Extending the Scope of Database Services

Daniel Barbará
Matsushita Information Technology Laboratory
182 Nassau Street, 3rd Floor
Princeton, NJ 08542-7072 USA

Abstract

A wide variety of important data remains outside of the scope of database management systems. In recent years researchers have been making efforts in two directions: first, developing database management systems that can support non-traditional data, and secondly, offering database services for data that remains under control of autonomous (not necessarily database) systems. The purpose of this paper is to provide a brief review of these efforts.

1 Introduction

Database technology has been available for three decades and it has enjoyed tremendous success in the commercial world. However, in spite of this fact and the enormous amount of research and development in the field, only a relatively small fraction of the data used by organizations around the world is actually kept under real database management systems (DBMS). Some of the reasons for this phenomenon are the following:

- Database management systems have been traditionally designed to provide support for business data processing applications in which fixed format records are to be stored and accessed. These records usually consist of numeric fields along with alphanumeric fields of bounded size. Other types of unstructured or semi-structured data, such as text, bitmaps, etc., have been neglected.
- Organizations have been accumulating data historically and, due to the lack of resources or good support, this data has been kept in plain files.
- Even with the database technology available, people keep designing applications in an ad-hoc manner, without using database software. The reason is that a large effort is usually needed to learn how to use the DBMS, and to setup and design the system.

Extending some of the services offered by DBMS to non-traditional data would radically improve the way people interact with it. Among these services we have:

- Data independence: most of the time, accessing non-traditional data implies at least some knowledge of where the data resides and the format in which it is stored. This is obvious for data under ad-hoc systems. It is also a problem when trying to access data offered by services available through public networks. For instance, consider the problem of finding a way to answer a question such as "find all the animated movies that are playing in town today."
- Atomicity, Consistency, Isolation and Durability (ACID) properties: These properties define the transaction model and although it is doubtful that all of them can be provided for non-DBMS, some coverage of them is important. For instance, consider the design of an application to set up meetings in a company. At some point, the application might need to access and update several data managers: electronic calendars of the individuals involved in the meeting, airline and hotel reservation systems and resource scheduler (for the place of the meeting and equipment needed). These systems might not be implemented under DBMS and yet the application might need to enforce some atomicity: a hotel reservation is worthless if no flight can be found for a participant. (Even if the all the managers are DBMS, the problem of guaranteeing ACID properties is far from trivial, as the research in heterogeneous databases evidences.)

Providing these and other database services to non-traditional data is an important area of research. The purpose of this paper is to provide a brief review of the most current work in this area. Of course, this review is far from being comprehensive and in some way reflects the author's preferences. The purpose of it is to put the work in context and offer some incentive for additional researchers to work in the topics presented here. In the next section we briefly discuss the issues involved in extending database services to non-traditional data. In Section 3 we review in more detail several selected papers. Finally Section 4 offers some conclusions and outlines some avenues of research.

2 Research Issues

The problem of extending databases services to non-traditional data can be viewed as twofold. The first problem is to provide DBMS that can handle more general types of data. The second one is to build tools to interoperate with data stored in autonomous systems that cannot be changed. In this section we outline the issues in both areas.

2.1 Non-traditional DBMS

In the recent past, there has been extensive work in implementing DBMS that can handle non-traditional data types.

On one hand we have the emergence of *object oriented database systems* (for a good survey see [22]), such as O_2 ([14]), ORION ([1]), GemStone ([15]) and Iris ([8]). Object oriented database systems are collections of objects that follow the object oriented model, a concept that evolved in the disciplines of programming languages and artificial intelligence. The key concepts in object oriented systems are the following:

- Encapsulation: the system encapsulates in an object some data and programs to operate on it.
- Extensibility: the system has the ability to extend without changes. This is achieved by including additional programs to operate on the data and by reusing the definitions and programs of an object to define new objects. This last concept is known as *inheritance.*

On the other hand, we have the appearance of *extensible databases.* These systems represent efforts in extending the relational model to support non-traditional applications. Projects in this area include EXODUS at the University of Wisconsin ([5]), STARBURST at IBM Almaden ([12]), GENESIS at UT-Austin ([2]) and POSTGRES at Berkeley ([20]). In the next section, we review POSTGRES in more detail.

2.2 Interoperability

Applications must deal with a wide variety of unstructured and semi-structured data (e.g., electronic mail, documents, spreadsheets) and services (e.g., electronic libraries, airline reservation systems) that are not under the control of DBMS. If database services are to be extended to this domain, it is important to study how to achieve interoperation in it. Although a lot of work has been done to provide interoperability in conventional DBMS (see for instance [6]), very little effort has been devoted to this wider domain.

We believe that due to the complex nature of this environment, interoperability will require human intervention. (As an example, consider an application that needs to interact with calendar systems and a user that keeps her or his calendar in paper.) Instead of fully automated interactions, it is best to think about tools to help the application to interoperate with the data and services, having the human being at the center of the application.

Among the tools that are needed for this task are the following:

- Tools for resource discovery: these tools will help the user to discover what resources and data are available to complete the task in hand. The following are examples of such tools:
  - The Wide Area Information System (WAIS) ([13]): WAIS is an Internet service that lets the user search for indexed articles, based on what they contain. The documents are indexed in servers that follow the WAIS protocol.
  - Gopher ([16]): a system that allows the user to browse for resources using a set of menus. For instance, by using Gopher, the user can access an on-line catalog at a university without having to know or use the Internet address of the server. (Thus providing some data independence.)
  - The World-Wide Web (WWW) ([4]): an information system that organizes Internet information (and local information as well) as hypertex documents.
  - Archie ([7]): a system that allows searching on indexes of files available on public servers on the Internet. The archie server periodically contacts every server that it knows about to find out about new entries.

  In [19], a comparison of these and other resource discovery systems can be found. Even with these tools, finding the needed data and resources can be an extremely difficult task. Browsing or following hypertex links can be extremely time consuming, due to the confusingly large number of resources. Systems like WAIS, on the other hand, are limited to servers that follow the WAIS protocol. Other systems try to solve this problem by the introduction of *agents*, i.e., entities that serve as a link between the applications and the information providers. Two proposals that follow this approach are those of G. Wiederhold ([21]) and the Networked Resource Discovery Project of M. Schwartz ([18]). In the next section we will review the later.
- Information Filtering: an important fraction of the non-traditional data reaches the user as a stream of incoming data (e.g., NetNews, electronic mail). Filtering is a technique to remove

data from the stream and keep only the information that is relevant for the user. (E.g., removing "junk mail.") Filtering techniques can also be used to access and retrieve information from remote sources (i.e., non-incoming data) by means of agents that look for the relevant information. As Belkin and Croft describe it in [3], "an information filtering system is an information retrieval system designed for unstructured or semi-structured data." The process of filtering information is based on the description of the user information preferences, often called *profiles*. A tool that implements information filtering can be viewed as a resource discovery tool that searches for the information described by the profiles. Tapestry ([11]), an experimental electronic mail system, implements a special type of profile called *continuous query*. A continuous query has the semantics of a query that is executed at every instant in time. We will review Tapestry in the next section.

- Activity management tools: These are tools that attempt to provide some or all of the ACID properties supported by DBMS while interoperating with the domain of non-traditional data. These tools can be used to run activities that query and update the data in a transactional manner. One of such tools is the Distributed Object Management System (DOMS) ([10]). DOMS supports a variety of extended transaction models to satisfy a wide spectrum of applications. We will review DOMS in the next section.

3 Review of the Current Work

In this section we review a paper that describe an extensible DBMS ([20]) and three papers that deal with interoperability tools: one on resource discovery ([18]), one on information filtering ([11]) and one on activity management ([10]).

3.1 POSTGRES

POSTGRES ([20]) is an extensible relational system designed to provided support for non-traditional data, such as bitmaps, icons, text and polygons and to provide the capacity of storing and enforcing rules that describe the semantics of the application.

Users of POSTGRES interact with their databases by using a set-oriented query language called POSTQUEL. Although POSTQUEL has navigational capabilities, the language is basically designed as a superset of a relational query language (such as QUEL).

The POSTGRES data model allow users to define two data types: *base types* and *constructed types*. Base types include bits, bitmaps, compressed integers, etc. and can be arbitrarily added to the system while executing. Constructed types (analogous to the concept of relations) are records (analogous to the concept of tuples) of base types and instances of other constructed types.

The data model also allows three different kinds of functions: *normal functions*, *operators* and POSTQUEL *functions*. Normal functions are user definable and their operands are base or constructed types. An example of a normal function is a function that maps an instance of a polygon to a floating point number that represents the area of the polygon. Operators are functions with one or two operands. For instance the operator "Area Greater Than" can be defined to compare the areas of two polygons and map the result into TRUE or FALSE, according to whether the area of the first is greater than the area of the second or not. POSTQUEL functions are packaged commands. For instance, a POSTQUEL command big-polygon can be defined to retrieve all polygons whose area is greater than 500.

Since POSTGRES implements abstract data types, classes, methods and inheritance of data and functions, it can, in some sense, be described as an object-oriented system. However, its main thrust lies on extending the capabilities of a relational system.

POSTGRES supports rules, allowing the user to do event-driven programming. Any POSTGRES command can be translated into a rule by adding the predicates *always* or *never*. (For instance "always make Joe's salary equal to Jim's," or "never make Joe's salary higher than Jim's.")

The paper offers a critique of the (according to its authors) weak points in POSTGRES. They point out flaws in the data model (such as the lack of union types and arrays of constructed types), rules system (complexity of the implementation) and storage system (instability under heavy load). Some of these problems were corrected in subsequent versions and in the upcoming commercial release of POSTGRES.

3.2 The Networked Resource Discovery

The goal of the Networked Resource Discovery ([18]) is to explore mechanisms to provide efficient ways of finding information and resources in the Internet.

Recognizing that building a global database of resources and data for such a vast collection would be expensive and very difficult to maintain, the project tries to find alternative ways of organizing and searching the space of resources.

The main idea in the paper is that of organizing the space using a flexible form of hierarchy, called *specialization graphs*. These graphs emulate the way human social networks evolve. Instead of using hierarchical (bureaucratic) mechanisms, humans often contact knowledgable intermediaries who refer them to other people of interest. A graph in which nodes are people and links represent one person knowing another commonly has a very small diameter (this is usually referred to as the "small world" phenomenon). Using a similar graph for resource networks, where the diameter refers to the "knows-about" relation, leads to efficient resource discovery.

In a specialisation graph, a resource is a member of multiple hierarchies and the graph can have back pointers and cycles. Nodes at the same nesting level are linked together indicated related resources. This graph can be viewed as following the same data organisation of the network database model. The difference is, of course, that the structure must evolve without the need for human intervention, as opposed to a network database design where a human sets all the links *a priori*.

Given a collection of information providers, or *Resource Information Repositories* (RIR) as the paper calls them, no initial set of relationships exists among them. Thus, the system must be provided with some entities that can build the relation graph. Such entities are called *agents*. The system also has *brokers* that encapsulate the heterogeneity and access methods to RIRs, and *clients* that start resource searches on behalf of users, by communicating with the agents.

When a RIR enters the network, its broker must find an agent to communicate to it the set of keywords that describe the RIR specialty (topics about which the RIR can answer queries). To make the graph evolve, agents broadcast the keywords they know about to other agents. In order to keep the system scalable, a *sparse diffusion multicast* algorithm is used: given a set of N agents, a message is sent to a subset of size $k + log(N)$, selected at random, where k is a tunable constant.

A search is initiated at some known agent. If this agent does not know about the keywords involved and does not know of other agents that know about them, a local broadcast can be used to contact nearby agents. This failing, a sparse diffusion multicast is initiated to search for other agents that can help. All these failing, the user is offered a menu of known words by the agent. Hopefully, as time goes by, the graph density increases (the diameter becomes smaller) and the agents become more knowledgable.

The Networked Resource Discovery Project is still experimenting with new techniques, such as probabilistic protocols to disseminate information.

3.3 Information Filtering

Tapestry ([11]) is an experimental system developed at Xerox Palo Alto Research Center. It is tailored to deal with electronic mail and other forms of incoming documents (such as newsgroups postings).

The main idea behind Tapestry is that a user can be better served in her/his information needs by means of *filters*. A filter scans repositories of documents to search for those that are of special interest to a user. Tapestry supports content-based filtering and *collaborative filtering*. The last term means that people help one another in filtering documents by means of *annotating* their reactions to the documents they have seen. Using this feature, a user could define a filter of documents that have been favorably endorsed by another.

In Tapestry, documents are indexed by their contents and placed in a repository that is append-only. (Thus, documents are immutable.) Another store keeps the annotations that users provide for such documents. A Filterer repeatedly runs a series of user-provided queries, placing documents that match a query in a user's owned box where the user can browse and read the documents.

Users can also issue ad-hoc queries, i.e., queries that will be answered only once. If the user is satisfied with the result of a query, he or she can install it as a filter, and from then on, the results of this query will be put in the user's box.

The Tapestry query language (TQL) allows boolean expressions that are similar to statements in first order predicate calculus. The expressions combine atomic formulas with boolean operators and can have free variables quantified by EXISTS and FORALL. TQL supports operations with sets, since many document fields are set-valued. (E.g., the field "To:" in a mail message is a set of strings.)

Tapestry's Filterer module executes filter queries giving them *continuous semantics*. By this semantics, a query has as results the set of data that would be returned if the query were executed at every instant in time. By doing this, the Filterer avoids unpredictable and erratic results. To see this, consider the query "Find all messages that have no reply." A document ceases to satisfy this query as soon as a reply appears. Instead of running this query, the Filterer rewrites it into "Find all messages that at some point in time have had no replies." Since the document repository is append only, this query guarantees that once a message satisfies it, it keeps satisfying it forever. The key to rewriting queries is to find a monotonic query that returns all documents that match the query or matched it at some time in the past. By doing this, the Filterer can afford to run queries periodically and still get the continuous semantics. Every time the query is evaluated, the Filterer only returns documents that satisfy the query now but did not satisfy the query the last time it ran. This is implemented efficiently in Tapestry by rewriting the query as an *incremental query* that can compute the new items (or an approximation of the set) quickly.

3.4 Activity Management

The Distributed Object Management System (DOMS) ([10]) is an environment currently under implementation at GTE Laboratories. DOMS goal is to support transactions that operate on three classes of objects:

- Objects under the control of systems that support transactions. (Autonomous and heterogeneous database systems.) These systems are referred to as *Local Database Systems* (LDBS).

- Objects under the control of systems that provide primitive atomic operations. (E.g., file systems.) These systems are referred to as *transactionless systems*.

- Objects that are maintained by DOMS itself.

DOMS aims to support two classes of complex transactions:

- Multi-system transactions: that are formed by simple (flat) transactions.

- Extended transactions: such as Nested Transactions ([17]) and Sagas ([9]).

By supporting a variety of transaction models, DOMS tries to satisfy the requirements of a wide variety of applications. To achieve this, DOMS implements a *Transaction Specification and Management Environment* (TSME), which provides a programmable transaction management mechanism. This module configures the appropriate run-time environment to support the specific transaction model required. The TSME is composed by a *Transaction Specification* module (TS), and a *Programmable Transaction Management* facility (PTM). The TS module allows the user to specify the requirements of the transaction management mechanism. These requirements are fed into the PTM which assembles and configures the run-time environment to enforce them.

Complex transactions are specified by enumerating its constituent transactions and the dependencies among them. (A constituent transaction might, in turn, be another complex transaction that needs to be specified.) Dependencies among transactions can be of two types: *transaction state dependencies*, which are conditions on transaction states that define the execution structure of complex transactions (e.g., "transaction $T2$ cannot begin before $T1$ commits"), and *correctness dependencies*, which specify which concurrent executions of complex transactions preserve correctness (e.g., "the history of operations performed by a set of transactions on a set of objects must be serializable"). Dependencies can be specified in DOM using *Dependency Specification* tuples.

Dependencies are implemented in the PTM module by means of schedulers. However, to accommodate to the needed flexibility, these schedulers need to be programmable, i.e., they must accept commands that alter their behavior. This is achieved by supporting rules that are triggered by *transactional events*, i.e., events that change the state of a transaction. However, some objects (such as those maintained by LDBS) implement their own, usually non-programmable schedulers. When dealing with those, the set of dependencies that can be supported may be restricted.

DOMS is currently under implementation and represents the first attempt to provide transactional aspect to a variety of objects, including those that are not under the control of transactional systems.

4 Conclusions

In this paper we have briefly surveyed the work that has been done in the two aspects of the problem of offering database services for non-traditional data: the implementation of new DBMS and the implementation of tools to interoperate with autonomous systems. We reviewed four selected papers in the area. It is our opinion that this is a very rich and exciting field in database technology. At the same time, it is an area in which the results have the potential to impact a very large community of users, given the increasing variety of electronic data and services being offered nowadays.

More research is needed in implementing and improving DBMS that can support a wider spectrum of applications such as CAD and software development.

The field of building tools for interoperability is just starting to emerge. More research is needed in the area to define other needed tools and to improve the ones that we have. The following is a non-exhaustive list of research needs:

- Providing the user with efficient and simple interfaces to find data and resources.

- Defining and implementing tools that capture users' information interests (i.e., information filters). Mechanisms to automate the creation of such tools are most welcome.

- Expanding the usability of some important concepts such as continuous queries, whose applicability is now limited to append-only information providers.

- Providing efficient implementation of general transaction models.

References

[1] J. Banerjee, H. Chou, J.F. Garza, W. Kim, D. Woelk, N. Ballou, and H. Kim. Data Model Issues for Object-Oriented Applications. *ACM Transactions on Office Information Systems*, 5(1):3–26, 1987.

[2] D.S. Batory, J.R. Barnett, Garza J.F., Smith K.P., K. Tsuku da, Twitchell B.C., and Wise T.E. Genesis: An extensible database management system. *IEEE Transactions on Software Engineering*, SE-14(11), 1988.

[3] N.J. Belkin and W. B. Croft. Information Filtering and Information Retrieval: Two Sides of the Same Coin? *Communications of the ACM*, 35(12):29–38, December 1992.

[4] T. Berners-Lee, R. Cailliau, J. Groff, and B. Pollerman. World-Wide Web: The Information Universe. In *Electronic Networking: Research, Applications and Policy, 2(1)*, pages 52–58. Meckler Publisher, Westport, CT, 1992.

[5] M. Carey, D. DeWitt, J.E. Richardson, and E.J. Shekita. Object and File Management in the EXODUS Extensible Database System. In *Proceedings of the Twelfth International Conference on Very Large Databases, Kyoto*, August 1986.

[6] A. Elmagarmid and C. Pu. Special issue on heterogeneous databases. *ACM Computing Surveys*, 22(3), September 1990.

[7] A. Emtage and P. Deutsch. Archie - An Electronic Directory Service for the Internet. In *Proceedings of the USENIX Winter Conference*, pages 93–110, January 1990.

[8] D.H. Fishman, D. Beech, J.P. Cate, E.C. Chow, T. Connors, J.D. Davis, N. Derrett, C.G. Hoch, W. Kent, P. Lyngbaek, B. Mahbod, M.A. Neimat, T.A. Ryan, and M.C. Shan. Iris: An Object-Oriented Database Management System. *ACM Transactions on Office Information Systems*, 5(1):48–69, January 1987.

[9] H. Garcia-Molina and K. Salem. Sagas. In *Proceedings of ACM-SIGMOD 1987 International Conference on Management of Data, San Francisco*, pages 249–259, 1987.

[10] D. Georgakopoulos, M. Hornick, and P. Krychniak. An Environment for Specification and Management of Extended Transactions in DOMS. In *Proceedings of the 3rd International Workshop on Interoperability in Multidatabase Systems, Vienna, Austria*, April 1993.

[11] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using the Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.

[12] L. Hass, W. Chang, and G.M. Lohman et al. Starbusrst mid-flight: As the dust clears. *IEEE Transactions on Knowledge and Data Engineering*, 2(1):143–160, March 1990.

[13] B. Kahle. Wide area information server concepts. Technical Report Anonymous FTP from think.com:/public/wais/wais-8-aXX.tar.Z, 1989.

[14] C. Lecluse, P. Richard, and F. Veles. O2, An Object-Oriented Data Model. In *Proceedings of ACM-SIGMOD 1988 International Conference on Management of Data, Chicago*, June 1988.

[15] D. Maier, J. Stein, A. Otis, and A. Purdy. Development of an Object-Oriented DBMS. *Proceedings of the First ACM Conference on Object-Oriented Programming Systems, Languages and Applications, SIGPLAN Notices*, 21(11):472–482, 1986.

[16] M. McCahill. The Internet Gopher: A Distributed Server Information System. *ConneXions - The Interoperability Report*, 6(7):10–14, July 1992.

[17] E. Moss. *Nested Transactions: An Approach to Reliable Distributed Computing.* MIT Press, 1985.

[18] M. F. Schwartz. The Networked Resource Discovery Project. In *Proceedings of the IFIP XI World Congress, San Francisco*, pages 827–832, August 1989.

[19] M. F. Schwartz, A. Emtage, B. Khale, and B. C. Neuman. A Comparison of Internet Resource Discovery Approaches. *Computing Systems*, 5(4), 1992.

[20] M. Stonebraker, L. A. Rowe, and M. Hirohama. The Implementation of Postgres. *IEEE Transactions on Knowledge and Data Engineering*, 2(1):125–142, March 1990.

[21] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25(3):50–62, March 1992.

[22] S. B. Zdonik and D. Maier. *Readings in Object-Oriented Database Systems.* Morgan-Kauffman, 1990.