Scheduling Adaptive Transactions in Real-Time Databases *

Erdoğan Doğdu
Department of Computer Science and Engineering
Case Western Reserve University
Cleveland, OH 44106
dogdu@ces.cwru.edu

Abstract. A new transaction model, called the Adaptive Transaction Model, is proposed for Real-Time Database Management Systems (RTDBMSs) applications. The Adaptive Transaction Model is an extended transaction model with a nested structure containing optional and required subtransactions. Adaptive Transactions (ATs) have time constraints to support real-time database applications. Optional subtransactions can be omitted during the execution if time does not permit. Scheduling issues for a special case of adaptive transactions, called chain-structured adaptive transactions, are discussed. Several priority-based scheduling policies are proposed and experimental results are reported under lock-based and timestamp-ordering concurrency control protocols. A priority assignment policy (MSF-MES) is found to provide superior (low) miss ratios compared to other policies.

1 Introduction

Timeliness is the key issue in scheduling transactions in a Real-Time Database System (RTDBMS). In an RTDBMS, transactions are attached with deadlines which can either be hard, soft, or firm. Transactions with hard deadlines are to complete their work before their deadlines, otherwise the results can be catastrophic. Soft deadline transactions are executed to the end regardless of their deadlines, but their contributed value to the system decreases as time progresses after the given deadline. In comparison, for an RTDBMS where transactions have firm deadlines, transactions are useless once they expire their deadlines, and therefore are aborted and their work is undone (rolled back) once they reach their deadline before committing. In this study, we consider RTDBMSs with firm deadline transactions.

Recent research in real-time systems and databases has focused on the idea of utilizing partial results so that jobs (such as transactions) meet their deadlines. In other words, transactions report estimate or approximate results when they cannot complete within their time quotas. Özsoyoglu et. al. partitions relations into semantically meaningful subsets in CASE-DB [Özs 95]. Queries process these subsets instead of all the tuples of relations to produce approximate results within given time quotas. Liu et. al. works on a similar idea, and defines an approximate relational model in which a relation consists of two sets of tuples: a

---

* This work has been done under the supervision of Prof. Gültekin Özsoyoglu of Case Western Reserve University. This research has been partially supported by the National Science Foundation grant IRI-92-24660. E. Dogdu has also been supported by a scholarship from the Ministry of Education, Turkey.
certain set and a possible set [Liu 95]. They also present approximate relational algebra operators that are used in approximate query processing. As operand relations get more data segments, query processor (named APPROXIMATE) incrementally reevaluates the query to provide a more precise result. Some other research focused on the processing of aggregate queries such as Count, Average, Max/Min in RTDBMS [Ozs 90, Hou 91, Lip 90]. An estimate of the aggregate function in the query is reported once the time deadline is reached.

The work in providing approximate or estimate results in RTDBMSs has focused on the query processing techniques only [Ozs 90, Liu 95, Ozs 95]. The idea has not been extended to a multi-programming environment where queries and transactions are not standalone, and share data and system resources with other transactions. Therefore, special handling is required in terms of providing transaction management, scheduling services to enhance the system and transaction performance. The extended transaction models [Elm 92] work towards that end by providing means for users to introduce a flexible workload to the system. Recent proposals have shown the applicability of extended transactions such as the nested transaction model in the RTDBMS area [Ram 92, Fort 94].

We propose the adaptive transaction model for RTDBMSs which falls under the category of extended transaction models. Adaptive transactions are nested transactions with firm deadlines. Subtransactions of an adaptive transaction are of two types: optional and required. Optional subtransactions and subtransactions originating from optional subtransactions can be cancelled during execution if time does not permit. Optional subtransactions strive to improve the result being provided to users. Data conflicts may also force transactions to cancel the execution of optional subtransactions which are not vital in the overall transaction work, and therefore can be skipped to improve the concurrent execution of transactions, leading to less transaction aborts and restarts. Vital and non-vital transactions introduced by [Gar 90] are similar to optional and required transactions of our model, yet they are not discussed in the context of real-time databases. Fortier et al has formalized flexible nested real-time transactions in the context of SQL transactions; but transaction management and scheduling issues are not considered [For 94].

In this paper, we concentrate on the scheduling problem for adaptive transactions. Priority assignment policies are proposed for the adaptive transaction model and is tested via simulation. In section 2, we define the adaptive transaction model, and give sample applications. Next, we discuss scheduling problem and present priority-based scheduling policies for chain-structured adaptive transactions (CAT) which is a simplified version of nested adaptive transactions. Concurrency control issues are discussed in section 4. Simulation results are summarized in section 5, and we conclude in section 6.

2 Adaptive Transaction Model

Adaptive transactions are nested transactions that are designed to suit the requirements of a real-time execution environment, and have timing constraints, i.e., an adaptive transaction meets its given completion time deadline.

Adaptive transactions, similar to nested transactions, are structured as transaction trees. The nodes of an adaptive transaction, which we call transaction subtransactions, have two different types, namely “required subtransactions” and “optional subtransactions”. Optional subtransactions are the stopping points of