Chapter 7.7 Tree Protocol

*Introduction:*

In this Chapter, mainly related with *Data of Tree Module*. However, the Tree Node has not formed the Level that contains Relation. *To some extant, Database Element are non - intersect Data Segment and the only way to reach the node is through its Parent Node.* *B - Tree* is one of the important example of this kind of Data.

Chapter 7.7.1 Lock Motive Based on Tree (Need to Re - read)

*Introduction:*

B - Tree Index treats single node (block) as the lockable Database Element. Tree Node is a correct Lock Granularity, since there has not any benefits to treat the whole Database Tree as one Database Element, also it prevents the development from happening through using the Lock Infrastructure which has been introduced before.

Since if we use the standard Shared Lock, Exclusive Lock, and Update Lock, such Locking Collection, and using the Two - Phase Lock, then it is impossible to use B - Tree structure concurrently. The reason is that each Transaction that used Index must start from Locking B - Tree Root Node. If Transaction is 2PL, then it can not release Root, till the Transaction releases B - Tree Root Node.

*In principle, any insertion or deletion transaction may need to rewrite the Root node of B - Tree, actually each transaction needs one Update Lock of the Root Node, or when the Update Node can not be reached, then it should get one Exclusive Lock. So, every time, there should only one non - read Transaction can access the B - Tree.*

*In the most of situations, we can conclude that B - Tree Node would not be rewritten, even we insert or delete the Tree Node.*

Normally, we can not release the Lock of the root since release the lock early would violate 2PL and we can not make sure whether schedule of several Transactions is Serializable. The Solution is to use the specialized protocol. This protocol violates the 2PL, but the fact that the access of Database Element must from top to the end is used to guarantee Serializable Transactions.

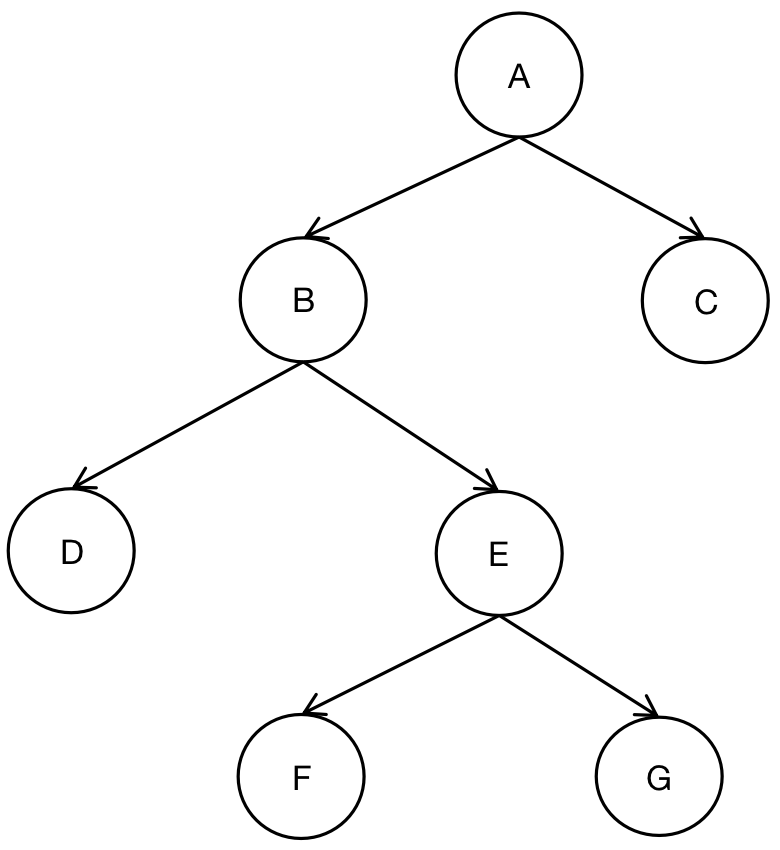
Chapter 7.7.2 Rule that Access to Data For Tree Structure  
*Definition:*

The constraint to the Lock has constructed the Tree Protocol. Here, we assume that there has only one type of lock which is li(X), however this kind of thinking pattern can be generalized to other kind of Locking Collection. We assume that Transaction is consistent, and Schedule is legal ( which means that Schedule can and only can grant Lock on Tree Node when there has no conflict for the granted Locks ), but there has no 2PL requirement on Transaction.

1. *The first lock of Transaction can be on Random Node of the Tree.*
2. *Only when Transaction gets Lock on its Parent Node then it can get subsequent lock on its Child Nodes.*
3. *The Node can be released at any time.*
4. *The Transaction can not get the Lock again on Node where Transaction has released Lock before, even if it still gets Lock of Parent Node.*

*Example:*

The image below gives the Level Structure of the node and the image below gives all behaviors of three Transactions.



|  |  |  |
| --- | --- | --- |
| Transaction T1 | Transaction T2 | Transaction T3 |
| l1(A); r1(A); |  |  |
| l1(B); r1(B); |  |  |
| l1(C); r1(C); |  |  |
| w1(A); u1(A); |  |  |
| l1(D); r1(D); |  |  |
| w1(B); u1(B); |  |  |
|  | l2(B); r2(B); |  |
|  |  | l3(E); r3(E); |
| w1(D); u1(D); |  |  |
| w1(C); u1(C) |  |  |
|  | l2(E) is declined; |  |
|  |  | l3(F); r3(F); |
|  |  | w3(F); u3(F); |
|  |  | l3(G); r3(G); |
|  |  | w3(E); u3(E); |
|  | l2(E); r2(E); |  |
|  |  | w3(G); u3(G); |
|  | w2(B);u2(B); |  |
|  | w2(E); u2(E); |  |

*Analysis:*

Transaction T1 starts from root A, continue going down to B, C, and D. Transaction T2 starts from node B and tries to move to E, but the initial request is declined by the conflict, since the the Lock E has already been applied by Transaction T3. Transaction T3 starts from E and moves to F and G. Attention that, Transaction T1 is not 2PL Transaction, since the Lock of Transaction A has been released before applying the Lock of Database Element D. Similar, Transaction T3 is not 2PL transaction, but Transaction T2 is 2PL.

Chapter 7.7.3 Reason why Tree Protocol Takes Effect

*Definition:*

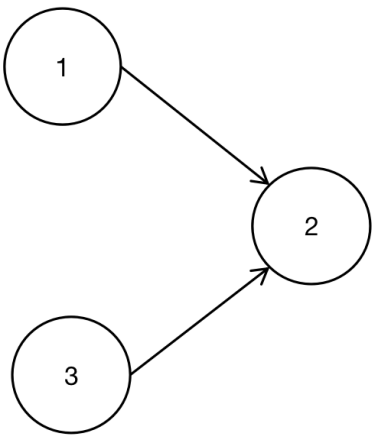
Transactions that are related in Tree Protocol must contains one Serial Sequence in one Schedule. We can define the sequence as below: if in the Schedule S, Transaction Ti and Tj have related to one common node, but Ti locks this node first, then we would say that Ti < s Tj.

*Example:*

1. We find that Transaction T1 and T2 all locked Database Element B, and Transaction T1 locked B first. So T1 < s T2.
2. We also find that T2 and T1 all locked Database Element E, but T3 locked E first. So we can conclude that T3 < s T2. However, there has no sequence between Transaction T1 and T3, since they have not the common locked Database Element.

|  |  |  |
| --- | --- | --- |
| Transaction T1 | Transaction T2 | Transaction T3 |
| l1(A); r1(A); |  |  |
| l1(B); r1(B); |  |  |
| l1(C); r1(C); |  |  |
| w1(A); u1(A); |  |  |
| l1(D); r1(D); |  |  |
| w1(B); u1(B); |  |  |
|  | l2(B); r2(B); |  |
|  |  | l3(E); r3(E); |
| w1(D); u1(D); |  |  |
| w1(C); u1(C) |  |  |
|  | l2(E) is declined; |  |
|  |  | l3(F); r3(F); |
|  |  | w3(F); u3(F); |
|  |  | l3(G); r3(G); |
|  |  | w3(E); u3(E); |
|  | l2(E); r2(E); |  |
|  |  | w3(G); u3(G); |
|  | w2(B);u2(B); |  |
|  | w2(E); u2(E); |  |

We conclude Final Sequence as picture below:



According to the priority picture, there does not exist any cycle, so we can claim that random topology sequence could be one equivalent serial schedule. For example, (T1, T2, T3) or (T3, T1, T2) are one equivalent serial schedule.The reason is that all nodes in this serial sequence that are triggered as the same sequence as their original schedule sequence.

In order to understand why the priority picture is always non - cyclic, there has one fact below:

* *If two Transactions have several elements that waited to be Locked, then the Lock sequence would be the same as the Lock Sequence.*

Consider the Random Transaction Collection T1, T2, ..., Tn, they obeys Tree Protocol and locks some nodes according to Schedule S.

At first, Transactions that have locked Tree Root, have been doing something according to the sequence, and the rule that we just find:

* *If Transaction Ti locked Tree Root before Tj, then Transaction Ti locked each node that Ti and Tj need to lock, which is to say Ti < s Tj, but not Tj < s Ti.*

*Example:*

Assume there have 10 Transactions including T1, T2, ..., T10, and among all these Transactions, T1, T2, and T3 locked Root sequentially. Also, we assume that Root has two Sub - Tree, the first one is locked by T1 to T7, the second one is locked by T2, T3, T8, T9 and T10. Assume that the first Sub - Tree consists of (T4, T1, T5, T2, T6, T3, T7); attention that this sequence needs to include T1, T2, and T3. Also Assume that Serial Sequence of the second Sub - Tree consists of (T8, T2, T9, T10, T3). The sequence to lock Root is the same as Sequence that they locked Root. The Serial Sequence is restricted as the image below.

