APPROXIMATE DATABASE QUERIES & UPDATES

J. W. S. LIU

UNIVERSITY OF ILLINOIS
DEPARTMENT OF COMPUTER SCIENCES
1304 WEST SPRINGFIELD AVE
URBANA, ILLINOIS 61801-2987

AFOSR/NM
110 DUNCAN AVE, SUITE B115
BOLLING AFB DC 20332-8080

SEE REPORT FOR ABSTRACT

19960726 038
FINAL REPORT ON

APPROXIMATE DATABASE QUERIES AND UPDATES

AFOSR Contract No: F49620-93-1-0060

Contract Period: November 15, 1992 — November 14, 1995

Principal Investigator

J. W. S. Liu 217 333-0135 janeliu@cs.uiuc.edu

Institution:

Department of Computer Science, University of Illinois
INTRODUCTION

This project is concerned with the design and implementation of update and query processing schemes that allow databases to be modified and queried in a monotone, incremental manner. It builds on our work on monotone approximate query processing [1,2]. (A monotone computation produces a better result when it is allowed to execute longer.) Specifically, a monotone approximate query processor provides approximate answers to database queries that improve monotonically in accuracy as more and more data is retrieved and processed. It returns an approximate answer when the exact answer to any query cannot be produced in time or when a failure causes the inaccessibility of some of the data required to produce the exact answer.

This research was motivated by the need for database systems suited for (hard) real-time applications, such as machine vision, multiple robots, and air traffic control. It is often difficult to meet two requirements of real-time, highly-dependable database systems: satisfying timing constraints of time-critical query and update operations and providing fault tolerance and graceful degradation in the presence of host and network faults. The imprecise computation technique was proposed in 1987 as a way to make meeting these requirements easier [4]. This technique relies on the use of well-behaved computational algorithms that produce acceptable, intermediate results in predictable, short amounts of time and better results when allowed more time. Until recently, such algorithms did not exist in the database domain. Traditionally, database query and update operations are atomic. An answer to a query is returned only after all the data required to answer the query are retrieved and processed. If a failure causes some of these data to become unavailable, no answer is returned. A partially completed update operation is either rolled back and aborted, or must eventually be completed. Atomicity of these operations is desirable for traditional database applications since one is willing wait for exact answers and the stored data remains good if it is not modified. In contrast, a late time-critical answer from a real-time database may be less accurate than a sufficiently good and timely approximate answer. Time-critical data stored in a database deteriorates with time as the real-world it models changes. It is often essential for time-critical data to be updated even when it is impossible for the entire update operation to be completed. To support imprecise computations in the database domain, query and update computations must be restructured to allow for their partial, approximate completion. This research focuses on the challenging problems of how to provide useful, approximate answers and how to update the stored information incrementally.

This report first summaries the problems addressed and results obtained in this project. It then provides the productivity measures of the project during the contract period from November 14, 1992 to November 15, 1995.

OBJECTIVES

The research carried out by this project has the following two parallel thrusts:

(1) extension, analysis and evaluation of the proposed monotone query processing scheme, and

(2) development of additional theoretical concepts and algorithms to support imprecise database updates.

This research intends to provide monotone schemes needed to make the application of the imprecise computation technique (also called the anytime and sufficiently good methods) feasible in the database domain. By allowing query processing and updates to terminate prematurely and providing the user with
usable, approximate information during failures or overloads, these schemes can enhance the availability and graceful degradation of a database system.

Our past work on monotone query processing focused on set-valued queries for which the answers are sets of objects that match the query qualifications. The objective of this project in this direction has been to determine how readily the proposed monotone query processing scheme can be extended to deal with arbitrary queries in real-life database systems encountered in different application domains. In particular, we wanted to determine the basic limitations of the proposed scheme and the cost of providing the capability for trading off between the quality of the answers with the time and resources used in query processing.

In the area of imprecise updates, our goal has been to develop strategies that allow knowledgebases/databases to be updated incrementally and imprecisely. Our attention has been directed towards databases that are used to provide decision support. We divide such databases into two types: traditional databases and "truth-maintenance systems".

In the case of a traditional database used in a decision support system, the decision maker examines the database and applies a decision policy based on the data in the database. The decision policy maps the set of database states to a set of decisions. When the database is updated, the update transaction completely specifies the final database state after the update is completed; independent of the current and past decisions made by the system. In other words, we can view the the database and decision maker as an "open-loop" system. Given a decision policy, an approximate database state A, reached when an update is only partially completed, is closer to the exact final state F than another approximate state B if the decision \( D_A \) based on A is closer to the precise decision \( D_F \) based on F than the decision \( D_B \) based on B. We say that the imprecise data in state A is better than the imprecise data in state B.

An example of the second type of databases is one that holds track records generated by multiple-hypothesis tracking algorithms. Here, the decision-making process is a part of the update process. Alternatively, we can view the the database and decision maker as an "close-loop" system.

**SUMMARY OF RESULTS**

The underlying principle of the monotone query processing scheme is the approximate relational model [1-3]. This model was initially developed as a rigorous framework for producing approximate answers to set-valued queries. Again, a set-valued query has as its exact answer a set of objects with properties given by the query qualifications; in the relational model, such an answer is a relation. A meaningful and useful set of approximate answers can be defined in terms of all the subsets and supersets of the exact answer. The approximate relational model formally captures this semantics of approximation. In particular, this model defines the approximations of any standard relation in terms of supersets and subsets of the relation, a partial-order relation over the set of all approximate relations for comparing them, and a complete set of new relational algebra operations on approximate operands. Every one of these relational algebra operations is monotone in the sense that the result of the operation is better when its operand(s) becomes better.

We have extended the semantics of approximation supported by the approximate relation model [3,6] to give meaningful approximation semantics for many frequently encountered types of single-valued queries in addition to set-valued queries. We have implemented a prototype query processor, called APPROXIMATE, to demonstrate the feasibility of monotone approximate query processing. APPROXIMATE accepts standard relational algebra queries. It assumes that the data is stored as
relations and the user's view remains that of the relational model. It can be implemented on a relational database system requiring little or no change to the underlying relational architecture. The query processor itself takes an object-oriented approach. Its views of the stored data are object-oriented. Such views provide it with the needed semantic support that is lacking in the relational model and enable it to keep the additional overhead in producing approximate answers small.

Our effort on incremental updates has been devoted to real-time databases that are independent of the decision maker and store time-critical data, i.e., databases in "open-loop" systems. We assume that individual update requests are precise. If they are all processed in time, the data would be precise, e.g., the stored information is as accurate as the designer intends it to be. Imprecision comes from selective processing of update requests.

Rather than attacking the problem in abstraction, we demonstrated the applicability of our approach and examined in depth the issues in approximate updates for two real-time databases: a furnace temperature monitoring system and a stock price quotation system [5]. The former is a representative of databases that contain time-continuous data about real-world states. The latter is a representative of real-time transaction systems. While some strategies thus obtained (e.g., (4) described below) are application-specific, others are applicable to the general problems of partial updating time-continuous data and data in real-time transaction systems.

Take former for example, the malfunction of a controller may cause the temperature of many furnaces to rise abruptly. The system is overloaded with requests for updates of individual furnace temperatures. When this occurs, the system can only grant some requests. The selected updates granted by the system are processed with precise sensing data, while the parts of the database not updated are left in imprecise states.

We have designed several update strategies that allow real-time databases containing time-continuous data to be updated incrementally and approximately. These strategies deal with the following issues:

1. the scheduling of update requests during normal conditions — This is a problem of selecting proper real-time scheduling algorithms for the specific application.

2. the detection of abnormal conditions — Abnormal conditions can sometimes be detected by monitoring consecutive misses of deadlines or buffer overflow.

3. algorithms to selectively grant update requests during abnormal conditions.

For example, for the furnace temperature remote sensing system, requests for temperature sensing are sent in packets, specifying furnace number and the seconds between temperature measurements. Streams of requests for individual furnaces can be modeled as independent periodic tasks assigned to a single processor when the temperature readings are stored in one database. For this problem, we explored and compared different algorithms for scheduling the update requests. Overload conditions are detected when several misses of deadline occur consecutively. Possible approaches to handle such overload conditions include:

1. reducing the periods of tasks of the highest priority,

2. involuntarily reducing the periods of tasks of the lowest priority,
(3) proportionally distribute the necessary amount of period reduction among all tasks, and

(4) shutting down the furnace which has the shortest sampling period.

The selection of a proper approach should be based on the length of the time interval during which each data item remains valid, the different levels of urgency of various sampling activities, the predictability of future performance and load, and the availability of features of fault tolerance. Our approximate update strategies are based on these considerations.

For the problem on stock market data propagation, we consider the system as being constantly under overload condition, i.e., there are always more incoming requests to update than the system resource can handle. In this case, the major concern is for the database of the local market site to selectively process requests to update the prices of various stocks from multiple remote market sites. We model the price change of a stock as a hybrid random process under constant forces, seasonal changes as well as random fluctuations, and use statistical inference to pick out abnormally active stocks for immediate update.

We are currently investigating how to provide flexibility in distributing resources for updating of different stock types and how to provide graceful degradation when the whole market becomes abnormally volatile. The other area of study is truth maintenance database system. For this type of systems, a possible approach to the problem of incremental updates of these systems might involve multiple hypothesis testing, constantly checking whether the current model is still valid, and the appropriate model changes in time. We are using radar tracking to demonstrate the applicability of our update strategies.

Our study of this application domain provided us with a deeper insight into the imprecise computation model in general and the need to model the effect of erroneous input in particular. (For example, when the information provided by a database is imprecise, the decision support system that use this information may need to compute for longer time in order to reach a decision of an acceptable quality. The model used in previous studies on imprecise computations typically ignores this type of dependency.) We have developed an extended imprecise computation model that take into account the effect of input error [14].

PUBLICATIONS

Below is a list of our publications on results produced by this project and on closed related work.


**PROJECT INFORMATION**

Number of researchers working on the project, including the Principal Investigator: 3

Postdocs: 0

Graduate Students: 2

Other (Specify): none

Major Professional honors and activities:


Member, Program Committee of IEEE Real-Time Systems Symposium, 1994 and 1996.

Chairman, Program Committee of the 15th ICDCS, 1995.