A real-time database system (RTDBS) is designed to provide timely response to the transactions of data-intensive applications. The transactions processed in a RTDBS are associated with real-time constraints typically in the form of deadlines. With the current database technology it is extremely difficult to provide schedules guaranteeing transaction deadlines. This difficulty comes from the unpredictability of transaction response times. Efficient resource scheduling algorithms and concurrency control protocols are required to schedule RTDB transactions so as to maximize the number of satisfied deadlines. In this paper, we describe several distributed, lock-based, real-time concurrency control protocols and report on the relative performance of the protocols in a distributed database environment. The protocols are different in the way real-time constraints of transactions are involved in controlling concurrent accesses to shared data. A detailed performance model of a distributed RTDBS was employed in the evaluation of concurrency control protocols.

A real-time database system (RTDBS) can be defined as a database system where transactions are associated with real-time constraints typically in the form of deadlines. A RTDBS is designed to provide timely information to data-intensive applications such as stock market, computer-integrated manufacturing, telephone switching systems, network management, and command and control systems (Ramanritham, 1992). Traditional database systems are designed to provide functionally correct information. Maintaining data consistency is the primary consideration in transaction scheduling. On the other hand, the basic issue considered in traditional real-time systems is the satisfaction of timing constraints associated with transactions. The problem of maintaining the consistency of shared data is usually not addressed. Design of a RTDBS requires the integration of scheduling concepts from both real-time systems and database systems to handle the timing and consistency requirements together. With the current database technology it is extremely difficult, if not impossible, to provide schedules guaranteeing transaction deadlines. This difficulty comes from the unpredictability of transaction response times. Each transaction operation accessing to a data item takes a variable amount of time due to concurrency control and disk IO (Stankovic & Zhao, 1988). The general approach to the scheduling problem in RTDBS’s is using existing techniques in CPU scheduling, buffer management, IO scheduling and concurrency control, and to apply time-critical scheduling methods to make a best effort to satisfy transaction deadlines. The performance goal in satisfying timing constraints can change depending on the application environment. If the only real-time parameter associated with each transaction is the assigned deadline, the goal is to minimize the number of transactions that miss their deadlines. A priority order is established among transactions based on their deadlines. There are RTDB applications where transactions may be assigned different values, where the value of a transaction reflects the return the application expects to receive if the transaction commits within its deadline (Biyabani et al., 1988; Haritsa et al., 1991; Huang et al., 1989). For such applications, the performance goal is to maximize the value realized by the in-time transactions. Transactions are assigned
priorities which are functions of both their values and deadlines.

Most of the recent research in RTDB transaction scheduling has concentrated on development and evaluation of concurrency control protocols. Each locking protocol proposed for RTDBS’s is based on either one of the following two schemes: priority inheritance and priority abort. The priority inheritance scheme allows a low priority transaction to execute at the highest priority of all the higher priority transactions it blocks (Sha et al., 1988). The priority abort scheme is based on aborting the lower priority transaction when priority inversion occurs. Development and evaluation of various lock-based concurrency control protocols are reported in (Abbott & Garcia-Molina, 1988, 1989; Agrawal et al., 1992; Huang et al., 1989, 1991a; Lin & Son, 1990; Sha et al., 1991; Son & Chang, 1990; Son et al., 1992; Ulusoy, 1992). Another class of proposed protocols is based on the optimistic method of concurrency control. Haritsa et al. (1990a, 1990b) and Huang et al. (1991b) present a set of optimistic protocols and provide the comparison of those protocols with locking. These performance comparisons do not completely agree due to the different types of systems used and the assumptions made in evaluating the protocols. Timestamp-based concurrency control protocols, which involve real-time priorities in constructing a timestamp order among transactions, are studied in (Son & Lee, 1990; Ulusoy, 1992).

Other recent research addressing the scheduling problem in RTDBS’s can be summarized as follows. Some new approaches to priority-based IO scheduling are discussed in (Abbott & Garcia-Molina, 1990; Carey et al., 1989; Chen et al., 1991; Kim & Srivastava, 1991). Huang et al. (1989) provides the development and evaluation of several real-time policies for handling CPU scheduling. Özsoyoglu et al. (1990) introduces new techniques to process database queries within fixed time quotas. Different degrees of accuracy of the responses to the queries can be achieved by using those techniques. Evaluation of priority-based buffer management policies is reported in (Carey et al., 1989; Huang & Stankovic, 1990).

In this paper, we focus on lock-based concurrency control protocols for distributed RTDBS’s. We describe several distributed, real-time concurrency control protocols and study the relative performance of the protocols in a nonreplicated database environment. The protocols aim to maximize the satisfaction of real-time requirements while maintaining data consistency via enforcing serializability. Concurrency control protocols are different in the way real-time constraints of transactions are involved in controlling concurrent accesses to shared data.

A detailed performance model of a distributed RTDBS was employed in the evaluation of concurrency control protocols. The performance model captures the basic characteristics of a distributed database system that processes transactions, each associated with a real-time constraint in the form of a deadline. A unique priority is assigned to each transaction based on its deadline. The transaction scheduling decisions are basically affected by transaction priorities. Various simulation experiments were carried out to study the relative performance of the protocols under many possible real-time and database environments. The performance metric used in evaluation of the protocols is success-ratio, which gives the fraction of transactions that satisfy their deadlines.

**Distributed Real-Time Concurrency Control Protocols**

Lock-based concurrency control in a distributed system is either centralized, where the lock management is provided by one of the sites, or distributed, where the lock managers are distributed along with the database. Our system model includes distributed concurrency control in which each scheduler manages locks for the data items stored at its site based on the two-phase locking rules (Eswaren & Gray, 1976).

The concurrency control protocols studied assume a distributed transaction model in the form of a master process that executes at the originating site of the transaction and a collection of cohorts that execute at various sites where the required data items reside. The master process is responsible for coordinating the execution of the cohorts as to be detailed in the section, A Distributed RTDBS Model. The cohorts carry the real-time priority of their transaction.

Each cohort process executing at a data site has to obtain a shared lock on each data item it reads, and an exclusive lock on each data item it updates. Conflicting lock requests on the same data item are ordered based on the real-time strategy implemented. Local serializability is provided by enforcing the rules of two-phase locking. In order to provide global serializability, the locks held by the cohorts of a transaction are maintained until the transaction has been committed. This constraint is enforced for the application of two-phase locking rules globally so that a transaction cannot obtain a lock after releasing a lock at another site.

The concurrency control protocols described in the following sections are distinguished based on whether they make use of a prior knowledge of data access patterns of transactions or not. The first group of protocols assume that data requirements of a transaction are not known before the execution of the transaction, while the second group of protocols assume that a list of data items to be accessed is submitted by each arriving transaction.

**Protocols with Unknown Data Requirements**

The first protocol is the basic version of two-phase locking and does not take real-time priorities into account in processing data access requests. Performance of basic two-phase locking provides a basis of comparison for studying the performance of the priority-based protocols.
Related Content

FOOM - Functional and Object-Oriented Analysis and Design of Information Systems: An Integrated Methodology
[www.igi-global.com/article/foom-functional-object-oriented-analysis/3258?camid=4v1a](www.igi-global.com/article/foom-functional-object-oriented-analysis/3258?camid=4v1a)

Formal Models and Hybrid Approaches for Efficient Manual Image Annotation and Retrieval
Rong Yan, Apostol Natsev and Murray Campbell (2009). *Semantic Mining Technologies for Multimedia Databases* (pp. 272-297).
[www.igi-global.com/chapter/formal-models-hybrid-approaches-efficient/28838?camid=4v1a](www.igi-global.com/chapter/formal-models-hybrid-approaches-efficient/28838?camid=4v1a)

Detecting Users' Dietary Preferences and Their Evolutions via Chinese Social Media

A Methodology of Constructing Canonical Form Database Schemas in a Multiple Heterogeneous Database Environment
[www.igi-global.com/article/methodology-constructing-canonical-form-database/51205?camid=4v1a](www.igi-global.com/article/methodology-constructing-canonical-form-database/51205?camid=4v1a)