Chapter 7.4 Locking System with Multi - Locking Mode

*Introduction:*

Main problem in Simple Lock System introduced before is that even Transaction T wants to read Database Element X but not read it, then it should request Lock on parameter X.

We can not escape getting the Lock. If not getting lock, then Non-Serializable Behavior would happen when another Transaction would write X when Transaction T is active.

*Outline:*

* *Two different complex Lock which is used to read (Shared Lock and Read Lock) and another is used to write (Exclusive Lock and Write Lock).*
* *Shared Lock can be upgrade into the Exclusive Lock.*
* *Increment Lock and can be used to write the Database Element incrementally. Increment Lock can be swapped.*
* *Compatibility Matrix is used to present what kind of Lock can be assigned on Database Element when there already has other locks already assigned on current Database Element.*

Chapter 7.4.1 Sharing Lock and Exclusive Lock

*Definition:*

When we write, Lock is much stronger than the Lock when we read, since the Database Element forbids reading and writing when we write. Let’s consider Lock Schedule which uses two different types Locks - Shared Lock and Exclusive Lock.

* *For any Database Element X, if there has one Exclusive Lock on X - which is used to Write on the Database Element X.*
* *For any Database Element X, there has no Exclusive Lock but there has random number Shared Lock on it - which is used to Read on Database Element X.*

*Representation:*

* *sli(X) is used to present ‘Transaction Ti applies one Shared Lock’ on Database Element X.*
* *xli(X) is used to present ‘Transaction Ti applies one Exclusive Lock’ on Database Element X.*
* *ui(X) is used to present ‘Transaction Ti releases the Lock’, which means Transaction release whatever lock it has on itself.*

Consistency of Transaction, 2PL of Transaction and Legality of Schedule have all been included in the respective Relation, here we can include all these as:

1. *Consistency of Transaction - Can not write only after has Exclusive Lock, and if has not got some locks, then can not read.*
2. *Before read behavior ri(X), there should have sli(X) or xli(X), and between them, there should has no ui(X).*
3. *Before write behavior wi(X), there should has xli(X), and between them, there should has no ui(X).*
4. *2PL of Transaction - Lock should before Release. More precisely, for random Database Element Y, uj(Y) should not appear right before any sli(X) or xli(X).*
5. *Legality of Transaction - One element can only be locked exclusively or can be shared locked, but can not have both. To be more precisely:*
6. *If there exists xli(X) in the Schedule, then for j != i, afterwords there should have not xlj(X) or slj(X), unless there has ui(X).*
7. *If there exists sli(X) in the Schedule, then for j != i, afterwords there should have not xli(X), unless there has uli(X).*

*( Attention, owning Shared Lock and Exclusive Lock is allowed on one Database Element, as long as the conflict would not happen with other Transactions. )*

*Example:*

Let’s consider to use Shared Lock and Exclusive Lock, below is the possible Schedule about two Schedules:

*T1: sl1(A); r1(A); xl1(B); r1(B); w1(B); u1(A); u1(B);*

*T2: sl2(A); r2(A); sl2(B); r2(B); u2(A); u2(B);*

Transaction T1 and T2 both read Database Element B, but only Transaction T1 writes B. Both do not write A. The Transaction Sequence is as table below.

|  |  |
| --- | --- |
| *Transaction T1* | *Transaction T2* |
| *sl1(A); r1(A);* |  |
|  | *sl2(A); r2(A);* |
|  | *sl2(B); r2(B);* |
| *xl1(B) is declined.* |  |
|  | *u2(A); u2(B);* |
| *xl1(B); r1(B); w1(B);* |  |
| *u1(A); u1(B);* |  |

*Explanation:*

Transaction T1 requests the Shared Lock on variable A, then Transaction T2 requests the Shared Lock on the variable A and B. Now Transaction T1 asks one Exclusive Lock on on variable B but this requirement would be declined, since Transaction T2 has already applied the Shared Lock in B. Transaction T1 needs to wait till the Shared Lock has been released. Then Transaction T1 can continue to proceed.

*The Schedule below is Conflict Serializable, the equivalent schedule is (T2, T1), since Transaction T2 releases Lock before Transaction T1.*

|  |  |
| --- | --- |
| *Transaction T1* | *Transaction T2* |
|  | *sl2(A); r2(A);* |
|  | *sl2(B); r2(B);* |
|  | *u2(A); u2(B);* |
| *sl1(A); r1(A);* |  |
| *xl1(B) is declined.* |  |
| *xl1(B); r1(B); w1(B);* |  |
| *u1(A); u1(B);* |  |

Chapter 7.4.2 Compatibility Matrix

*Definition:*

If we use several kinds of Lock Type, then Schedule may need one known Strategy about when to grant Lock for Transaction when already have other types of Locks. *Compatibility Matrix is one simple method which describes Lock - Management Strategy. Each Lock method has one line and one row. The row means that Lock which another Transaction already has been granted for Database Element X, while column means the application Transaction Type for Database Element X.* The rule can be described as:

* *The C type Lock is only granted when each types of Lock that granted on other Transactions on Database Element X has corresponding column R, the value of which on the C type row is ‘YES’.*

*Example:*

Table 7 - 16 is the Compatibility Matrix which includes Shared Lock and Exclusive Lock. About the Column of Shared Lock means that if there has only Read Lock on the Database Element, then we can grant Shared Lock on it. Only when there has no other Locks that has been granted, then we can grant Exclusive Lock on it.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Applied Lock | |
| Shared Lock | Exclusive Lock |
| Locks that already have been granted | Shared Lock | YES | NO |
| Exclusive Lock | NO | NO |

Chapter 7.4.3 Upgrade Lock

Chapter 7.4.4 Update Lock

Chapter 7.4.5 Increment Lock