Chapter 7.9 Concurrency Control by Validation

*Definition:*

* *Validation Method is another Optimistic Concurrency Control*, among which, we enable Transactions do not need to lock Database Element, but when in the appropriate time, we need to check *whether Transactions can be ran in Serializable Way.*

*Difference:*

* *For Validation, Schedule needs to maintain all records which are used to check what all Active Transactions are doing.*
* *For Time Stamp, Schedule needs to save Write Time and Read Time of all Database Elements.*

*Process:*

When Transaction is trying to write value for Database Element, it needs to go through ‘*Validation Period*’, at this time, it *compares all Read and Written Database Element Collection with Write Collection of other Active Transactions. If there do exist Non - Realizable Behavior Risk, then the Transaction should be roll back.*

Chapter 7.9.1 Confirm Structure based on Validation

*Definition:*

When Validation is used on Concurrency Control Mechanism, then for each Transaction, *Schedule needs to be informed Database Element Collection which needs to be written and read in Transaction*, and they are *Read Collection RS(T)* and *Write Collection WS(T)*. The Transaction can be divided into three phases to execute:

1. *Read - In the first phase, Transaction reads all Database Elements Collection which are waiting to be read. Transaction needs to calculate all values which is waiting to be written to its Local Address Space.*
2. *Validation - In the second phase, Schedule needs to compare the Collection of all Database Elements which are waiting to be read and written with all other Transactions to confirm the effectiveness of the current Transaction.* This process would be described later. *If the validation process failed, then the Transaction needs to be Rolled back; Otherwise, the Transaction goes to the Third Step.*
3. *Write - In third phase, Transaction needs to write all Database Elements from Collection which are waiting to be wrote into the Database System.*

To be directly, each Transaction validates in just one moment. Therefore, the Schedule which is based on Validation would have an assumed Serializable Sequence for the Transaction, and it would go to check whether this Transaction Behavior is effective or not to decide whether this Transaction is effective or not.

*In order to support the decision about whether the Transaction is effective or not, the Schedule needs to maintain three collections:*

1. *START* - *The Transaction Collection which are already start but has not finished Validation.* For each Transaction in the Transaction Collection, Schedule needs to maintain *START(T)*, which is to say, *the start time of Transaction*.
2. *VAL (Validation)* - *The Transaction Collection which has already confirmed the effectiveness but has not stepped into third step.* For each Transaction T in the Collection, the Schedule needs to maintain *START(T) and VAL(T)*, which is to say, Validation Time of Transaction T. Attention that, *VAL(T) is the Execution Time of Transaction T* which has already assumed in the Serializable Sequence.
3. *FIN (Finish)* - *The Transaction has already finished the third step.* *For all Transactions, Schedule needs to record START(T), VAL(T) and FIN(T), which stands for the finishing time of Transaction T.* In principle, the Collection would increase, but just as we see, for random active Transaction U ( Which means that any Transaction U in the START or VAL. ), the Transaction T may satisfy the condition FIN(T) < START(T), then we do not need to remember the Transaction T. The Schedule can delete FIN Collection periodically, in case this may enlarge and exceed it’s size.

Chapter 7.9.2 Validation Rule

Chapter 7.9.3 Comparative among three kinds of Concurrency Control