PURDUE UNIVERSITY

CS54200: Distributed Database Systems

Serializability Theory

Chris Clifton

Indiana Center for Database Systems

Transactions

- A transaction consists of read and write operations on database objects.
- It also specifies an order in which the operations are executed. This may be a partial order, i.e. some pairs of operations are not strictly ordered in time. The order describes the *"happened-before"* relationship.

Transactions

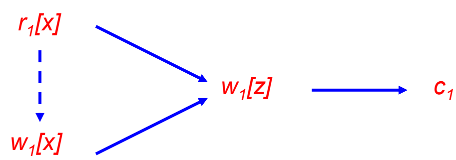
- Each operation of a transaction will be represented by the following symbols:
  - $r_1[x]$ – txn T_1 reads data item x
  - $w_1[x]$ – txn T_1 writes data item x
  - c_1 – txn T_1 commits
  - a_1 – txn T_1 aborts
  - The start of a txn is implicit

Transaction

- A transaction T_i is a partial order with ordering relation $<_i$, where
  - $T_i \subseteq \{r_i[x], w_i[x] \mid x \text{ is a data item}\} \cup \{a_i, c_i\}$;
  - $a_i \in T_i$ iff $c_i \notin T_i$;
  - If t_i is c_i or a_i (whichever is in T_i), for any other operation $p \in T_i$, $p <_i t_i$;
  - If $r_i[x], w_i[x] \in T_i$, then either $r_i[x] <_i w_i[x]$ or $w_i[x] <_i r_i[x]$.

Transactions

- A partial order can be represented by a directed acyclic graph (DAG).
- E.g.



Transactions

- Ignore all other actions of txns
- Can model input values as read statements, output as write

Histories

- A history captures the execution of several transactions.
- Histories are collections of the partial orders of txns and are partial orders too.
- They need to be more that just the sum of the partial orders of their constituent transactions though – they MUST order conflicting operations
- A pair of operations *conflict* if they both operate on the same data item and at least one of them is a write.

Complete Histories

- Let $T = \{T_1, T_2, \ldots, T_n\}$ be a set of transactions. A *complete* history H over T is a partial order with ordering relation $<_H$ where:
  - $H =$ [redacted] T_i;
  - $<_H \supseteq \bigcup_{i=}^{n} <_i$; and
  - For any two conflicting operation, $p, q \in H$, either $p <_H q$ or $q <_H p$.
- A *history* is simply a prefix of a complete history.

Example

- $T_1 = r_1[x] \rightarrow w_1[x] \rightarrow c_1$
- $T_2 = r_2[x] \rightarrow w_2[y] \rightarrow w_2[x] \rightarrow c_2$
- $T_3 = r_3[y] \rightarrow w_3[x] \rightarrow w_3[y] \rightarrow c_3$

$r_1[x] \rightarrow w_1[x] \rightarrow c_1$

$r_3[y] \rightarrow w_3[x] \rightarrow w_3[y] \rightarrow c_3$

$r_2[x] \rightarrow w_2[y] \rightarrow w_2[x] \rightarrow c_2$

Orders implied by transitivity are omitted.
For total orders, we can drop the arrows.

Committed Projection of a History

- The committed projection of a history H, denoted *C(H)*, is the history obtained from *H* by deleting all operations that do not belong to committed txns.
- This is important for the definition of serializable histories.

Equivalent Histories

- We want to allow only those histories that are EQUIVALENT to some serial history.
- We define two histories H and H' to be equivalent ($\equiv$) if
  - They are defined over the same set of transactions and have the same operations; and
  - They order conflicting operations of non-aborted transactions in the same way; that is, for any conflicting operations p_i and q_j belonging to transactions T_i and T_j (respectively) where $a_j, a_i \notin H$, if $p_i <_H q_j$ then $p_i <_{H'} q_j$.

Serializable Histories

- Because only the complete execution of txns represents a consistent state, we define a history to be serializable (SR) if its committed projection, $C(H)$, is equivalent to some serial history H_s.
- A serialization graph can be used to determine whether a history is serializable.

Serialization Graph (SG)

- The *SG(H)* is a directed graph whose nodes are committed txns in *H*, and whose edges are $T_i \rightarrow T_j$ such that one of T_i's operations precedes and conflicts with one of T_j's operations in *H*.

E.g.

$r_3[x] \rightarrow w_3[x] \rightarrow c_3$

$r_1[x] \rightarrow w_1[x] \rightarrow w_1[y] \rightarrow c_1$

$r_2[x] \rightarrow w_2[y] \rightarrow c_2$

$T_2 \rightarrow T_1 \rightarrow T_3$

NOTE: SG may not be transitive!

Serializability Theorem

A History *H* is serializable iff *SG(H)* is acyclic

Serializability Theorem

- Theorem: A history H is serializable iff $SG(H)$ is acyclic.
- Proof: IF
- Suppose H is a history over $T=\{T_1, T_2, \ldots, T_n\}$.
- WLOG assume $T_1, T_2, \ldots, T_m$ ($m \leq n$) are all txns in T that are committed in H.
- Thus $T_1, T_2, \ldots, T_m$ 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 $T_{i1}, T_{i2}, \ldots, T_{im}$ is a topological sort of $SG(H)$.
- Let Hs be the serial history $T_{i1}, T_{i2}, \ldots, T_{im}$.
- We claim that $C(H) \equiv$ Hs.
- 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, $p_i <_{Hs} q_j$.
- 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

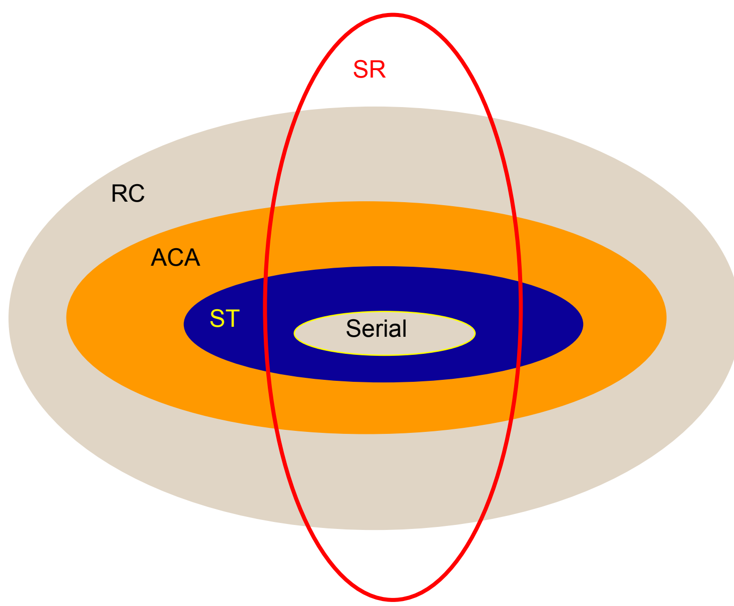
- Thus we see that if $T_i \rightarrow T_j$ is in $SG(H)$, then T_i precedes T_j in H_s.
- Suppose that there is a cycle in $SG(H)$, say $T_1 \rightarrow T_2 \rightarrow \ldots \rightarrow T_k \rightarrow T_1$
- This implies that T_1 appears before itself in H_s, which is absurd.
- Thus no cycle can 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, $c_j < r_i[x]$.
- 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

- $T_1=w_1[x]\ w_1[y]\ w_1[z]\ c_1$
- $T_2=r_2[u]\ w_2[x]\ r_2[y]\ w_2[y]\ c_2$
- $w_1[x]\ w_1[y]\ r_2[u]\ w_2[x]\ r_2[y]\ w_2[y]\ c_2\ w_1[z]\ c_1$
- 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

	Read	Write	Increment	Decrement
Read	Y	N	N	N
Write	N	N	N	N
Increment	N	N	Y	Y
Decrement	N	N	Y	Y

View Equivalence

- So far, we have based equivalence of histories on the fact that the ordering or 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_1 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 Hs 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 $w_j[x] <_{C(H')} w_k[x] <_{C(H')} r_i[x]$.
- Hs must order these in the same way i.e. $w_j[x] <_{Hs} r_i[x]$, and no intermediate $w_k[x]$. Hence they have the same reads-from relationships.
- Similarly for final writes.

VSR ≠ CSR

$w_1[x]\ w_2[x]\ w_2[y]\ c_2\ w_1[y]\ w_3[x]\ w_3[y]\ c_3\ w_1[z]\ c_1$