CS216: Data-Intensive Computing Systems

Concurrency Control (II)

Shivnath Babu
How to enforce serializable schedules?

*Option 1:* run system, recording P(S); at end of day, check for P(S) cycles and declare if execution was good
How to enforce serializable schedules?

*Option 2:* prevent P(S) cycles from occurring

![Diagram showing DB, Scheduler, and transactions T1, T2, ..., Tn]

- DB
- Scheduler
- Transactions T1, T2, ..., Tn
A locking protocol

Two new actions:

lock (exclusive): \( li_1 (A) \)

unlock: \( ui_1 (A) \)
Rule #1: Well-formed transactions

Ti: ... li(A) ... pi(A) ... ui(A) ...
Rule #2  Legal scheduler

\[ S = \ldots li(A) \ldots \ldots \ldots \ldots ui(A) \ldots \ldots \]

no \( l_j(A) \)
Exercise:

- What schedules are legal?
  What transactions are well-formed?

\[ S_1 = \text{l}_1(\text{A})\text{l}_1(\text{B})\text{r}_1(\text{A})\text{w}_1(\text{B})\text{l}_2(\text{B})\text{u}_1(\text{A})\text{u}_1(\text{B}) \]
\[ \text{r}_2(\text{B})\text{w}_2(\text{B})\text{u}_2(\text{B})\text{l}_3(\text{B})\text{r}_3(\text{B})\text{u}_3(\text{B}) \]

\[ S_2 = \text{l}_1(\text{A})\text{r}_1(\text{A})\text{w}_1(\text{B})\text{u}_1(\text{A})\text{u}_1(\text{B}) \]
\[ \text{l}_2(\text{B})\text{r}_2(\text{B})\text{w}_2(\text{B})\text{l}_3(\text{B})\text{r}_3(\text{B})\text{u}_3(\text{B}) \]

\[ S_3 = \text{l}_1(\text{A})\text{r}_1(\text{A})\text{u}_1(\text{A})\text{l}_1(\text{B})\text{w}_1(\text{B})\text{u}_1(\text{B}) \]
\[ \text{l}_2(\text{B})\text{r}_2(\text{B})\text{w}_2(\text{B})\text{u}_2(\text{B})\text{l}_3(\text{B})\text{r}_3(\text{B})\text{u}_3(\text{B}) \]
Exercise:

• What schedules are legal?
  What transactions are well-formed?

S1 = l_1(A)l_1(B)r_1(A)w_1(B)l_2(B)u_1(A)u_1(B)
  r_2(B)w_2(B)u_2(B)l_3(B)r_3(B)u_3(B)

S2 = l_1(A)r_1(A)w_1(B)u_1(A)u_1(B)
  l_2(B)r_2(B)w_2(B)l_3(B)r_3(B)u_3(B)

S3 = l_1(A)r_1(A)u_1(A)l_1(B)w_1(B)u_1(B)
  l_2(B)r_2(B)w_2(B)u_2(B)l_3(B)r_3(B)u_3(B)
## Schedule F

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_1(A)$; Read(A)</td>
<td>$l_2(A)$; Read(A)</td>
</tr>
<tr>
<td>$A \leftarrow A + 100$; Write(A); $u_1(A)$</td>
<td>$A \leftarrow Ax2$; Write(A); $u_2(A)$</td>
</tr>
<tr>
<td>$l_1(B)$; Read(B)</td>
<td>$l_2(B)$; Read(B)</td>
</tr>
<tr>
<td>$B \leftarrow B + 100$; Write(B); $u_1(B)$</td>
<td>$B \leftarrow Bx2$; Write(B); $u_2(B)$</td>
</tr>
</tbody>
</table>
# Schedule F

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>( l_1(A); \text{Read}(A) )</td>
<td>( l_2(A); \text{Read}(A) )</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>( A \leftarrow A + 100; \text{Write}(A); u_1(A) )</td>
<td>( A \leftarrow A x 2; \text{Write}(A); u_2(A) )</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>( l_1(B); \text{Read}(B) )</td>
<td>( l_2(B); \text{Read}(B) )</td>
<td></td>
<td>250</td>
</tr>
<tr>
<td>( B \leftarrow B + 100; \text{Write}(B); u_1(B) )</td>
<td>( B \leftarrow B x 2; \text{Write}(B); u_2(B) )</td>
<td></td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>250</td>
<td>150</td>
</tr>
</tbody>
</table>
Rule #3  Two phase locking (2PL)  
for transactions

\[ T_i = \ldots \text{li}(A) \ldots \ldots \ldots \text{ui}(A) \ldots \ldots \]

no unlocks  no locks
# locks held by Ti

Growing Phase

Time

Shrinking Phase
Schedule G

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>l₁(A); Read(A)</td>
<td>l₂(A); Read(A)</td>
</tr>
<tr>
<td>A ← A + 100; Write(A)</td>
<td>A ← A x 2; Write(A)</td>
</tr>
<tr>
<td>l₁(B); u₁(A)</td>
<td>Delayed</td>
</tr>
</tbody>
</table>
Schedule G

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>l₁₁(A); Read(A)</td>
<td>l₂₁(A); Read(A)</td>
</tr>
<tr>
<td>A ← A + 100; Write(A)</td>
<td>A ← Ax2; Write(A)</td>
</tr>
<tr>
<td>l₁(B); u₁(A)</td>
<td>l₂(B); delayed</td>
</tr>
<tr>
<td>Read(B); B ← B + 100</td>
<td>Write(B); u₁(B)</td>
</tr>
</tbody>
</table>
Schedule G

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( l_1(A); \text{Read}(A) )</td>
<td>( l_2(A); \text{Read}(A) )</td>
</tr>
<tr>
<td>( A \leftarrow A + 100; \text{Write}(A) )</td>
<td>( A \leftarrow A \times 2; \text{Write}(A) ); ( l_2(B) )</td>
</tr>
<tr>
<td>( l_1(B); u_1(A) )</td>
<td>delayed</td>
</tr>
<tr>
<td>Read((B)); B \leftarrow B + 100</td>
<td>( l_2(B); u_2(A); \text{Read}(B) )</td>
</tr>
<tr>
<td>Write((B)); u_1(B)</td>
<td>( B \leftarrow B \times 2; \text{Write}(B); u_2(B) );</td>
</tr>
</tbody>
</table>
Schedule H  (T$_2$ reversed)

T1

l$_1$(A); Read(A)
A$\leftarrow$A+100; Write(A)

l$_1$(B)  delayed

T2

l$_2$(B); Read(B)
B$\leftarrow$Bx2; Write(B)

l$_2$(A)  delayed
• Assume deadlocked transactions are rolled back
  – They have no effect
  – They do not appear in schedule

E.g., Schedule H =

This space intentionally left blank!
Next step:

Show that rules #1,2,3 ⇒ conflict-serializable schedules
Conflict rules for \( l_i(A), u_i(A) \):

- \( l_i(A), l_j(A) \) conflict
- \( l_i(A), u_j(A) \) conflict

Note: no conflict \( < u_i(A), u_j(A) >, < l_i(A), r_j(A) > \),...
**Theorem**  Rules #1,2,3 \(\Rightarrow\) conflict

(2PL) serializable schedule

To help in proof:

**Definition**  \(\text{Shrink}(T_i) = \text{SH}(T_i) = \) first unlock action of \(T_i\)
Lemma

$Ti \rightarrow Tj$ in $S \Rightarrow SH(Ti) <_S SH(Tj)$

Proof of lemma:

$Ti \rightarrow Tj$ means that

$$S = \ldots p_i(A) \ldots q_j(A) \ldots; \ p, q \text{ conflict}$$

By rules 1,2:

$$S = \ldots p_i(A) \ldots u_i(A) \ldots l_j(A) \ldots q_j(A) \ldots$$

By rule 3:

$$SH(Ti) \leftarrow$$

$$SH(Tj)$$

So, $SH(Ti) <_S SH(Tj)$
**Theorem**  Rules #1,2,3 $\implies$ conflict

(2PL) serializable schedule

**Proof:**

(1) Assume $P(S)$ has cycle

$T_1 \rightarrow T_2 \rightarrow \ldots \rightarrow T_n \rightarrow T_1$

(2) By lemma: $SH(T_1) < SH(T_2) < \ldots < SH(T_1)$

(3) Impossible, so $P(S)$ acyclic

(4) $\implies$ $S$ is conflict serializable
• Beyond this simple 2PL protocol, it is all a matter of improving performance and allowing more concurrency....
  – Shared locks
  – Multiple granularity
  – Inserts, deletes, and phantoms
  – Other types of C.C. mechanisms