Database Systems — Breaking Out of the Box*

Avi Silberschatz
Bell Laboratories
Murray Hill, NJ 07974
avi@bell-labs.com

Stan Zdonik
Brown University
Providence, RI 02912
sbz@cs.brown.edu

1 Introduction

The field of database systems research and development has been enormously successful over its 30 year history. It has led to a $10 billion industry with an installed base that touches virtually every major company in the world. It would be unthinkable to manage the large volume of valuable information that keeps corporations running without support from commercial database management systems (DBMS).

Today, the field of database research is largely defined by its previous successes, and much current research is aimed at increasing the functionality and performance of DBMS's. A DBMS is a very complex system incorporating a rich set of technologies. These technologies have been assembled in a way that is ideally suited for solving problems of large-scale data management in the corporate setting. However, a DBMS, as does any large tool, places some requirements on the environment in which it is being used. The DBMS will impose some execution overhead, often requires a fairly high level of expertise to install and maintain, and only manages data that is in fairly specific file formats.

At the same time, the data that needs managing is changing radically and is being stored in places other than database systems (e.g., files). It is also obtained in large volumes from external sources like sensors. While the trend of building more powerful database management systems has a place, there is also a need for data management in contexts that cannot cope with the overhead of a full-blown DBMS; many environments call for a much lighter-weight solution.

Sometimes, instead of using an existing tool in a new application, it is better to embed reusable components in order to make the resulting system more responsive. In some cases, it is the techniques that a tool embodies that are most reusable. We argue that this observation is true in many new data intensive applications. We would like to reuse database system components, but when that is inappropriate we must be willing to reuse our techniques and our experience in new ways.

If we look around at information that people use, we see many examples in which database systems are conspicuous by their absence. One of the most compelling examples is the World Wide Web. While

*Participants in this workshop were José Blakeley, Peter Buneman, Umesh Dayal, Tomasz Imielinski, Sushil Jajodia, Hank Korth, Guy Lohman, Dave Lomet, Dave Maier, Frank Manola, Tamer Ozsu, Raghu Ramakrishnan, Krithi Ramamritham, Hans Scheck, Avi Silberschatz (co-chair), Rick Snodgrass, Jeff Ullman, Jennifer Widom, and Stan Zdonik (co-chair).

it is true that DBMS vendors are making their products *web-enabled*, the approach they are taking is to provide better web servers. This capability is only a very small step in the direction of managing the huge volume of nonstandard data that exists on the web. It is doubtful that this move will cause the hundreds of thousands of web sites to shift to the use of a full-featured database system whose target market is business data processing.

Other examples of applications that could benefit from data management techniques but that typically do not make heavy use of database products include personal information systems, news services, and scientific applications. In the case of personal information systems, one only has to think about the information that is found on the typical PC. Electronic mail is of great personal value to many users, but when messages are saved, they are most often stored in the file system. It would be extremely useful to have DBMS facilities such as indexing and querying available for use on e-mail. While some support for a more organized approach to storage and retrieval of e-mail is emerging (e.g., Lotus Notes), sophisticated querying is not well developed.

Other recent reports [Gray95, SSU91, SSU95] have charted the course of database research and did an excellent job prioritizing current research topics and delineating new influences with respect to their impact on the database system industry. This report takes a somewhat different tack. Our theme is that database research should be devoted to the problems of data management no matter where and in what form the data might be found. We should not be defined strictly by the current product space or by the commonly held notion that our job is to manage very large collections of structured records within a controlled environment. Instead, we should apply our skills to new data management environments that potentially require radically new software architectures.

2 Background

The database field was born in the late 60's with the release of IMS, an IBM product that managed data as hierarchies. While hierarchies later proved to be too restrictive, the key contribution of IMS was the widespead revelation that data has value and should be managed independently of any single application. Previously, applications owned private data files that often duplicated data from other files. With a DBMS, data should not be logically replicated, making it easier to maintain. Creating shared databases required analysis and design that balanced the needs of multiple applications, thereby improving the overall management of data resources

Both the IMS data model and its most well-known successor, CODASYL, were based on graph-based data structures. While the idea of traversing links was intuitively attractive, it made it difficult to express database interactions independently of the actual algorithms that were need to implement them.

In 1970, Ted Codd published a landmark paper [Codd70] that suggested that data could be managed at a much higher level by conceptualizing it in terms of mathematical relations. This paper sparked a great deal of interest within the research community thoughout the 1970's to make this notion practical. At this point in time, the relational model is most commonly supported among commercial database vendors.

Because of its simplicity and its clean conceptual basis, an active theoretical community developed around the relational model. This community has contributed many important results including database design theory, a theory of query language expressibility and complexity, and an extension to relational languages called Datalog. Theoretical work continues in many forms including topics such as constraint databases and queries with incomplete information.

In the early 80's, a new data model emerged that was based on object-oriented programming principles. The object-oriented data model was the first attempt at providing an extensible data model. Data abstraction was used to allow users to create their own application-specific types that would then be managed by the DBMS. In the last five or six years, several object-oriented database companies have emerged and a committee made up of representatives from these vendors has produced a standard (ODMG). More recently, a hybrid model, commonly known as the object-relational model, has emerged that embeds object-oriented features in a relational context.

The use of objects has also been demonstrated as a way to achieve both interoprability of heterogeneoud databases and modularity of the DBMS itself. The object model provides very powerful tools for creating interfaces that do not depend on representational details. Heterogeneity in object representations can be paved over by overlaying an object-oriented schema on top of the actual stored data. DBMS modules can be descibed in object-oriented terms making them easier to export to other systems.

3 Our Skills

Database management systems have been largely concerned with the problems of performance, correctness, maintainability, and reliability. High performance must be achievable even when the volume of data is far greater than what fits in physical memory, and even when the data is distributed across multiple machines. Correctness is achieved by the enforcement of integrity constraints (e.g., referential integrity) and by serializable transactions. Maintainability is achieved by separation of logical and physical data structures, as well as by a large collection of tools to facilitate such functions as database design and system performance. Reliability is typically provided by combining a mechanism, such as write-ahead logging, with transactions that can maintain data consistency in the face of hardware and software failures.

Database research and development has explored these problems from the point of view of relatively slow-memory devices that must be shared by multiple concurrent users. Database systems have also developed in a context where there is no control over the execution of their clients. This approach has led to a particular set of skills and techniques that can be applied and extended to other problems. We describe these skills and techniques below.

- **Data modeling.** A *data model* consists of a language for defining the structure of the database (data definition language) and a language for manipulating those structures (data manipulation language, e.g., a query language). A *schema* defines a particular database in terms of the data definition language. By requiring that all data be described by a schema, a DBMS creates a separation between the stored data structures and the application-level abstractions. This *data independence* facilitates maintenance, since stored structures can be changed without any impact on applications.

  A good data model should be sufficiently expressive to capture a broad class of applications, yet should be efficiently implementable. While the relational model has dominated the field for the last decade, there is clear indication that more powerful and flexible models are required. The design and use of data models are important topics of study within the database community, and the extension of these models to incorporate more challenging types such as spreadsheets and videostreams is an important line of study for future applications.

- **Query languages.** A query is a program that is written in a high-level language to retrieve data from the database. The structure of a database query is relatively simple, making it easy to understand, generate automatically, and optimize. Many modern query languages (e.g., SQL) are declarative, in the sense that they express what should be returned from the database without any reference to storage structures or the algorithms that access these structures. Since implementation choices cannot show through at the query level, the query processor is free to choose an evaluation strategy. Moreover, the separation of request from implementation means that the storage structures can change without invalidating existing query expressions.

- **Query optimization and evaluation.** Relational databases became a commercial reality because of the maturation of optimizers for relational query languages and the development of efficient query evaluation algorithms. The ability to compile queries into a query execution plan based on the form of the query as well as the current storage structures on the disk is an important part of database system development. Optimization technology is particularly important for data retrieval and manipulation, whenever the stakes of picking an inefficient strategy are high and whenever the environmental conditions on which the execution plan are based may change.

- **State-based views**. It is possible to define a restricted and possibly reorganized view of the database using the query language. These state-based views are often used to limit the access to data. For example, we can limit use to a view containing the average salary by department excluding departments with fewer than three employees. In file systems, authorization is typically handled by access privileges associated with each file independent of the contents of the file.

- **Data management.** Database systems have always paid special attention to the automatic maintenance of data structures like indices and the efficient movement of data to and from system buffers. Typically these data management techniques are highly tuned for the particular storage devices involved. This approach to careful resource usage can be extended to other areas in which the devices include things like communication links and tertiary storage.

- **Transactions.** The database community developed the notion of transactions as a response to correctness problems introduced by concurrent access and update. By adopting a correctness criterion based on atomicity, transactions simplify programming, since the programmer need not worry about interference from other programs.

  The transaction has also been used as the unit of recovery. Once a transaction is committed, it is guaranteed to be permanent even in the presence of any hardware or software failure.

  Recently, other looser notions of transaction have been investigated. These typically are based on a user-supplied notion of correctness.

- **Distributed systems.** Database systems must deal with the problems introduced by having data distributed across multiple machines. The *two-phase commit* protocol allows systems to retain the advantages of atomic transactions in the face of distributed and possibly failure-prone activities. Other areas that have been studied in the distributed context include query processing, deadlock detection, and integration of heterogeneous data.

- **Scalable systems.** Databases have always been concerned with very large data sets. For the most part, database systems have been tuned to efficiently and reliably handle data volumes that exceed the size of the physical memory by several orders of magnitude. It is primarily for this reason that database systems have been successful in real commercial environments.

This list is not meant to be exhaustive, but rather illustrates some of the major technologies that have been developed by database research and development. Researchers have also investigated other areas including active databases and data mining.

4 Scenarios

In this section, we describe two applications of database technology that illustrate the directions we are advocating in this report. These are meant to be suggestive only. We believe the capabilities represented here point the way for future data management systems, and the technology that would support these scenarios constitutes a research agenda for the next decade.

4.1 Instant Virtual Enterprise

An "instant virtual enterprise" (IVE) is a group of companies, that do not routinely function as a unit, coming together to respond to a customer order or request for proposal. Computer integrated manufacturing (CIM) is a prominent example of an environment requiring IVE cooperation. The CIM environment encompasses many dedicated departments and subsystems. The engineering side includes computer-aided design, production, and quality assurance while the administrative side includes product planning, production control and resource management. Dedicated subsystems belong to different organizations, each with its own user interface, data model, specialized operations, and storage organization.

In many business areas, it will be necessary for the companies in an IVE to exchange and cooperatively manage large amounts of data. It is unlikely that the information systems will be integrated with each other at the time a decision is made to collaborate on an offer or a bid. Even within one CIM company, many heterogeneous databases will exist. Yet, sharing and exchanging data between the participating organizations and coordinating this information is critical.

In the following we present an example scenario of a CIM IVE. We then use examples from this scenario to illustrate areas where database functionality is needed for data that is not necessarily under the aegis of a DBMS.

Company A is building an oil pipeline and needs 600 large-diameter valves for the project. They solicit bids by issuing an RFP specifying dimensions, coupling mechanism, operating temperatures, pressure ranges, corrosion resistance and so forth. Company Q, an engineering firm, wants to put together an IVE to respond to the RFP. Engineers at Q use the Internet to search for companies that already have a design for a similar valve that can be used. It turns out that Company R is willing to license such a design. Company Q plans to do the design modification work themselves, but will contract with Company S to do an engineering analysis of the resulting design and convert the design to manufacturing plans. Company T is brought in to do the actual fabrication, but will contract out to Company U for die-making and casting. Finally, Company V and Company W will also cooperate: Company V provides a design file conversion service; it will be used for converting design files for the CAD package that Company R uses into the format for the CAD system that Company Q plans to use. Company W provides a documentation and archiving service for documents such as instruction and maintenance manuals.

We now give examples of the kinds of database capabilities needed here, both in putting together the bid, and in fulfilling the contract (if awarded).

When Company Q looked for an existing design of a valve, they were executing a query. A number of aspects of this query are particularly challenging: parts of it are based on closest match rather than exact match; the query asks about designs from many companies which presumably reside in many different repositories; and the design from Company R may not be stored in a DBMS, but rather in individual files for which there might not be the analog of a database schema. Similarity search requires sophisticated indexing, based on many descriptors and high-dimensional feature vectors. This aspect is already challenging. Moreover, the interesting point here is that we must provide query and indexing support on external objects. Whatever sophisticated index support we invent must keep track of changes of external objects and keep the index consistent with them.

In order for Company S to estimate a cost for engineering analysis and manufacturing plans, they need to see the original design, but in a form compatible with their tools. Thus, in putting together a bid, there is a need for data translation services such as those provided by Company V. However, they need to know the format of Company R's design files. It is possible those files are in a self-descriptive data interchange format, but it is also possible that descriptive information will have to be added. Often standards such as STEP/EXPRESS are used for the description and for the exchange of CIM product data. However, additional mechanisms must be provided in order to allow Company R to restrict the information to be given out to a "need-to-know" subset of the schema. We can hardly imagine that Company R will give away all the details just for the purpose of putting together a bid.

If the bid is awarded to the IVE led by Company Q, there will be a need for coordination and configuration management, as the original design is initially modified to meet specifications, and then further modified based on analysis by Company S and feedback on manufacturability by Company T and Company U. Various dependencies between data in the different IVE companies must be coordinated (i.e., coordination between objects in different subsystems). Relationships and (referential) integrity constraints must be modeled and maintained without requiring a traditional, global database. Changes to an object in one subsystem require changes to one or more related objects in other subsystems. Changes in external systems need to be monitored and potentially propagated to other systems. For example, if Company U changes the spindle of the valve, the related documentation about the valve must be changed by Company W. Access to the spindle in Company U might be restricted until the documention is updated by Company W. Again, only "need-to-know" information (i.e., information necessary to update the documentation) is exported by Company U.

Assume that Company Q decides to replace the spindle t provided by Company T by a spindle t' of another company T', because t' is equivalent in some sense but cheaper. This change may cause changes in all valve type designs where t was used resulting in a conflict between the marketing decision of Company Q and the design activities of Company S. Supporting actions to resolve such conflicts is critical to the IVE. For example, in this case, a decision must be made to determine whether all valve type designs must change and t' used instead of t, or whether some valve designs might continue without change, making re-negotiations necessary with T. Monitoring relevant changes and detecting conflicts is the DB functionality to be used here.

While the IVE operates, there is also a need for security and access control over the information. For example, it may be the case that Company R and Company T are competitors, so R is willing to let Q and S see the original design, but does not want T to have access.

Finally, Company A needs assurance that information on design and manufacturing of the valves is available even after the IVE disbands. Thus, there is a need for archiving of information that is possibly independent of any of the IVE companies. Such archiving is a database.

4.2 Personal Information Systems

A *personal information system* is a system that providing information tailored to an individual and delivered directly to that individual via a portable, personal information device (PID) such as a personal digital assistant, handheld PC, or a laptop. The PID can be either carried by the individual or mounted in an automobile, and will be equipped with a wireless network connection. It will also have network ports for "plugging in" when a stationary network connectionis available.

A user equipped with a PID will, in the near future, have access to the Internet from anywhere at any time. However, the characteristics of the physical link will vary widely in terms of characteristics such as bandwidth (several kilobits/sec to several megabits/sec) and prevailing error rates. Tariffs and charging schemes for information will also vary widely; some providers may charge per packet while others may charge by connection time. In addition, the method of information delivery will cover a wide spectrum of possibilities from periodic broadcast (satellite networks, pointcast, etc.) to standard, request-driven, client-server scenarios. Also, global positioning systems will be widely available, and there is every reason to assume that in a few years every laptop will have a GPS card. Thus, location will become an important parameter in selecting information, especially for location-dependent information services.

We envision a personal information service to be tightly integrated with an individual's activities from the time of waking up in the morning, through the person's daily activities, up to bed time. These services would work on behalf of the person even while asleep.

In the morning, the services could include a local weather report, a list of reminders about special events of the day (such as birthdays or anniversaries of friends and relatives), a list of morning work meetings and appointments (e.g., dentist), and suggested diet for the day from a personal health advisor.

Delivery of personal information services will continue as the person commutes to work. The PID can provide the best route from home to work based on up-to-date traffic conditions with expected delays displayed on a city map (the best route may include a combination of private as well as public transportation). It may provide personalized news headlines from national newspapers such as the New York Times and the Wall Street Journal as well as international headlines from papers such as The Globe and Mail; on Mondays, the report will include a summary of international, weekend sporting events (e.g., Italian soccer league). It could provide a personalized investment report, with recommended investments for that day provided by a personal financial advisor. By the time the person arrives at work, he is completely up to date on the events and news of interest.

Personal information services will continue throughout the day. Upon arrival at the office, the services could deliver a list of tasks for the day, a list of customers to contact, a reminder to set up an appointment for a periodic dental examination, a summary of breaking news of interest, information about the start of a sale from a local furniture store on a particular piece of home furniture, and a notification about the best airplane ticket to purchase for an upcoming vacation. If the person drives anywhere during the day, the services will provide best driving routes always based on up-to-date traffic information.

At the end of the day, the personal information service will provide a preview of the next day's activities, the person's daily diet balance statement from the personal health advisor, as well as appointments, and activities for the next day.

The PID must continuously query remote databases and monitor broadcast information. Thus, personalized information systems will magnify today's client-server performance, scalability, and reliability problems. Servers that both disseminate (push) information to clients and respond to client requests (pull) will play an important role in the delivery of personalized services. The load on these servers will potentially be much higher and the requests will likely be more sophisticated. Among the architectural problems that arise in an environment like this are questions of whether data should reside on the PID or on the server, and which tasks should be performed on the PID as opposed to the server.

5 Barriers

A DBMS provides a tightly controlled and highly uniform environment. All access to data passes through the upper levels of the system, making it relatively easy to control all occurrences of certain classes of event. For example, updates can be detected easily, thus making index maintenance manageable. The layout of data in files is known, the contents of data buffers is under strict control of the DBMS, and all stored data corresponds to an explicit schema.

Life becomes more complex when we try to provide database functionality outside of the confines of a DBMS. We are talking about moving to an environment in which there may be no central point of control and in which there may not be a great deal of uniformity. Thus, conventions and assumptions that held in a DBMS and could be exploited by its components now need to be negotiated. Similarly, uniformity either must be discovered post-facto, or new ways of providing functionality that can cope with variance must be devised.

In order to address adequately he vision represented in our scenarios, a number of technical barriers must be removed. These barriers typically result from new application requirements that have not yet been addressed or from the need for new DBMS architectures. In this section, we will outline some of these barriers.

5.1 Overhead

A modern DBMS is a software engineering tour de force. It represents hundreds of person-years of effort and a very mature technology base. Managing a corporate information system without such a device would be folly. Creating a special-purpose DBMS is typically an unjustifiable investment.

However, many application builders are turning their backs on this industry because the modern database system is a heavyweight resource. The overhead in terms of system requirements, expertise, planning, data translation, and monetary cost is too great for many emerging applications. For example, a builder of a personalized newspaper service might choose not to use a DBMS because she has no need for many of the advanced features. They are only interested in doing filtering of stream-oriented data (as is done, for example, in the wire services).

A subset of the traditional database services are needed, though, by many new applications. An ideal world would offer a collection of database modules that one could mix and match to produce a configuration that is as lean or as full-featured as is needed. For example, a wire service only needs a common data model and stream-based querying.

5.2 Scale

The database environments of interest require us to rethink expectations concerning size. Some applications manage quite small databases for which the management overhead of a full DBMS is overkill. Indeed, in many instances the benefits of a DBMS are not used simply because the overhead of the DBMS is too large.

At the other end of the spectrum, the volume of data in future applications may be many orders of magnitude greater than what database applications routinely deal with today. If we are going to locate information on the Internet, we must be prepared, at least conceptually, to handle many petabytes of data. This data grows at unpredictable rates over which we have no control.

The number of client and server sites is also many times greater than what we have dealt with in any corporate network. In current client-server systems, there are typically a very small number of servers (often one) to supply data to a modest client population. In our scenarios, there could be a hundreds of thousands of servers, and the client population could be even larger.

Distribution patterns in the new world are more geographically dispersed than anything we are used to. Information suppliers could be anywhere in the world. The unrestricted use of sites in distant places means that the cost of accessing an information source can depend on the available bandwidth into and out of those sites. This effective bandwidth can vary depending on the time of day and the popularity of the site.

Since all of these parameters create an optimization nightmare, it will become imperative that large unrestricted searches of many sites are avoided. Instead, it must be possible to precompute much of the information of interest and store it in a few more convenient places.

In the personal information system scenario, it is clear that servers will need to handle several orders of magnitude more requests than do today's servers. Consider a personal information device in every car continuously requesting information from a server or servers geographically distributed in a city. Robust and scalable server designs, where the volume of requests handled increases with the amount of server resources available, will be needed.

Occassionally, there will be servers that become hot spots, such as the 911 emergency service or a server close to a football or baseball stadium (overloaded when there is a game). In such cases, broadcast rather than point-to-point communication may be an alternative to satisfying commonly expected requests while reducing the workload on the server. Understanding when to broadcast, how to organize a broadcast, and how best to use local client memories become important issues.

5.3 Schema Organization

The standard database paradigm involves first creating a schema to describe the structure of the database and then populating that database through the interface provided by the schema. The DBMS maps the input data to actual storage structures.

Increasingly, we will no longer have this luxury of an a priori schema. Many applications currently create data independently of a database system [1] (e.g., scientific applications), and as information gets easier to collect, transmit, and store, this mismatch will only get worse. Thus, there is a need for externally generated data to be mapped to a schema (and possibly to new storage structures) after the fact. This bottom-up approach to populating databases is not often supported in current systems. It is crucial, however, to provide simple mechanisms for making foreign data sources available to database systems in order to realize something like the IVE scenario. Such a facility involves complex mapping procedures. We are talking about creating what is, in effect, a database view of the foreign data, but the view must be constructed over data in arbitrary formats.

The data that is received from a source like a Web site may appear to have some structure. Pieces of text are coded with tags describing their role. Unfortunately, the use of these tags may be quite varied. The fact that one page uses an *H3* tag for headings does not necessarily carry across to other pages, perhaps even from the same site. This variation in the use of text coding makes it difficult to construct something we normally think of as a schema to describe things like web pages.

Moreover, as new data is added to these data sources, we may find that a schema is incomplete or inconsistent. Thus, the current rigidity of database schemas becomes an impediment to using database systems to address the needs of many information systems. We need schema management facilities that can adapt gracefully to the dynamic nature of foreign data. Moreover, the schema must allow different formats and different sets of properties for the data as it appears in the DBMS.

5.4 Data Quality

Information accessed from a widearea network may be of varying quality. Quality relates to the timeliness, completeness, and consistency of the data. Future information systems must be able to assess and react to the quality of the data source. Often the source of the data will give clues regarding data quality. Quality-related metadata must be captured and processed in a way that is as transparent as possible to the user.

Current database technology provides little support for maintaining or assessing data consistency. Constraint maintenance in commercial systems is limited to a few simple constraint types such as the uniqueness of keys and referential integrity. Even if there were a way to include a quality metric with

[1] This situation is a major reason why a large fraction of these applications do not use database systems today.

data values, there is no way in current systems to include it in processing the data from disparate sources. For example, we might not want to have two values participate in a join if their quality metrics are significantly different.

5.5 Heterogeneity

The database community has recognized for some time that data exists in many forms. These dissimilar formats must be integrated to allow applications to access combined data sources in a high-level and uniform way. The autonomy of information sites makes it impossible for any centralized authority to mandate standardization.

Imagine an archive of newspaper stories that covers the last twenty years. The archive also contains descriptive information about when and where the stories appeared, the source of the articles, the author, and other related articles. It would be very difficult to provide a single interface to all of this information because of its *semi-structured* nature. Semi-structured refers to the fact that the structure of the data is less uniform than what we might find in a conventional DBMS. (e.g., fileds may be routinely missing or have varying semantics)

While there has been a great deal of research in integrating data and operations from heterogeneous sources, products are only just beginning to emerge. The dominant approach seems to be distributed object management as manifested in products such as CORBA, SOM, and OLE. Each of these approaches provides an object-oriented model as the common language for describing distributed object interfaces. While these standards, and the systems that support them, go a long way in integrating different software systems, they are best suited for providing uniform syntactic interfaces to new or existing applications. These approaches provide a common protocol for passing messages between objects in a distributed environment, but they do not tackle the difficult problem of resolving semantic discrepancies. They also cannot be used directly to integrate or create uniformity of data from different sources. In general, sophisticated tools for dealing with data heterogeneity still need to be layered above CORBA, SOM, or OLE interfaces.

5.6 Query Complexity

The problem of query optimization takes on some very different characteristics in future environments, making conventional optimizers inadequate. Firstly, the types that must be considered for our new world include diverse bulk types such as sequences, trees, and multidimensional arrays. Secondly, other types that are stored will be highly application-specific; they will be instances of arbitrary abstract data types.

Conventional query optimization tries to minimize the number of disk accesses. Network optimization might be based on quite different criteria. For example, a user might be more interested in getting an answer in a way that minimizes the total "information bill" for that request. Given two sources that can handle a request, the optimizer should pick the one that will result in a lower charge. This charge can include cost components from processing, data usage, and communication.

Also, optimizers will need to employ different strategies to account for the new forms of data and the characteristics of new computing environments. Standard query optimization techniques have little to offer for a query over a large time series or for a query that may have to translate several data sets into a canonical form before producing a result. Situations like these are very likely to arise in the IVE scenario.

If we consider the personal information systems scenario, for example, we see a need for more flexible query optimization techniques, which will consider changes in the cost of available broadcast medium (e.g., radio, cellular) as the PID moves. The degree of detail or accuracy provided by the server may be based on the amount of money the person is willing to pay. Thus, query optimization models must take into account not only the formulation of the requests, but also a description of optimization goals. These goals might be couched in terms of resource consumption (e.g., optimize for minimum memory consumption and maximum network use) or as execution limits based on accuracy of answer or allowable resource consumption.

5.7 Ease of Use

Even though there has been tremendous improvement in the ease of installation, management, and use of DBMSs, especially those that run on personal computers or workstations, many applications still prefer to use a file system rather than a DBMS. There is an implicit assumption that a DBMS will be managed by a highly-trained, full-time staff, yet most database users have no training in database technologies. Users still find it difficult to connect to a DBMS, to find the right catalog or database name space where data is stored, and to formulate queries and updates to the database. The file system connection and access paradigm is easier to understand. Database systems that are easier to use would present an opportunity for their more pervasive use.

If a complex and time-critical application like the one presented in the IVE sceanrio required a complex programming activity, it would never be workable. Instead, a simple set of interfaces are required to allow managers of the IVE to specify high-level requirements on things like the design of the needed valve. The mapping and matching of data from many distributed sources that is needed to locate relevant designs all must occur transparently.

A database systems would be easier to use, for example, if it were to adapt to individual user interests. For a personal information system, there could be a way for the server to handle different personal profiles. The personal profiles could include travel itineraries within a city at various times of the day, week, or month (e.g., home to work in the morning and evening, visiting client A on Wednesday, etc.), bank branch locations, cash machine locations, favorite restaurants or movie theaters. The server could send the PID time-varying information that is relevant to the user profile, and this information could be displayed on a map of the city. This makes a view of the database available to the user in a form that is easy to apply to the problem of managing one's schedule.

5.8 Security

The World Wide Web (WWW) supports quick and efficient access to a large number of distributed, interconnected information sources. As the amount of shared information grows, the need to restrict access to specific users or for specific use arises. The non-uniformity of WWW documents, and the physical distribution of related information, make such protection difficult.

Authorization models developed for relational or object-oriented database management systems cannot be adapted to securing hypertext documents for a number of reasons. First, in defining a suitable authorization model, the semantics of the data elements must be clearly defined and the possible actions that can be executed on them must be identified. The definition and semantics of the conceptual elements of a hypertext document are not uniform and vary from system to system. A second difficulty derives from the fact that the "data elements" of a hypertext document are not systematically structured as in the data in a database management system. As noted earlier (Section 5.3), there is no equivalent of the "database schema." This makes it more difficult to administer authorizations. Third, an authorization model for hypertext needs to support different levels of granularity for both performance and user convenience. For example, it should be possible to assign authorizations not only for a single hypertext node, but also for a part of a node without being forced to break the node unnaturally into multiple pieces.

5.9 Guaranteeing Acceptable Outcomes

Transaction management provides guarantees that user activities will leave the database in an acceptable state. Committed transactions take a database from one acceptable state to another. Otherwise, an aborted transaction is guaranteed to leave the database in its pre-transaction state. Only acceptable states are made visible to concurrent users.

Transaction management is an extremely successful field with a well-understood and sound theory, and sophisticated techniques for high-performance implementations. Its success is documented by the impressive transaction rates in existing database products.

Despite its successes, transaction management can become a barrier to both system performance and the ability to specify acceptable outcomes. Today's transactions link together atomicity, isolation, and persistence, and this linkage imposes both performance overhead and rigidity in what it requires of transaction outcomes. Moreover, transaction management is currently database-centric; that is, most transactional data is "in the box".

New applications and system environments require new or enhanced transaction technology. Long running applications need to define acceptable outcomes that are weaker than serializability because making data unavailable (isolation) for long periods of time is unacceptable. Further, aborting entire transactions in the face of potentially unacceptable outcomes is too draconian. We need to avoid losing useful work and free the end user from dealing with unsuccessful transaction outcomes, e.g. those requiring resubmission.

Today, wide area networks, of which the Internet is the prime example, are making it possible to do business between widely separated individuals and organizations. However, today's standard protocol for distributed transaction processing (two-phase commit) imposes a barrier to the participation of component systems because it is a blocking protocol that compromises the autonomy of the participants. This poses an even larger problem when the component systems are only intermittently connected, or are of highly variable reliability and trustworthiness. For these reasons, today's transaction management facilities are often considered inappropriate for modern distributed applications such as those discussed in Section 4.

5.10 Technology transfer

In addition to the specific barriers listed above, there is also a barrier between research and industry. There is insufficient knowledge by researchers of the techniques and solutions needed by industry, and insufficient utilization of the results of research by industry. The monolithic structure of a DBMS contributes to the problem. Each improvement has an impact on many portions of the code base, rendering vendors hesitant to apply insights generated by the academic community. Researchers generally have little understanding of these complex interactions. Finally, much of database technology available commercially is dictated by standards, which have little input from the research community.

6 Research

In order to achieve our vision and overcome these barriers, a number of central research topics must be addressed. We enumerate the most prominent of these below.

- **Extensibility and componentization.** While this report has argued that database components be used for lightweight support of new applications, there is also a related need to approach the construction of DBMS in a modular way. We are beginning to see the emergence of lighter-weight database engines from some vendors which begin to address this concern. Even for applications that need the full functionality of a database management system, there is often a need to extend that functionality with application-specific support.

  Even though there are extensible DBMSs today that allow the definition of new data types (ADTs) or that provide native support for new types such as text, spatial data, audio, and video, these extensions and services are available in closed, proprietary ways. We need to create systems that make it easier for developers to incorporate new data types, developed outside the DBMS, that can be manipulated inside a database as first-class native types. Similarly, we need to look for ways to open the architecture of DBMSs in such a way that new services can be incorporated and that database functionality can be configured in more flexible ways according to application needs.

Research is also required to find ways for DBMS components to cooperate or be integrated with non-DBMS components such as operating systems, programming languages, and network infrastructures. For example, query processing and data movement components should be able to take advantage of, and cooperate with, advanced network facilities in order to negotiate quality-of-service and bandwidth allocation.

- **Imprecise results.** In today's DBMSs, we expect 100% accurate results; that is, we assume that there is a single correct and complete answer to a query. In the Web or other large information sources, this level of accuracy may not be possible or desirable. In fact, many search engines for text and multimedia types do not provide 100% accuracy. Research has been done on similarity queries, but in general these results are isolated and are based on peculiarities of specific data types (e.g., images, text). There is nothing to tie the type-specific techniques together; we need to develop a general theory of imprecision.

- **Schemaless databases.** In order to apply database facilities to data created outside of a DBMS, we will need sophisticated data mapping facilities. Ideally, these mapping tools would be declarative and, thus, combinable with a query language such as is done in SQL.

  When the structure of data is dynamically evolving, it is difficult to capture it with a fixed schema. The Web is a good example of such data. Nevertheless, extensions to existing database techniques can be used to query and transform this kind of unstructured data.

- **Ease-of use.** Better database interfaces are required if we are to get the kind of penetration into personal computing that other tools like spreadsheets and word processors have had. We cannot expect users to write SQL. Similarly, it is important to translate theoretical notions to usable techniques. For example, *functional dependencies* were developed in relational database theory. They underlie many of the PC DBMS design aids without the users needing to be expert at the theory.

- **New transaction models.** New transaction models allow user-defined notions of correctness and allow transactions to be nested. Often they decouple atomicity from isolation. They typically allow notions like semantic serializability and semantic atomicity. The models make it possible to specify compensation/rollback that is local to a scope. We need to design mechanisms for these models to support partial rollbacks followed by an ability to go forward to an acceptable state that not only leaves the database consistent but that also accomplishes useful work for the end user.

  New transaction models also try to overcome blocking in the 2PC protocol in that they allow more autonomy by early commits at the cost of potential compensation. We need to investigate requirements on the properties of a subsystem in order to include it in such a distributed transaction. We also need to study the scheduling and correctness requirements that can be taken "outside the box".

- **Query optimization.** Query processing will have to be extended to cover more data types than those handled in today's database products. For example, queries involving sequences (e.g., time series) are becoming more important. Optimization over these structures will require new indexing methods and new query processing strategies.

  Also, optimization criteria may change. In the past, optimizers tried to reduce overall response time by reducing the total resource consumption (possily dominated by the number of disk accesses) required to process the query. Users may wish to minimize their overall information bill by using sources that are cheaper but may give slower response time. Alternatively, a user may care more about accuracy and completeness than cost, thus requiring that the optimizer find the most reliable and up-to-date sources.

In addition, in nomadic or wireless computing, query optimization must be sensitive to bandwidth and power considerations. Satellite broadcast might be required in order to achieve the necessary bandwidth to deliver large amounts of data in a mobile environment. In addition, query processing algorithms must be sensitive to battery consumption issues on the mobile computer.

- **Data movement.** In a highly distributed environment, the cost of moving data can be extremely high. Thus, the optimal use of the communication lines and caches on various intermediate nodes becomes an important performance issue. While these considerations are related to distributed query optimization, here we must consider overall system access patterns as opposed to the processing of a single request. We must also consider existence of asymmetric communication channels introduced by low bandwidth lines and/or highly loaded servers.

- **Security.** Issues related to access control in distributed hypertext systems include (1) formulation of an authorization model, (2) extension of the model to take distribution aspects into consideration, (3) interoperability between different security policies, and (4) investigation of credential-based access control policies.

- **Database mining.** Database mining is another rapidly growing research area which can also be thought of as "out of the box". It is a synergy of machine learning, statistical analysis and database technologies. Discovery tasks such as rule (association) generation, classification and clustering can be viewed as ad hoc queries leading to new families of query languages. Evaluation of such queries require running inductive machine learning algorithms on large databases. Research challenges include the design of an adequate set of simple query primitives and a new generation of query optimization techniques.

Solutions in some of the above areas will also have the positive effect of enabling the transfer of newer technologies. For example, extensibility will permit novel, as yet undeveloped indexing approaches to be incorporated into a database system, without having an impact on the other components of the existing DBMS. Moreover, the research community needs to participate more fully in standardization efforts and to form a closer partnership with industry.

7 Conclusions

In this report, we argued thats database research must be more broadly defined than it has been in the past. We discussed the idea that the database community must apply its experience and expertise to new problem areas that will likely require new solutions packaged in ways that may not resemble existing database systems.

The long-term view is that the database community can contribute a great deal to the very general problem of scalable, efficient, and reliable information systems. Information must be defined in the broadest of terms to include a large variety of semantic types that are obtained in many forms. The vision is an integration that supports the application of database functionality in small modules that give us just the right capability. These modules should also represent a unified theory of information that allows for the querying information of all types without having to switch languages or paradigms.

Acknowledgements

The editors would like to emphasize that this report was developed through discussions, comments, and direct contributions from the members of the working group. We would also like to thank David DeWitt, Jim Gray, Gail Mitchell, and Peter Wegner for helpful comments on earlier drafts of this report.

References

[Codd70] E. F. Codd, "A Relational Model for Large Shared Databanks", Communications of the ACM, **13**:6, (June 1970), pp. 377-387.

[Gray95] J. Gray, `http://www.cs.washington.edu/homes/lazowska/cra/database.html`

[SSU91] A. Silberschatz, M. Stonebraker, and J. Ullman, "Database Systems: Achievements and Opportunities," SIGMOD Record, **19**:4, pp. 6-22. Also in CACM **34**:10 (Oct., 1991), pp. 110-120.

[SSU95] A. Silberschatz, M. Stonebraker, and J. Ullman, "Database Systems: Achievements and Opportunities Into the 21st Century", `http://www.cs.stanford.edu/pub/papers/lagii.ps`.

[TY95] J. Toole and P. Young, `http://www.hpcc.gov/cic/forum/CIC_Cover.html`.