数据库week14

18.1

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

We proof it by contradiction.

Assume **two-phase locking protocol** doesn't ensure conflict serializability and a non-serializability transaction sequence $T_1, T_2, T_3 \ldots\ldots T_N$which obeys the **two-phase locking protocol**.

If a sequence is non-serializable,we can draw a precedence graph to represent it. Without loss of generality , we assume it$T_1 \rightarrow T_2 \rightarrow T_3 \ldots\ldots \rightarrow T_N \rightarrow T_1$.

According to above,we can infer that the lock points of these transactions LP_i(The Lock point of i_{th} transaction) should obey the rule(If $T_i \rightarrow T_j$,then $LP_i < LP_j$).

So we get $LP_1 < LP_2 < LP_3 < \ldots\ldots < LP_N < LP_1$

Since of conflict of $LP_1 < LP_1$,we can infer the assumption is wrong, which means **the two-phase locking protocol** ensures **conflict serializability**

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.

	S	X	I
S	true	false	false
X	false	false	false
I	false	false	true

Figure 18.25 Lock-compatibility matrix with increment lock mode.

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

The operation **increment** is same to operation **Write**. So the operation sequence should be sort by Lock points too. Because of **15.1**'s proof, it can be proof similarly that two-phase locking ensures serializability.

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

The increment lock mode being compatible with itself allows multiple incrementing transactions to take the lock simultaneously, thereby improving the concurrency of the protocol. While in other situations, the **false** truth value can guarantee the data consistency for increased concurrency.

18.18

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

- **Guarantees serializability** . It means that it can serialize concurrent transactions into a sequence that is equivalent to some serial (non-concurrent) execution of those transactions.
- **Ensures recoverability.** In the event of system crashes, power outages, or other types of failures, a recoverable schedule enables the database system to recover and bring the database back to a known good state. This is often achieved through the use of logs that record the changes made by each transaction, which can be used to undo or redo operations as necessary.
- **Avoids cascading roll-backs.** Each transaction should be an independent unit of work. Avoiding cascading roll-backs ensures that the failure of one transaction does not unduly affect others, maintaining the atomicity and isolation of individual transactions.