Database Research at the Indian Institute of Technology, Bombay

D. B. Phatak N. L. Sarda S. Seshadri S. Sudarshan

Department of Computer Science and Engineering
Indian Institute of Technology, Powai, Bombay 400076, India
{dbp,nls,seshadri,sudarsha}@cse.iitb.ernet.in

1 Introduction

The Indian Institute of Technology, Bombay is one of the leading universities in India. Located in Powai, a suburb of the vibrant city of Bombay (which is soon to revert to its original name, Mumbai), it is a scenic campus extending over 500 acres on the shores of Lake Powai. The institute has a faculty strength of about 400, and has about 2500 students. The Department of Computer Science has a faculty strength of 25, and around 150 undergraduate and 70 postgraduate students.

The Database Group in the Department of Computer Science and Engineering is the largest database group in India. The group currently has four faculty members, D. B. Phatak, N. L. Sarda, S. Seshadri and S. Sudarshan. The group also currently has three research scholars, ten Masters students, ten undergraduate students and nine project engineers.

Apart from engaging in active basic research, group members also interact with many industries and government organizations for consultancy and sponsored research projects. Consultancy projects have been carried on for the State Bank of India (the largest bank in India), ACC Ltd. (the largest cement manufacturing company in India), Hewlett Packard HCL Ltd., Coromandel Software Ltd, Mafatlal Consultancy, and various other financial sector industries.

In the rest of the document, we describe the major research activities of the group. Further information about the group is available on the World Wide Web at the URL http://www.cse.iitb.ernet.in/dbgroup.

2 Management of Time and History Data

Principal investigator(s): N. L. Sarda.

The need for adding time as a dimension to databases has long been felt by the researchers as well as the user community. An ongoing project here has constructed a prototype historical database management system on a Netware platform using B-trieve as a storage manager. The novel features of this system include:

- Definition of an extended relational data model for modelling time and history data.
- Extensions to SQL for temporal support.
- Strategies for optimization of historical queries based on a historical algebra.

The prototype, named HDBMS, is based on the concept of an extended relation, called a 'historical' relation, which provides support for valid time and for history consisting of past states. A number of new algebra operations, such as coalesce, expand, time-slice, and concurrent join, were proposed. They formed a basis for extending SQL for retrieval of current and history data, as well as for optimization of queries. The prototype has also been used as a testbed for further research into various aspects related to historical data management, such as:

- A new storage structure, called 'time-grid', for historical relations.
- Support for transaction time in HDBMS using system logs.
- Extensions to concurrency control and recovery techniques for taking advantage of inherent timestamps and past data.

The management of time in real-world is often more complex than assumed by many researchers in this field. An organization must deal with future data and the uncertainties associated with future events. It also needs to store data under many alternative 'happenings' in the future. It must be able to interrogate its database with respect to some time instant in future. To support these requirements, we have explored modeling of 'branching' time for supporting future events

and alternatives. We are extending our historical data model and the query language for handling branching time.

Yet another practical aspect to be considered in building information systems that have a long life is to manage changes to database schema. A temporal database must support these changes as well as migration from one schema to the next. Some work on schema evolution and associated data management has been recently reported. Our work in this area is focused on broadening the extent of schema changes and on providing a seamless interface between versions.

1. Sarda, N.L., "HSQL : A Historical Query Language," Chapter 5 in *Temporal Databases : Theory, design and Implementation*, Tansel et al (ed), Benjamin/Cummings, New York, 1993.

2. Nadkarni, H. and Sarda, N.L., "Future Valid Time in Temporal Databases", Technical report, Dept. of Computer Science and Engg., IIT Bombay, June 1994.

3. Sarda, N.L., "Time-rollback Using Logs in Historical Databases", *Information and Software Technology*, Vol (35), No. 3, March 1993, pp. 171-180.

4. Jensen C.S. et al. (ed): "A consensus Glossary of Temporal Database Concepts", SIGMOD RECORD, Vol 23(1) March 1994, pp. 52-64. (Includes N.L. Sarda as one of the co-authors).

5. Sarda, N.L., "New Paradigms for Design of Information Systems", Key-note address at *International Conference on Management of Data 1993 (COMAD'93)*, Calcutta.

6. Tutorial on "Temporal Databases" by N.L. Sarda at DOOD'95 (4th Int'l Conf. on Deductive and Object-oriented Databases), Dec. 4-6, 1995, Singapore.

3 Distributed Database Systems

Principal investigator(s): N. L. Sarda

The advent of fast networks has heralded the entry of distributed databases into the commercial world. The relational vendors are once again showing a lot of interest in adding good distributed features like locational transparency, distributed transaction management etc. We have studied the issues of optimization of update queries in distributed databases, and strategies for allocation of rules in a distributed deductive database system.

While query optimization in distributed databases has been extensively studied, very little work has been reported in optimization of updates in distributed databases where fragments have qualifications associated with them. An update on global relation could decompose into insert in one fragment and delete from another fragment. We have proposed new operators in translation of update requests. The algebraic properties of these operators were studied for the purpose of identifying optimizing transformations. These led to formulation of criteria for efficient translation of global updates into updates on fragments.

We also studied the problem of allocation of rules to sites in a distributed database, based on minimization of communication cost during rule execution. The problem is formulated as a directed acyclic graph, where nodes represent rules or relations, and edges represent either dependencies between rules or between rules and relations. A number of heuristics are proposed and analyzed both for non-replicated and replicated allocation. We also attempt to characterize rulebase hierarchies in terms of height and inherent clusters, and study performance of algorithms for different types of rulebases.

1. Ravikanth K.V., Manohar R., Ratna Babu K., "Translation of Global Updates in Distributed Databases", *Proc. of Int'l Conf on Information Systems and Management of Data (CISMOD'93)*, New Delhi, Oct. 6-8, 1993.

2. Mohania M.K. and Sarda N.L., "A Selective Enumeration Based Heuristic for Rule Allocation", *Journal of Information and Software Technology*, Oct. 1994.

3. Mohania M.K. and Sarda N.L., "Rule Allocation in Distributed Deductive Database Systems", *Journal of Data and Knowledge Engineering*, Dec. 1994.

4. Mohania M.K. and Sarda N.L., "A Framework for Cooperative Deductive Database Systems", To appear in *Journal of Computers and Artificial Intelligence.*

5. Mohania M.K. and Sarda N.L., "Some Issues in Design of Distributed Deductive Databases", in *20th International Conf. on Very Large Databases*, Santiago, Chile, 1994.

4 Migrating Legacy Applications

Principal investigator(s): D. B. Phatak and N. L. Sarda

The goal of this project is to automate to the extent possible the process of migrating legacy COBOL

applications to a current environment like a relational database management system, by extracting the design information and reshaping it in an object oriented paradigm. We have developed tools that take as input COBOL sources, and

- generate data flow and control flow graphs
- extract data definitions and their dependencies,
- generate normalized relations given the appropriate functional dependencies
- throw up slices of COBOL programs corresponding to identified object attributes, that serve as a basis for writing methods.

The forward engineering is done using an existing commercial tool that generates embedded SQL programs to complete the migration to a relational back-end.

1. Baliga Ramdas, "Reverse Engineering Tools", Technical Report, Jan. 1995.
2. Harish Idnani, "Cobol Slicer and MERIT Data Dictionary", Technical Report, Jan. 1995.
3. Vernon, M., "Implementation of the data dictionary", Technical Report, Jan. 1995.

5 Benchmarking Relational Database Management Systems

Principal investigator(s): D. B. Phatak

Performance evaluation of a database management system is a very critical operation for database vendors to identify bottlenecks and evaluate new implementation algorithms. It is equally critical to the user who has to decide amongst multiple vendors. We have designed a benchmark suite which can generate realistic workload, varying the number of users, the types of queries running etc. An extensive data generator generates synthetic data with specified distribution, selectivity and inter-relationships, from DDL statements in SQL-2. The performance indices of throughput and turnaround-time are found to be particularly useful in a mixed load environment of OLTP and long queries, an issue not addressed by TPC suites so far.

1. Kekre A.A., Phatak D. B. "Design of a Benchmark for Performance Evaluation of Multiprocessor UNIX Systems for Data Processing Environment", In *Proceedings of International Conference on Management of Data*, (COMAD) December 1994.
2. Anirudhha Joshi, "Performance Measurement of Computer Systems", Technical Report, Jan. 1993.
3. Atul Pandit, "System Performance Evaluation through Benchmarks", Technical Report, Jan. 1995.
4. Ashish Sonkusare, "Performance issues in client server systems", Technical Report, Jan. 1995.

6 Object Oriented Databases

Principal investigator(s): S. Seshadri and S. Sudarshan

The object-oriented database market has been growing steadily for several years now, and the area has seen a lot of interest in the past few years. The long term goal of our object-oriented databases project is to develop a high-performance object storage system which supports parallel I/O as well as multiple processors in a shared-disk configuration. An implementation is in progress. The topics of research in this area include the following.

- Indexing for class hierarchies: We have proposed a new index structure called hcC-trees whose performance is superior to other structures proposed in literature. We have also proposed a concurrency control and recovery algorithm for hcC-trees.
- Clustering for data structures: Many new applications that use object oriented databases require new data structures like interval trees, segment trees, quad trees etc. However, these trees are designed as main memory structures and have many small nodes connected by pointers, and have relatively long path lengths. Efficient pagination of these data structures is crucial for efficient traversal of these structures if they are stored on disk with pages retrieved on demand. We have developed techniques for paginating such main memory structures to minimize path lengths for lookups, and have shown significant improvements over other clustering techniques.
- Declustering objects across multiple disks: A natural question that arises in parallel object oriented database systems is how to decluster objects across multiple disks. While declustering is a well studied topic in the context of relational databases, much remains to be studied in object oriented databases. We are investigating the problem of clustering objects into pages and

declustering pages across disks in a parallel object oriented database. Clustering and declustering decisions are made based on information about access patterns of the objects. The problem in full generality is a hard one, and we have developed several heuristics and are currently studying their performance.

- Buffer management: A good buffer manager can drastically reduce the number of page faults in a database system. We have analyzed the performance of various well known algorithms in an object oriented context and shown that they do not perform very well. We have also suggested some new algorithms which perform much better.

- Garbage collection: We are currently investigating efficient mechanisms for collecting garbage (objects not reachable from a persistent root) in an object oriented database system based on an augmented reference count mechanism that also detects cycles.

1. B. Sreenath and S. Seshadri, "hcC-trees: An Efficient Index Structure For Object Oriented Databases," In *Proceedings of Twentieth International Conference on Very Large Data Bases*, Santiago, Chile, September 1994.

2. S. Seshadri and B. Sreenath, "S-trees: A High Performance Index Structure For OODB," In *Proceedings of Fifth International Conference on Management of Data*, Calcutta, Dec 1993.

3. Sanjeeva Rane, S. Seshadri and C. Mohan. Concurrency Control and Recovery Algorithms for hcC-trees, Technical Report, Indian Institute of Technology, Bombay, February 1995.

4. A. A. Diwan, Sanjeeva Rane. S. Seshadri and S. Sudarshan. "Clustering Techniques For Minimizing External Path Length." Submitted for publication.

5. Rina Panigrahy, S. Seshadri and Sundar Vishwanathan, "Competitive Online Buffer Management." Submitted for publication.

7 Real Time Databases

Principal investigator(s): S. Seshadri

Real-time database systems are designed to handle workloads where transactions have completion deadlines and the goal is to meet these deadlines. The problems investigated include the following:

- Many real-time database environments are characterized by workloads that are a mix of real-time and standard (non-real-time) transactions. We have come up with a new database system architecture in which the real-time transactions use optimistic concurrency control and, simultaneously, standard transactions use locking. We prove that our architecture maintains data integrity and show, through a simulation study, that it provides significantly improved performance for the standard transactions without diminishing the real-time transaction performance. We have also showed, more generally, that the proposed architecture correctly supports the coexistence of any group of concurrency control algorithms that adhere to a standard interface.

- We have developed real-time variants of several classical B-tree concurrency control algorithms and compared their performance using a detailed simulation model of a firm-deadline real-time database system. The experimental results showed that the performance characteristics of the real-time version of an index concurrency control algorithm could be significantly different from the performance of the same algorithm in a conventional (non-real-time) database system. In particular, B-link algorithms, which are reputed to provide the best overall performance in conventional database systems, perform poorly under heavy real-time loads. We present and evaluate a simple load-adaptive variant of the B-link algorithm called LAB-link – this algorithm provides the best performance over the entire loading range for a variety of real-time transaction workloads.

- We are currently investigating the performance of commit protocols and concurrency control policies in a distributed real time database.

1. S. Seshadri, Shiby Thomas and J. R. Haritsa, On Integrating Standard Transactions Into a Real Time Database. To Appear in *Information Systems.*

2. S. Seshadri, Shiby Thomas and J. R. Haritsa, Extensible Concurrency Control Mechanisms In *Proceedings of International Conference on Management of Data*, Bangalore, India, December 1994.

3. Brajesh Goyal, Jayant Haritsa, S. Seshadri and V. Srinivasan, Index Concurrency Control in Firm Real–Time Database Systems In *Proceedings of Twenty First International Conference on Very*

Large Data Bases, Zurich, Switzerland, September 1995.

4. Jayant Haritsa and S. Seshadri, Real Time Index Concurrency Control, SIGMOD Record, To Appear.

8 Main-Memory Databases

Principal investigator(s): S. Seshadri and S. Sudarshan

Main-memory databases differ from disk databases in that data is completely (or almost completely) resident in shared main-memory. Such databases are important for real-time applications such as telecommunications and control applications. The design of the database has to provide support for versioning and user controlled concurrency control, to support real-time transactions. Processes must be allowed direct access to data, to avoid delays in a server process. Memory corruption becomes a possibility due to such open access, and hence error detection is a priority. Similarly, process failures must be detected and recovered from, so that the system is not brought to a halt by an errant process. The fact that data is memory resident can be utilized to significantly improve performance over disk databases, where accesses have to go through a buffer manager, and where dirty pages may have to be written to disk at any time to make space for new pages.

Work in this area is being carried out in association with researchers in AT&T Bell Labs., where the Dalí main-memory database project was initiated by S. Sudarshan and several colleagues, while Sudarshan was working there. Earlier work in the Dalí project had shown how to use characteristics of memory resident data to reduce logging overheads. We have explored techniques for versioning of data in a main-memory database. We are now working on extensions of the recovery algorithm to provide other services, such as operation logging to provide higher concurrency. We are also exploring query evaluation in a main-memory environment, with the goals of minimizing memory usage while providing good performance.

1. P. Bohannon, D. Leinbaugh, R. Rastogi, S. Seshadri, S. Sudarshan and Avi Silberschatz. "Logical and Physical Versioning For Real Time Readers in a Main Memory Database System." Submitted for publication.

9 Cost Estimation using Sampling

Principal investigator(s): S. Seshadri

Among the crucial inputs to a query optimizer are estimates of the number of distinct values in a column, the distribution of values of an attribute, and the size of various subqueries. We have proposed new sampling techniques and compared various techniques for estimating the number of distinct values in a column, the size of queries, and the percentiles of an attribute of a relation. Percentiles are useful for load balancing for parallel joins and parallel sorting. The sampling techniques that we have proposed are fast, accurate, and robust in the presence of correlations.

1. Peter J. Haas, Jeffrey F. Naughton, S. Seshadri and Arun Swami, Selectivity and Cost Estimation for Joins based on Random Sampling To Appear in *Journal Of Computer and System Sciences.*

2. Peter J. Haas, Jeffrey F. Naughton, S. Seshadri and Lynne Stokes, Sampling-Based Estimation of the Number of Distinct Values of an Attribute. In *Proceedings of Twenty First International Conference on Very Large Data Bases*, Zurich, Switzerland, September 1995.

3. Peter J. Haas, Jeffrey F. Naughton, S. Seshadri and Arun Swami, Fixed Precision Estimation of Join Selectivity In *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Washington D.C., May 1993.

10 Evaluation and Optimization of Queries

Principal investigator(s): S. Seshadri and S. Sudarshan.

The evaluation and optimization of complex queries has been an on-going area of research for S. Seshadri and S. Sudarshan. Areas studied in the past include evaluation and optimization of complex queries in deductive databases. Areas covered more recently include the following:

- We have investigated the algebraic optimization of SQL queries using the theta-semijoin operator, and shown its connections with Magic sets optimization, and have demonstrated its use in optimizing complex queries.
- Incremental evaluation of queries in the presence of updates is required for maintenance of materialized views as well as for verification of integrity constraints. We have shown how the maintenance of extra materialized views can speed up incremental evaluation significantly. This is particularly important since SQL-92 allows arbitrarily

complex queries to specify integrity constraints. We have also presented algorithms and heuristics for finding the optimal set of extra views to materialize and maintain.

- We argue that database compression is attractive from a query processing viewpoint also and should therefore be implemented even when disk storage is plentiful. Compression can therefore be a performance enhancement tool if implemented properly. We have developed a modified attribute level compression algorithm, based on non-adaptive arithmetic compression, called **COLA**. We demonstrate for a range of relational queries that *COLA* simultaneously provides good query processing performance and reasonable compression ratios.

- We are currently investigating the decomposition and the scheduling of a query in a parallel database management system.

1. S. Seshadri and Jeffrey F. Naughton, On the Expected Size of Recursive Datalog Queries. *Journal Of Computer and System Sciences*, Vol. 51, No. 2, Oct. 95.

2. Gautam Ray, Jayant Haritsa and S. Seshadri, Database Compression - A Performance Enhancement Tool In *Proceedings of International Conference on Management of Data*, Pune, India, December 1995.

3. Divesh Srivastava, Peter Stuckey and S. Sudarshan, "Optimizing Complex SQL Queries: The Magic of Theta-Semijoins", Submitted for publication.

4. Kenneth Ross, Divesh Srivastava and S. Sudarshan, "Materialized View Maintenance and Integrity Constraint Checking: Trading Space for Time." Submitted for publication.