8: Transaction Model

Page Model

- All operations on data will be eventually mapped into read and write operations on pages.
- To study the concurrent execution of transactions it is sufficient to inspect the interleavings of the resulting page operations.
- Independently whether a page resides in cache memory or resides on disk, read and write are considered as indivisible.
Parallelism as prerequisite for distributed execution

A transaction \( T \) is a partial order \( <^1 \) of actions in \( OP, \ T = (OP, <) \), where \( OP \) is a finite set of \( T \)'s actions \( RX \) and \( WX \), where \( X \) is a data item.

Moreover, \( < \subseteq OP \times OP \) is a partial order on \( OP \) which fulfills the following properties:

- Each data item is read and written by \( T \) at most once.
- If \( p \) is a read action and \( q \) is a write actions of \( T \) and both access the same data item, then \( p < q \).

Complete transaction

We call a transaction \textit{complete}, if its first action is begin \( b \) and its last action either is commit \( c \) or abort \( a \).

\[^1\text{A binary relation is a partial order}, \text{ if it is reflexive, antisymmetric and transitive.}\]
A parallel debit/credit transaction. \( b: \text{BEGIN}; c: \text{COMMIT.} \)

When transactions are depicted as directed graphs, we omit transitive edges.

Two parallel debit/credit transactions, each prepared for parallel execution.

\[\rightarrow\] Definition of a schedule? Definition of serializability?
Two parallel debit/credit transactions, each prepared for parallel execution.

Transaction $T_1$

<table>
<thead>
<tr>
<th>PA</th>
<th>RA</th>
<th>b</th>
<th>RB</th>
<th>PB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WA</td>
<td></td>
<td></td>
<td>WB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R_1 A$</td>
<td>$W_1 A$</td>
<td>$R_2 A$</td>
<td>$W_2 A$</td>
<td></td>
</tr>
</tbody>
</table>

Transaction $T_2$

<table>
<thead>
<tr>
<th>PA</th>
<th>RA</th>
<th>b</th>
<th>RB</th>
<th>PB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WA</td>
<td></td>
<td></td>
<td>WB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R_2 B$</td>
<td>$W_2 B$</td>
<td>$R_1 B$</td>
<td>$W_1 B$</td>
<td></td>
</tr>
</tbody>
</table>

Locally observable schedules of the two transactions when executed in parallel by CPU PA and CPU PB.

(i) PA : $R_1 A$ $W_1 A$ $R_2 A$ $W_2 A$
     PB : $R_1 B$ $W_1 B$ $R_2 B$ $W_2 B$

(ii) PA : $R_1 A$ $W_1 A$ $R_2 A$ $W_2 A$
      PB : $R_2 B$ $W_2 B$ $R_1 B$ $W_1 B$

On each CPU in both cases the local schedules are serializable - however, globally, in the second case the transactions are not executed in a serializable manner!
Histories and schedules

Let $\mathcal{T} = \{T_1, \ldots, T_n\}$ be a (finite) set of complete transactions, where for each $T_i$ we have $T_i = (OP_i, <_i)$.

A history of $\mathcal{T}$ is a pair $S = (OP_S, <_S)$, where
- $OP_S = \bigcup_{i=1}^{n} OP_i$ and $<_S$ a partial order on $OP_S$ such that $<_S \supseteq \bigcup_{i=1}^{n} <_i$.
- Let $p, q \in OP_S$, where $p$ and $q$ belong to distinct transactions, however access the same data object. If $p$ or $q$ is a write action, then either $p <_S q$ or $q <_S p$; we say, $p$ and $q$ are in conflict; if $p <_S q$ and $p$ and $q$ are in conflict, we write $(p, q) \in \text{conf}(S)$.

A schedule of $\mathcal{T}$ is a prefix of a history.\(^2\)

Conflict graph

The conflict graph of a schedule $S$ is given as $G(S) = (V, E)$, where $V$ is the set of transactions in $S$ and the set of edges $E$ is given by the conflicts in $S$: $T_i \rightarrow T_j \in E$, iff there are conflicting actions $p \in OP_i$, $q \in OP_j$ and $p <_S q$.

\(^2\)A partial order $L' = (A', <')$ is a prefix of a partial order $L = (A, <)$, if $A' \subseteq A$, $<' \subseteq <$, for all $a, b \in A'$: $a <' b$ if $a < b$, and for all $p \in A, q \in A'$: $p < q \Rightarrow p <' q$. 
A schedule/history of the two parallel debit/credit transactions.

The schedule is not serializable as its conflict graph is cyclic.

Serializability

- A schedule \( S = (\text{OP}_S, <_S) \) is *serial*, if for any two transactions \( T_1, T_2 \) appearing in \( S \), \( <_S \) orders all actions of \( T_1 \) before all actions of \( T_2 \), or vice versa.

- A schedule is called (conflict-)serializable,\(^3\) if there exists a (conflict-)equivalent serial schedule over the same set of transactions.

- A schedule \( S = (\text{OP}_S, <_S) \) is serializable, iff its conflict graph is acyclic.

\(^3\)We consider only conflict-serializability and therefore talk about serializability in the sequel, for short.