Report on the Workshop on Heterogenous Database Systems held at Northwestern University Evanston, Illinois, December 11-13, 1989 Sponsored by NSF

General Chair: Peter Scheuermann
Program Chair: Clement Yu

Program Committee

Ahmed Elmagarmid
Hector Garcia-Molina
Frank Manola
Dennis McLeod
Arnon Rosenthal
Marjorie Templeton

Executive Summary

Advances in networking and database technology during the past decade have changed dramatically the information processing requirements of organizations and individuals. An organization may have heterogenous database systems which differ in their capabilities and structure and which may be dispersed over a number of sites. In addition to these characteristics of heterogeneity and distribution, the ever larger number of databases available in the public and private domain makes it imperative to allow shared access to these databases in such a way that individual systems maintain their autonomy. Thus it becomes necessary to develop new techniques and provide new functionality to support the interoperability of autonomous database systems without requiring their global integration. Furthermore, the demands for interoperability extend beyond database systems to include office information systems, information retrieval systems and other software systems.

Research into the interoperability of heterogenous database systems plays an important role in the development of high level open systems. This important fact has been recognized not only in the United States but also in Japan and in Europe, with Japan having allocated around $120M over five years for research and development in this area.

The objective of this workshop was to explore current approaches to interoperability of autonomous information systems and to identify the most important research directions to be pursued in this area. This report summarizes our discussions and broadly classifies the issues into the following categories: [1]

1. Semantic Heterogeneity and Schema Integration
2. The Role of the Object-Oriented Approach
3. Transaction Processing
4. Query Optimization
5. Standardization Efforts

[1] Any opinions, findings, conclusions, or recommendations expressed in this report are those of the panel and do not necessarily reflect the views of the National Science Foundation.

Introduction

We are currently at a crossroads in the development of heterogenous distributed database systems. Advances in the networking and database technology have added two new dimensions to the problems to be solved, namely size and autonomy. The proliferation of public and private databases implies that for effective sharing of information we must provide tools to help users locate the information sources and learn something about their contents. In addition, organizations must maintain a certain degree of autonomy over their data in order to allow access to their information. We can distinguish different types (degrees) of autonomy: design autonomy, communication autonomy and execution autonomy. Design autonomy refers to the capability of a database system to choose its own data model and implementation procedures. Communication autonomy means the capability of a system to decide what other systems it will communicate with and what information it will exchange with them. Execution autonomy refers to the ability of a system to decide how and when to execute requests received from another system. Design autonomy usually has been assumed in distributed database systems, and this assumption brought with it the issue of heterogeneity. Here we can distinguish between data heterogeneity and system heterogeneity. Examples of system heterogeneity are differences in data model, data manipulation language, concurrency control mechanism, etc.

The workshop examined the impact of heterogeneity, autonomy and size on the development of federated database systems, in particular with respect to schema integration, transaction processing and query processing. We use the term federated database system or multidatabase system to refer to a collection of predefined database systems, called local database systems, that cooperate to achieve various degrees of integration while preserving the autonomy of each local database system. We explored the techniques and functionalities required to support the interoperability of federated database systems as well as the interoperability of database systems with other software systems.

This report summarizes the invited talks and position papers presented as well as the open discussion held with all the participants on the last day of the workshop.

Semantic Heterogeneity and Schema Integration

Each local database system in a federated architecture has its own conceptual schema, which describes the structural and dynamic properties of its information. Structural properties refer to the specification of object types and their relationships and constraints at various levels of abstraction. Structural description included both data and meta-data specifications. Dynamic properties describe how constraints are to be enforced and give the rules for update propagation in response to various operations.

We can observe a spectrum of database coupling that has been proposed (at the schema level) to support the interoperability of pre-existing database systems in a federation. At one end of the spectrum we find advocates of total integration, one global federated schema constructed under the responsibility of a global administrator. At the other end, we find systems that use partial integration, with the users themselves specifying which subsets of

the conceptual schemas should be grouped into partially federated schemas. While total integration may be feasible for a small number of databases, it appears that the partial integration approach is more desirable for an environment with many databases, some of which may appear and disappear on a daily basis. However, the problems of enforcing constraints and view update propagation across a number of partially federated schemas remain to be solved.

The schema integration process, whether it results in one or multiple federated schemas, presents a number of problems caused by various aspects of semantic heterogeneity and design autonomy. Schema integration includes the resolution of naming conflicts (e.g. homonyms and synonyms), scale differences, structural differences and missing data. Some tools for schema integration are being proposed to aid in identifying various object relationships such as equality, overlap, containment, etc. An issue that has not been addressed is that of determining relationships among objects that also exhibit behavioral abstractions as is the case in object-oriented systems. More importantly, it remains to be seen to what extent these tools can be automated and to verify their validity on real life systems.

If the federated database schema(s) uses a different data model from the local conceptual schemas, a schema translation module(s) must be provided for in the federated architecture. Efforts have been reported towards the development of specification languages for mapping among different data models. Although the problem of translation among data models received much attention in the early 1980's, the usual approach requires that a new mapper be implemented any time a new DBMS is added in the federation. The advantage of a specification language is that it would enable automatic generation of model mappers. While this seems a promising approach to the schema translation issue, it is not always possible to map from one data model to another and the extent to which this process can be automated for arbitrary models also is not known.

The approach to federated integration discussed so far assumes that the users or the database administrators have complete knowledge of the local conceptual schemas. However, in an environment consisting of hundreds of databases, the first step prior to any possible integration is to learn what information sources exist, where they are located, and what their contents are. A number of concepts have been proposed, but it appears necessary to couple these with AI techniques in order to help incorporate semantic knowledge into this learning process.

The Role of the Object-Oriented Approach

Object-oriented approaches are being considered as potential solutions to problems that exist in a number of software environments. Object-oriented DBMSs (OODBMSs) are being developed to allow databases to include "unconventional" data types such as text, voice, graphics, CAD data, etc. In a more general context, several papers at the workshop addressed the problems of using object-oriented approaches to provide interoperability of heterogenous computing resources, including both data and software components, and of integrating object-oriented databases with conventional relational database systems. Technology supporting these types of data and component integration will be crucial to the development of

the National Collaboratory, an infrastructure proposed by NSF to foster remote interaction between multi-disciplinary teams of scientists. The use of object-oriented approaches both complicates and eases the integration of heterogenous databases with other components.

First, increasing use of object-oriented systems will increase the complexity of the problem. The enhanced capabilities of object-oriented systems create the potential for increased heterogeneity of systems, since a richer collection of data types, as well as software components, will be included. Moreover, distributed object-oriented systems create the potential for users and programs to access vast areas of resources. This implies the need for increased assistance in simply selecting, let alone using, appropriate resources within the network. This problem was also mentioned in papers at the workshop.

Second, object-oriented approaches provide a natural framework for use in integrating heterogenous components. As a design approach, thinking of the components in a federation as objects or collections of objects allows a common design methodology to be applied to objects at all levels of granularity. As an implementation approach, an object-oriented approach provides a rich data model for use in problems of semantic heterogeneity. Behavioral modeling provides a framework for procedures required in inter-object communication, such as data conversion procedures, to be incorporated directly in the objects. Inheritance facilities found in most object systems provide a means of organizing similar data found in heterogenous components.

However, the papers at the workshop suggest that object-oriented approaches, at least so far, have had relatively little impact on some key aspects of problems in heterogenous systems. The architectures currently proposed for object-oriented heterogenous systems seem reasonably straightforward extension of those found in many existing heterogenous database systems. Although the use of objects provides a natural framework for including additional metadata such as units, time information, etc. that may be useful in attribute integration it is not clear how the complexity of this problem is simplified in object-oriented systems. Similarly, considerably more research is required in such problems as query optimization and concurrency control in the context of both object-oriented and heterogenous database systems. At the same time, since OODBMSs are, in a sense, inherently heterogenous (since they may include data of widely varying structure), work on OODBMSs and heterogenous databases will be mutually supportive in these key areas.

In addressing these problems, it will be necessary to make use of work in related technologies. Since object-oriented approaches deal with objects that include both procedures and data, the required technology overlaps such areas as database, programming languages, and operating systems technologies. Research in these areas is already becoming interrelated, as illustrated by emerging work in areas such as persistent programming languages and object-oriented distributed operating systems. It will be necessary to determine how these related technologies interact with the specific problems of heterogenous DBMSs. For example, as already mentioned, the development of advanced modeling facilities in object-oriented systems may well help in heterogenous database system development. On the other hand, work on data storage mechanisms in OODBMSs may have less effect, since the storage requirements in heterogenous systems will be handled primarily by the underlying local DMBSs.

It also will be necessary for researchers to gain more experience with real systems that include many large databases of realistic complexity. It is only this way that both the scope of the problems, and those aspects of real systems that sometimes allow for simplifying

assumptions, will become evident.

Transaction Processing

One of the key issues in federated database systems is transaction management. The problem is to make a collection of different database systems, running on different computers, cooperate and execute a transaction.

In conventional systems, a transaction is a collection of database actions that must be executed with three properties:

1. Atomicity: The entire transaction must be completed, or none of its actions should be executed.

2. Serializability: The execution of a transaction must be isolated from other concurrent transactions.

3. Durability: The values written by a transaction must persist in the database after the transaction completes.

Guaranteeing these properties in a federated system is difficult mainly for three reasons:

1. The actions of a transaction may be executed in different systems, each of which has different mechanisms for ensuring the properties of transactions. For instance, one system may use locks to guarantee the serializability property, while another may employ timestamps.

2. Guaranteeing the properties of transactions may restrict node autonomy, which may be undesirable in a federated system. For example, to guarantee the atomicity property, the participating systems must execute some type of a commit protocol. During this protocol, some systems must become subordinate to other systems, and the subordinates may not unilaterally release the resources, thereby compromising autonomy.

3. The local database systems may not provide the necessary "hooks" (functionality) to implement the required global coordination protocols. Again referring to the commit protocol, it is usually necessary for local systems to become "prepared," guaranteeing that the local actions of a transaction can be completed. Existing systems may not allow a transaction to enter this state (without committing all changes of the transaction to the local database), and providing this functionality may violate design autonomy.

Current efforts in this area may be classified into four (not necessarily mutually exclusive) approaches:

1. Develop strategies for meshing together existing but different transaction processing mechanisms. For example, some researchers have looked into mixed concurrency control algorithms (e.g., locking, timestamps, optimistic) and mixed commit protocols (e.g., centralized, distributed). Each local database system continues to use its native strategy, although there may have to be modifications so that the global mechanism can work.

2. Coordinate existing systems without any modifications. It is usually assumed that the systems share some basic concurrency control and recovery strategies, but that they must be globally coordinated. Each local system receives transactions either locally or from a global execution component; it treats all transactions in the same fashion. The global execution component must assure to it that the transaction properties are guaranteed for non-local transactions.

3. Weaken the properties that are guaranteed for transactions. In other words, new concepts are defined that encompass some, but not all, of the desirable properties of transactions. These new models make it easier to execute "transactions" in a heterogeneous environment.

4. Restrict the types of transactions and/or when they can run. If we can limit transactions a-priori, then it may be easier to guarantee the desired properties. As a very simple example, suppose that node a contains object x, while node b contains y. Local transactions at a and b can read and write the local objects. Suppose there is a single type of global transaction, which only reads x and writes y. In this case no global coordination is required. Each site can use its local concurrency control mechanism, and the resulting schedule will be serializable.

We note that among these four approaches, the first violates design autonomy, the second may violate execution autonomy, and the last two approaches aim to preserve autonomy at all levels.

Research in this area is at an early stage. A number of solutions to the heterogeneous transaction management problems have been suggested, but the solution space has not been fully explored. The exploration of weaker transaction models is at a particularly early stage. It is clear that the payoff in this direction can be significant, but a critical problem is finding a new transaction model that is useful in practice while at the same time allows efficient execution in a heterogenous environment. An almost completely open problem is the comparison of proposed solutions. A first step would be the definition of meaningful metrics for this environment. A second step would be a comparison of the options and tradeoffs.

Query Processing and Optimization

Query optimizers are optimizing compilers with the one notable difference that many query optimizers use quantitative estimates of operation costs to choose among alternative execution plans. Query optimization in homogenous distributed database systems has been an area that has received considerable study. The optimization techniques developed include, at one end, heuristic solutions that cannot guarantee optimality and, at the other end, exhaustive enumeration techniques that potentially may be very slow. In the middle of this spectrum we find techniques based on semijoin algorithms that obtain exact solutions in polynomial time for restricted types of query classes but are also of heuristic nature for arbitrary queries.

When dealing with a single real world object that comes from multiple local databases, the global query processor must solve the problem of data integration, e.g., how to assign ap-

propriate values to attributes in a global request when some of the underlying local databases overlap and disagree on the information or when some of it is missing. The outerjoin operation has been introduced to model an extended join operation in which null values are assigned to missing data. While it is easy to provide a reasonable implementation of the outerjoin, optimization of queries involving outerjoins is an unsolved problem. In particular, expressions that involve several outerjoins, joins and selections may need a new parenthesization (i.e. reassociation) to avoid computing large intermediate results, and the necessary theory is progressing slowly. Some papers at the workshop proposed more sophisticated interpretation of information combination by which some information that is not explicitly represented in the outerjoin can be deduced, rather than simply set to null. But these schemes will see little use if they require a complete overhaul of the query evaluator of the database system - they need to be implemented as extensions rather than replacements for the current algebras that underlie query processing.

One of the major problems in query optimization in a federated architecture is to develop an appropriate cost model, because many local systems have no facilities for estimating or communicating cost information. One option is to emphasize global optimization techniques that rely only on qualitative information such as semantic query optimization (perhaps using information from a generalization hierarchy). Another option is to implement a global optimizer based on a set of parameterized, modifiable cost equations. But customizing this optimizer to a particular local database system is difficult and must be repeated for every new database system release. In either case, cost information in a federated database system is likely to be inaccurate so it may be desirable to incorporate into the optimizer techniques for learning from past experiences.

Early federated database systems used a fixed high-level language, such as DAPLEX, as the multidatabase query language. A query in the multidatabase language must be translated into subqueries of the local database systems. Different translations can yield executions with widely different costs since some older DBMSs have little optimizing capability. Thus, it is important to provide translators that yield optimal or near-optimal subqueries in the local database systems.

Another important issue to be addressed in query optimization and processing is extensibility. It had been assumed that the multidatabase query language had more power than any of the local query languages. But new local systems may support additional operators on attributes that correspond to new data types or on entire relations (e.g., outerjoin, recusion) that are not available in the multidatabase query language. Researchers in extensible optimizers are examining ways in which one may declare the properties of new operators so that they can be exploited by the global optimizer. Thus, a new operator may require extensions to the multidatabase query language, to the optimizer's model of the local system's capabilities, to its cost model and its ability to exploit the new operator's properties. It is imperative to minimize the amount of effort it takes to modify the optimizer in response to the addition of a new local system or a new operator.

Standard optimization algorithms may also need changes to take into account hitherto unconsidered operations. Current optimizers assume that predicate evaluation should be done as early as possible because attribute comparison is cheap; this assumption may fail

for spatial and other user-defined data types.

Standardization Efforts

Although the goal of standardizing all hardware and software interfaces cannot be achieved, without some basic standards federated database systems are not feasible.

Standardization of network interfaces, file transfer formats, and mail formats has allowed extensive interconnection of computers for loosely coupled applications such as electronic mail. The next step is to move toward closer coupling at the data and program level, ultimately allowing programs to process data from any location and to pass partial results from an application on one computer to a second computer. This objective requires well defined standards for locating data, for understanding the semantics of the data, for understanding the capabilities of the data source, for connecting to the data source, for specifying data to be retrieved or updated, for transferring data, for controlling data access, for accounting for resource usage, and for dealing with errors.

Current efforts towards standardization are well underway in some important areas:

- RDA (Remote Data Access) defines a generic interface to a DBMS including database open/close, command execution, and transaction start/end.
- SQL defines a standard data manipulation language.
- TP (Transaction Processing) defines the behaviour and language for transaction processing.
- RPC (Remote Procedure Call) defines interprocess communication at the program call level.
- API (Application Program Interface) is being defined by a group of vendors.

The challenge is to extend standards to meet new requirements without making the systems that implement the standards obsolete. The SQL standard is a good example of a standard that is difficult to extend. SQL is based on the relational model and does not address DBMSs with extended functionality such as OODBMSs or older DBMSs and file systems which will continue to held data for many years.

Concurrent processing is not well supported by the present standards. RPC is a blocking protocol designed for client-to-server communication. RDA is also a peer-to-peer protocol. These standards need to evolve to support a model of multiple autonomous cooperating processes.

The TP protocol needs to support multiple models of transaction processing that can be negotiated between the client and server(s). Traditional protocols use serializability as the criterion for correctness. This may be too slow and too restrictive for some applications. CAD/CAM applications require very long transactions where availability for concurrent updates may be more important than serializability. Real-time applications may have hard time constraints that are more important than data consistency.

In addition to the extensions required in the current standards, new standards are needed. Tension between users who want easy access to all resources and owners of resources who

want to protect their resources. Before we can expect resource owners to open their systems to remote users, access control standards must be devised. Current data dictionaries provide insufficient semantic information and new standards for data definition are needed to include such information as level of abstraction, granularity and so on. Finally, we need standards for describing the capabilities and behaviour of systems, as well as standards for systems to advertize their data and services and for negotiating shared access.

Acknowledgement

Dr. Maria Zemankova, NSF Program Director of the Database and Expert Systems Program, recognized the substantial potential of this area and actively supported our efforts.