Transaction Systems
Exercise Session 4

Andrey Gubichev

November 11, 2013
Homework: Task 1

- Prove that $s \in CSR \iff (\forall T \subseteq \text{trans}(s)) \Pi_T(s) \in VSR$
  - $(\Rightarrow)$ If $s \in CSR$, then $CG(s)$ is acyclic. Note that $CG(\Pi_T(s))$ is a subgraph of $CG(s)$, i.e. it is acyclic as well, and the schedule is in $CSR \subseteq VSR$
  - Lemma: Let $s_1$ be a schedule with transactions from $T_1$. Let $CG(s_1)$ have an edge $T_i \rightarrow T_j$ because of $x$, and no other transaction writes $x$. Then in any serial view equivalent schedule: $T_i < T_j$
  - Proof of the Lemma: three cases
    - $r < w$
    - $w < r$
    - $w < w$
Homework: Task 1 (cont.)

- Prove that $s \in \text{CSR} \iff (\forall T \subseteq \text{trans}(s)) \Pi_T(s) \in \text{VSR}$
  - Lemma: Let $s_1$ be a schedule with transactions from $T_1$. Let $CG(s_1)$ have an edge $T_i \rightarrow T_j$ because of $x$, and no other transaction writes $x$. Then in any serial view equivalent schedule: $T_i < T_j$
  - $(\Leftarrow)$ Suppose there is a cycle in $CG(s)$. Get the shortest cycle $\{T_1, \ldots, T_k\}$. Let $s_1$ be a projection of $s$ on that cycle
    - $k = 2$: apply Lemma directly
    - $k = 3$: $T_1 \rightarrow T_2$ because of $x$. Show that $T_3$ can not write $x$. Apply Lemma
    - $k > 3$: same as $k = 3$
Homework: Task 2

- $w_1(x)r_2(y)r_1(x)c_1r_2(x)w_2(y)c_2$
- 2PL, S2PL, SS2PL
Plan for today

- Snapshot Isolation
- Deadlocks in 2PL
Snapshot Isolation

Relaxes serializability:

- all read operations see a consistent view of the database
- Informally: transaction operates on a ”snapshot” taken at its start
- write operations acquire locks
- more specifically, \( T \) can update \( x \) if no one else updated \( x \) and committed while \( T \) was running
SI: Pros and Cons

+  
  ▶ no dirty reads  
  ▶ reads are repeatable  
  ▶ no phantom reads

–

  ▶ Write skew: example about two doctors from the lecture
Complex example: Batch processing

Transaction system with two tables:

▶ **receipts**: tracks the receipt of today, each receipt has a batch number. Batch number is increased once a day.

▶ **control**: holds the today’s batch number

There are three types of transactions:

▶ **New-Receipt**: read controls, insert a new row into receipts with the current batch number

▶ **Close-Batch**: increment the batch number in control at the end of the day

▶ **Report**: get the current batch number, report the total of the previous batch receipts
Complex example: Batch processing

<table>
<thead>
<tr>
<th>Step</th>
<th>$T_1$ Report</th>
<th>$T_2$ New-Receipt</th>
<th>$T_3$ Close-batch</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td></td>
<td>$x \leftarrow \text{SELECT} \text{ current_branch}$</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>$x \leftarrow \text{SELECT} \text{ current_batch}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 5.   | $\text{SELECT SUM}(\text{amount})$  
FROM receipts  
WHERE batch=$x-1$ |                   |                  |
| 6.   | COMMIT      |                   |                  |
| 7.   |             |                   |                  |
| 8.   |             | $\text{INSERT INTO}$ receipts  
VALUES $(x, \text{somedata})$  
COMMIT |                  |

▶ what’s the problem with execution under SI?
<table>
<thead>
<tr>
<th>Step</th>
<th>$T_1$ Report</th>
<th>$T_2$ New-Receipt</th>
<th>$T_3$ Close-batch</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>$x \leftarrow \text{SELECT current}_\text{batch}$</td>
<td>$x \leftarrow \text{SELECT current}_\text{branch}$</td>
<td></td>
</tr>
</tbody>
</table>
| 5.   | $\text{SELECT SUM}(\text{amount})$  
$\text{FROM receipts}$  
$\text{WHERE batch}=x-1$ |                  | **INCREMENT**  
$\text{current}_\text{batch}$  
**COMMIT** |
| 6.   | **COMMIT**  |                   |                  |
| 7.   |             |                   |                  |
| 8.   |             | **INSERT INTO**  
$\text{receipts}$  
**VALUES**  
$(x, \text{somedata})$  
**COMMIT** |                  |

- what’s the problem with execution under SI?
- note: $T_1$ is read-only! But without $T_1$ the history is in CSR
- what would 2PL do?
2PL vs SI

\[ s_1 = r_2(y)r_3(y)w_3(y)c_3r_1(y)r_1(x)c_1w_2(x)c_2 \]
\[ s_2 = r_1(x)w_2(y)r_2(x)r_3(y)w_1(z)r_1(y)w_3(z)c_1c_2c_3 \]
\[ s_3 = r_1(x)r_2(x)r_1(y)r_2(y)w_1(y)c_1w_2(x)c_2 \]
\[ s_4 = r_1(x)r_3(x)r_4(x)w_4(x)w_4(y)c_4r_2(y)w_2(z)c_2c_2w_3(z)c_3r_1(y)c_1 \]
2PL: handling deadlocks

- 2PL can lead to deadlocks
- Wait-for graph: nodes are active transactions, edges: $T_i$ waits for $T_j$.
- Cycle in the WFG means deadlock
- $s = r_1(x) r_2(x) w_3(x) w_4(x) w_1(x) c_1 w_2(x) c_2 c_3 c_4$
  - $T_1$ waits for $T_2$ and vice versa
Homework: Task 1 (Experimental)

Get the PostgreSQL and do the following:

Log into the PostgreSQL database twice (with two different SQL windows of PGAdminIII) and try to update your City table in the first window while concurrently looking at the database content in the second window (how exactly – see next slide).

Initial scripts to populate the tables will be given on the website.
Homework: Task 1.1, Transaction Isolation

Execute the following two transactions and write down the results of the SELECT COUNT() statements:

<table>
<thead>
<tr>
<th>Window 1</th>
<th>Window 2</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0    BEGIN TRANSACTION;</td>
<td>BEGIN TRANSACTION;</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>SELECT COUNT(*) FROM City;</td>
<td>—</td>
</tr>
<tr>
<td>2</td>
<td>INSERT INTO City VALUES (...)</td>
<td>—</td>
</tr>
<tr>
<td>3</td>
<td>SELECT COUNT(*) FROM City;</td>
<td>—</td>
</tr>
<tr>
<td>4</td>
<td>INSERT INTO Person VALUES(..)</td>
<td>—</td>
</tr>
<tr>
<td>5</td>
<td>SELECT COUNT(*) FROM Person</td>
<td>—</td>
</tr>
<tr>
<td>6</td>
<td>COMMIT;</td>
<td>—</td>
</tr>
<tr>
<td>7</td>
<td>SELECT COUNT(*) FROM Person;</td>
<td>—</td>
</tr>
</tbody>
</table>

Which of the four ACID properties are reflected in the system responses? What would have happen if you did rollback instead of commit in Window 1?
**Homework: Task 1.2, Isolation Level**

Execute the following two transactions and write down the results of the `SELECT COUNT()` statements:

<table>
<thead>
<tr>
<th>Window 1</th>
<th>Window 2</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BEGIN TRANSACTION; SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>SELECT COUNT(<em>) FROM City; SELECT COUNT(</em>) FROM Person</td>
<td>—</td>
</tr>
<tr>
<td>2</td>
<td>INSERT INTO City VALUES (...)</td>
<td>—</td>
</tr>
<tr>
<td>3</td>
<td>SELECT COUNT(*) FROM City;</td>
<td>—</td>
</tr>
<tr>
<td>4</td>
<td>SELECT COUNT(*) FROM City;</td>
<td>—</td>
</tr>
<tr>
<td>5</td>
<td>INSERT INTO Person VALUES(...)</td>
<td>—</td>
</tr>
<tr>
<td>6</td>
<td>SELECT COUNT(*) FROM Person</td>
<td>—</td>
</tr>
<tr>
<td>7</td>
<td>SELECT COUNT(*) FROM Person;</td>
<td>—</td>
</tr>
<tr>
<td>8</td>
<td>COMMIT;</td>
<td>—</td>
</tr>
<tr>
<td>9</td>
<td>SELECT COUNT(*) FROM City;</td>
<td>—</td>
</tr>
<tr>
<td>10</td>
<td>COMMIT;</td>
<td>—</td>
</tr>
<tr>
<td>11</td>
<td>SELECT COUNT(*) FROM City;</td>
<td>—</td>
</tr>
</tbody>
</table>

Discuss the difference to the previous experiment.
Homework: Task 2, Programming Exercise

- Write a program (C++/Java/Python) that for a given history identifies whether it belongs to FSR, VSR or CSR
- Input format: the number of transactions and the history in the form

  \[ r_{1}x r_{2}y w_{1}z w_{2}y \ldots \]

- Submit: Makefile-based project (C++) or Eclipse project (Java) or Python file(s) (.py)
- Submission should have README file on how to run the program
Consider the following input schedules for Ordered 2PL (O2PL). What are the output schedules? Give details about when locks are requested, granted, attempted to be released and eventually released.

- $s = w_1(x)r_2(x)c_2r_3(y)c_3w_1(y)c_1$
- $s = w_1(x)r_2(x)r_3(y)c_3r_2(z)c_2w_1(y)c_1$

What is the relationship between Gen(2PL) and Gen(Basic Timestamp Ordering)? (proofs and examples)

- equal?
- intersect?
- subset? not subset?
Grading

- Task 1: 10 points
- Task 2: 15 points
- Task 3 + 4: 10 points
Info

- Exercises due: 9 AM, December 2, 2013