Chapter 9 Concurrency Control

T1 T2 ... Tn

DB (consistency constraints)
Concepts

*Transaction*: sequence of $r_i(x)$, $w_i(x)$ actions

*Conflicting actions*:

```
(1) r1(A) < W2(A) < W1(A)
(2) W2(A) < r1(A) < W2(A)
```

*Schedule*: represents chronological order in which actions are executed

*Serial schedule*: no interleaving of actions or transactions
A ReCap

- Examples of Main Concepts are given next.
Example:

T1: Read(A)  T2: Read(A)
   A ← A+100  A ← A×2
   Write(A)  Write(A)
   Read(B)  Read(B)
   B ← B+100  B ← B×2
   Write(B)  Write(B)

Constraint: A=B
Schedule C

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Read(A); A ← A+100</td>
<td>Write(A);</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Write(A);</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Read(B); B ← B+100;</td>
<td>Write(A);</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Write(B);</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>25</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>25</td>
<td>250</td>
<td></td>
</tr>
</tbody>
</table>

Read(A); A ← A×2;
Write(A);

Read(B); B ← B×2;
Write(B);
Example:

$Sc = r_1(A)w_1(A)\ r_2(A)w_2(A)\ r_1(B)w_1(B)\ r_2(B)w_2(B)$

$Sc' = r_1(A)w_1(A)\ r_1(B)w_1(B)\ r_2(A)w_2(A)\ r_2(B)w_2(B)$

$T_1 \quad T_2$
Returning to Sc

\[ Sc = r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B) \]

- no cycles \( \Rightarrow \) Sc is “equivalent” to a serial schedule,
  I.e., in this case \( (T_1,T_2) \).
# Schedule D

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read(A); A ← A+100</td>
<td></td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Write(A);</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read(A); A ← A×2;</td>
<td></td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>Write(A);</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read(B); B ← B×2;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write(B);</td>
<td>B ← B+100;</td>
<td></td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>150</td>
</tr>
<tr>
<td></td>
<td></td>
<td>250</td>
<td>150</td>
</tr>
</tbody>
</table>
Now for $S_d$:

$$S_d = r_1(A)w_1(A)r_2(A)w_2(A) r_2(B)w_2(B)r_1(B)w_1(B)$$

$$S_d = r_1(A)w_1(A) r_1(B)w_1(B)r_2(A)w_2(A)r_2(B)w_2(B)$$
Or, let’s try for $S_d$:

$$S_d = r_1(A)w_1(A)r_2(A)w_2(A) \quad r_2(B)w_2(B)r_1(B)w_1(B)$$

$$S_d = r_2(A)w_2(A)r_2(B)w_2(B) \quad r_1(A)w_1(A)r_1(B)w_1(B)$$

T1

T2
For Schedule D:

\[ S_d = r_1(A)w_1(A)r_2(A)w_2(A) r_2(B)w_2(B)r_1(B)w_1(B) \]

- there seems to be no save way to transform this S-D into an equivalent serial schedule?
For Schedule D:

$S_d = r_1(A)w_1(A)r_2(A)w_2(A) \ r_2(B)w_2(B)r_1(B)w_1(B)$

$T_1 \rightarrow T_2 \quad T_2 \rightarrow T_1$

$S_d$ cannot be rearranged into serial schedule
Definition

$S_1$, $S_2$ are conflict equivalent schedules if $S_1$ can be transformed into $S_2$ by a series of swaps on non-conflicting actions.
Definition

A schedule is conflict serializable if it is conflict equivalent to some serial schedule.
How determine this?

Answer: A Precedence Graph!
Precedence graph $P(S)$ ($S$ is schedule)

Nodes: transactions in $S$

Arcs: $T_i \rightarrow T_j$ whenever

- $p_i(A), q_j(A)$ are actions in $S$
- $p_i(A) <_S q_j(A)$
- at least one of $p_i, q_j$ is a write
Exercise:

• What is $P(S)$ for $S = w_3(A) \, w_2(C) \, r_1(A) \, w_1(B) \, r_1(C) \, w_2(A) \, r_4(A) \, w_4(D)$

• Is $S$ serializable?
Another Exercise:

• What is $P(S)$ for $S = w_1(A) \ r_2(A) \ r_3(A) \ w_4(A)$?

• Is $S$ serializable?
Lemma

\( S_1, S_2 \) conflict equivalent \( \Rightarrow P(S_1) = P(S_2) \)

Proof:

Assume \( P(S_1) \neq P(S_2) \)

\( \Rightarrow \exists T_i: T_i \rightarrow T_j \) in \( S_1 \) and not in \( S_2 \)

\( \Rightarrow S_1 = \ldots p_i(A) \ldots q_j(A) \ldots \) \[ \begin{array}{c} p_i, q_j \text{ conflict} \\ \end{array} \]

\( S_2 = \ldots q_j(A) \ldots p_i(A) \ldots \)

\( \Rightarrow S_1, S_2 \) not conflict equivalent
Note: \( P(S_1) = P(S_2) \not\Rightarrow S_1, S_2 \) conflict equivalent

**Counter example:**

\[
S_1 = w_1(A) \, r_2(A) \, w_2(B) \, r_1(B)
\]

\[
S_2 = r_2(A) \, w_1(A) \, r_1(B) \, w_2(B)
\]
Theorem

$P(S_1)$ acyclic $\iff S_1$ conflict serializable

($\Leftarrow$) Assume $S_1$ is conflict serializable

$\Rightarrow \exists S_s: S_s, S_1$ conflict equivalent

$\Rightarrow P(S_s) = P(S_1)$

$\Rightarrow P(S_1)$ acyclic since $P(S_s)$ is acyclic
Theorem

$P(S_1)$ acyclic $\iff S_1$ conflict serializable

($\implies$) Assume $P(S_1)$ is acyclic

Transform $S_1$ as follows:

1. Take $T_1$ to be transaction with no incident arcs
2. Move all $T_1$ actions to the front
   
   $S_1 = \ldots \ q_j(A) \ldots p_1(A) \ldots$

3. we now have $S_1 = < T_1 \text{ actions } > <\ldots \text{ rest } \ldots >$
4. repeat above steps to serialize rest!
How to enforce serializable schedules?

Next Time ...