

- a schedule is an ordering of the execution of its xacts
- in an ideal world, we would have only a serial schedule
  - one xact finishes before another starts
  - T1; T2; T3
  - no interference guaranteed
  - very slow
- instead we aim for a serializable schedule
  - equivalent to a serial schedule
  - different types of equivalence
sample transactions

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R(X)</td>
<td>R(X)</td>
</tr>
<tr>
<td>X=X-N</td>
<td>X=X+M</td>
</tr>
<tr>
<td>W(X)</td>
<td>W(X)</td>
</tr>
<tr>
<td>R(Y)</td>
<td></td>
</tr>
<tr>
<td>Y=Y+N</td>
<td></td>
</tr>
<tr>
<td>W(Y)</td>
<td></td>
</tr>
</tbody>
</table>

T1 and T2 are sample xacts. Let’s look at some schedules, using values X=Y=100, N=10, and M=5.

Note that in both serial schedules T1;T2 and T2;T1, we end up with X=95 and Y=110.
problem with concurrent execution

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R(X)</td>
<td>R(X)</td>
</tr>
<tr>
<td>X=X-N</td>
<td>X=X+M</td>
</tr>
<tr>
<td>W(X)</td>
<td></td>
</tr>
<tr>
<td>R(Y)</td>
<td>W(X)</td>
</tr>
<tr>
<td>Y=Y+N</td>
<td></td>
</tr>
<tr>
<td>W(Y)</td>
<td></td>
</tr>
</tbody>
</table>

- T1 and T2 handle their work in local memory
- End result is X=105, Y=110
- Not consistent
- Example of the *lost update problem*
dirty read problem

### Problems
- if T1 aborts, then T1 is rolled back, forcing us to rollback T2 as well
  - this is called *cascading rollback*
- if T2 had committed, we would violate the durability property

---

**Definition**

A **recoverable schedule** is one where a xact commits only after all xacts whose changes they read also commit.
locking

• we apply locks to data items
• lock granularity
  • database
  • table
  • page
  • row
  • column
• lock uses
  • read lock (shared)
  • write lock (exclusive)
• requires dbms have a lock manager
two-phase locking

• lock no good if not held long enough
• two phase locking used
  1. growing phase: all locks acquired
  2. shrinking phase: locks may be released
• no locks can be acquired once any lock is released

• theorem any schedule that uses two-phase locking is conflict serializable
conflict serializable

• we say two schedules are conflict equivalent \( S_1 \approx_c S_2 \) if they involve the same set of actions on the same xacts and they order every pair of conflicting actions in the same way

• a schedule is conflict serializable if it is conflict equivalent to some serial schedule

• this can be tested in poly time

• view equivalence is more general (and less restrictive)

• testing for view serializability is NP-complete
test for conflict serializability

**algorithm:**

1. build a conflict graph
   - look at each pair of xacts Ti and Tj
   - draw an edge from Ti to Tj if
     - both xacts act (R/W) on the same data item X
     - one action is a W
     - Ti does it before Tj

2. test to see whether the graph G has a cycle
   - if cycle, “NO”
   - else “YES”
cascading rollback

- using 2PL does not avoid the dirty read problem
- introduce **strict 2PL**
- most commercial DBMS follow strict 2PL

- results in strict schedule
- strict implies ACR (avoid cascading rollback) implies recoverable
schedule properties
deadlock

• all of these protocols may induce deadlock
• optimistic protocol
  • hope it won’t happen
  • test for it, if it happens kill xact
• pessimistic protocol
  • follow a set of rules that avoid it
• conservative 2PL
  • all locks must be acquired at start of xact
  • an atomic request, all granted or all refused
  • deadlock free
• could now follow conservative and strict 2PL
  • too slow!!