Chapter 8.2 Deadlock

*Definition:*

*Deadlock is kind of Database Status which belongs to Currency Transactions because of resource competition: Each Transaction among several transactions is waiting for the resource which has been occupied by other Transactions, therefore each Transaction can not any progression.*

*Scenario:*

1. Even normal Operation of Two - Phase Locking (2PL) Transaction may cause Deadlock. The reason is that one Transaction has locked the resource which has been locked by another Transaction.
2. When the Lock has been updated from the Shared Lock into Exclusive Lock, then this may cause Deadlock. The reason is that, each transaction want to upgrade from the Shared Lock into Exclusive Lock of the Common Element.

*Method to Deal with Deadlock:*

*There have two methods to deal with Deadlock.*

* *The first one is to detect Deadlock and repair it.*
* *The other one is to manage Transaction, even Deadlock may not happen forever.*

Chapter 8.2.1 Timeout Deadlock Detection

*Principle:*

When exists Deadlock, then it is possible to repair all related Transactions and make them continue to execute. Since, one of the Transaction needs to abort and restart again.

*Method:*

*The simplest method to solve Deadlock is to utilize Timeout. We need to limit the time of Active Transaction, if the Transaction has exceeded this time, then this Transaction needs to be rolled back.*

*Supplement:*

Attention that, when one Deadlock Transaction has exceeded time and finished rolled back, then this Transaction needs to release its lock and all other Resources. Therefore, some other Transactions may finish before they reach the time limit.

Chapter 8.2.2 Waiting Image

*Principle:*

By using the Waiting Image to detect the Deadlock problem since one Transaction is waiting for the Lock occupied by another Transaction. The Waiting Image represents which Transaction is waiting for locks occupied by other Transactions. The Waiting Image can be used to detect Deadlock after it has been formed. Also, the Waiting Image can be used to prevent the formation of Deadlock. We assume that the latter one happens, in any time, we need to maintain the Waiting Image and decline the formation of Cycle.

*Definition:*

Waiting Image maintains Holding Lock and Waited Lock for each Transaction to be one Node. For each Node/Transaction T and U, if exists some Database Element which makes:

1. The Lock that Transaction U keeps on Database Element A.
2. The Lock that Transaction T waits for Database Element A.
3. Unless Transaction U release its Lock on A first, otherwise Transaction T can not get corresponding Lock.

As long as these three conditions have been satisfied, then there exists one arch from Transaction T to U.

* If there does not exist cycle in the Waiting Image, then each Transaction would finish at last. At least, one Transaction does not wait for another Transaction, then this Transaction can be finished.
* If there has cycle in the Waiting Image, then any Transactions in Cycle can not get progression, therefore exists the Deadlock.At that time, one Strategy to avoid Deadlock is to rolled back the Transaction which requirement may cause the cycle exists.

*Example:*

Assume that there exist four Transactions, each Transaction read one Element and write another Element:

T1: l1(A); r1(A); l1(B); w1(B); u1(A); u1(B);

T2: l2(C); r2(C); l2(A); w2(A); u2(C); u2(A);

T3: l3(B); r3(B); l3(C); w3(C); u3(B); u3(C);

T4: l4(D); r4(D); l4(A); w4(A); u4(D); u4(A);

We are using one simple Locking System. The table below is the most start part of the Schedule among four Transactions.

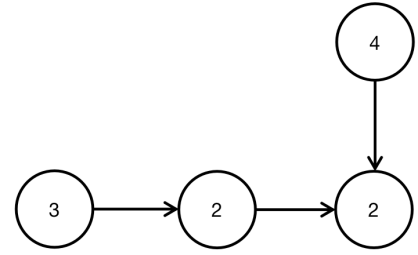
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Transaction T1 | Transaction T2 | Transaction T3 | Transaction T4 |
| 1 | l1(A); r1(A) |  |  |  |
| 2 |  | l2(C); r2(C); |  |  |
| 3 |  |  | l3(B); r3(B); |  |
| 4 |  |  |  | l4(D); r4(D); |
| 5 |  | l2(A) is declined. |  |  |
| 6 |  |  | l3(C) is declined; |  |
| 7 |  |  |  | l4(A) is declined; |
| 8 | l1(B); is declined; |  |  |  |

*Analysis:*

In the first four step, each Transaction gets the required Lock on which the Transaction wants to read.

In the fifth step, Transaction T2 tries to lock Database Element A, but since Transaction T1 has kept Lock on Database Element A, therefore the requirement has been declined. Therefore, Transaction T2 is waiting for Transaction T1. There would exists one Arch from Transaction T2 to Transaction T1.

In the sixth step, Transaction T3 tries to lock Database Element C, but since Transaction T3 has kept Lock on Database Element C, and the requirement has been declined. Similarly, in the seventh step, Transaction T4 required Lock which has been kept by Transaction T1. At that moment, the Waiting Image is shown as below, in this image there has no cycle.



Chapter 8.2.3 Prevent Deadlock through Sorting Database Element

Chapter 8.2.4 Deadlock Detection through Time Stamp

Chapter 8.2.5 Comparison of Deadlock Management Mechanism