Serializability

Theorem 2

Serializability Theorem

- **Theorem**: A history $H$ is serializable iff $SG(H)$ is acyclic.
- **Proof**: IF
  - Suppose $H$ is a history over $T=\{T_1, T_2, ..., T_n\}$.
  - WLOG assume $T_1, T_2, ..., T_m$ ($m \leq n$) are all txns in $T$ that are committed in $H$.
  - Thus $T_1, T_2, ..., T_m$ are the nodes in $SG(H)$.
  - Since $SG(H)$ is acyclic, it can be topologically sorted.
Serializability Theorem

• Let $i_1, i_2, \ldots, i_m$ be a permutation of $1, 2, \ldots, m$ such that $T_{i_1}, T_{i_2}, \ldots, T_{i_m}$ is a topological sort of $SG(H)$.
• Let $H_s$ be the serial history $T_{i_1}, T_{i_2}, \ldots, T_{i_m}$.
• We claim that $C(H) \equiv H_s$.
• Let $p_i \in T_i$ and $q_j \in T_j$, where $T_i, T_j$ are committed in $H$.
• Suppose $p_i, q_j$ conflict and $p_i <_H q_j$.
• By the definition of $SG(H)$, $T_i \rightarrow T_j$ is in $SG(H)$.

Serializability Theorem

• Therefore in any topological sort of $SG(H)$, $T_i$ must appear before $T_j$.
• Thus in $H_s$ all operations of $T_i$ must precede all operations of $T_j$, and in particular, $p_i <_{H_s} q_j$.
• Thus any two conflicting operations are ordered in the same way in $C(H)$ as $H_s$. Thus $C(H) \equiv H_s$, which is serial, therefore $H$ is SR.
Serializability Theorem

• **ONLY IF:**
  - Suppose $H$ is SR. Let $H_s$ be a serial history equivalent to $C(H)$.
  - Consider an edge $T_i \rightarrow T_j$ in $SG(H)$.
  - Thus there are two conflicting operations $p_i, q_j$ of $T_i, T_j$ (respectively), such that $p_i <_H q_j$.
  - Because $C(H) \equiv H_s, p_i <_{Hs} q_j$.
  - Because $H_s$ is serial, and $p_i$ precedes $q_j$, it implies that $T_i$ precedes $T_j$ in $H_s$.

Serializability Theorem

• Thus we see that if $T_i \rightarrow T_j$ is in $SG(H)$, then $T_i$ precedes $T_j$ in $H_s$.
• Suppose that there is a cycle in $SG(H)$, say $T_1 \rightarrow T_2 \rightarrow \ldots \rightarrow T_k \rightarrow T_l$
• This implies that $T_i$ appears before itself in $H_s$, which is absurd.
• Thus no cycle can exist in $SG(H)$ if $H$ is SR.
• QED
Recoverable Histories

• A txn $T_i$ reads $x$ from $T_j$ in history $H$ if
  - $w_j[x] < r_i[x]$;
  - NOT $(a_j < r_i[x])$ and
  - If there is some $w_k[x]$ such that $w_j[x] < w_k[x] < r_i[x]$, then $a_k < r_i[x]$.

• A history is Recoverable (RC) if, whenever $T_i$ reads from $T_j$ ($i \neq j$) in $H$, $c_i \in H$, $c_j < c_i$.

• A history Avoids Cascading Aborts (ACA) if, whenever $T_i$ reads $x$ from $T_j$ ($i \neq j$) in $H$, $c_i < r_i[x]$.

• A history $H$ is Strict (ST) if whenever $w_j[x] < o_i[x]$ ($i \neq j$), either $a_j < o_i[x]$ or $c_j < o_i[x]$, where $o_i[x]$ is $r_i[x]$ or $w_i[x]$.

Examples

• $T_1 = w_1[x] w_1[y] w_1[z] c_1$
• $T_2 = r_2[u] w_2[x] r_2[y] w_2[y] c_2$
• $w_1[x] w_1[y] r_2[u] w_2[x] r_2[y] w_2[y] w_1[z] c_2 w_1[z] c_1$
• Not RC
• $w_1[x] w_1[y] r_2[u] w_2[x] r_2[y] w_2[y] w_1[z] c_1 c_2$
• RC, not ACA
• $w_1[x] w_1[y] r_2[u] w_1[z] w_2[x] c_1 r_2[y] w_2[y] c_2$
• RC, ACA, not Strict
Prefix Commit-closed

• A property of a history is called prefix commit-closed if, whenever the property is true of history $H$, it is also true of history $C(H')$, for any prefix $H'$ of $H$.

• Since failures may occur when a prefix of an acceptable history has been processed, DBMS schedulers and recovery managers must satisfy prefix commit-closed properties for CC and recovery, i.e. every $C(H')$ must be acceptable too.
Theorem

• Serializability is a prefix commit-closed property.
• **Proof:** Since $H$ is **SR**, $SG(H)$ is acyclic. Consider $SG(C(H'))$ where $H'$ is any prefix of $H$.
• If $T_i \rightarrow T_j$ is an edge of this graph, then we have two conflicting operations $p_i, q_j$ belonging to $T_i, T_j$ (respectively) with $p_i <_{C(H')} q_j$.
• But then clearly $p_i <_H q_j$ and thus $T_i \rightarrow T_j$ exists in $SG(H)$.
• Therefore $SG(C(H'))$ is a subgraph of $SG(H)$.
• If $SG(H)$ is acyclic, so must $SG(C(H'))$, hence $C(H')$ is **SR**.

Other Operations

• So far, we have limited ourselves to reads and writes.
• However, serializability does not limit us to these.
• We just need to redefine conflicting operations as any pair for which the result, in general, depends upon the order of their execution.
• Effect is: value returned, and final value of data.
• Thus we need only define the notion of conflict appropriately. For example, we could add Increment and Decrement as basic (atomic) operations. Assume they do not return a value.
### Compatibility Matrix

<table>
<thead>
<tr>
<th></th>
<th>Read</th>
<th>Write</th>
<th>Increment</th>
<th>Decrement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Write</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Increment</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Decrement</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

### View Equivalence

- So far, we have based equivalence of histories on the fact that the ordering of writes with respect to other operations on the same object should be the same.
- We can say that the effects are simply the values read and the final values of data objects. If these are the same in two histories, then they are declared to be view equivalent.
View Equivalence

- The final write of $x$ in a history $H$ is the operation $w_i[x] \in H$, such that $a_i \notin H$ and for any $w_j[x] \in H$ ($j \neq i$) either $w_j[x] < w_i[x]$ or $a_j \in H$.

- Two histories $H, H'$ are view equivalent if
  - they are over the same set of txns and have the same operations;
  - For any $T_i, T_j$ such that $a_i, a_j \notin H$ (hence $a_i, a_j \notin H'$) and for any $x$, if $T_i$ reads $x$ from $T_j$ in $H$ then $T_i$ reads $x$ from $T_j$ in $H'$ and
  - For each $x$, if $w_i[x]$ is the final write of $x$ in $H$ then it is also the final write of $x$ in $H'$.

View Serializability

- A history, $H$, is defined to be view serializable (VSR) if for any prefix $H'$ of $H$, $C(H')$ is view equivalent to some serial history.
- We need to ensure prefix commit closure
  - $w_1[x] w_2[x] w_2[y] c_2 w_1[y] c_1 w_3[x] w_3[y] c_3$
  - The complete history is view equiv. to $T_1 T_2 T_3$.
  - However, upto $c_i$ it is not view equiv. to either $T_1 T_2$ or $T_2 T_1$!
CSR vs. VSR

- **Theorem**: If $H$ is conflict serializable then it is view serializable. The converse is not, generally, true.

- **Proof**: Suppose $H$ is CSR. Let $H_s$ be a serial history equivalent to $C(H')$.

- If $T_i$ reads $x$ from $T_j$ in $C(H')$, then $w_j[x] <_{C(H')} r_i[x]$ and there is no $w_k[x]$ such that $w_j[x] <_{C(H')} w_k[x] <_{C(H')} r_i[x]$.

- $H_s$ must order these in the same way i.e. $w_j[x] <_{H_s} r_i[x]$, and no intermediate $w_k[x]$. Hence they have the same reads-from relationships.

- Similarly for final writes.

\[\begin{align*}
w_1[x] & \quad w_2[x] & \quad w_2[y] & \quad c_2 & \quad w_1[y] & \quad w_3[x] & \quad w_3[y] & \quad c_3 & \quad w_1 \\
[z] & \quad c_1
\end{align*}\]