Serializability Summary

- As transactions execute concurrently, we must guarantee isolation, i.e., we only want to allow "good" schedules.
- "Good" schedules, or rather, schedules that guarantee isolation, means that the resulting schedules are equivalent to some serial schedule.
- Any schedule that is conflict serializable is equivalent to some serial schedule.
- Any schedule that is view serializable is equivalent to some serial schedule.
- Schedules exist which are neither view nor conflict serializable, but are equivalent to some serial schedule.
- Schedules exist which are view serializable but not conflict serializable.
- Concurrency control schemes/algorithms are required that ensure either conflict or view serializability.
Testing for View Serializability

- Let $S$ be a schedule consisting of transactions $\{T_1, T_2, \ldots, T_n\}$.

- Construct a labeled precedence graph as follows.

- First, add two more “dummy” transactions $T_b$ and $T_f$.
  - $T_b$ issues write($Q$) for each $Q$ accessed in $S$.
  - $T_f$ issues read($Q$) for each $Q$ accessed in $S$.
  - $T_b$ is inserted at the beginning of $S$.
  - $T_f$ is inserted at the end of $S$. 
Testing for View Serializability, Cont.

1. Add an edge \( T_i \rightarrow T_j \) if transaction \( T_j \) reads the value of data item \( Q \) written by transaction \( T_i \).

2. Remove all the edges incident on useless transactions. A transaction \( T_i \) is useless if there exists no path, in the precedence graph, from \( T_i \) to transaction \( T_f \).

3. For each data item \( Q \) such that \( T_j \) reads the value of \( Q \) written by \( T_i \), and \( T_k \) executes \text{write}(Q) \) and \( T_k \neq T_b \), do the following:
   a) If \( T_i = T_b \) and \( T_j \neq T_f \), then insert the edge \( 0 \ T_j \rightarrow T_k \) in the labeled precedence graph.
   b) If \( T_i \neq T_b \) and \( T_j = T_f \), then insert the edge \( 0 \ T_k \rightarrow T_i \) in the labeled precedence graph.
   c) If \( T_i \neq T_b \) and \( T_j \neq T_f \), then insert the pair of edges \( p \ T_k \rightarrow T_i \) and \( p \ T_j \rightarrow T_k \) in the labeled precedence graph where \( p \) is a unique integer larger than 0 that has not been used earlier for labeling edges.
Testing for View Serializability, Cont.

- Meaning of rules 3a-3c:

  - Rule 3a) ensures that if a transaction reads an initial value of $Q$ in schedule $S$, then it also reads that same value in any view-equivalent schedule.

  - Rule 3b) ensures that if a transaction writes the final value of $Q$ in schedule $S$, then it also writes that same value in any view-equivalent schedule.

  - Rule 3c) ensures that if a transaction $T_i$ writes a data item that $T_j$ reads, then any transaction $T_k$ that writes the same data item must either come before $T_i$ or after $T_j$ in any view-equivalent schedule.
Testing for View Serializability

Example #1

- Consider the following schedule:

<table>
<thead>
<tr>
<th></th>
<th>T3</th>
<th>T4</th>
<th>T7</th>
</tr>
</thead>
<tbody>
<tr>
<td>read(Q)</td>
<td>write(Q)</td>
<td>read(Q)</td>
<td>write(Q)</td>
</tr>
<tr>
<td>write(Q)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Diagram]
Testing for View Serializability - Example #2

- Consider the following schedule:

<table>
<thead>
<tr>
<th>T3</th>
<th>T4</th>
<th>T7</th>
<th>T8</th>
<th>T9</th>
<th>T10</th>
</tr>
</thead>
<tbody>
<tr>
<td>read(Q)</td>
<td>write(Q)</td>
<td>read(Q)</td>
<td>write(A)</td>
<td>read(A)</td>
<td>write(A)</td>
</tr>
<tr>
<td>write(Q)</td>
<td>read(Q)</td>
<td>write(A)</td>
<td>write(A)</td>
<td>read(A)</td>
<td>write(A)</td>
</tr>
<tr>
<td>write(Q)</td>
<td>write(B)</td>
<td>write(A)</td>
<td>write(A)</td>
<td>read(A)</td>
<td>write(A)</td>
</tr>
</tbody>
</table>

![Diagram](image)