Database Research at The University of Oklahoma

Le Gruenwald, Leonard Brown, Ravi Dirckze, Sylvain Guinepain
Carlos Sanchez, Brian Summers, Sirirut Vanichayobon

The University of Oklahoma
School of Computer Science
Norman, OK 73019
gruenwal@cs.ou.edu

1. INTRODUCTION

Database management is one of the main areas of research of the School of Computer Science at The University of Oklahoma (OU). The objective of the database research team at OU (OUDB) is to help solve the many issues and challenges facing the database research community, especially with respect to emerging technology. Currently, many projects are being conducted in the following areas: real-time databases, object-oriented databases, mobile databases, multimedia databases, data mining and data warehouses. These projects have been funded by federal and state agencies as well as private industries such as National Science Foundation, the U.S. Department of Education, Oklahoma State Department of Environmental Quality, and Objectivity, Inc. The purpose of this paper is to document some of our past and current projects. More information can be found at http://www.cs.ou.edu/~database.

2. REAL-TIME DATABASES

In a Real-Time DataBase Management System (RTDBMS), transactions must not only maintain the consistency constraints of the database but also satisfy their timing constraints. In addition, a RTDBMS often involves processing both temporal and persistent data. A RTDBMS is appropriate for applications such as air traffic control, stock trading, telecommunications, flexible manufacturing and aircraft flight programs. Its main goal is to meet the timing constraints of data transactions regardless of system or transaction failures. Our project aims at the development of techniques to manage recovery and nested transactions.

2.1. Recovery in Real-Time Databases

To prepare a RTDBMS for coping with failures, we have developed techniques to handle three recovery activities: logging, checkpointing, and reloading. The objectives of these techniques are to reduce recovery time, minimize the percentage of transactions missing their deadlines, and minimize the percentage of temporal data becoming invalid.

Our logging technique called Transaction Type Logging (TTL) takes transaction types and data types into consideration and employs both immediate and deferred update at the same time. Transactions are divided into six types depending on their characteristics. The logging scheme dynamically changes from performing no logging activities to performing either immediate or deferred update depending on the transaction types. We conducted an analytical study comparing the performance of TTL with that of the best technique called VILMB (Valid-Invalid Logging with Multiple Buffers using deferred update). The results showed that TTL reduces the REDO cost and number of memory references during normal operation in most cases at the cost of a slight increase of log space. When the amount of temporal data in the database exceeds 50%, the performance of TTL is consistently better than VILMB's [GC97].

Our checkpoint scheme considers data partitioning, update frequency, temporal data valid interval, and data priority. The main memory database is divided into persistent and temporal data partitions with different update frequencies and valid intervals. Our algorithm checkpoints partitions of high update frequency, short temporal data valid intervals, and high data priority more often than other partitions. The simulation results showed that our technique outperforms the conventional fuzzy checkpoint algorithm. In the cases of high transaction load and system failure rate, our technique also outperforms the scheme that considers update frequency and temporal data valid interval, but not data priority [HG96a].

We developed two reload algorithms, partition reload and data priority reload, for Real-Time Main Memory DataBases (RTMMDBs) that aim at minimizing the number of timing constraints violated in addition to reducing system recovery time. Common features of the proposed reload algorithms include: 1) the system is brought on-line before the entire database is reloaded into main memory in order to reduce down time, 2) transaction execution priorities are considered so that high priority transactions have more opportunities to meet their deadlines, and 3) data accessed frequently are reloaded before other data. The two reload algorithms differ from each other based on their choice of recovery unit and reload priority. In order to evaluate the performance of the proposed schemes, we conducted extensive simulations. The results show that: (1) The data priority reload algorithm, which considers transaction priority, data access frequency, reload prioritization, and preemption during reloading, provides a significant performance improvement over the conventional reload algorithm. (2) The system load, database size, and system failure rate determine whether the efficiency of different reload algorithms is crucial to the overall performance. (3) The key design factors of RTMMDB reload algorithms are system unavailability, transaction execution priority, reload threshold, and recovery unit [HG96b].

2.2. Processing Nested Transactions in RTDBMSs

Extensive work has been done in the area of real-time transaction scheduling on single-level transaction models, but only limited work has been done on nested models. For many advanced applications, transactions are long, complicated, and access data items at various network sites. In conventional real-time single-level transaction models, a transaction is considered as a flat single unit of tasks that consists of a sequence of primitive actions with a given deadline. Priorities in such systems are usually assigned according to the given deadlines of individual transactions. Schedulers may use these priorities to determine how to allocate resources. Because of the atomicity requirement of transactions, if there is a failure during one's execution, it is rolled back and restarted. Therefore, even if all transactions are initially schedulable, these rollbacks and restarts may cause them to miss their deadlines. Thus, a transaction model beyond the flat model is desired to maximize system performance.

In this project, we have provided solutions for the scheduling problem that exists in distributed real-time database systems using a nested transaction model [CG96]. In this model, a transaction may consist of several subtransactions, each of which acts as a unit of work. We have proposed two priority assignment schemes to derive CPU usage and data conflict resolution priorities that are based on either top-level transactions' deadlines or individual subtransactions' work amounts. Nested two-phase locking is used for concurrency control. As this locking protocol is integrated with the priority-driven preemptive scheduling approach, a problem known as priority inversion may arise. In order to prevent this problem, two resolution protocols, priority abort and priority inheritance, are most widely used for flat transactions. In this project, we have developed two conflict resolution schemes based on these two protocols for a real-time nested transaction environment. We have conducted simulations to measure the overall system performance when integrating the priority assignment schemes and conflict resolution protocols. The simulation results indicate the following: 1) For nested transactions, if there is high data contention, the priority assignment scheme which assigns all members of a transaction with the same deadline works best to derive priorities for both CPU usage and data conflict resolution; otherwise, individual subtransactions' deadlines based on their execution times should be used for CPU usage priority assignment. The abort-based conflict resolution protocol performs better when the system load is low, while the block-based protocol performs better when the load is high. 2) For flat transactions, the priority scheme that assigns the same deadline to all members of a transaction has the best performance, and the abort-based conflict resolution protocol consistently outperforms the block-based one.

3. OBJECT-ORIENTED DATABASES

Object-Oriented DataBase Management Systems (OODBMSs) have been used for many advanced applications requiring modeling power that can represent complex data and complex relations among data. Those application areas include computer-aided design, computer-aided manufacturing, and artificial intelligence. Our research has focused on the following aspects: concurrency control, data clustering, trigger scalability, and temporal indexing.

3.1. Concurrency Control in OODBMSs

Concurrency Control (CC) is a mechanism used to synchronize accesses to the database to maintain its consistency. Due to the complexity in OODBMSs, CC is more complicated than in conventional databases, and therefore may severely degrade system performance if not designed carefully.

In this project, we have developed a locking-based scheme to increase concurrency among methods [JG98]. Our model is based on multi-granularity locking. It utilizes a rich set of lock modes with different locking granularities and the concept of commutativity among methods. Two methods are said to commute if their execution orders do not affect their results and to conflict otherwise. Therefore, if two methods commute, the transactions invoking the methods can run in parallel. For instance access methods, our scheme has several important characteristics. First, it does not put the burden of determining commutativity for methods on application programmers. Second, it provides more concurrency by taking fine locking granularity. Third, it reduces deadlocks due to lock escalation. Finally, it makes use of run-time information in method commutativity construction to improve concurrency. For class definition access methods, it allows them to run concurrently by taking fine locking granularity. Also, it allows more parallelism between class definition access methods and instance access methods. Our comparison study has shown the superiority of our technique over many existing ones.

3.2. Data Clustering in Object-Oriented Databases

Data clustering is a technique that can be used to improve the performance of an OODBMS. When data cannot fit in the main memory, they are stored on the hard disks. Without data clustering, accessing two related objects usually requires two disk I/Os because they are not stored in the same page. This degrades the performance of an OODBMS because accessing a hard disk is slow. In contrast, the main memory can perform very fast random access. Thus, related objects should be stored close to each other in order to maximize the amount of relevant information returned when a page is loaded from the disk.

We have conducted a comprehensive comparison of existing clustering techniques and developed a new one that offers several innovative features [DG96, GG97]. First, the required storage size of our technique is linear with the number of objects. Second, we developed a replication strategy adaptable for each class of objects. Objects are duplicated only when they are read accessed to increase object locality. Third, our technique is flexible and can be tuned to avoid big overhead when the OODBMS is overloaded. Fourth, the clustering process can be tuned by using a reduced set of statistics which is easier to set compared with those required in existing techniques. The simulations have shown that our technique outperforms many static and dynamic clustering techniques.

3.3. Trigger Scalability in Active OODBMSs

In the 1990s, active features in database systems have become common place. Most commercial systems now support some kind of trigger processing. A scalable trigger system is an active database that can support multiple triggers on a single data element. Although trigger support has become more popular, scalability in these systems is non-existent.

From an application point of view, scalability allows for the implementation of workflow modeling in an active database system. Scalable trigger support allows for workflow diagrams to be compiled and implemented by the active database system. Several commercial products already do this, but are at the application level. These workflow diagrams would run substantially faster inside an active database [BBC98].

There are several issues facing scalability being investigated at OU. Some issues include trigger priority and selection during execution such as whether they can be checked in parallel. Another is determining the capacity of active (running) triggers in the system. Some triggers are used more frequently than others are, so trigger-caching strategies must be implemented. Semantic analysis of triggers prior to execution can be done to minimize the number of them in a system. Composite event handling must also be examined for possible performance benefits. Also, predictability of trigger execution can be a great tool for optimization. Strategies for synchronous as well as asynchronous trigger support must be investigated. In addition, recovery schemes for scalable trigger systems will need to be rethought. To solve these issues, a conceptual model for scalability trigger support will first be constructed. The plan is to use Informix as the host database system and construct a scalable trigger support module with its datablade API package.

3.4. Indexing in Temporal OODBMSs

A temporal database supports the storage and querying of information that varies over time. Each data value is associated with a time interval corresponding to the transaction time, valid time, or both in the case of bitemporal databases. Data are usually not removed. Instead, updates are made by adding new records. Such a database is expected to be much larger than its conventional counterpart; so its indexing is much more critical. Conventional indexing techniques such as B+-trees and hash-based indexes are not particularly useful for indexing interval data.

Our goal is to develop an object-oriented indexing technique that accommodates temporal characteristics so as to capture sophisticated semantics and provide a close model of future real-world applications. Specifically, we are striving to develop an indexing and storage technique for object-oriented temporal data that would be efficient for temporal queries while remaining optimal for non-temporal ones.

One issue that arises is how to evaluate an indexing and storage scheme, such as determining the different criteria that make it efficient and if those criteria are the same for temporal and non-temporal indexes. In the case of a static database where all the data have already been collected and the database is of fixed size, the efficiency problem can be addressed by analyzing the data and their associated queries statistically so as to store them optimally. However, for most applications, temporal databases are dynamic, and it is practically impossible to reorganize the storage and indexing each time new data values become available.

Another issue in dynamic temporal databases is to store the data so as to minimize the disk I/O for both temporal and non-temporal queries. This could be done by grouping successive updates of a given object or category of objects together within the same disk sector. Also, we could place the most frequently accessed information in main memory. The issue, then, is to find out what information is accessed most frequently. In our model, such information consisted of the index and the most recent data values.

In the case of bitemporal databases, the temporal ordering according to the valid time may not be the same as the ordering according to the transaction time. In that case, an issue is determining if it is still possible to index both time axes optimally.

All of the issues present in OODBMSs are also issues in temporal OODBMSs. For instance, the model representing the objects in the database may affect the indexing scheme, as may the class hierarchy and use of nested predicates and methods in queries. References between objects may be another important factor.

4. MOBILE DATABASES

A MultiDataBase System (MDBS) is a federation of pre-existing database systems. In [BBC98], the authors state that “in the future, billions of web clients will be accessing millions of databases, and that the World Wide Web will be one large federated system”. Rapid advances in wireless communication technology dictate that these static database systems extend their services to mobile users. Our research concentrates on transaction management and security in such systems.

4.1. Transaction Management in Mobile Databases

Existing transaction management techniques in the Mobile MultiDataBase (MMDB) environment do not address two key issues. First, the techniques do not address the Isolation property of global transactions. It is difficult to enforce the Isolation property in MMDBs due to the large number of dispersed databases in the federation. Second, they fail to address disconnection that represents catastrophic failures.

In this project, we have proposed a Pre-Serialization (PS) technique for MMDBs [DG98, DG99]. This technique allows site-transactions to commit independently so that resources may be released in a timely manner. Two new states, Disconnected and Suspended, are introduced to fully address disconnection and migration. A toggle operation is used to minimize the ill effects of the prolonged execution due to disconnection of mobile transactions. A Partial Global Serialization Graph (PGSG) commit algorithm that enforces a wide range of correctness criterion with respect to the Atomicity and Isolation properties has been proposed and its correctness has been proven. This algorithm is ideally suited for the MMDB environment as it is de-centralized, and does not require the cooperation of all sites. We have also developed an analytical model to compare PS with other mobile transaction management techniques. We have concluded that for very little additional overhead, the PS technique offers substantial benefits over existing techniques such as fully supporting disconnection and verifying Isolation. Currently, we are developing a simulation model for further analysis.

4.2. Security in Mobile Databases

Database federations have achieved a high degree of sophistication when implementing solutions for data protection and sharing. However, these solutions have fallen short when mobile databases are involved or make part of the federation. Therefore, we are exploring what security issues arise, especially in the area of access controls, when mobile databases are included as part of normal functioning of a database federation [SG99]. We have focused our attention to the development of a security model for mobile database federations. Currently, we are exploring how a mobile federation would behave under different configurations or scenarios. For instance, we have determined the advantages and pitfalls that arise when a mobile database joins the federation either tightly or loosely. In the former case, the mobile database gives up part of its autonomy and most of its security operations are controlled by a central (distributed) entity. In the latter case, the mobile database retains its autonomy and the federation must synchronize all security activities with it for a correct execution. We have also identified how the federation is affected when mobile users' access permissions are altered. Moreover, we have analyzed how access right propagation plays a role in keeping the consistency of access permissions between the federation and the mobile database components that have joined it.

For future work, we still have to explore how role-based access is affected when there are mobile users participating in the federation. There is no clear understanding of how the great variety of databases involved in the system affect the system performance and the security policies implemented in the federation. In addition, not much work has been done in the development of strategies that support multi-level security for mobile databases. Finally, it remains to be explored how emerging technologies such as trust-based systems and the World Wide Web are changing the overall picture of security and its implementation for mobile database federations.

5. MULTIMEDIA DATABASES

In many areas of applications such as medicine, law enforcement, video game development, and web design, users may create new (derived) multimedia objects by editing existing (base) ones. In order to save space, a derived object can be stored as the set of editing operations used to create it along with a reference to its base [GS96]. So, the binary format of the derived object does not have to be physically stored in the database. When a user wants to retrieve such an object, the system accesses the referenced base object, then applies the associated editing operations on it. This storage format is called a *specification*.

The goal of this project is to demonstrate that specifications can be used to improve Content-Based Retrieval (CBR). Knowledge about the editing operations stored in specifications allows us to perform two aspects of CBR more efficiently, feature extraction and similarity search. This project is a collaborative effort between OU and Baylor University.

Conventional multimedia database management systems generally extract the set of features used for querying from each object as it is inserted into the database. This can be extremely time-consuming, especially if the features are extracted manually. If the derived objects are stored as specifications, however, we can use the information in that storage format to determine the objects' features. Therefore, only the base images have to be analyzed manually. To automatically extract features from any specification, we must determine the effects of all possible editing operations on the set of features used for querying. Because of this, the set of editing operations, S, used in our specifications must be *complete*, which means that it can represent all possible transformations from one object to another. In addition, S should be *minimal*, which means that no subset of it is complete. We have developed methods to test a set of image editing operations for these properties [BGS97, BG98].

The second aspect of this project concerns improving similarity search using specifications. Finding the k-nearest neighbors of some query object involves making several distances computations. Calculating these distances can be very expensive. Using specifications, we can reduce the number of times that such expensive distance functions must be computed for the entire database. An upper bound on the actual distance from a query object, Q, to a derived object, D, is the distance from D to its base plus the distance from its base to Q. D's distance to its base can be determined directly from the editing operations stored in its specification. Specifically, each operation is assigned a certain weight, so, the distance between D and its base is the sum of the weights of the operations contained in D's specification. Our goal is to use such information to develop an algorithm for satisfying nearest neighbor queries that minimizes the number of times expensive distance functions are calculated.

6. DATA MINING AND DATA WAREHOUSE

In this project, we proposed a feature classification scheme that can be used to study data mining software. This scheme is based on the software's general characteristics, database connectivity, and data mining characteristics. We then applied our scheme to investigate 43 software products, which are either research prototypes or commercially available. We also derived features that we consider important for data mining software in order to accommodate its users effectively, as well as issues that are either not addressed or insufficiently solved [GG99]. We further explored the use of data mining in data warehouse applications.

The vast majority of requests for information from a data warehouse involve dynamic ad hoc queries [APB98]. The ability to extract data to answer such queries quickly is a critical issue in the data warehouse environment. Proper indexing is crucial to avoid I/O intensive scans against the large data warehouse tables. To support the dynamic nature of the ad hoc queries, the index has to be scalable. The cost of building indexes using all the important attributes is prohibitive. Hence, the challenge is to find the subset of indexes that would improve the ad hoc queries' performance automatically. This is our project's goal.

The major problem in selecting indexes is workload determination since it is unknown until users start using the data warehouse. However it may be possible to estimate it [Win98]. Knowing what the data warehouse's workload looks like plays a major role in determining the attributes considered by the index selection algorithm.

To derive an algorithm for the auto-selection of an index, our project is currently conducting the following tasks: 1) identifying different transaction classes, 2) identifying the characteristics of each class, 3) building models from predefined training queries that will classify the user queries, and 4) selecting indexes based on the estimated workloads of the transaction classes.

To identify the transaction classes and their characteristics, users' requirements need to be understood. Reasonable assumptions have to be made to define the characteristics of a transaction class and periodical reviews are needed to improve them [Win98]. By applying a data mining technique to the training queries of the transaction classes [CH96], the class models are generated. These models are then used to generate the likely workload from the query history. The index attributes that would best suit the need of the user queries are selected from the estimated workload by using an algorithm based on a data mining technique.

There are many issues that still need to be addressed. These include determining how long to keep the history, how to measure the classification accuracy of the models, how to detect runaway queries, and how to manage resources for such queries.

REFERENCES

[APB98] OLAP Council, "*APB-1 OLAP Benchmark Release II*", November 1998, http://www.olapcouncil.org.

[BBC98] Bernstein, P. et al, "*The Asilomar Report on Database Research*", SIGMOD Record, 27(4), 1998, pp.74-80.

[BG98] Brown, L. and L. Gruenwald, "*Determining a Minimal and Independent Set of Image Processing Operations for a Multimedia Database System*", Proceedings of the 1998 Energy Technology Conference and Exhibition, February 1998.

[BGS97] Brown, L., L. Gruenwald, and G. Speegle, "*Testing a Set of Image Processing Operations for Completeness*", Proc. 2nd Conf. on Multimedia Information Sys., April 1997, pp. 127-134.

[CG96] Chen, Y. and L. Gruenwald, "*Effects of Deadline Propagation on Nested Transactions in Real-Time Database Systems*", Information Systems Journal, A Special Issue on Real-Time Database Systems, 21(1), March 1996, pp. 103-124.

[CH96] Chen, M. and J. Han, "*Data Mining: An Overview from a Database Perspective*", IEEE Trans. on Knowledge and Data Engineering, 8(6), Dec 1996.

[DG96] Darmont, J. and L. Gruenwald, "*A Comparison Study of Clustering Techniques for Object-Oriented Databases*", Information Sciences - An International Journal, 94(1-4), 1996, pp. 55-86.

[DG98] Dirckze, R. and L. Gruenwald, "*A Toggled Transaction Management Technique for Mobile Multidatabases*", ACM Intl. Conf. on Information and Knowledge Management, November 1999.

[DG99] Dirckze, R. and L. Gruenwald, "*A Pre-Serialization Transaction Management Technique for Mobile Multidatabases*", Submitted to Journal on Special Topics in Mobile Networking and Apps, 1999.

[GC97] Gruenwald, L. and J. Cheng, "*A Logging Technique Based on Transaction Types for Real-Time Databases*", Real-Time Database and Information Systems-Research Advances, editors Azer Bestavros and Victor Fay-Wolfe, Kluwer, 1997, pp. 379-392.

[GG97] Gay, J. and L. Gruenwald, "*A Clustering Technique for Object-Oriented Databases*", Intl Conf. on Database and Expert Systems Applications, LNCS 1308, Springer, September 1997, pp. 81-90.

[GG99] Goebel, M. and L. Gruenwald, "*A Survey of Data Mining and Knowledge Discovery Tools*", to appear in ACM SIGKDD Explorations, 1999.

[GS96] Gruenwald, L. and G. Speegle, "*Research Issues in View-Based Multimedia Database Systems*", Proc. 2nd World Conference on Integrated Design and Process Technology, December 1996, pp. 331-336.

[HG96a] Huang, J. and L. Gruenwald, "*An Update-Frequency-Valid-Interval Checkpoint Technique for Real-Time Main Memory Database Systems*", International Workshop on Real-Time Database Systems, March 1996 , pp. 135-143.

[HG96b] Huang, J. and L. Gruenwald, "*Impacts of Timing Constraints on Real-Time Main Memory Database Recovery*", Workshop on Databases: Real-Time and Active (Concepts Meet Practice), November 1996, pp. 63-38.

[JG98] Jun, W. and L. Gruenwald, "*Semantic-Based Concurrency Control in Object-Oriented Databases*", Journal of Object-Oriented Programming, 10(8), January 1998, pp. 33-39.

[SG99] Sanchez, C. and L. Gruenwald, "*Mobile Federated Database Security*", Tech. Rep., Computer Science Dept, The Univ. of Oklahoma, May 1999.

[Win98] Winter, R., "*Defining Your Data Warehouse Workload*", VLDB Vision, Sept. 1998.