An Overview of Real-Time Database Systems

Ben Kao[1,2] *and Hector Garcia-Molina*[2]

[1] Princeton University
Princeton, NJ 08544
U.S.A.
[2] Stanford University
Stanford, CA 94305
U.S.A.

1 Introduction

Traditionally, real-time systems manage their data (e.g. chamber temperature, aircraft locations) in application dependent structures. As real-time systems evolve, their applications become more complex and require access to more data. It thus becomes necessary to manage the data in a systematic and organized fashion. Database management systems provide tools for such organization, so in recent years there has been interest in "merging" database and real-time technology. The resulting integrated system, which provides database operations with real-time constraints is generally called a real-time database system (RTDBS) [1].

Like a conventional database system, a RTDBS functions as a repository of data, provides efficient storage, and performs retrieval and manipulation of information. However, as a part of a real-time system, whose "tasks" are associated with time constraints, a RTDBS, has the added burden of ensuring some degree of confidence in meeting the system's timing requirements.

Example applications that handle large amounts of data and have stringent timing requirements include telephone switching (e.g. translating an 800 number into an actual number), radar tracking and others. Arbitrage trading, for example, involves trading commodities in different markets at different prices. Since price discrepancies are usually short-lived, automated searching and processing of large amounts of trading information are very desirable. In order to capitalize on the opportunities, buy-sell decisions have to be made promptly, often with a time constraint so that the financial overhead in performing the trade actions are well compensated by the benefit resulting from the trade. As another example, a radar surveillance system detects aircraft "images" or "radar signatures". These images are then matched against a database of known images. The result of such a match is used to drive other system actions, for example, in choosing a combat strategy.

Conventional database systems are not adequate for this type of application. They differ from a RTDBS in many aspects. Most importantly RTDBSs have different performance goals, correctness criteria, and assumptions about the applications. Unlike a conventional database system, whose main objective is to

W. A. Halang et al. (eds.), *Real Time Computing*

provide fast "average" response time, a RTDBS may be evaluated based on how often transactions miss their deadlines, the average "lateness" or "tardiness" of late transactions, the cost incurred in transactions missing their deadlines, data external consistency (how current the values of data are in reflecting the state of the external world), and data temporal consistency (values of data in the database should be taken from the external world at similar times) [37].

As a real-time system, specifications related to timing constraints are usually supplied by the application designers. For most cases, these timing requirements are expressed as deadlines for transactions. Transactions of this sort, with which explicit time constraints are associated, are termed real-time transactions.

As mentioned above, a RTDBS can be viewed as a value-added database system that supports real-time transactions. A real-time transaction has to be completed by its deadline to be of full benefit to the system. Such guarantees are usually hard to ensure. In case a transaction's deadline is not met, the transaction is called a tardy transaction.

Real-time database systems differ in the way tardy transactions are handled, and this issue is generally referred to as the overload management problem. A tardy transaction may carry positive, zero, or negative residual value to the system. For the positive case, even though the benefit obtained by completing the tardy transaction is usually less than its full fledged value, the system should still complete it, if possible. The system may, however, choose to lower the transaction's priority so that non-tardy transactions are given preferential treatment, for example, in accessing system resources. When a tardy transaction completely loses its value (zero residual value case), it should be dropped to free system resources for the benefit of other transactions. Finally, when a tardy transaction carries negative value, the system may choose to raise the transaction's priority so that it can be completed as soon as possible to diminish the cost incurred due to it tardiness. On the other hand, the system may lower the transaction's priority or even drop it so that other transactions have a better chance of meeting their deadlines. The decision is dependent upon the application semantics. In the extreme case that a system cannot afford having a tardy transaction (e.g. in nuclear power plant control), the system is said to be a hard real-time database system; otherwise, if tardy transactions are tolerated even though they may be undesirable (e.g. arbitrage trading), we say that the system is a soft real-time database system.

It is argued in [38] that with current technology, it is very hard to provide an absolute guarantee on meeting transaction deadlines, and therefore, RTDBSs are mostly limited to soft real-time systems. There are several factors that make it hard for a RTDBS to meet all deadlines. Firstly, the executions of database transactions are usually data and resource dependent. To guarantee satisfaction of transaction deadlines requires enormous excess resource to accommodate for the highest system load. Secondly, full transaction support involves many database protocols which are highly unpredictable in their execution times. Concurrency control protocols, for example, often introduce blocking and restart of transactions over resource contention. Thirdly, disk-based database systems interact heavily with the I/O subsystem. Problems such as disk seek time variation, buf-