Database Research at the IBM Almaden Research Center

Laura M. Haas, Patricia G. Selinger
IBM Almaden Research Center, San Jose, CA 95120

Introduction

Research on database at the IBM Almaden Research Center covers a broad range of topics. We feel the main challenges in the database area are to enable increased performance both for transaction processing and for complex, *ad hoc* queries, and to raise the level of abstraction provided by database systems, enabling databases to serve a broader range of applications. We are addressing the first with work in advanced database algorithms and on high performance architectures (including parallelism and large semiconductor memories). For the second we are exploring extensions to relational capabilities, investigating new, object-oriented approaches, and experimenting with a variety of technologies for some classes of applications.

Our work also ranges from pure exploratory to applied research. We are collaborating with IBM product development laboratories in a joint effort, the **Database Technology Institute**, that was established to promote the invention of database technologies and their rapid use in products and to carry out large scale database experimentation. The institute, located in our research center, brings together researchers and product developers to work on topics of mutual interest. Many of the advances covered below are the result of this joint work.

System Architecture and Performance

Database Algorithms

Every component of current database systems could be enhanced with faster, more efficient algorithms. We want to develop techniques that optimize performance globally rather than just at the component level. Our work on extensible database systems is giving us the opportunity to investigate new and better techniques, particularly for the areas of query analysis, optimization, and compilation. Some examples are rule-based optimization and magic set transformations on queries for more efficient execution. An example of work in the area of concurrency and recovery is a technique whereby record-level locking can be combined with write-ahead logging without risk of log overflow. We are also exploring database access methods that directly affect query execution performance. For example, we have improved the B-Tree access method and developed a new access method that extends the types of objects supported by traditional B-Trees by using highly compressed encodings ("signatures") of data.

Chang, W. and H. J. Schek, *A Signature Access Method for the Starburst Database System*, Proc. 15th VLDB, Amsterdam (Aug 1989).

Lohman, G., *Grammar-Like Functional Rules for Representing Query Optimization Alternatives*, Procs. ACM SIGMOD, Chicago (May 1988).

Lee, M., J.C. Freytag, and G.M. Lohman, *Implementing an Interpreter for Functional Rules in a Query Optimizer*, Proc. 14th VLDB, Long Beach (Aug 1988).

Mohan, C. et al., *ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging*, IBM Research Report RJ6649 (January 1989). To appear in *ACM Transactions on Database Systems.*

Mohan, C., *ARIES/KVL: A Key-Value Locking Method for Concurrency Control of Multiaction Transactions Operating on B-Tree Indexes*, Proc. 16th VLDB, Brisbane, Australia (Aug 1990).

Mohan, C., D. Haderle, Y. Wang, and J. Cheng, *Single Table Access Using Multiple Indexes: Optimization, Execution, and Concurrency Control Techniques*, Proc. Intl. Conf. on Extending Data Base Technology, Venice, Italy (Mar 1990).

Mohan, C., *Commit_LSN: A Novel and Simple Method for Reducing Locking and Latching in Transaction Processing Systems*, Proc. 16th VLDB, Brisbane, Australia (Aug 1990).

Mohan, C., and H. Pirahesh, *ARIES-RRH: Restricted Repeating of History in the ARIES Transaction Recovery Method*, Proc. 7th Data Engineering, Kobe, Japan (Feb. 1991).

Mumick, I., H. Pirahesh, and R. Ramakrishnan, *The Magic of Duplicates and Aggregates* Proc. 16th VLDB, Brisbane, Australia (Aug 1990).

Parallelism and Coupling

Large database systems supporting high rates of simple transactions have traditionally run on large computers, as have relational database systems supporting complex queries. Even the largest uniprocessor machines, however, will not be able to meet future demands for higher throughput of simple transactions or faster response time for queries of ever increasing complexity. We are studying two strategies to solve this problem: *parallelism* and *coupling*.

Parallelism that is achieved by partitioning the database over multiple processors, each with their own disks ("shared nothing" architecture), promises potentially linear speedup of complex query response time or nearly linear increases in transaction throughput. To achieve these gains, we must contend with data skew, either where some rows are much more frequently referenced than others (causing hot spots or even hot processors) or where the combination of query predicates and data correlations leads to unpredicted load imbalances between processors executing a query in parallel. We are investigating how data partitioning, optimization, compilation, and execution techniques can prevent as many of these situations as possible, how to recognize an imbalanced situation during execution, and how to correct it dynamically. Together with people from IBM's T.J. Watson Research Center, we are designing and building a prototype parallel system called Persist for a shared nothing architecture. We are also obtaining advice and assistance from database product developers in such non-traditional research areas as multi-system management, which we believe are very important to parallel systems.

We are also exploring coupling of systems through shared disk architectures for both complex query parallelism and increased throughput of simple transactions. We have studied how to decentralize important database functions such as concurrency, logging and recovery and how to maintain coherency across multiple buffers. For example, we are examining how to continue processing after one of N processors has failed. In such a case, the committed transactions from the failed processor must be redone (if necessary) by one or more of the N-1 working processors, and its uncommitted transactions must be rolled back. This is not hard if all work stops on N-1 processors while the recovery of the failed processor takes place, but we'd like to accomplish it while new work continues to be processed by the N-1 processors.

Both shared nothing and shared disk systems require high performance, parallel execution algorithms able to process not only complex queries on large amounts of data but also simple transactions affecting only one or two records. There are many open research topics: parallel algorithms for sort and join, static and dynamic data partitioning, task scheduling, load balancing, hot spot accommodation, efficient commit protocols, scalability, fault tolerance and query optimization.

Iyer, B. and D. Dias, *System Issues in Parallel Sorting for Database Systems*, Proc. 6th Intl. Conf. on Data Engineering, Los Angeles (Feb. 1990).

Lorie, R. and H. Young, *A Low Communication Sort Algorithm for a Parallel Database Machine*, Proc. 15th VLDB, Amsterdam (Aug 1989).

Lorie, R., J. Stamos, J-J Daudenarde and H. Young, *Exploiting Database Parallelism in a Message-Passing Multiprocessor*, IBM Research Report RJ8202 (Jun 1991).

Mohan, C. and I. Narang, *Recovery and Coherency Control Protocols for Fast Intersystem Page Transfer and Fine-Granularity Locking in Shared Disks Transaction Environment*, Proc. 17th VLDB, Barcelona (Aug 1991).

Stamos, J. and F. Cristian, *A Low-Cost Atomic Commit Protocol*, Proc. 9th Symposium on Reliable Distributed Systems, Huntsville (Oct 1990).

Pirahesh, H. et al., *Parallelism in Relational Data Base Systems: Architectural Issues and Design Approaches* Proc. 2nd Intl. Sym. on Databases in Parallel and Distributed Systems, Dublin, Ireland (Jul 1990).

Remote Site Recovery

Our research in this area is concerned with databases that must be kept running despite disasters or planned shutdowns, for example, an earthquake, or a power shutdown over a holiday weekend. Even if the production computing center is destroyed, no data should be lost, nor should a significant service disruption occur. Users of the system, such as reservation clerks or teller machine clients, must not be inconvenienced, regardless of what happens to the production site's database system or the machine(s) it runs on.

We have designed and built a manageable system called Lifeboat, which has a low overhead mechanism that supports very high transaction rates and high availability. We view remote site recovery as a natural extension of database and transaction recovery, but with new and unique problems. Our approach, which takes advantage of advances in fiber optics telecommunications technology, is to transport an active system's recovery log to one or more tracking systems at remote sites. The tracking sites are far enough away from the production system that they are unaffected by the same disaster or shutdown. The interesting issues we are addressing include real-time tracking and recovery of changes to data, and parallel catchup techniques to be used after the link between production and tracking systems has been cut and then restored. The main paths of the Lifeboat system are now running and will be installed as an experimental system in a bank so that we can validate and improve our techniques.

Mohan, C., K. Trieber, and R. Obermarck, *Algorithms for the Management of Remote Backup Data Bases for Disaster Recovery*, IBM Research Report RJ7885 (Dec 1990).

Burkes, D. and K. Treiber, *Design Approaches for Real-Time Transaction Processing Remote Site Recovery*, Spring COMPCON (Feb 1990).

Massive Databases

As the size of databases in many enterprises grows beyond current DBMS limits, issues such as performance, availability, cost of storage and addressing arise. For massive databases we are exploring such issues as how to address data beyond current limits, how to archive inactive data, building indexes while allowing concurrent updates, and allowing data access during re-organization. In the area of archiving inactive data, we are investigating automatic extraction of inactive data from active data, its storage on cheaper media, and how to process queries against it. One of the key challenges in using archival data is how to achieve query performance against data stored in possibly slower, mountable devices such as optical libraries.

Starburst

The goal of the Starburst project has been to build a relational database management system that can be easily extended to support applications and technologies not typically supported by today's relational systems, and that can serve as a flexible testbed to pursue our diverse research interests. In addition, we wanted to re-examine traditional system structures and algorithms, with the goal of improving their performance. Starburst is written in a mixture of C and C++, and runs under AIX on IBM's RT/PC and RS/6000 workstations. Designed to be a fully functional relational database management system, Starburst can be extended by highly skilled programmers with expertise in database systems. It is accessed using an extended version of SQL, including several advanced features such as user-defined functions and gigabyte-long strings. An extensible system structure provides powerful capabilities to extenders. While we are still working on a few missing pieces (such as a complete application programming interface, documentation, and features such as an extensible type system), the system is sufficiently operational for us to experiment with it, and we have already built and tested several extensions, including advanced features such as the Starburst long field manager. One extension, the so-called IMS attachment for support of clustering and parent-child links, was made by university researchers unfamiliar with Starburst, proving that Starburst is extensible by others.

The next few years should produce interesting results, as we use Starburst as a testbed for many of the other efforts described in this paper. Recursion, user-defined datatypes, complex objects, support for active database systems, main-memory database, parallelism and the optimization of large joins are some of the technologies we expect to investigate using Starburst. We are starting to explore the use of Starburst for searching large image databases based on image fragments ("query by image content") and are planning to investigate how Starburst's extension mechanisms can be used to support geographic applications.

Carey, M. et al., *An Incremental Join Attachment for Starburst*, Proc. 16th VLDB, Brisbane, Australia (Aug 1990).

Haas, L. M., et al., *Starburst Mid-Flight: As the Dust Clears*, IEEE Trans. on Knowledge and Data Engineering 2:1 (Mar 1990).

Haas, L. and W. Cody, *Exploiting Extensible DBMS in Integrated Geographic Information Systems*, Design and Implementation of Large Spatial Databases, Springer-Verlag, Second Symposium SSD Proceedings, Zurich (Aug 1991).

Lehman, T. and B. Lindsay, *The Starburst Long Field Manager*, Proc. 15th VLDB, Brighton (Aug 1989).

Lindsay, B. and L. Haas, *Extensibility in the Starburst Experimental Database System*, Database Systems of the 90s, Proc. of Int'l Symposium, Berlin, FRG, Springer-Verlag 466, A. Blaser, ed. (Nov 1990).

Lohman, G. et al., *Extensions to Starburst: Objects, Types, Functions and Rules*, Comm. of the ACM 34:10 (Oct 1991).

Improving the Quality of Data

One of the major themes of our research has been to provide better quality data to users. *Data quality* includes all those characteristics -- such as additional structure, semantics, behavior, reliability and performance -- that make data more useful, for more applications. Several of our efforts to enhance the quality of relational data by adding more behavior and semantics are sketched below.

An Extensible Type Facility

We are working on a user-defined type facility, known as Polyglot, that will capture the basic features of object-oriented technology, including data abstraction, subtyping, multiple inheritance, and encapsulation. Operations on objects (instances of a type) are defined by ***generic functions*** that can be implemented by a set of methods, each defined for a set of arguments of particular types. Unlike the methods of many object-oriented languages, Polyglot methods are ***multi-methods***, dispatched according to the types of all arguments, not just one. We have invented an algorithm to statically type-check these operations.

Polyglot is designed to be used from multiple application programming languages. One unusual feature of Polyglot is that it makes the behavior (type-specific operations) of database types available both in the database (within queries) and in the application program. Instances of database types can also be converted into application language representations if desired, so that existing libraries of functions in that language can be used. Polyglot also aims to be ***platform neutral***: while it is being prototyped initially in Starburst, it should be possible to incorporate Polyglot into other relational and non-relational systems as well.

Agrawal, R., L. G. DeMichiel and B. G. Lindsay, *Static Type-Checking of Multi-methods*, Proc. ACM OOPSLA, Phoenix (Oct 1991).

Support for Complex Objects

Many applications need to be able to store and manipulate objects with complex structures. In a relational system, these complex objects may span records of one or more relations. We will support such objects in Starburst through an extension to the query language to allow the return of structured query results. Objects may be defined intensionally by these structured queries. This query language extension is known as ***XNF***, for ***eXtended Normal Form***. An XNF query consists of (1) definitions of the component tables (known as ***nodes***) and (2) definitions of directed relationships between component tables. The component tables are defined using standard (i.e., flat) queries. The relationships are defined by predicates that link a parent node with zero or more elements from a child node, using left outer join semantics. These resulting nodes and relationships must form a connected acyclic graph, with a unique root. XNF queries can be combined (to form new composite objects), projected, and restricted. XNF objects can be traversed using cursors, and loaded into memory to be navigated by the application. We intend to support caching of materialized objects for fast access, and to investigate versioning, concurrency control, and other aspects of object management. Further research will look at providing a type system for these objects with many of the object-oriented benefits described above.

Active Databases

The Starburst Rule System

Starburst supports set-oriented production rules that allow users to define a series of actions to be performed in response to specific changes in the stored data. The Starburst rules respond to aggregate or cumulative state changes -- sets of changes, in keeping with the spirit of the relational model. Rules are associated with a table and have three parts: a trigger, a condition and an action. The ***trigger*** is a list of operations on the

table (e.g., **inserted, updated**). The occurrence of one of these operations indicates that the associated rule may be applicable when rules are next evaluated (asserted). The ***condition*** is a predicate over the database state and over logical "transition tables" that encapsulate changes to the trigger table that have occurred since the last time the rule was considered. If the rule condition is satisfied, the rule's list of ***actions*** is executed. Actions may be any database statement and may refer to transition tables. Arbitrary sets of rules may be defined. Rule processing is fully integrated with query and transaction processing.

We are currently using production rules to support referential integrity in Starburst. We plan also to use them for a general-purpose dependency tracking system. For example, when an index referenced in a query plan is deleted, the query should be automatically recompiled and a different plan should be chosen. Other applications of production rules include enforcing constraints, maintaining materialized views, and simulating deductive database capabilities. We are experimenting with these and other applications of our rule system.

Ceri, S. and J. Widom, *Deriving Production Rules for Constraint Maintenance*, Proc. 16th VLDB, Brisbane (Aug 1990).

Ceri, S. and J. Widom, *Deriving Production Rules for Incremental View Maintenance*, Proc. 17th VLDB, Barcelona (Aug 1991).

Widom, J. and S.J. Finkelstein, *Set-Oriented Production Rules in Relational Database Systems*, Proc. ACM SIGMOD, Atlantic City (May 1990).

Widom, J., B. Lindsay and R. Cochrane, *Implementing Set-Oriented Production Rules as an Extension to Starburst*, Proc. 17th VLDB, Barcelona (Aug 1991).

Alert

Alert is an extension architecture for experimentation with active databases. It provides a layered architecture that allows the semantics of a variety of production rule languages to be supported on top. Triggers may be specified on user-defined and built-in operations, and on abstract objects (e.g., views) as well as on tables. Both synchronous and asynchronous event monitoring are possible. Alert, like the production rules facility described above, is implemented as an extension to Starburst, and offers another approach to providing active database capabilities.

Alert is an extension of passive DBMS facilities, and re-uses as many of these as possible. Events are modeled as tuples in special "active" tables. Thus much of the standard storage management for tables can be used to keep track of sets of events. Efficient event detection is provided through indexes and query optimization techniques. Alert is integrated with concurrency control and recovery for use in a shared environment. Future research in this area will explore additional mechanisms for efficiently monitoring changes, multiple query optimization of active queries, and parallel execution of conditions and actions.

Schreier, U., H. Pirahesh, R. Agrawal and C. Mohan, *Alert: An Architecture for Transforming a Passive DBMS into an Active DBMS*, Proc. 17th VLDB, Barcelona (Aug 1991).

Support for New Application Areas

Querying By Image Content

Today's hardware technology enables us to acquire, store, manipulate and transmit large numbers of images. Applications are amassing vast image databases that are expected to grow in importance and volume in the near future. Yet we lack a natural means of querying and accessing these image databases. In addition to the simple text-based queries that can be handled today, we wish to allow users to efficiently search through very large image databases using sketches, layout or structural descriptions, texture, images, and other iconic and graphical information to specify the images desired. We also wish to provide search by similarity, rather than only by exact matching. A typical query might be to find all images with a pattern similar to the example provided by the user. Patterns might include part of another image (e.g., a photograph of a face or an X-ray image), a menu of candidate patterns (e.g., of textile texture), or a user-drawn sketch (e.g., of a dress, building, fabrics, ancient vase, etc.).

We want to provide truly integrated image databases with a completely new and more powerful query capability that will enable much wider and more efficient use of computers in handling the vast amount of image data that we can collect and store today. However, fundamental problems must be solved before this is feasible. We need to be able to represent images in a way that captures more of their semantic content than the raw bits, making it possible to search efficiently. This efficiency is essential, as there will be many images (potentially millions), each quite voluminous in stan-

dard (i.e., pixel) representations. We also need to understand what it means for one image to be "similar" to another. This will typically vary depending on the application, user and the context of the query, so in addition to the set of measures to be used, we must also investigate how to interact with the user to determine an appropriate measure for a particular query. Since these problems are difficult, we will aim to discover the general principles, but at the same time we will identify target application(s) for which we will prototype, evaluate, and test concrete pilot system(s). Possible applications include: education, journalism, museum cataloging, document processing, biomedical, etc. We are currently investigating applications and looking for partners with expertise in these areas.

Support for Database Mining

Database mining refers to extracting and using information from historical data to aid decision making, and is emerging as a major application area for databases. Cited applications for database mining include target marketing, customized newsletters and magazines, credit approval, fraud detection, franchise planning, market projections, order planning, etc. Databases involved in these applications are mammoth.

In Quest, a project on database mining, we are investigating how database technology should be enhanced to address this problem. We are studying representative database mining problems to develop appropriate abstractions and approaches. We have identified classification, queries on sequences, successive query refinements, query language extensions and optimization as some of the key technical issues in this area. We have developed an algorithm for classification based on sample instances that is particularly suited for interfacing with databases. We have also developed an algorithm for discovering rules (patterns) in a historical database of customer transactions.

Information Overload

Software technology has not evolved to accommodate the rapidly increasing volume and diversity of on-line data. Finding data is often very difficult, and once found, data from different sources is often difficult to combine because of a lack of common formats and semantics. The advent of distributed systems has added to the problem by increasing the amount of on-line data that is readily available.

Rufus

To address these problems, the Rufus project has created a system that has an understanding of a user's data and that provides the user with an integrated view of that data. In particular, users are offered the ability to search over their data, using properties of the data to guide their search as appropriate, and to operate on data using type-appropriate operations. Rufus does not disturb existing data and applications. Attributes are extracted from a user's data into an object-oriented data repository without affecting the native files. Rufus provides tools to search and organize data across the extracted structures, and provides connections among these extracted objects, the original data files and the applications that operate on the data.

To do this, we have developed and prototyped an object-oriented data model, storage system, and associated search and display methods for a variety of user data types (around 20 so far). A user's data files are automatically classified and type-specific attributes are extracted and imported into the Rufus storage system. We have also built a general purpose X-windows application for browsing, filtering, searching and acting on data objects. We plan to add library support for more specialized applications. The prototype runs on an IBM RISC System/6000 workstation on top of the AIX operating system.

Melampus

Melampus has taken a revolutionary approach to the problem of information overload. One contributor to the problem is the lack of a comprehensive model in which to describe and manipulate the range of entities in a system. The overall goal of Melampus is to define such a data model, to build a prototype system that implements the model, and to build sample applications that highlight both strengths and weaknesses of the model. We hypothesize that by providing a system-wide framework in which to describe the structure and behavior of data and by providing efficient associative access to all entities, we will significantly improve the ability of users to exploit the system. By providing global system coherence, we will enable the use of data in unanticipated ways, allow rapid retrieval of data, ease the formation of new relationships among data, and promote the sharing of data between applications.

To date, we have defined an initial data model and query algebra, with special mechanisms to accommodate evolution and type migration. We are currently focused on defining the overall architecture of the sys-

tem, and are investigating access control mechanisms for such a system.

Richardson, J.E. and P.M. Schwarz, *Aspects: Extending Objects to Support Multiple Independent Roles*, Proc. ACM SIGMOD, Denver (May 1991).

Richardson, J.E. and P.M. Schwarz, *MDM: an Object-Oriented Data Model*, Proc. 3rd Workshop on DB Programming Languages, Nafplion, Greece (Aug 1991).

Heterogeneous Distributed Databases

We have a long history of research in distributed relational databases, beginning with the *R** research project which was aimed at achieving both site autonomy and location transparency for homogeneous distributed databases. We are now addressing similar issues for heterogeneous distributed relational databases, a much tougher challenge. The problems include naming, partitioning, replication, query optimization, compilation, execution techniques, deadlock detection and transaction management.

Choy, D. and P. Selinger, *A Distributed Catalog for Heterogeneous Distributed Database Resources*, IBM Research Report RJ8109 (May 1991).