Transaction Systems
Exercise Session 8

Andrey Gubichev

December 23, 2013
Task 1

- Consider a database with a Person table (unique Name, City attribute). Two operations:
  - select * from Person where City = c
    - select(c)
  - update Person set City = c where Name = n
    - update(n,c)
- B+-tree index for both attributes have depth 2 (i.e., root and leaves)
- operations: lookups (search(key)), record fetch (fetch(rid)), record modification (modify(rid)), index maintenance (insert(key,rid) and delete(key,rid))
- all of them are transformed into page reads and writes
- we consider two transactions: (i) find all persons from Los Angeles and New York, (ii) move a couple of people Liz and Jerry Smith from LA to NYC
- model them as 3-level transactions, show possible schedules
- give non-serial examples for 3-level schedules that are (a) tree-reducible, (b) not tree-reducible
Task 1

- **Tree-reducible:**
  
  \[
  \text{search}_1(LA) \text{search}_1(NY) \text{fetch}_1(LA) \text{search}_2(Liz) \\
  \text{search}_2(Jerry) \text{fetch}_1(NY) \text{rest of } T_2
  \]

- **Not tree-reducible:**
  
  \[
  \text{search}_1(LA) T_2 \text{search}_1(NY) \ldots
  \]

- **Not tree-reducible:**
  
  \[
  \text{search}_1(LA) \text{update}_2(Liz, NY) \text{search}_1(NY) \text{update}_2(Jerry, NY) \ldots
  \]
Task 2

- Semi-queue: Enqueue like usual queue, Dequeue non-deterministically selects and removes an arbitrary entry from the queue.

- Construct the return value commutativity table for a semi-queue

- Show (by example) that semi-queues allow higher concurrency than FIFO-queues

<table>
<thead>
<tr>
<th></th>
<th>Enq!OK</th>
<th>Enq!one</th>
<th>Deq!OK</th>
<th>Deq!empty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enq!OK</td>
<td>+</td>
<td>imp</td>
<td>+</td>
<td>imp</td>
</tr>
<tr>
<td>Enq!one</td>
<td>-</td>
<td>imp</td>
<td>-</td>
<td>imp</td>
</tr>
<tr>
<td>Deq!OK</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Deq!empty</td>
<td>imp</td>
<td>-</td>
<td>imp</td>
<td>+</td>
</tr>
</tbody>
</table>
Task 3

Consider the following execution of operations on an initially empty queue \( q \), where \( a, b, c \) are added:

\[ \text{enq}_1(q, a)\text{enq}_2(q, b)\text{deq}_3(q)\text{enq}_1(q, c)\text{deq}_3(q) \]

Is it serializable, assuming (a) general commutativity, (b) return value commutativity for queues, (c) return value commutativity for semi-queues?

no, no, yes
Homework: Task 1

- Investigate what kind of deadlocks can arise in a layered system with layered 2PL, and how they can be detected and eliminated.
Consider two counter objects $x$ and $y$, with initial values $x = 100$ and $y = 50$. Both counters have zero as a lower bound and no upper bound. Apply the escrow locking method to the following schedule of three transactions, one of which aborts:

$$
decl(x, 60)\ incr(x, 20)\ incr(x, 10)\ de克拉(x, 50)\ de克拉(y, 60)\ incr(y, 20)\ a2\ de克拉(y, 10)\ c1\ c3$$
Info

- Exercises due: 9 AM, January 13, 2014
- Submit to andrey.gubichev@in.tum.de