An Approach to Reduce Resource Contention while Scheduling Time Constrained Long Running Activity in Active Database System

D.S. Yadav  
U P Technical University  
Lucknow UP, India  
e-mail: divakar_yadav@rediffmail.com

R.C. Sarswat  
U P Technical University  
Lucknow UP, India  
e-mail: rcs@biet.ac.in

G. Chandra  
U P Technical University  
Lucknow UP, India  
e-mail: get_gchandra@yahoo.co.in

R. Agrawal  
Department of Computer Science  
Kettering University  
Michigan, USA  
e-mail: ragrawal@kettering.edu

Abstract
Active database system with ECA rule has been found to provide an elegant framework to capture the semantics of many real life applications. In active database system, the long running activities can be modeled as long duration transaction where a transaction may be viewed as collection of sub transactions. A Real Time Database Management System functions as a repository of the data and it provides the efficient storage and retrieval of data. However, real time database systems have an added burden of processing the transaction and meeting the deadline. Performance of real time database system largely depends on contention for resources. Higher the resource contention among the concurrent transaction, leads to high probability of slipping the deadlines. In this paper, we present how can we model long running activities and how the contention for resources may be reduced by exploiting cooperation semantics.

1. Introduction
Conventional passive database management systems are inadequate for time critical applications, because either they do not provide timely response to critical situations or they compromise the modularity [1]. Traditional database systems are called passive because they execute the transaction or query only when the user or application program explicitly requests them. Some of the time critical application developed with active databases are hospital information system [2], work flow management [3], trading and stock control [4]. For such time critical applications, it is necessary to monitor the conditions defined on the states of the database and whenever these conditions are found to be satisfied, an appropriate action is invoked subject to the timing constraint. In time critical operations, correctness of the result depends not only on logical correctness of computation or proper interleaving of various operations being done concurrently but also on the timeliness of the result [5]. In Active Databases, the transactions have ability to monitor the database objects and to take specific action when a event of interest is found to occur.

A real time database management system [6] can be considered 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 [7]. Active databases
have been found to provide a framework to capture the occurrence of database and external events and to provide timely response to these events [3]. In active database, transactions have ability to monitor the database events and on occurrence of such events, the defined conditions are evaluated. If these conditions are found to be true then specific action plan is invoked subject to timing constraints. These action plans may be invoked in separate transaction or within the triggering transaction.

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. The active databases provide the constructs to capture an event but do not guarantee timely processing of real time transaction as time constraints may not be explicitly defined. To cater to the time constrained real life application, it requires both active and real time characteristics[6]. Therefore, in such cases, a real time database system requires underlying active database.

Next we presents a model for transaction processing serving cooperative and long duration transaction and show how cooperation semantics is exploited to reduce resource contention.

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 [1,5]. 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:

Rule < rulename >
   On < Event Expression >
   If < Condition Expression >
   Then < Action Expression >
   < rule mode >

The 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.

3. Scheduling Long Duration Transaction

3.1 Long Duration Transactions

Long duration transactions take relatively longer time to complete its 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[8] 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 [9], Sagas [10], Cooperative Transaction [11] 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 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.

3.3 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 complex transaction may be made available to other concurrent transaction, if they are cooperating.

4. Transaction Management

4.1 Transaction Types

A base transaction type is a collection of database object operation, which has to be executed as an atomic transaction. A base transaction commits atomically to its parent complex transaction. A base transaction can not start its own base transaction. A Compensating base transaction is executed if the effect of committed base transaction needs to compensated. A base transaction is a fired instance of base transaction type.

Complex Transaction Type is defined as set of base transactions, a set of detached mode ECA rules and the state transition model of complex transaction type. 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 and a
state transition model. The state transition model of complex transaction type defines the execution sequences of complex transaction type. A complex transaction is a fired instance of complex transaction type.

In this scheme, the firing of complex transaction forms a tree of height two and all leaf nodes are base transaction. When the complex transaction enters into transaction processing system, it fires its base transaction. These base transactions acquire the lock as per two phase locking protocol and commits atomically to complex transaction. The complex transaction inherits the lock acquired by base transaction and conditional access to these lock data object is given to cooperative complex transactions. A typical diagrammatic representation of a complex transaction in running state is shown in the figure 1.

![Figure 1 Complex Transaction](image1)

**4.2 Abort of Transaction Full Abort**

A complex transaction needs to be fully aborted during its execution if a fired vital base transaction of its get aborted. In case of full abort compensating transactions are executed for each committed base transaction.

**Partial Abort**

A complex transaction needs to be partially aborted during its execution if a fired base transaction other then vital base transaction of its get aborted. In this case a compensating base transaction is executed to compensate the effect of forward execution of base transaction fired after previous vital base transaction. Essential difference between full and partial abort is that after a failure of vital base transaction in case of full abort, the complex transaction is aborted but in case of partial abort, a complex transaction may further proceed after executing compensating base transaction.

A partial abort may lead to full abort of complex transaction subject to temporal constraints.

**4.3 Scheme for Compensation**

Traditional notion of serializability is too restrictive for long duration transaction. To avoid this bottleneck, Saga[10] model of the transaction is proposed to represent long duration transaction, which support the relaxed correctness criterion. A Saga can be expressed as series of sub transaction or component transaction. These component transactions may be interleaved with other concurrently running component transaction. If any of the component transaction needs to be aborted, a compensatory action may be taken for committed component transaction. These transaction are called compensating transactions. Figure-2 present the forward execution of saga from initial state $S_i$ to state $S_k$ with state $S_j$ as one of the intermediate state. If state $S_i$ is a marked state and saga aborts at state $S_k$, then compensating transactions will be executed to compensate the effect of forward execution from state $S_i$ to state $S_j$. Further execution of this complex transaction may lead to the state $S_m$.

![Figure 2 Compensation](image2)

**4.4 Lock Acquiring Mechanism**

A base transaction is executed atomically i.e. either it is committed or aborted. A base transaction acquires and releases locks following conventional two phase locking. When a base transaction releases the lock, it does so within the scope of parent complex transaction. This process is known as inheriting the lock by parent complex transaction. This concept is similar to the concept of inheriting the locks as in Moss locking protocol [9].

A data object X may be locked explicitly by the base transaction or may be locked implicitly by complex transaction. The complex transaction maintains an interleaving set of complex transaction on X which is denoted as InterleaveSet(CT, X). This set contains those complex transaction types whose instances may interleave with the complex transaction on X i.e. can access data object X concurrently with the complex transaction. Locking mechanism as proposed in QUAD-LOCK concurrency control scheme[8] for transaction management in active database management system, is used in this model. This scheme uses four modes of locks, denoted as shared (S), exclusive (E), relative shared (RS) and relative exclusive (RE). Relationship among the four modes of locks is given below.

The base transaction acquires lock on data object in shared (S) or exclusive (E) modes. 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.

Compatibility among the locks is shown in figure 4.3.

In the Figure-3 a symbol Y means that lock requested by the base transaction BTi on data object X is compatible with the lock held by base transaction BTj or by complex transaction CTk i.e. lock requested may be granted. Similarly symbol N means that lock requested is denied. The symbol C denotes that lock may or may not be compatible with that of held by complex transaction CTk. The lock may be granted to the requesting base transaction subject to satisfaction of following condition.

- Let the base transaction BTi be sub-transaction of the complex transaction CTi and the complex transaction CTk be in the state S.

$$\text{IF COOP}(\text{CTk}, S) \in \text{TYPE}(\text{CTi}) \text{ Then C=Y Else C=N}$$

This means that if the complex transaction type CTk enlists the complex transaction type of CTi on the state S as a cooperating transaction types then the lock requested by the BTi may be granted.

<table>
<thead>
<tr>
<th>The Lock held by Base Transaction BTi</th>
<th>The Lock held by Complex Transaction CTk</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>E</td>
</tr>
<tr>
<td>S</td>
<td>Y</td>
</tr>
<tr>
<td>E</td>
<td>N</td>
</tr>
</tbody>
</table>

**Figure 3 Compatibility among Locks**

5. Conclusions

Long duration transactions are more vulnerable to failures as they take longer time to execute. It has been emphasized that a long running activity may be modeled as a transaction. 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. Failures of long duration transaction in the real time database system may have serious impact on the performance. In the proposed model blocking and abort of the transaction can be restricted because of the cooperation semantics. We have shown that a long duration transaction may be modeled as complex transaction and transactions are allowed to share the locks with the concurrent cooperative transactions. This semantics reduces the resource contention and a transaction waiting to lock a data object may proceed even if it is currently held by concurrently running cooperative transaction. This process therefore reduces blocking of transactions.

**References**