Transaction Systems
Exercise Session 7

Andrey Gubichev

December 6, 2013
Versions for
\[ r_1(x)r_2(x)r_3(y)w_2(x)w_1(y)c_1w_2(z)w_3(z)r_3(x)c_3r_2(y)c_2 \]
\[ x_0 << x_2: \ t_1 \rightarrow t_2, \ t_0 \rightarrow t_2 \]
\[ y_0 << y_1: \ t_0 \rightarrow t_1, \ t_3 \rightarrow t_1 \]
\[ h(r_3(x)) = w_0(x_0) \]
\[ h(r_2(y)) = w_1(y_1) \text{ (or } w_0(y_0)\text{)} \]
\[ r_i(x) \rightarrow r_i(x_k), \quad x_k \text{ has the largest timestamp before } t_i \]

\[ w_i(x): \text{ if too late } (ts(t_k) < ts(t_i) < ts(t_j) \text{ and there is } r_j(x_k)), \text{ reject and abort} \]

\[ \text{otherwise, } w_i(x) \rightarrow w_i(x_i) \]

\[ \text{delay the commit } c_i \text{ until all transactions } T_j \text{ that have written } \text{new versions of data items read by } T_i, \text{ have committed} \]
Example 1

\[ w_1(x)c_1 r_2(x)r_3(x)c_2 r_4(x)w_3(x)c_4 c_3 \]
Example 1

- $w_1(x)c_1r_2(x)r_3(x)c_2r_4(x)w_3(x)c_4c_3$
- $T_3$ aborted
Example 2

- $r_1(x)r_2(x)r_3(y)w_2(x)w_1(y)c_1w_2(z)w_3(z)r_3(x)c_3r_2(y)c_2$
Example 2

- \( r_1(x) r_2(x) r_3(y) w_2(x) w_1(y) c_1 w_2(z) w_3(z) r_3(x) c_3 r_2(y) c_2 \)
- \( T_1 \) aborted, \( T_3 \) waits for \( T_2 \)
every data item has exactly two versions (before update, after update)
have to make sure at most one uncommitted version is present
read the last committed version only
locks for read, write, commit
commit lock is set on every item that was written by transaction

<table>
<thead>
<tr>
<th></th>
<th>$r_l(x)$</th>
<th>$w_l(x)$</th>
<th>$c_l(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_l(x)$</td>
<td>√</td>
<td></td>
<td>–</td>
</tr>
<tr>
<td>$w_l(x)$</td>
<td>√</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>$c_l(x)$</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
Example 1

- $r_1(x)w_2(y)r_1(y)w_1(x)c_1r_3(y)r_3(z)w_3(z)w_2(x)c_2w_4(z)c_4c_3$
Example 1

- $r_1(x)w_2(y)r_1(y)w_1(x)c_1r_3(y)r_3(z)w_3(z)w_2(x)c_2w_4(z)c_4c_3$
- $T_2$ has to wait before committing
- $T_4$ has to wait before writing $z$
Example 2

\[ r_1(x)r_2(x)r_3(y)w_2(x)w_1(y)c_1w_2(z)w_3(z)r_3(x)c_3 r_2(y)c_2 \]
Example 2

- $r_1(x)r_2(x)r_3(y)w_2(x)w_1(y)c_1w_2(z)w_3(z)r_3(x)c_3r_2(y)c_2$
- deadlock
Example 3

\[ r_1(x)w_1(x)r_2(x)w_2(y)r_1(y)w_2(x)c_2w_1(y)c_1 \]
Example 3

- $r_1(x)w_1(x)r_2(x)w_2(y)r_1(y)w_2(x)c_2w_1(y)c_1$
- deadlock
No blind model, action model

- No blind model: $\text{MVSR} = \text{MCSR}$
- Action model: $\text{MVSR} = \text{VSR}$
Homework: 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
Homework: 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
Consider the following execution of operations on an initially empty queue \( q \), where \( a, b, c \) are added:

\[
enq_1(q, a)\ enq_2(q, b)\ deq_3(q)\ enq_1(q, c)\ deq_3(q)
\]

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

- Exercises due: 9 AM, December 23, 2013
- Submit to andrey.gubichev@in.tum.de