Database Challenges in Global Information Systems

Joann J. Ordille *Barton P. Miller*
`joann@cs.wisc.edu` `bart@cs.wisc.edu`

Computer Sciences Department
University of Wisconsin-Madison
1210 W. Dayton Street
Madison, Wisconsin 53706
USA

1. INTRODUCTION

The global Internet provides users with an information labyrinth – rich in resources, yet confusing and difficult to navigate. Many researchers are responding with a desire to integrate all files, indeed all information, into one global (file) tree. We know (and wonder how the rest of the world fails to see) that hierarchical navigation is an inadequate query facility. Non-procedural languages, like relational query languages, offer users the hope of breaking through the labyrinth to access information quickly and directly. The door is open for the database community to make a major impact on the structure of global computing, but current technology is inadequate to the task. New research must overcome the problems of scale, autonomy and availability to make global information systems a reality.

Traditionally, distributed name services have addressed the problems of locating resources, people, and information in a network. The most successful name services are hierarchical. For example, the X.500 directory standard [4] provides a tree of objects; each object has a type and a collection of attributes. The authority to create and control naming information is allocated by assigning subdirectories of the tree to different organizations. Since each organization has freedom in structuring its subtree, the global network, with tens of thousands of organizations, generates a non-uniform tree structure. Users must either guess the unique pathname for an object, or must navigate and list the tree to find the object. Few of us can find a particular file in our own directory tree, so it is no surprise that users cannot find objects in a global tree-structured information space.

This work was supported in part by an AT&T Ph.D. Scholarship, National Science Foundation grants CCR-8815928 and CCR-9100968, Office of Naval Research grant N00014-89-J-1222, and a Digital Equipment Corporation External Research Grant.

The strength of the hierarchical organization of information comes from the ease with which it distributes authority to create and control information. Heterogeneous distributed database systems (multidatabases) also distribute authority to their component systems, and they improve on hierarchical services by providing user-friendly non-procedural query languages. Multidatabases are positioned to help integrate the recent proliferation of name services [16]. They could integrate hierarchical name services, centralized name services with a more relational flavor like the CSNET name service [11], and sources of public files like Archie [7] into one descriptive (or relational) name service to make resource location simpler for everyone. Several efforts already support descriptive queries over file trees and other hierarchical name spaces [9, 17, 21]. The challenge is not the integration of the data models of the component systems, but the performance of the resulting name service.

The most frequent query for any descriptive name service is the global selection query. To answer a global selection query, the name service must locate objects within a search space of tens or even hundreds of thousands of database partitions. The scale, autonomy and availability characteristics of the environment make brute force selection techniques, exhaustive searches, and global synchronization impossible. The bottleneck for query performance is not disk access or cpu power, but communications latency and throughput. The challenges of global information systems force us to reexamine our model of consistency and the role of databases in supporting information-based enterprises. The very meaning of the database is different in this global arena, because the database is not an authority for database objects (e.g. your bank balance) but a hint for finding objects globally without excessive cost. Database and name service insights together provide a foundation for meeting the challenges of world computing.

The following sections describe global information sharing problems and possible research directions in more detail. Section 2 describes the environment and the challenges it raises. Section 3 examines how models of consistency impact the possible solutions. Finally, Section 4 summarizes frontiers of research for the information infrastructure of the future.

2. GLOBAL INFORMATION ENVIRONMENT

Current multidatabase research focuses on problems of data integration and transaction processing [2, 6, 12, 22]. Data integration research seeks to combine information from different databases despite differences in vocabulary, completeness and representation. Transaction processing research seeks to combine independent transaction processing systems without major loss in the autonomy of those systems. Even if we assume that we know how to solve these problems (which are areas of active research), have we reached the point where we can organize the information in the world so we can find just the objects or data we need? The answer is clearly NO. Scale, autonomy and availability prevent multidatabases from delivering global computing.

2.1. Scale

The scale of the global Internet is immense, both in number of objects and in geographic dispersion. There are now a million hosts on the Internet belonging to tens of thousands of organizations [13]. There are millions of Internet users distributed across every continent and terabytes of publicly available data [18]. Moreover, potential sources of information, like hosts and organizations, are growing exponentially.

In this environment query processing can take hours or even days. A query site that must contact tens of thousands of database partitions will have major bottlenecks that result from sending and receiving messages sequentially. Indices could reduce the problems of scale by isolating queries to a subset of the data partitions, but maintaining them raises additional autonomy issues (see below). The increased scale in the number of users makes global query rates of hundreds per second a real possibility. Although contacting all data partitions may be possible for a single query, it is certainly impossible for any large number of queries. Research must identify methods for delivering more parallelism in global query processing, possibly through a tree of query processing engines whose sole role is the forwarding of subqueries to individual sites and the coalescing of results.

The geographic dispersion of data partitions around the planet produces long latencies. Latencies on the order of hundreds of milliseconds are a physical limitation of the network. Name services can use disks as well as memory to cache information near to the client, because wide area latencies exceed disk latencies by at least an order of magnitude. By reducing global system load, caches can improve response times for all queries in the multidatabase. The question is: what to cache, where to cache, and how to cache? Do we cache data or meta-data like indices? Do we cache information at individual workstations or at some higher level of organization? How do we keep caches consistent? Current database caching research does not answer these questions [1], because it assumes a local area environment where multicast and low latency communication can be used to maintain cache consistency.

2.2. Autonomy

Global name services are autonomous and heterogeneous. Organizations protect both their privacy and the operational integrity of their environment by administering name services locally. A name service is a valuable organization resource that must be protected from tampering. To understand this, one need only imagine the disastrous affects of failure to translate host names into addresses on a network, or locate essential personnel in a time of crisis. Organizations make some of their information available for external communities to facilitate electronic mail connectivity, interactive collaboration, marketing, publication, and other activities. Organizations only share information if it does not threaten their privacy, jeopardize the availability of local name services, or require excessive software modifications and support.

Autonomy exacerbates problems of scale. Scale and performance improve if we create a global index that accepts a selection predicate and constrains it to a subset of the partitions in the global multidatabase [15, 17]. Autonomy makes such an index virtually impossible, because maintaining the traditional level of consistency (serializability) of the index introduces a global synchronization bottleneck and requires changes in the component systems. Interestingly, this tension between autonomy and consistency is present in the struggle to achieve global transactions. The traditional model of consistency causes us to violate the autonomy of component systems to provide transactions, and now tempts us to violate autonomy to support indices. This continual tension calls us to reexamine the traditional model of consistency (see Section 3).

Any successful multidatabase name service will require minimal cooperation from the existing name services, but will reap the benefits of cooperation when it occurs. We have been so caught up in requiring cooperation that we have given little thought to encouraging it. The Digital Library Project sees the Internet as the great electronic publisher of the future. [10] The network locates and supplies relevant documents to users; users pay fees to publishers for the documents they receive. Certainly publishers would cooperate if updating indices meant selling books or articles. Commerce is just one of the models of cooperation we should consider for the new global multidatabases.

2.3. Availability

Availability should perhaps be called "unavailability" when we refer to the global network. If the probability that a partition is available is .999, then the probability that 10,000 independent partitions are available simultaneously is .0000452 (or 24 minutes of availability per year). Only 45% of X.500 partitions in the United States were available at one time during experiments in 1991 [17]. Measurements of the DNS indicate that more than 10 percent of the partitions at educational institutions are not replicated, so the unavailability of a single system can make one of those partitions unavailable [5]. The reasons for unavailability are failure, abandonment, and intermittent availability. As scale grows, the chance of failure in at least one system or network component increases dramatically. Failure can sometimes be addressed by replication, but there is a surprising absence of replication in essential services. Data partitions are often abandoned by their owners when interest in the application wanes. The owners do not update the global catalog of partitions, and users continue to query the partition but are never answered. Organizations often make partitions unavailable intermittently due to maintenance schedules or privacy concerns. We have a slightly pessimistic, but not unrealistic rule of thumb: at any given time, at least some of the partitions that you wish to query will be unavailable.

Chronic problems with availability require new approaches to consistency and transaction processing. If transactions require the availability of all data partitions, transactions will never complete. We need a model of consistency that allows incremental processing. A user can be given partial results for a query, and the option to have the query completed in the background when partitions become available. In the absence of interdependencies, updates can be processed as independent subtransactions on the component systems. Automatic techniques for identifying and removing component systems that will never answer or have completely unreliable data are also important.

3. MODELS OF CONSISTENCY

The difficulties of maintaining serializability in the presence of scale, autonomy and chronic unavailability lead us to reexamine this traditional model of consistency. Some current name services do not use complete query responses, but instead depend on samples of the name space. With a flair for science fiction, name service researchers have proposed sending processes out to scour the network and return with useful tidbits of information [10]. One name service snoops in news headers in an effort to locate people [20]. Other name services look around and return some answers to a query, but not all answers and not necessarily the same ones as last time [19, 20]. We could describe the semantics of this type of query processing as *luck semantics.* Even if we must depart from the traditional model of consistency, we would like to improve on luck by supplying results that can be interpreted and used without frustration.

Traditional consistency has three important characteristics in the the global name service environment. Traditional consistency treats the database as the authority for information, views the database as a sequence of consistent states in time and transactions as transformations between those states, and maintains consistency in the presence of any data interdependencies. In this model for example, the database is an authority for bank balances. It starts in a consistent state and faithfully reflects bank deposits and withdrawals. If your bank balance is wrong, you need to expend considerable effort to prove the error and correct the balance. Name services differ in substantial ways from these characteristics of authority, consistent states in time, and interdependencies. We can use these differences to develop alternative models of consistency and query processing with *better than luck semantics*.

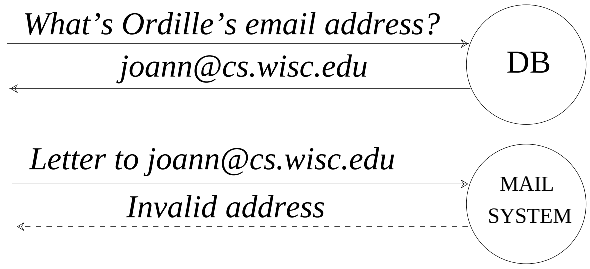

Figure 1.
The name service database supplies a hint.
If the hint is invalid, the mail system notifies the user.

3.1. Name Service Characteristics

In name services the object, not the database, is the authority for information. Name service responses are hints [23] that speed binding with the object. Users validate the hints when they contact the object, and recover if the hint is incorrect. Since serializability of query responses is so costly (if not impossible to obtain), systems with weaker consistency greatly improve performance by supplying hints that are seldom wrong and methods for recovering from the errors that do occur. For example, consider Figure 1. In this example a user asks the database for Ordille's electronic mail address. The user validates the electronic mail address received by sending mail to that destination. If the mail is not returned, then the address is valid. Externalities can cause name service information to be invalid because object naming and binding are separate operations. For example, it is possible for a person's electronic mail address to change between the time of the name service query and the mail system's subsequent attempt to deliver a message. It is even more common for a person's address to change before the new address is recorded in the name service. Problems with invalid data can be reduced if external systems provide forwarding services to translate old binding information to new binding information. In a system where the database is inconsistent with the world, worries about getting a true picture of a strictly consistent database seem pointless. Moreover, such a state is hard to define; problems of scale, autonomy, and availability make global synchronization impossible.

This is not say there is no sense of consistency in contemporary name services. Name services guarantee *tuple level consistency*, that a tuple will describe the state of an object at some time. (Forwarding can be used to correct invalid hints, because each tuple is consistent.) For example, an electronic mail address might be incorrect, but it was once the address of the person to be contacted. Name service queries are weaker than weakly consistent t-bound queries [8], providing consistency at the level of a single tuple rather than a collection of tuples. Collections of tuples are not updated in a single transaction, because name service information has no interdependencies that require full transactions. The creation, deletion, updating and reading of a tuple is atomic. Replicas of partitions can diverge from the collection level consistency of transactions as long as tuple level consistency is maintained. In the Domain Name System [14], local caches use timeouts to maintain tuples that contain host name to address translations. If a query requests a tuple that has timed out in the cache, the name service retrieves the tuple again from the source partition. When a tuple is updated in Grapevine [3], the system timestamps the change and propagates the tuple to its replicas. Two different partitions can update an attribute in the same tuple without knowing about each other's update; Grapevine uses the timestamp to identify the most recent change, which it then keeps. When tuples are read from many partitions in a name service, the collection of tuples represents information that may never have been current at the same time.

3.2. Better than Luck Semantics

Tuple consistency still leaves us with luck semantics, because we do not have a good idea of how the database changes with time. When we suspect that tuples are missing from a response or find that a tuple is invalid, we would like a clearer notion of how to obtain the correct information from the name service. Consider the response in Figure 2 for a query about Ordille's electronic mail address. What information might we add to this response to get a clearer understanding of its usefulness? This is the information we would like to capture in the model of consistency. If the name service makes information about consistency visible to the user, the user will be better able to recover from bad hints.

Name	Email Address
Joann J. Ordille	joann@kontiki.cs.wisc.edu
Joann J. Ordille	joann@cs.wisc.edu
Joann J. Ordille	joann@research.att.com

Figure 2.
Name service responses containing Joann J. Ordille's electronic mail address.

Temporal information provides one possible measure of consistency. If the information in Figure 2 included the last modification time for each tuple, it would be possible to choose the electronic mail address from the most recent tuple. Tuple modification times are already supplied by name services like X.500 and the CSNET name service. A name service could also provide the modification time for the collection of tuples in a response; tuples added after the collection modification time might not be reflected in the response. The collection modification time can indicate when a global index was last constructed or when the collection was created for the local cache. The collection modification time is simple to obtain from a database with traditional consistency because each replica is current. It is also simple to obtain from systems with timestamped replicas like X.500, but may not be simple to obtain in a system like Grapevine where updates arrive out of order. This measure of modification time differs from the types of consistency discussed for quasi copies [1] because the user discovers the inconsistency of the hint and demands more current information. In quasi copies, the system automatically updates caches to reflect a consistency requirement specified in advance by the user. Updating the user's information on demand consumes less system resources and is simpler for the user than specifying abstract consistency requirements.

Visible measures of consistency, such as modification time, make it possible to process parts of queries in the background. If a partition is unavailable, partial results can be given to the client and optionally the remainder of the results can be delivered when they become available. The results are easy to interpret because they contain their own consistency information. In the case where consistency is measured with modification times, users can request more recent information from the name service when they suspect information to be missing or invalid. Visible measures of consistency also provide more freedom for addressing the problems of scale and autonomy. If modification time measures are used, indices can be constructed periodically using the same query facilities that component systems supply to users. Since individual tuples in the name service change infrequently, the name service can periodically update its indices and caches while still maintaining high accuracy for users [23]. Autonomy is not violated, global synchronization is not necessary, and query performance is improved.

Other types of consistency are possible. Terry uses the expected lifetime of an object to estimate the probability that information about the object is current [23]. If we automate the validation of tuples (perhaps by using methods in an object-oriented paradigm), we could record the relative frequency of invalid information in different components of the multidatabase. This could lead to a consumer rating service for the information in component systems. As we consider expanding name services to support electronic publishing, rating the reliability and completeness of competing indices of the same information would be invaluable. The ultimate goal should be a systematic treatment of measures of consistency, their interrelationships, and effects on performance. It may be possible to structure the consistency of global multidatabases hierarchically. The top level of the multidatabase would support a weak form of consistency that is derived from stronger forms of consistency in the component systems. For example, modification time consistency information is simple to obtain from component databases that are strictly consistent. This type of approach is more likely to respect the autonomy of component systems, and remove the constant tension between consistency and autonomy in multidatabase research.

4. SUMMARY

Global computing presents new challenges of scale, autonomy, and availability to multidatabase systems. Database research can make a significant impact on global computing by providing fast selection queries in search spaces of tens of thousands of database partitions. New wide area techniques for query processing parallelism, caching, and partially delayed queries are necessary. Name service applications provide a good initial arena for wide area research, because they require a simpler model of consistency than traditional serializability. As we develop name services for global computing, we can develop an understanding of different measures and models of consistency, their interrelationships, and their effects on performance. It is reasonable to expect that the semantics of query processing will be significantly different in the global computing environment. The new semantics will rely on weaker forms of consistency, new models of cooperation, and visible scale, availability and consistency information.

5. REFERENCES

[1] R. Alonso, D. Barbara, and H. Garcia-Molina, ''Data Caching Issues in an Information Retrieval System,'' *ACM Transactions on Database Systems* **15**(3), pp. 359-384 (September 1990).

[2] C. Batini, M. Lenzerini, and S. B. Navathe, ''A Comparative Analysis of Methodologies for Database Schema Integration,'' *ACM Computing Surveys* **18**(4), pp. 323-364 (December 1986).

[3] A. D. Birrell, R. Levin, R. M. Needham, and M. D. Schroeder, ''Grapevine: An Exercise in Distributed Computing,'' *Communications of the ACM* **25**(4), pp. 260-274 (April 1982).

[4] International Telegraph and Telephone Consultative Committee (CCITT), ''The Directory,'' Recommendations X.500, X.501, X.509, X.511, X.518-X.521 (1988).

[5] P. B. Danzig, K. Obraczka, and A. Kumar, ''An Analysis of Wide Area Name Server Traffic,'' *ACM SIGCOMM Symposium on Communications Architectures and Protocols*, Baltimore, pp. 281-292 (August 1992).

[6] A. K. Elmagarmid, ed., *Database Transaction Models for Advanced Applications*, Morgan Kaufmann Publishers, Palo Alto, CA (1992).

[7] A. Emtage and P. Deutsch, ''archie – An Electronic Directory Service for the Internet,'' *Usenix Conference*, San Francisco, pp. 93-110 (Winter 1992).

[8] H. Garcia-Molina and G. Wiederhold, ''Read-Only Transactions in a Distributed Database,'' *ACM Transactions on Database Systems* **7**(2), pp. 209-234 (June 1982).

[9] D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. W. O'Toole, Jr., ''Semantic File Systems,'' *Thirteenth ACM Symposium on Operating Systems Principles*, Pacific Grove, California, pp. 16-25 (October 1991).

[10] R. E. Kahn and V. G. Cerf, *The Digital Library Project*, Corporation for National Research Initiatives, Reston, VA (March 1988).

[11] L. Landweber, M. Litzkow, D. Neuhengen, and M. Solomon, ''Architecture of the CSNET Name Server,'' *SIGCOMM Symposium on Communications Architectures and Protocols*, Austin, pp. 146-149 (March, 1983).

[12] W. Litwin, L. Mark, and N. Roussopoulos, ''Interoperability of Multiple Autonomous Databases,'' *ACM Computing Surveys* **22**(3), pp. 267-293 (September 1990).

[13] M. Lottor, ''Internet Growth (1981-1991),'' Request for Comments 1296, DDN Network Information Center, SRI International, Menlo Park, CA (January 1992).

[14] P. V. Mockapetris, ''Domain Names - Concepts and Facilities,'' Request for Comments 1034, DDN Network Information Center, SRI International, Menlo Park, CA (November 1987).

[15] J. J. Ordille and B. P. Miller, ''Distributed Active Catalogs and Meta-Data Caching in Descriptive Name Services,'' *Thirteenth International IEEE Conference on Distributed Computing Systems*, Pittsburgh(May, 1993).

[16] J. J. Ordille and B. P. Miller, ''Lost in a Labyrinth of Workstations,'' *Third Workshop on Workstation Operating Systems*, Key Biscayne, pp. 52-55 (April, 1992).

[17] J. J. Ordille and B. P. Miller, ''Nomenclator Descriptive Query Optimization in Large X.500 Environments,'' *ACM SIGCOMM Symposium on Communications Architectures and Protocols*, Zurich, pp. 185-196 (September, 1991).

[18] L. Press, ''The Net: Progress and Opportunity,'' *Communications of the ACM* **35**(12), pp. 21-25 (December 1992).

[19] M. F. Schwartz, ''The Networked Resource Discovery Project,'' *IFIP XI World Congress*, San Francisco, pp. 827-832 (August 1989).

[20] M. F. Schwartz and P. G. Tsirigotis, ''Experience with a Semantically Cognizant Internet White Pages Directory Tool,'' *Internetworking: Research and Experience* **2**(1), pp. 23-50 (March 1991).

[21] S. Sechrest and M. McClennen, ''Blending Hierarchical and Attribute-Based File Naming,'' *Twelfth International IEEE Conference on Distributed Computing Systems*, (1992).

[22] A. P. Sheth and J. A. Larson, ''Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases,'' *ACM Computing Surveys* **22**(3), pp. 183-236 (September 1990).

[23] D. B. Terry, ''Caching Hints in Distributed Systems,'' *ACM Transactions on Software Engineering* **13**(1), pp. 48-54 (January 1987).