Towards a Model of Concurrency to achieve Temporal Consistency in Long Duration Transaction Processing in Real Time Active Database System

D S Yadav ,A K Majumdar, R C Sarswat,D S Chauhan

Institute of Engineering & Technology , Lucknow
Indian Institute of Technology , Kharagpur

Abstract

This paper presents modeling of the transaction in the real time active database management systems. Active database with ECA rule has been found to provide an elegant framework to capture semantics of many real life applications. Real time database management system functions as a repository of data and provides storage and retrieval of data. Transactions, which update the database, are assigned deadlines with suitable slackness. Performance of real time database system mainly depends on how often the transaction misses their deadline. In the applications like situation assessment and tracking, trading and stock control needs actions to be taken in stringent time frame for the full benefit of the system. A long duration transaction takes relatively long time to complete its computation even without interference from other transactions. This makes long duration transaction vulnerable to failures and consequent recovery procedure is also complicated. In this paper, we will discuss how long duration transactions derives their deadlines and how ECA rules of active databases helps in processing long duration transactions.

1. Introduction

Conventional passive database management system are inadequate for time critical applications [1,2]. In time critical applications it is essential to take suitable action whenever an event occurs in the database. These events may be periodic, aperiodic or they may occur external to the database. Conventional database does not provide timely response to critical situations or they compromise modularity [2]. Conventional database system are called passive because they executes the transaction or query when they are explicitly requested by user or application program. For such time -constrained applications, it is important to monitor the events defined on the states of database. On occurrence of such events the conditions are evaluated, if conditions are found to be true then specific action plan is invoked subject to timing constraints. For such time-constrained application, correctness of result depends not only on the correctness of computation but also on the timeliness of the result. There exist two approaches [1,2] to handle such time-constrained applications in traditional database system. None of the approach is satisfactory because they do not guarantee to take action in correct time window. A Real time database management system [3] functions as a repository of the data and it provides the efficient storage and retrieval of data. However, RT database systems has an added burden of processing the transaction and meeting the deadlines. In Real time database system, timing constraints are defined by means of associating deadline with a job. More specifically the applications like obstacle detection and avoidance on the floor of an automated factory, radar tracking and recognition of the objects; involves gathering the data from the environment, processing the gathered data in context of the information retrieved from the environment in the past and subsequently invoke an action plan. This action plan must be executed meeting the deadline.

2. ECA Rules

Central to the Active database is the concept of Event Condition Action (E C A) Rule. ECA rules can be used for condition monitoring. Based on the application requirement, an event to be monitored can be categorized in to following three categories; Database Event e.g. insert, delete and update operation on database objects; Temporal Event e.g. absolute and relative time events; External Event e.g. the events detected outside the scope of DBMS.

An ECA rule of has the following form:

\[
\text{Rule} < \text{rulename} > \\
\text{On} < \text{Event Expression} > \\
\text{If} < \text{Condition Expression} > \\
\text{Then} < \text{Action Expression} > \\
< \text{rule mode} >
\]

Where Event expression is defined in the terms of observed database object events generated by the transaction. The Condition Expression defines the precondition for the execution of the action plan. The Action Expression defines a set of object operations to be executed as a as a part of the transaction The Rule Mode allows user the flexibility to control the coupling mode by making the placement of condition evaluation and action execution relative to the occurrence of event as an explicit part of the rule definition. These are called Immediate, Deferred and Detached or Decoupled mode [2].
3. Issues in long duration transaction processing

3.1 Long Duration Transactions
Long duration transactions take relatively longer time to complete their computation thus they are more vulnerable to failures. In cases of aborts of Long duration transactions, a large amount of work done by the transaction has to be undone [4] thus making recovery procedures in real time systems more complicated and a costly affair. Traditional notion of serializability as a correctness criterion turns out to be too restrictive and bottleneck for long duration transactions. In order to avoid these bottleneck different kinds of extended transaction models like Nested transaction [5], Sagas [6], Cooperative Transaction [7] has been proposed. These Transaction model supports relaxed correctness criterion and uses relaxed version of ACID (Atomicity, Consistency, Isolation and Durability) properties for concurrent execution of transaction.

3.2 Cooperative Transactions
An active transaction may interact with other concurrent transaction by making its changes in the database accessible to other concurrently running transaction provided they are allowed to cooperate. Thus transaction processing system requires a controlled cooperation among the concurrent transactions. This model provides higher degree of concurrency as the data objects locked by component transaction are released and are made available to cooperating transactions.

3.3 Processing of Long Duration Transaction
In the proposed model, processing of long duration transaction supports forward execution from one state of transaction to another state. Out of these state some are marked states. Marked states indicate that in case of aborts, committed base transactions are compensated only up to the marked state. Figure 1 shows forward execution of complex transaction from initial state Si to state Sk with Sj as one of the intermediate state. Let Sj be as marked state. In case of abort at state Sk, Compensating transaction will be executed to compensate the effect of forward execution from state Sk to state Sj.

4. System Model

4.1 Data Characteristic in RT Database System
A Real time consists of a controlling agent and environment. Controlling agent contains the transaction processing system. Controlling agent (Fig 2) gathers the data from environment or field by means of sensors. It is a centralized unit capable of capturing the data from sensors, computing the data stored in persistent database and provides data to actuators to control environment.

The state of environment as reflected from the data gathered from sensors must confirm the state reflected from the database. This needs to define temporal consistency.

4.2 Temporal Consistency
A data object D in real time database is defined by three attributes data value, validity interval and Observation time where

\[ D \text{[data value]} = \text{Current value of D} \]
\[ D \text{[Validity Interval]} = \text{Length of Time when D is considered to have absolute validity.} \]
\[ D \text{[Observation Time]} = \text{Time of observation relating to D.} \]

Current value of data object becomes inconsistent after time \( t \), where \( t = D \text{[Validity Interval]} + D \text{[Observation Time]} \). Similarly, A set of such data object may be used to derive a new data which has its own validity interval. Validity Interval of derived data is obtained from the validity interval and time of observation of the entire data object being used to derive new data as shown in figure 3.
Figure 3 shows the validity interval of Data object D1, D2 and D3. \(<t1,t2>, <t3,t4>, <t5,t6>\) are validity interval of data object D1, D2 and D3. Time of observations of relating to data objects D1, D2 and D3 are t1, t3 and t5 respectively. Any data derived from the data objects D1, D2 and D3 will have validity interval denoted by T. Therefore all the computations done on temporal data has temporal validity. This further imposes timing constraints on the transactions.

4.2 Transaction Types

This section presents concept of complex transaction type and base transaction type in object oriented database management system.

4.2.1 Base Transaction Type

A base transaction type is a collection of database object operation, which has to be executed as an atomic transaction. A base transaction is a fired instance of base transaction type.

4.2.2 Complex Transaction Type

A set of base transactions, a set of detached mode ECA rules and the state transition model of complex transaction type specifies a complex transaction. The complex transaction may be viewed as a collection of related base transaction types, which are triggered as consequence of firing detached mode ECA rule. The state transition model of complex transaction type may define the execution sequences of complex transaction type. A complex transaction is a fired instance of complex transaction type.

4.3 Compatibility Among Locks

Base transaction releases the lock within the scope of parent complex transaction. This process is known as inheriting the lock by parent complex transaction. This is similar to the concept of inheriting the locks as in Moss locking protocol [5]. Locking mechanism as proposed in QUAD-LOCK concurrency control scheme [8,9] for transaction management in active database management system is used in this model. These scheme uses four modes of locks, denoted as shared (S), exclusive (E), relative shared (RS) and relative exclusive (RE). Shared and exclusive locks have similar meaning as in two-phase locking protocol. At the commit of base transaction the locks acquired by the base transactions are inherited by the complex transaction. Shared and exclusive locks are modified to relative shared (RS) and relative exclusive (RE) respectively at the time inheriting the locks. In figure 4 a symbol Y means that lock requested by the base transaction BT\(i\) on data object X is compatible with the lock held by base transaction BT\(j\) or by complex transaction CT\(k\). The symbol N means that lock requested is denied. Therefore for the successful completion of complex transaction CT\(k\) the lock requested by the base transaction BT\(i\) must be granted.

Let the base transaction BT\(i\) be subtransaction of the complex transaction CT\(i\) and the complex transaction CT\(k\) be in the state S.

\[
\text{IF } \text{COOP}(CT_k, S) \in \text{TYPE}(CT_i) \text{ THEN C=Y Else C=N}
\]

Thus if the complex transaction type CT\(k\) enlists the complex transaction type of CT\(i\) on the state S as a cooperating transaction types then the lock requested by the base transaction BT\(i\) may be granted.

5. Transaction Scheduling and Concurrency Control

This section presents priority driven transaction scheduling and concurrency control scheme for managing the execution of concurrent long duration transaction. In our model, long duration transaction is modeled as a complex transaction, condition -action part of the ECA rule fires sub transaction called base transactions. Thus the fired instances of complex transaction forms a transaction tree of height two. A typical diagrammatic representation of a complex transaction is shown in the figure 5.

Let T be a complex transaction in the active state. \(t_1\) is the time when its execution started. \(t_2\) is the deadline associated with the complex transaction. This means that its computation must be finished before time \(t_2\) for the full benefit of the system.

Length of the time within which T must complete computation (\(\tau\)) = \(t_2 - t_1\)

Let s and c are the estimated computation time and the slack time associated with the complex transaction T. Therefore for the successful completion of complex transaction T following inequality must hold.
\[ s + c \leq \tau \] .................................(1)

Let \( m = \) Total no. of Base transaction to be fired by the complex transaction \( T \). Say these base transactions are \( B_1, B_2, \ldots, B_m \) respectively. Following condition must be satisfied for the firing of base transaction.

\[ \tau \geq \sum_{i=1}^{m} (c_i + s_i) \] .................................(2)

where \( s_i = \) slack time for base transaction \( B_i \)
\( c_i = \) Computation time of base transaction \( B_i \)

5.1 Modifications in the slack time

Suppose a base transaction \( B_i \) takes the time more than its computation time \( c_i \) and its slack time \( s_i \) then it affects the slack time of all the base transaction which has to be executed in order to complete complex transaction. I.e.

If time taken by base transaction \( (T_{bi}) \geq c_i + s_i \), then new slack time for \( k \)th base transaction may be modified as

\[ s_k = s_k - \frac{(T_{bi} - (c_i + s_i))}{(m - I)} \]

Where \( m \) is the no. of base transaction to be fired by complex transaction and \( I \) is number of base transaction fired so far. Thus the extra time taken by the base transaction to complete its execution will be compensated by reduction in slack time of the remaining base transaction.

Similarly, if a base transaction \( B_i \) finishes its computation before its deadline then the slack time of remaining base transactions may be increased. i.e.

if \( (T_{bi}) < c_i + s_i \), then new slack time for \( k \)th base transaction may be modified as

\[ s_k = s_k + \frac{(c_i + s_i - T_{bi})}{(m - I)} \]

Where \( m \) is the no. of base transaction to be fired by complex transaction and \( I \) is number of base transaction fired so far.

5.2 Condition for Abort

Let \( n = \) total no. of complex transaction in the transaction processing system. Abortion of the complex transaction will be determined by the total remaining time to complete the execution and the time already lapsed in the processing. Suppose \( L \) is the length of time remaining to complete the execution of the complex transaction. Thus,

\[ L = \tau - \text{Time already lapsed in the computation} \]

If \( L \leq \sum_{i=1}^{I} (c_i + s_i) \) .................................(3)

then complex transaction must be aborted, as in this case the complex transaction will not be able to meet the deadline. Here \( I \) denotes remaining number of base transaction to be fired in order to complete the remaining computation of complex transaction.

5.3 Priority Inversion

In real time systems the scheduling of the transactions must be done on the basis of urgency and criticalness. A transaction higher in priority is given upper hand in gaining the access to the system resources. Relatively less number of transactions will be blocked in this model, because the lock acquired by the transaction may be shared by the concurrent transactions provided they are cooperating transactions. For determination of priority of the transaction, it has been suggested [10], that a combination of criticalness, deadline, slackness, amount of work already done may be used.

\[ \text{Priority} (\rho) = f (\lambda, D, s, L) \]

Where \( \lambda = \) criticalness of the transaction. More critical a transaction higher is its priority.
\( D = \) Deadline of the transaction. Earlier the deadline higher is its priority
\( s = \) Slackness of transaction. Tighter the slackness, higher is its priority.
\( L = \) Amount of work already done.

In the real time database systems, abortion of long duration transaction may be a costly affair and compensation procedure may be too complex. Thus a high weightage must be assigned to amount of work already done in order to give high priority to long duration transaction. A transaction should be aborted only if condition of abortion is satisfied. To ensure non-blocking of higher priority transaction by a lower priority transaction priority inversion [11] in real time database system is required. Solution to this problem is to hoist the priority of lock holder to that of requester. E.g.

Consider complex transaction \( T_1 \) and \( T_2 \).

\[ \rho(T_1) > \rho(T_2) \]

Let \( T_2 \) has acquired a lock for which \( T_1 \) is waiting. In such case, either of two options may be chosen.

a. Abort transaction \( T_2 \)
b. Allow \( T_2 \) to run at elevated priority equal to \( \rho(T_1) \)

Since first strategy is not desirable because it will be a costly affair to abort long duration transaction. Thus second strategy is preferred. With this strategy, rollback of the transaction may be restricted thereby saving the cost of execution of the compensating action.

6. Conclusions

Increasing the concurrency and maintaining the consistency has been two conflicting goals in the design of database systems. This problem is more of importance if the system is serving long duration transaction. Long duration transactions are more vulnerable to the failures. Failures of long duration transaction in the real time database system may have serious impact on the performance. Central to our model is concept of ECA rules. Active database allows taking action in the correct time window, if the event of interest has found to occur. In the proposed model blocking and abort of the transaction can be restricted because of the cooperation semantics. Since transactions are allowed to share the locks with the concurrent cooperative transactions, degree of
multiprogramming may be improved. Conditions for the abortion of transaction suggest that an active transaction should be aborted only in the case when there is no possibility to meet the deadline. In such cases the compensating action plan should be invoked. For scheduling of the transaction, technique uses does not suggest abort of the transaction. We have also presented the method to modify the slack time when a base transaction finishes its execution before the deadline. Similarly method to compute the penalty in terms of reduction in slack time has also been presented, if a base transaction misses the deadline.

References: