DB - HW14

18.1

Show that the two-phase locking protocol ensures conflict serializability and that transactions can be serialized according to their lock points.

Answer:

$$
\begin{aligned}
&Assume\ two-phase\ locking\ doesn't\ ensure\ serializability.\\
\therefore\ &There\ exists\ a\ set\ of\ transections\ \ T_0,\ T_1,\ \dots\ ,\ T_{n-1}\\
&that\ obey\ 2PL\ and\ produce\ non-serializable\ schedule.
\end{aligned}
$$

$$
\begin{aligned}
&A\ non-serializable\ chedule\ must\ have\ a\ cycle\ in\ its\ precedence\ graph.\\
\&\ &We\ should\ show\ that\ 2PL\ couldn't\ produce\ cycles\ like\ this.
\end{aligned}
$$

$$
\begin{aligned}
&Let's\ assume\ that,\ in\ the\ precedence\ graph,\ there\ exists\ a\ cycle\ like:\\
&\qquad T_0 \to T_1 \to T_2 \to\ \dots\ \to T_{n-1} \to T_0\\
&Assume\ LockPoint\ of\ T_i = \alpha_i,\ i \in [0,\ n],\ i\ \in\ \mathbb{N}
\end{aligned}
$$

$$
\begin{aligned}
&\because\ (T_i \to T_j) \Rightarrow \alpha_i < \alpha_j\\
&\therefore\ \alpha_0 < \alpha_1 < \alpha_2 <\ \dots\ < \alpha_{n-1} < \alpha_0\ \ Obviously,\ \alpha_0 < \alpha_0\ is\ a\ contradiction.\\
&\therefore\ 2PL\ cannot\ produce\ non-serializable\ shedule.
\end{aligned}
$$

$$
\begin{aligned}
&\because\ (T_i \to T_j) \Rightarrow \alpha_i < \alpha_j\\
&\therefore\ Transaction\ in\ lock\ point\ order\ is\ also\ in\ topological\ order.\\
&\Rightarrow\ Transactions\ can\ be\ serialized\ according\ to\ its\ lock\ points.\\
&\Rightarrow\ Two-phase\ locking\ protocol\ ensures\ conflict\ serializability\\
\&\ \ &Transactions\ can\ be\ serialized\ according\ to\ their\ lock\ points.
\end{aligned}
$$

18.2

Consider the following two transactions:

$$
\begin{aligned}
T_{34}: &\ read(A);\\
&\ read(B);\\
&\ if\ A\ =\ 0\ then\ B :=\ B\ +\ 1;\\
&\ write(B).
\end{aligned}
$$

$$
\begin{aligned}
T_{35}: &\ read(B);\\
&\ read(A);\\
&\ if\ B\ =\ 0\ then\ A :=\ A\ +\ 1;\\
&\ write(A).
\end{aligned}
$$

Add lock and unlock instructions to transactions T_{31} and T_{32} so that they observe the two-phase locking protocol. Can the execution of these transactions result in a deadlock?

Answer:
Lock & Unlock instructions are as follows:

$$
\begin{aligned}
T_{34} : & \ Lock-S(A); \\
& \ read(A); \\
& \ Lock-X(B); \\
& \ read(B); \\
& \ if\ A = 0\ then\ B := B + 1; \\
& \ write(B); \\
& \ Unlock(A); \\
& \ Unlock(B).
\end{aligned}
$$

$$
\begin{aligned}
T_{35} : & \ Lock-S(B); \\
& \ read(B); \\
& \ Lock-X(A); \\
& \ read(A); \\
& \ if\ B = 0\ then\ A := A + 1; \\
& \ write(A); \\
& \ Unlock(B); \\
& \ Unlock(A).
\end{aligned}
$$

The execution of these could result in a deadlock, the case is as follows:

T_{31}	T_{32}
Lock-S(A)	
	Lock-S(B)
	read(B)
read(A)	
Lock-X(B)	
	Lock-X(A)

18.7

Consider a database system that includes an atomic increment operation, in addition to the *read* and *write* operations. Let *V* be the value of data item *X*. The operation

$$increment(X)\ by\ C$$

sets the value of *X* to *V+C* in an atomic step. The value of *X* is not available to the transaction unless the latter executes a *read(X)*.

Assume that increment operations lock the item in increment mode using the compatibility matrix in Figure 18.25.

a. Show that, if all transactions lock the data that they access in the corresponding mode, the two-phase locking ensures serializability.

b. Show that the inclusion of increment mode locks allows for increased concurrency.

Answer:

a.

$\because$ *if 2 transactions have an $I-$mode lock on the same item,*
the increment opertaions can be swapped.
& if any pair of conflicting operations is serialized in the order of
lock point of corresponding transactions
then we can obtain an serialized sequence.

$\therefore$ *if all transactions lock the data that they access in the corresponding mode,*
the two $-$ phase locking ensures serializability

b.

- The **increment** lock mode allow multiple incrementing transaction to take the same lock simultaneously and it's compatible with itself, thereby improve the concurrency.
- If we don't take this mode, each transaction that wants to increment the the data item should take an exclusive mode. And an exclusive lock being incompatible with itself would prolong the waiting time and hinder the overall progress of the concurrent schedule.
- Thus, the **increment** lock mode allows for increased concurrency by increasing the true entries in the compatibility.

18.18

Most implementations of database systems use strict two-phase locking. Suggest three reasons for the popularity of this protocol.

Answer:

1. Two-phase locking protocol is relatively simple to implement.
2. Two-phase locking protocol usually allows an acceptable concurrency.
3. Because its cascadelessness, two-phase locking protocol imposes less rollback.