# Modeling Long Duration Transactions with Time Constraints in Active Database

DS Yadav[*], Rajeev Agrawal[#], DS Chauhan[*], RC Saraswat[*], AK Majumdar[§]

[*]*Institute of Engineering & Technology, U P Technical University, Lucknow-21, INDIA*
[#] *Department of Computer Science, Kettering University, Flint, Michigan, USA*
[§]*Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur-2, INDIA*

{divakar_yadav, pdschauhan}@rediffmail.com , ragrawal@kettering.edu, akmj@cse.iitkgp.ernet.in

## Abstract

*The applications like situation assessment, object tracking, trading & stock control, and workflow management require actions to be taken in stringent time frame for full benefit of the system. These applications also require identifying the occurrence of desired event, to take appropriate action when an event of interest is found to have occurred while maintaining timing constraints. Active database with ECA rule provides the mechanism to capture the different database events and provide timely response but do not guarantee timely processing of real time transactions because of their inability to express time constraints explicitly. Most of the time constrained real life application requires both active and real time characteristics. Long duration transactions with explicit time constraints are more vulnerable to failures and they are subject to heavy compensations in case of aborts. The paper intends to investigate the construct required for modeling of transactions with timing constraints, cooperation semantics and run time monitoring of these constraints in active database.*

## 1. Introduction

Traditional Database management systems are not considered fit for time critical applications [1]. A time-constrained application requires the suitable actions to be taken in correct time window whenever an event occurs in the database. These events may be periodic, aperiodic or may occur external to the database. For such time-constrained application, correctness of result depends not only on the logical correctness of computation but also on the timeliness of the result. A real time database management system [5] can be considered as a repository of data, which, provides the efficient storage and retrieval of data and has an added capability of processing transactions within the time constraints [13]. In real time database system, timing constraints are defined by means of associating deadline with a transaction [7,8]. Whereas, active databases have been found to provide a framework to capture the occurrence of database & external events and also provide timely response to these events.The basic constructs provided in Active Database for maintaining the integrity constraints are ECA (Event-Condition-Action) rules. The ECA rules are defined on the state of database and monitors the database events occurring due to transaction execution [3]. In the following sections, we discuss the construct required to model a long duration transaction with full and partial aborts and the mechanism for enforcement of temporal constraints, maintaining the deadlines and early detection of aborting transaction.

## 2. Transaction Model

**Complex Transaction Types**

Traditional notion of the serializability is too restrictive and a bottleneck for long duration activities [10,16]. A number of extended transaction model like cooperative transaction [6], SAGA [14], nested transaction [17] have been proposed which addresses long running activities. Saga is a long duration transaction model, which can be expressed as a series of base transactions. These base transactions may be interleaved with other concurrently running base transactions. A base transaction type may be defined as collection of database object operation, which has to be executed as an atomic transaction [12].

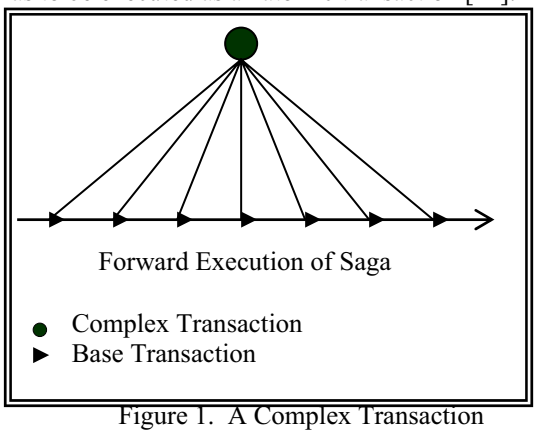


Figure 1. A Complex Transaction

A time constrained long duration activity may be expressed as a complex transaction that consists of a set of base transaction where timing constraints may further be



Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)
0-7695-2108-8/04 $ 20.00 © 2004 IEEE

IEEE COMPUTER SOCIETY

specified on base transaction. In this model, a fired instance of complex transaction forms a transaction tree of height two [4] as shown in Figure 1.

**Cooperation Semantics**

Long duration transactions are more vulnerable to failures and, therefore, in case of aborts, a large amount of work has to be undone [4].Reducing the resource contention has been one of the major design issues in real time systems. Reduced resource contention also minimizes blocking of transactions. It has been suggested that prior resource reservation is the best policy to meet the real time characteristic of a job [5,15]. We suggest that higher degree of concurrency along with minimized resource contention may be obtained by capturing cooperation semantics among the transactions[11]. In [6] a concept of Cooperative Transactions is proposed which uses relaxed version of ACID properties for concurrent execution. A cooperative transaction allows other blocking transaction to access the data object locked by it. The cooperation semantics reduce the contention for the resources. As shown in the fig. 2, if two concurrent complex transaction ($T_1$,$T_2$) request for same data object, and if $T_1$ succeeds in getting a lock on data object, same may also be granted to $T_2$ if they are cooperating. $R_1$ and $R_2$ in the fig.2 shows the requests for locks to the data object by transactions $T_1$, $T_2$ respectively. Grant to access the data object is given by $G_1$, $G_2$ to the transactions[20].

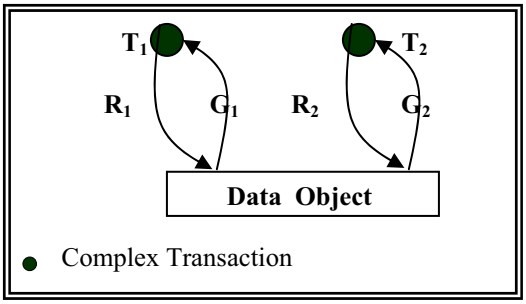


Figure 2**:** Cooperative Transaction

**Timing Constraints on Transaction**

Timing constraints imposes temporal restriction on system and its users. Two timing constraints namely performance constraints and behavioral constraints has been proposed for real time systems. Performance constraint limits the system itself, while behavioral constraints are applicable to users [2]. For the purpose of setting limits to the transactions, we have adopted following timing constraints:

***Maximum Timing Constraints***: The constraint imposes temporal restriction on maximum time between occurrences of two events.

***Minimum Timing Constraints***: The constraint sets the restriction on minimum time elapsed in occurrence of two events.

***Durational Constraints***: The restriction states that an event must occur for specified time duration.

Either one or a combination of these timing constraints may be applicable to complex transaction. Suppose 'A' and 'B' represent start and termination event of a complex transaction. Time[A] function return time of occurrence of event 'A' i.e. start time of complex transaction. Similarly, Time[B] represent the termination time of complex transaction. Maximum timing constraint imposes temporal restriction on maximum time elapsed between the occurrence of the events 'A' and 'B'. Let $T_{Max}$ denotes maximum time which may be elapsed between event 'A' & 'B'. To ensure temporal correctness, the Transaction Processing System (TPS) must execute complex transaction such that following condition is satisfied:

$$\text{Time [B]} - \text{Time [A]} <= T_{Max}$$

Similarly, if $T_{Min}$ denotes the minimum time which must be elapsed between the start and termination event of complex transaction and $T_{Due}$ denotes the time duration the complex transaction must be executed, then following conditions must be satisfied to ensure temporal correctness:

$\text{Time [B]} - \text{Time [A]} >= T_{Min}$ & $\text{Time [B]} - \text{Time [A]} = T_{Due}$

If any of the time constraints is violated then system may decide to roll back the transaction. The TPS must schedule the transactions in such a way that they should meet all the timing constraints specified with the complex transaction. In order to avoid slipping the deadline and violation of durational timing constraints, a complex transaction may be executed at elevated priority. Priority driven transaction scheduling [8,15,16] allows the transaction to change the priorities at run time according to the criticalness. A high priority transaction gets the resource earlier than low priority transaction. This semantics allows the transactions to meet the deadline even if they are at the verge of slipping the deadlines.

## 3. Enforcement of Temporal Constraints and slack time modifications.

Suppose a complex transaction (CT) fires 'm' base transactions (BT) $BT_1$, $BT_2$,…$BT_m$ as a consequence of forward execution. Let 'A' and 'B' are events of start and termination of complex transaction and execution of complex transaction start at time $t_1$ and it completes the computation at $t_2$. Therefore,

$$t_1 = \text{Time[A]} \ , \ t_2 = \text{Time[B]}$$

Let 'C' & 'S' are estimated computation time and slack time for complex transaction.

**Maximum Timing Constraints**

Maximum timing constraints may be specified by a parameter $\tau$ . It represents the maximum time length for completion of computation of a complex transaction. Therefore following inequality must be satisfied.

$$t_2 - t_1 <= \tau \ , \ S + C <= \tau$$



Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)
0-7695-2108-8/04 $ 20.00 © 2004 IEEE

IEEE COMPUTER SOCIETY

Similarly, if '$c_i$' and '$s_i$' are estimated computation time and slack time of $i^{th}$ base transaction ($BT_i$), the following inequalities must also hold.

$$\tau >= \sum_{i=1}^{i=m} ( ci + si )$$

Suppose the actual time and maximum allotted time for $i^{th}$ base transaction are $T_{Act}(BT_i)$, $T_{Max}(BTi)$ respectively. We propose that the maximum allowable time for computation ($\tau$) of complex transaction (CT) must placed to various base transactions such that,

$$\tau = \sum_{i=1}^{i=m} ( T_{Max}( BTi) )$$

where $T_{Max}(BTi) = ci + si$ .

Following conditions must be monitored during complex transaction execution to ensure temporal correctness.

1. $T_{Act}(BT_i) > T_{Max}(BTi)$ i.e. $BT_i$ takes larger time than maximum allowable time for computation.
2. $T_{Act}(BT_i) < T_{Max}(BTi)$ i.e. $BT_i$ takes lesser time then maximum allowable time for computation.
3. $T_{Act}(BT_i) = T_{Max}(BTi)$ i.e. $BT_i$ 's actual time to complete its computation is exactly equal to maximum allowable time.

In event of completion of computation of a base transaction if the conditions 1 & 2 of above are satisfied then slack time for remaining base transactions must be modified according to following policy. Consider first condition when a base transaction BTi's actual execution time exceeds the maximum allowed time $T_{Max}(BTi)$. The extra time taken by the base transaction to complete its execution should be compensated by reduction in slack time of the remaining base transactions [4]. Therefore, If $T_{Act}(BT_i) > T_{Max}(BTi,)$, then new slack time for $k^{th}$ base transaction may be modified as

$$s_k = s_k - (( T_{Act}( BT_i ) - (ci + si) ) / ( m - I ))$$

where 'm' is the total number of base transaction to be fired by complex transaction and 'I' is number of base transaction fired so far. Consider the second condition where a base transaction BTi finishes its computation earlier then its maximum allotted time $T_{Max}(BTi)$. The time saved in computation of $i^{th}$ base transaction must be added to slack time of remaining base transaction. Therefore, If $T_{Act}(BT_i) < T_{Max}(BTi)$ ,then new slack time for $k^{th}$ base transaction may be modified as

$$s_k = s_k + (( ci + si) - T_{Act}( BT_i )) / ( m- I ) )$$

The third condition mentioned above is considered ideal and no slack time modification is required.

**Minimum Timing Constraints**

Minimum timing constraints imposes restrictions on minimum time elapsed in two events. Let $T_{Min}$ denotes minimum time which must be elapsed between event 'A' and 'B'. Suppose ci is estimated execution time of $i^{th}$ base transaction (BTi) of complex transaction CT. Following inequalities must be satisfied for ensuring temporal correctness with minimum timing constraints.

$$T_{Min} <= \sum_{i=1}^{i=m} ( c_i )$$

Suppose $T_{Act}(BT_i)$ is the actual time taken by $i^{th}$ base transaction and its minimum execution time is $T_{Min}(BTi)$. There exist following three conditions to be monitored,

1. $T_{Act}(BT_i) > T_{Min}(BTi)$ i.e $BT_i$ takes larger time than minimum allowable time for computation.
2. $T_{Act}(BT_i) < T_{Min}(BTi)$ i.e $BT_i$ takes lesser time than minimum allowable time for computation.
3. $T_{Act}(BT_i) = T_{Min}(BTi)$ i.e. $BT_i$'s actual time to complete its computation is exactly equal to minimum allowable time.

First condition is desirable one, as base transaction execution takes more time than its minimum allowable time. Similarly, the third condition is also desirable as a $BT_i$ 's actual time to finish its computation is exactly equal to minimum allowable time. In second case where $BT_i$ takes lesser time than minimum allowable time for computation, the execution of $BT_i$ must be delayed until following is satisfied,

$$T_{Act}(BT_i) >= T_{Min}(BTi)$$

It may be noted that slack time is insignificant while imposing minimum timing constraints.

**Durational Timing Constraints**

Durational timing constraints impose the restriction on duration of occurrence of an event. Let $T_{Due}$ denotes the time for which the complex transaction must be executed. Following condition must be satisfied for temporal correctness,

$$t_2 - t_1 = T_{Due} , \; S + C = T_{Due}$$

Similarly, if '$c_i$' and '$s_i$' are estimated computation time and slack time of $i^{th}$ base transaction ( BTi ) and $T_{Due}$ (BTi) denotes the time duration for which BTi must be executed, following must hold ,

$$T_{Due} = \sum_{i=1}^{i=m} ( ci + si )$$

where 'm' is total number of base transactions to be fired for a complex transaction. During the complex transaction processing, it is still possible to satisfy the durational timing constraints even if any base transaction finishes its computation earlier or its execution is delayed. Let $T_{Act}(BT_i)$ is the actual time taken by $i^{th}$ base transaction. Following conditions must be monitored at run time in order to maintain these constraints.

1. $T_{Act}(BT_i) > T_{Due}(BTi)$ i.e $BT_i$ takes larger time then its allowable duration.
2. $T_{Act}(BT_i) < T_{Due}(BTi)$ i.e $BT_i$ takes lesser time then its allowable duration for computation.



Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)
0-7695-2108-8/04 $ 20.00 © 2004 IEEE

IEEE COMPUTER SOCIETY

3. $T_{Act}$ ( $BT_i$ ) = $T_{Min}$ ($BT_i$) i.e. $BT_i$ 's actual time to complete its computation is exactly equal to allowable duration for computation.

The first and second condition defined above requires modification in the slack time while third is an ideal case.

**Slack Time Modification**

Consider the first condition where $BT_i$ takes larger time than its allowable duration. The slack time for the remaining base transactions may be modified in a way similar to maximum timing constraints. In this case slack time of rest of the base transaction to be executed is reduced. Slack time for $k^{th}$ base transaction to be executed may be modified as follows,

$$s_k = s_k - ((T_{Act}(BT_i) - (c_i + s_i)) / (m - I))$$

where 'm' is the total number of base transaction to be fired by complex transaction , 'I' is number of base transaction fired so far. Consider the second condition where a base transaction completes its computation earlier i.e. $BT_i$ finishes its computation earlier than its allowable duration $T_{Due}(BT_i)$. Either of following option may be chosen to maintain durational timing constraints. First option is to modify the slack time for $k^{th}$ base transaction. The time saved in computation of $i^{th}$ base transaction must be added to slack time of remaining base transaction. Slack time for remaining base transactions may be modified as following,

$$s_k = s_k + ((c_i + s_i) - T_{Act}(BT_i)) / (m - I))$$

where 'm' and 'I' have similar meaning as stated above. Second option is to delay the execution of $BT_i$ until following is satisfied,

$T_{Act}(BT_i) = T_{Due}(BT_i)$ .Consider the third condition where $T_{Act}(BT_i) = T_{Min}(BT_i)$ i.e. $BT_i$'s actual execution time equals allowable duration for computation. This condition is a desirable condition and no modification in transaction profile is required.

## 4. Condition for Abort

Abortion of the complex transaction is determined by the total remaining time to complete the execution and the time already lapsed in the processing. This process allows the early detection of those transactions, which cannot meet the timing constraints.

**Maximum Timing Constraints.**

Suppose L is the length of time remaining to complete the execution of the complex transaction. Thus,

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

where, $\tau$ is maximum allowable time to complete execution of complex transaction. Following condition must be monitored for early detection of complex transaction.

$$\text{If } L < \sum_{i=p}^{i=m} (c_i + s_i)$$

then complex transaction must be aborted. Here 'p' denotes number of remaining base transaction to complete computation of complex transaction.

**Minimum Timing Constraints**.

Let $T_{Min}$ is minimum time which must be elapsed in complex transaction execution. Following inequalities must be satisfied for ensuring temporal correctness with minimum timing constraints.

$$T_{Min} < \sum_{i=1}^{i=m} (T_{Act}(BT_i))$$

Conditions where $T_{Act}(BT_i) > T_{Min}(BT_i)$ & $T_{Act}(BT_i) = T_{Min}(BT_i)$ are desirable and corresponds to minimum timing constraints. The case where $BT_i$ takes lesser time then minimum allowable time for computation, the execution of $BT_i$ must be delayed until following is satisfied,

$T_{Act}(BT_i) >= T_{Min}(BT_i)$ . Therefore, this policy ensures that no abortion of transaction is required while maintaining minimum timing constraints.

**Durational Timing Constraints.**

As mentioned in previous section , following condition must hold to maintain the durational timing constraints.

$$T_{Due} = \sum_{i=1}^{i=m} (c_i + s_i)$$

where $T_{Due}$, $c_i$, $s_i$ have usual meaning. The case where $T_{Act}(BT_i) = T_{Due}(BT_i)$ is desirable to satify the durational time constraint.The case where $T_{Act}(BT_i) < T_{Due}(BT_i)$, the base transaction execution may be delayed until following is satisfied.

$$T_{Act}(BT_i) = T_{Due}(BT_i)$$

Under the two conditions mentioned above, there is no need for complex transaction abortion. However, for the third condition where $T_{Act}(BT_i) > T_{Due}(BT_i)$, the slack time for the remaining base transaction can be modified subject to satisfaction of following,

$$L >= \sum_{i=p}^{i=m} (c_i + s_i)$$

where 'p' is number of remaining base transactions and 'L' is length of time remaining to complete complex transaction execution. Length of remaining time may be computed as shown below.

$$L = T_{Due} - \sum_{i=1}^{i=p} (c_i + s_i)$$

Therefore, the complex transaction need to be aborted only if following inequality is satisfied after slack time modifications.

$$L < \sum_{i=p}^{i=m} (c_i + s_i)$$



Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)
0-7695-2108-8/04 $ 20.00 © 2004 IEEE

IEEE COMPUTER SOCIETY

## 5. Conclusions

In the proposed model, we have modeled long duration activity as complex transaction, which follows forward execution from one state to another by firing base transactions. The base transactions are fired as a consequence of condition-evaluation and action-taken part of detached mode ECA rule. The temporal constraints may be explicitly specified on complex transaction. We have proposed a method for enforcement of maximum, minimum and durational timing constraints on complex transactions. Run time monitoring of these constraints in active database can be done for early detection of an aborting transaction. The model also suggests that the blocking of concurrent transaction can be restricted by capturing cooperation semantics among the complex transactions. Since transactions are allowed to share the locks with the concurrent cooperative transactions, contentions for resource are reduced. This semantic enable the transactions to meet timing constraints. 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 timing constraints. In such cases the compensating action plan should be invoked. While defining the various timing constraints, this model ensures best utilization of slack time even though a base transaction fails to meet timing constraints .We have also suggested the conditions when a complex transaction can satisfy the timing constraints even if some of its base transaction fails to meet timing constraint.

## References


1. U Dayal. "Active Database Management System", in Proc. 3rd Intl. Conference of Data and Knowledge Bases, pp 150-169, 1988.
2. B Dasarathy. "Timing constraints of real time system: construct for expressing them, Methods of validating them", IEEE Transactions on Software Engineering, 11(1):80-86, Jan 1985.
3. M Hsu, R Ladin and D McCarthy. "An Execution Model for Active Database Management System", in Proc. 3rd Intl. Conference of Data and Knowledge Bases, 1988.
4. DS Yadav, etal. "Towards a Model of Concurrency", Proc. 3rd Intl. Conf. on Information, Communication and Signal Processing (ICICS 2001), Singapore, 15-18 Oct 2001.
5. Purimetla, Rajendrandran, Ramamirtham, Stankovic. "Real Time Databasse: Issues & Application", Editor Sang Son in Advances in Real Time System, Prentice Hall, pp 487-505, 1995.
6. F Korth, G Speegal. "Formal aspects of concurrency control in long duration transaction system – using NT/PV model", ACM Transaction on Database System, 19: 492-535, 1994.
7. Ben Kao, H Garcia-Monlina. "Overview of Real Time Database System", Real Time Computing, NATO ASI Series F, Vol 127, Berlin Springer Verlag, pp 261-282, 1994.
8. R Abott, H Garcia-Monila."What is a Real Time Database System", Abstracts of 4th workshop on Real Time Operating Systems, IEEE, pp 134-138, 1987.
9. U Dayal, M Hsu, R Ladin. "Organising long running activities with triggers and transaction", Proc. ACM SIGMOD, Intl. Conference on management of data, Atlanta, 1990.
10. P Kangsabanik, R Mall, AK Majumdar. "Modelling Long Duration and Cooperative Transactions in Active Object Oriented Database System ", in Proc. Intl. Workshop on Advanced Transaction Model and Architectures, India, 1996.
11. SK Madria. "A Study of Concurrency Control & Recovery algorithms in nested transaction environment", The Computer Journal, 40(10), 1997.
12. P Kangsabanik , R Mall , AK Majumdar., "Concurrency control of nested cooperative transaction in Active DBMS", in IEEE 4th Intl Conference on High performance computing ( HiPC-1997), India.
13. J Stonkovic, Zhao. "On Real Time Transaction", ACM SIGMOD record, 17: 4-18, 1988.
14. H Garcia Monila, K Salem. "SAGAS", Proc. of ACM SIGMOD, Intl Conference on Management of Data, pp 249-259, 1987.
15. Taylor. "Introducing real time constraints in to requirement and high level design of operating system", Proc. National telecom Conference, Houston, 1:18.5.1–18.5.5, 1980.
16. L Sha, R Rajkumar, JP Lehoczky. "Priority Inheritance protocol, an approach to real time synchronization", IEEE Trans. On computers, 39(9):1175-1185, 1990.
17. J Moss. "Nested Transactions: An approach for reliable distributed computing", Proc. ACM SIGMOD, Intl. Conf. On management of Data, Atlantic city, 1990.
18. P Kangsabik, R Mall , AK Majumdar. "Semantic Based Concurrency Control of Open Nested Transactions in Active Object Oriented Database Management Systems", Distributed & Parallel Databases, 8(2):181-222, 2000.
19. P Kangsabik, R Mall, AK Majumdar. "A Technique for Modelling Applications in Active Object Oriented Database Management Systems", Information Sciences, 102(1-4):67-103, 1997.
20. D S Yadav, Rajeev Agrawal, R C Sarswat, "An approach to reduce resource contention while scheduling time constrained long running activity in Active Database System", 5th International Workshop on Computer & Information Technologies, Proceedings CSIT 2003, ,Ufa, Russia , Vol 1 , PP162-166, 16-18 Sept , 2003




Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)
0-7695-2108-8/04 $ 20.00 © 2004 IEEE

IEEE COMPUTER SOCIETY