Many computer systems that automatically control financial applications are required to handle large amounts of data with timing constraints imposed on the data and on the transactions that access the data. Traditional database technology, however, is not designed to manage perishable (time-constrained) data of financial applications, such as “current” stock prices and trading volumes in index arbitrage transactions. Furthermore, traditional database systems are not capable of scheduling the time-constrained transactions in financial applications, such as transactions that must be performed within time bounds of guaranteed price quotations. This paper presents a schema model of real-time object-oriented database that addresses these weaknesses and shows how this model can support a form of intelligent program stock trading called index arbitrage. This paper also illustrates the use of a protocol called timed atomic commitment, which is an extension of traditional distributed database atomic commitment protocol, to perform index arbitrage transactions using our real-time object-oriented database model.

In many applications computer systems manage large amounts of data where there are timing constraints on the data and on the transactions that access the data. For instance, in database systems that support intelligent program stock trading [Miller, Miller, & Brennan, 1991], “current” stock prices and volumes of stocks can only be considered valid for short periods of time and are thus time-constrained data. Similarly, database transactions initiated by a computer program for stock trading may be constrained to be completed before the deadline imposed by guaranteed price quotations from brokers. Such applications require a real-time database management system. Traditional database management systems (DBMS) are concerned with maintaining logical consistency of persistent data and providing good average response time for transactions. Although real-time DBMSs may still have persistent data, some of their data may also be perishable because it may become invalid after a certain amount of time (e.g., stock prices). Consequently, real-time DBMSs must be concerned with both logical consistency and temporal consistency of data. Also, since the success of a transaction in a real-time DBMS can depend on meeting its timing constraints, the real-time DBMS must be concerned with scheduling transactions to meet timing constraints instead of simply optimizing average transaction response time.

Recently, there has been limited research directed towards developing relational real-time databases that can support such real-time applications [Ramamirtham, 1993; Son, Yannopoulos, Kim & Iannaccone, 1992]. Although the relational data model is useful for many applications, we believe that it is not as well-suited as an object-oriented data model [Zdonik & Maier, 1990] for many applications that require complex data, complex relationships among data, and first-class support for timing constraints. As we show later, the encapsulation and complex data representation of the object-oriented model facilitates the expression and enforcement of complex constraints found in financial databases. Unfortunately, although there has been significant recent work done in developing object-oriented databases, there has been little work done on real-time object-oriented databases. This paper describes the schema model of a real-time object-oriented database that incorporates the ability to express absolute temporal consistency constraints on a data object attribute,
relative temporal consistency among attributes of a data object, and relative temporal consistency among attributes of different data objects. In addition to expressing temporal consistency of data, our real-time object-oriented database model supports expression and enforcement of transaction timing constraints.

As an example of an application with time-constrained transactions, consider a form of program trading called index arbitrage where an investor uses program trading to take a position in a stock index futures contract on an exchange, such as the Chicago Mercantile Exchange (CME), and simultaneously takes an opposite position in a basket of stocks that replicates the underlying index of the futures contract on a stock exchange, such as the New York Stock Exchange (NYSE). We assume that the system’s database is tied directly to price information from the stock market and to the futures market through a quotation services company’s reporting system [Spectrum Staff, 1987]. We further assume that the system is tied to an automatic order execution system (DOT and Super DOT are examples of automated order execution systems in use at the NYSE.) that would accept orders in electronic form for the exchange-listed stocks.

Using the data stored in the database, the system computes the difference between a futures contract at the futures exchange and the actual stocks at the stock exchange. If the price difference is more than the carrying cost (the cost of short term money less the dividends received during the holding period) plus the transaction cost (broker’s fees and the cost of conducting the exchange), the computer will initiate a transaction with brokers at respective exchanges. However, when the orders are actually placed, the prices for the stocks and the futures may have already changed; if the transaction is not executed while the price data is valid, the investor may turn a profit opportunity into a loss. In order to reduce the risk posed by perishable data, the investor must ensure that he can buy and sell the stocks and futures at the prices he gets from his database. One way to facilitate ensuring a guaranteed profit is for the investor to obtain a guaranteed quotation from his brokers. A guaranteed quotation is a typical price quotation along with a guarantee that the quotation will be good for a certain time interval (e.g. from the time of the quotation for 30 minutes).

The data’s timing constraints from the guaranteed quotations in the index arbitrage application, as well as external timing constraints that may be imposed for other reasons (e.g. a deadline to obtain capital for another purchase, or to complete by the close of the exchange), must be used to schedule the index arbitrage transaction. There have been several techniques developed to support real-time scheduling of database transactions [Abbott & Garcia-Molina, 1988; Harista, Livny & Carey, 1990]. Most of these techniques use a deadline-driven scheduling heuristic. Unfortunately, determining how to decompose execution into transactions and how to establish the deadline for each transaction is non-trivial.

Our solution to performing such an index arbitrage uses a traditional database protocol called atomic commitment [Berstein, Hadzilacos, & Goodman, 1986], and a recent real-time extension called timed atomic commitment [Davidson, Lee, & Wolfe, 1991]. Traditional atomic commitment requires that either all of a set of actions, such as committing writes to data objects, is performed, or none are performed. In timed atomic commitment, coordinating all-or-nothing behavior must be done under timing constraints. Index arbitrage can be considered a timed atomic commitment: Before carrying out the buying and selling of futures and stocks, the investor’s program must determine from the database that a suitable profit is guaranteed and that the buy/sell actions can be performed within timing constraints. If these conditions can not be met, the investor does not initiate a purchase and loses nothing. Using the terminology of traditional atomic commitment: if both the buying and selling of futures and stocks occur, the system has committed; if neither is carried out, then the system has aborted. In an timed environment, there is another outcome: If one or both parts of the transaction miss their deadline (perhaps due to a communication failure), a loss may occur and the system is in an exception state calling for recovery. A centralized two-phase commit protocol has been developed for timed atomic commitment [Davidson, et al., 1991]; in this paper, we adopt this protocol to express a timed-constrained transaction for index arbitrage in our real-time object-oriented database schema model.

Application and Related Work

To motivate the need for real-time object-oriented database support for program stock trading, we first discuss the requirements of an example index arbitrage application. We then review related work on databases for financial applications, real-time databases, and object-oriented real-time databases. This review shows that current financial applications do not take advantage of new database technology like real-time databases and that current real-time databases can better support financial applications through the use of the object-oriented data model.

Index Arbitrage

Index arbitrage is one of seventeen strategies implemented using program stock trading [Market Volatility and Investor Confidence Panel, 1990]. It involves taking a position in a derivative index product, such as stock index futures, and simultaneously taking an opposite position in a basket of stocks that replicates the underlying index of the futures contract. This transaction is carried out in order to profit from the price difference between the basket and the derivative product. Suppose, for example, a “current” S&P 500 futures contract is priced lower than the underlying basket of actual stocks. An investor could make profit by “simultaneously” buying the futures contract and selling the underlying stocks.
Related Content

Co-creation and Collaboration in a Virtual World: A 3D Visualization Design Project in Second Life
www.igi-global.com/chapter/creation-collaboration-virtual-world/63672?camid=4v1a

Distributed Fragmented Database Set Queries
www.igi-global.com/article/distributed-fragmented-database-set-queries/51080?camid=4v1a

Object-Process Methodology Applied to Modeling Credit Card Transactions
Dov Dori (2001). Journal of Database Management (pp. 4-14).
www.igi-global.com/article/object-process-methodology-applied-modeling/3257?camid=4v1a

Where Do Time Constraints Come From? Where Do They Go?
Krithi Ramamritham (1996). Journal of Database Management (pp. 4-11).
www.igi-global.com/article/time-constraints-come-they/51161?camid=4v1a