Transaction Systems
Exercise Session 11

Andrey Gubichev

January 27, 2014
Homework, Task 1

- Given is the $B^+$-tree index on AccountNumber (see next slide)
- All nodes have capacity of 4
- Which locks need to be requested for the following transaction
  
  ```sql
  begin transaction;
  select Count(*) From Accounts
  where AccountNumber between 11 and 25;
  insert into Accounts (AccountNumber,...) values (27,...);
  commit transaction;
  ```

- Assume incremental key range locking at the access layer and lock coupling at the page layer
Task 1: Index
Lock coupling

- Range_Search: read lock on the node. Check for conflicts
- Insert: write lock on the node. Conflicts!
- Search: release the lock once we got the lock of the child
- Range_Search: or on the predecessor
Levels of isolation

- read uncommitted: write locks only (S2PL)
- read committed: write locks plus short read locks (only during the operation)
- repeatable read: only phantoms are allowed, no other inconsistencies
Task 2:

- Give examples of schedules that fall into the following five isolation-level classes (but not into the next larger, surrounding class):
  - not even read uncommitted
    - \( r_1(x)r_2(x)w_2(x)w_1(x)c_2c_1 \)
  - read uncommitted
    - \( r_1(x)w_1(x)r_2(x)a_1w_2(x)c_2 \)
  - read committed
    - \( r_1(x)r_2(x)w_2(x)c_2w_1(x)c_1 \)
    - \( r_1(x)w_1(x)w_2(y)c_2r_1(y)c_1 \)
  - repeatable read
    - \( r_1(x)w_2(x)c_2r_1(x)c_1 \)
  - serializability
Homework, Part 2

- Determine the necessary log actions (during normal work) and recovery actions (during restart). Redo-history

<table>
<thead>
<tr>
<th>Action</th>
<th>Change of cached DB [PageNo:SeqNo]</th>
<th>Change of stable DB [PageNo:SeqNo]</th>
<th>Log added to buffer [LogSeqNo:SeqNo]</th>
<th>Log added to Log [LogSeqNo’s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: begin($t_1$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2: write($p, t_1$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3: write($q, t_1$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4: commit($t_1$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5: flush($p$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6: begin($t_2$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7: write($p, t_2$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8: write($r, t_2$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9: checkpoint</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10: commit($t_2$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11: begin($t_3$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12: flush($p$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13: write($p, t_3$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14: write($q, t_3$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15: flush($q$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16: write($r, t_3$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... system crash</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Homework, Part 2, Cont.

- First consider the case where heavyweight checkpoints are used and flush actions are not logged; then consider lightweight checkpoints; finally, discuss the additional effect of keeping flush actions
Info

- Exercises due: 9 AM, February 3, 2014
- Submit to andrey.gubichev@in.tum.de