Bulletin of the Technical Committee on

Data Engineering

March, 1993 Vol. 16 No. 1 **IEEE Computer Society**

Letters

Special Issue on Scientific Databases

Conference and Journal Notices

Important Membership Message

To Current Members of the Technical Committee on Data Engineering:

There is both good news and bad news in this letter. The good news is that we are well on our way to being able to distribute the Bulletin electronically. This low cost method of distribution should ensure the long term economic survival of the Bulletin. It should permit us to continue to provide the bulletin to all members free of charge. The bad news is that if you do not re-enroll as a member of the Technical Committee, then the June 1993 hardcopy issue of the Bulletin is the last copy of the Bulletin that you will receive.

Our annual revenue, which comes almost entirely from sponsoring the Data Engineering Conference, is not sufficient to cover the costs of printing and distributing four issues of the Bulletin a year. Four issues is, we agree, the minimum publication schedule that is reasonable in order to bring you timely information on technical and professional subjects of interest in data engineering. Long term survival then requires that we limit free distribution to those that can receive the Bulletin electronically. We are working on arranging hardcopy distribution via paid subscription for those that are not reachable electronically or who simply prefer receiving hardcopy. The annual subscription fee for four issues is expected to be in the $10 to $15 range.

Please be aware that failure to enroll means that you will not remain a TC member, and hence that you will no longer receive the Bulletin.

Electronic Enrollment Procedure

Electronic enrollment is the preferred method of becoming a member of the Technical Committee on Data Engineering. To enroll electronically, please use the following procedure:

1. Send e-mail to TCData@crl.dec.com and include in the subject header the word "ENROLL".

2. You will then be sent via an e-mail reply, an electronic membership form that will request:

   Name, IEEE membership no., postal address, e-mail address

   In addition to the above information, you will be asked a number of questions on issues that will affect how the Bulletin's distribution will be handled. These questions will include whether you are interested in subscribing to the hardcopy version of the bulletin. This information will enable us to plan our print runs and determine the parameters of electronic distribution.

3. You should then e-mail the electronic application form to TCData@crl.dec.com, with the word "APPLICATION" in the subject header.

Alternative Enrollment Procedure

In the event that you cannot reach the above electronic mail address, you may enroll in the Technical Committee on Data Engineering by sending postal mail to David Lomet, whose address is given on the inside cover of this issue. Please clearly state that you want to enroll as a member of the TC. A hardcopy application will then be sent to you requesting the same information as described above.

Please note that if you are unreachable via electronic mail, that you will not receive a free copy of the Bulletin. Only hardcopy will be available to you which can be obtained only via paid subscription.

Rakesh Agrawal, David Lomet
TCDE Chair, Editor-in-Chief

The Evolution of the Bulletin

The current issue of the Bulletin represents a substantial step forward in our effort to ensure that we can continue to bring you on a regular basis the special issues that are unique to the Bulletin. This entire issue has been assembled electronically. We will be sending it out on a trial basis to a select list of TC members so as to test our ability to distribute the Bulletin electronically. This will permit us to gain confidence in this process and to plan for a much wider electronic distribution of the June issue. The June issue will also be published and distributed in hard copy to our entire current membership list. However, the June issue is the last one that we will distribute in that fashion, as indicated in the preceding membership announcement.

I would like to take this occasion to thank two people who contributed substantially to the progress described above. First, Mark Tuttle, a theory researcher and colleague of mine at Digital's Cambridge Research Lab spent many hours carefully crafting latex definitions so as to provide the professional appearance that you see. His work also automated much of my task of assembling each issue of the Bulletin, a contribution that readers do not see but that I greatly appreciate. There must be some deep neurological connection between theory and "Tex" proficency for Mark has surely done a superlative job. Second, I want to thank Yannis Ioannidis, the issue editor, for patiently dealing with all the problems that arise in the first trial of a brand new process. This was in addition to the normal substantial effort that is required to produce an issue of the Bulletin.

As you can see, the current issue is devoted to scientific databases. It contains many figures and three bit-mapped medical images. I hope that we can continue to provide this high level of graphics. Our trial electronic distribution is intended to test that aspect among others. Issues of the Bulletin that are planned include one on transaction models and a second on geographic databases. So I have real confidence that the Bulletin will continue its valuable role of documenting progress in areas of current interest to the database technical community.

The current issue also contains two conference notices. Both conferences are affiliated with the Technical Committee. The TC sponsors the Data Engineering Conference and provides cooperation status for VLDB. Providing publicity furthers our intent to lend our support to such quality conferences.

Let me close, as I did in the previous issue, by urging all of you to follow the re-enrollment procedure and thus ensure your continued receipt of the Bulletin.

David Lomet
Editor-in-Chief, DE Bulletin

Letter from the Special Issue Editor

If one were to conduct a survey among researchers from all sciences asking for the greatest challenges in their particular discipline, one would expect very different answers from scientists of different backgrounds. Although this is true in general, there is at least one issue that is currently perceived as a major challenge in most scientific disciplines: data management. Traditional techniques employeed by scientists to manage their data are no longer adequate. On the other hand, current database systems are not the appropriate tools either. Several research issues must be addressed before the huge gap between the needs of scientists and what database technology can offer closes.

Scientific databases have become an important new area of database research, covering a wide variety of problems, e.g., management of unprecedented amounts of data, mass storage systems, unconventional data types, new forms of user-system interaction, heterogeneity, imprecise data, missing and/or redundant data, collaborative work, data mining, data compression, and data visualization. The papers in this issue deal with some of the key research ideas that are being pursued regarding several of these problems. I hope that they will spur the interest of others to join in this effort to advance database technology so that it can serve the needs of many other sciences. Below I give a brief overview of the essence of each paper.

The first four papers describe efforts that touch upon several scientific disciplines. Shoshani describes an evolutionary approach that the LBL group has taken to develop tools for scientific data management and the variety of scientific disciplines in which these tools are being used. Cushing et al. present their efforts to use object-oriented databases to address problems of program and data interoperability, with specific emphasis on computational chemistry experiments and protein structure verification. Pfaltz and French discuss the ADAMS language, which allows scientists to directly access possibly shared data from a variety of programming languages. Ioannidis et al. highlight the key issues involved in the development of a desktop experiment management system that helps scientists in managing their experimental studies.

The next three papers primarily address data management problems in the earth sciences. Stonebraker gives an overview of the Sequoia 2000 project, whose aim is to build a better computing environment for global change researchers. Hachem et al. provide the main features of the Gaea system, a spatio-temporal database system targeted for use by geographers involved in global change studies. Smith et al. describe their efforts to design and develop a modeling and database system that aids scientists involved in modeling various activities in the Amazon basin.

The next two papers focus on dealing with medical images. Arya et al. discuss the main features of QBISM, a system whose goal is to efficiently handle 3-D spatial data sets created from 2-D medical images, emphasizing querying and physical database design. Cardenas et al. address a similar problem, with an emphasis on integration and interoperation of multiple existing medical data repositories.

Finally, Wolniewics and Graefe describe their use of the Volcano extensible query optimizer generator to optimize combinations of database retrievals and numerical computations.

The authors of all these papers have put much effort in preparing them. I want to thank them for the excellent work that they did to make this issue a reality.

Yannis Ioannidis
University of Wisconsin, Madison

A Layered Approach to Scientific Data Management at Lawrence Berkeley Laboratory

Arie Shoshani
Lawrence Berkeley Laboratory
Berkeley, California 94720

1 Introduction

Over the last few years, we have been involved with several scientific projects that require data management support. These include databases for genomic, epidemiological, climate modeling, and superconducting magnet applications. Our research program is oriented towards the development of data management techniques and tools in support of such scientific applications.

In the past, database management systems (DBMSs) were rarely used in scientific applications. Typically, scientists dealt directly with files whose structure was specialized for the application. Recently, it has been recognized that the complexity and the amount of data being collected require data management tools to keep track of the data being generated by experiments, simulations, and measurements.

There are three aspects of "large" that occur naturally in scientific applications. The most obvious is the large volume due to automated measuring devices or simulations. The second aspect is the complexity of the information structure itself, which involves a large number of objects and associations between them. For example, a typical chromosome database may includes 40-50 objects, such as genes, markers, maps, various sequence types, and their associations. The complexity of the information structure is, in general, independent of the volume of the data. The third aspect of "large" is the number of datasets that exist in an application, their origin, whether they were derived from each other, when and where they were collected or generated, etc. It is not uncommon to have in a single database hundreds of such datasets. As we discuss below, we address the first aspect of "large" by developing a seamless interface to specialized mass storage access, the second aspect by providing a migration path to new and more powerful data management systems, and the third aspect by modeling the information about datasets as a metadatabase in its own right.

2 Approach

Most scientific applications require support for specialized data structures and operations. Examples are multi-dimensional data structures in space and time (such as climatology applications), or sequence structures (such as DNA sequences). Operations, such as selecting a "slice" in space and time, or searching for partially "overlapping" sequences are natural in the application domain. Such structures and operations are not supported by commercial relational DBMSs. Consequently, there is much interest in new database management technologies, such as "Extended Relational Database Systems", and "Object-Oriented Database Systems". Such systems permit the definition of application specific data structures and objects. While using such new technologies seems to be a promising approach, we have found from our involvement with various scientific applications that, in practice, scientific users are still struggling with ways to capture the conventional aspects of the data, such as the information about experiments being conducted, keeping track of laboratory equipment, personnel, references to

relevant articles, etc. Also, they tend to select and use existing proven technology (because of short term pressures), which means, of course, commercial relational database management systems. Since the most important need is to have systems that store and manage the information, scientists are content to perform the specialized operations (such as a spatial search, or finding overlaps between DNA sequences) outside the data management system, that is, in their application programs.

These observation led us to adopt an evolutionary approach to the support of scientific applications. In the short term, we are developing techniques and tools that make it possible for scientists to use existing commercial relational database technology. At the same time, we need to have a way of migrating, in the future, to new technologies. This is achieved by insulating the application programs from the specific underlying database system by using an intermediate object model. Currently, we are developing translators from object models to commercial relational data management systems. In the future, we plan to implement the same object models on top of new database systems. This will provide a smooth migration environment, since existing application programs will continue to interact with the same model. However, because the underlying technology will be richer (and extensible) it will be possible to enrich the object model as well.

A second problem is one of efficiency. It is often the case that the use of commercial general purpose software provides less efficient data access than the specialized file level software written for an application. For example, current database management systems do not provide direct access to tertiary storage, and specialized software is used to access files from mass storage. In addition, there may be packages that provide efficient computation for some specific operation, such as sequence matching. The problem is how to interface these specialized software to work with other database management software. Our approach is to use a single object level interface which, in turn, can access multiple underlying software modules to achieve greater data access efficiency for applications with special computation and/or storage requirements. Our approach is discussed below in the framework of a "layered database technology".

3 The advantages of a layered database technology

It is convenient to think about database support in terms of layers of software: the file layer, the data system layer, the object layer, and the application layer.

Most scientific application programs have been developed directly on top of file systems. Typical file systems (provided by the underlying operating system) hide from the user the details of where the file actually resides (on disk or some tertiary storage, such as a tape robot). The user is also unaware of complex file migration policies.

A level above the file system is the data system level. Database management systems which were developed in order to provide richer data models to describe complex applications provide this level. This level provides other desired functionality, such as crash recovery, concurrency control, and maintenance of integrity constraints. Some database systems have been developed on top of the file systems provided by the operating system, but because of efficiency and portability considerations modern database management systems manage physical devices, such as disks, directly.

The level above the data system level is the object level. While relational database systems have become commercially successful, they are still unnatural for the application scientist. The main reason is that users are forced to use database-specific concepts and operations, such as "normalized relations" and "join" (or "outer join") operations. These concepts are not only foreign to application scientists, but also do not correspond directly to application-specific data structures and operations. Consequently, object level models that are closer to the way users view their data have been introduced. These models facilitate the representation of information in terms of objects (or entities), their attributes, and their or associations with other objects. Interacting at the object level has several potential advantages: 1) users deal with simpler, more intuitive concepts, 2) data definition and queries are more concise (about an order of magnitude gain), and 3) the application programs become independent of the particular underlying database system. However, in order to gain the above benefits these models need to be

implemented as independent systems or on top of existing systems. Unfortunately, object level models (e.g., the Entity-Relationship model) became popular only as database design tools, and thus it was still necessary for users to use the database language (such as SQL) in order to define the relational schema and to query the relational database. New technologies (such as Object-Oriented database systems) may provide the means for supporting richer semantic data models at the object level.

Another layer above the object layer is necessary since even a rich semantic model may not be sufficient to model scientific applications. From our experience with scientific projects, we have learned that some applications have requirements that cannot be described directly in terms of databases objects. For example, scientific protocols and laboratory experiments are best described in terms of processes, where a process step may have several inputs and generate several outputs. Processes are best described in terms of data flow diagrams or similar representations. Even if the object model is extended to support such a functionality, customization at the application level will still be necessary, such as providing specialized graphical interfaces for chemical or biological structures.

The main advantage of working with a "layered database technology" concept is that in scientific applications it is useful to provide interfaces to these various levels. For some applications, interfacing directly to files is more appropriate. For example, there is extensive amount of software for climate modeling that interfaces directly to files. Such applications need an intelligent, efficient mass storage server to manage vast amounts of data rather than the use of a data management system. Some applications may benefit from interfacing to a "customized" data management system that supports specialized data structures, such as DNA sequences. Those application level interfaces that can be designed to be supported by the object level system would enjoy the benefit of being independent of the specific database technology used.

4 Technology migration

There are many proposals for object level models, often called semantic data models. When such models are developed on top of some data management system (rather than being used only as an abstract model for database design), they are limited by what can be supported through translators. We have shown that a fairly powerful model (such as an Extended Entity-Relationship (EER) model that models generalization and full aggregation [1]), can be correctly and fully implemented on top of a relational database management system. However, it will be necessary over time to migrate to a more powerful database system in order to accommodate more complex scientific applications.

Our goal is to initially support applications through the object level on top of relational systems. This in itself provides the clarity, conciseness, and data management independence described above. However, by using an object level model we gain upward compatibility with future more powerful data models that will additionally support objects for representing complex structures such as spatial, temporal, and sequence structures. We believe that such models can be built on top of future technology, such as Object-Oriented database systems. However, we foresee that in the interim period there would be a need for special modules that will support the definition of specialized structures outside the DBMS. For example, a special module to support the definition of DNA sequences may be developed outside a relational DBMS. The object model could then be extended to include DNA sequence structures and operators, by using a translator that knows how to interface with both subsystems.

Our work in the past concentrated on the identification and characterization of the functional requirements of scientific applications. Based on these characterizations, we have developed conceptual models and physical data management techniques to support these requirements. For example, we have developed conceptual models for statistical (summary) data, temporal data, sequence data, and multidimensional data, and various physical methods for the efficient implementation of these models [e.g. 2,3,4,5,6]. In general, once a conceptual model has been defined for a special scientific need, then methods for its efficient implementation need to be developed.

Such methods can then be implemented as separate modules that can be used in conjunction with commercial relational systems to augment the object level model.

5 Existing and planned projects

We describe here briefly several projects currently taking place at Lawrence Berkeley Laboratory. These projects follow the approach mentioned above.

One project is devoted to the development of generic object level tools. So far, we have developed schema translators from the EER model to several popular relational database management systems, including Oracle, Ingres, Sybase, and Informix. We also developed a query translator from an object level query (which refers directly to the objects in the EER model) to SQL (so far only SYBASE was targeted) [7]. In addition, we have developed graphical editors for specifying the database structure and for specifying queries at the object level. These tools have been used by projects at LBL and are used at over 30 institutions in various countries. Our current and planned work with several scientific applications is described below.

There are two aspects of the Human Genome project that we are involved in at this time. The first is a Chromosome Information System (CIS), which uses the EER model to describe the application [8]. The object structure definition (schema) is fairly complex, and contains about 35 objects. We have used database tools developed at LBL to translate this object structure definition into a relational database definition. The resulting relational schema was over 3000 lines of code in SQL, and required the use of the trigger mechanism of Sybase. The generation of such code by hand would have been an error prone and complex task that would require a database expert. Instead, a biologist familiar with the application was able to describe the schema graphically and concisely in a matter of hours.

Another activity associated with the Human Genome project is the Laboratory Information Management System (LIMS). The purpose of this project is to provide software tools for the rapid development of databases supporting a variety of sequencing protocols. The LIMS databases developed using these tools will allow scientists to specify and query the databases in terms of protocols and objects, and will be easily adaptable to protocol changes. The LIMS database tools are based on a data model called the Object-Protocol Model (OPM) developed at LBL [9]. OPM provides constructs for directly modeling object and protocol structures specific to LIMS databases. Since protocols often involve a series of steps which are also protocols, OPM supports the specification (expansion) of protocols in terms of alternative and sequences of component (sub) protocols. Protocol expansion allows protocol designers to progressively specify protocols in increasing levels of detail. The OPM-based database tools target commercial relational DBMSs, because of the widespread use of such DBMSs in molecular biology laboratories and genome centers worldwide. This entails translating OPM database specifications and data manipulation operations into relational database specifications and queries. Our approach to this translation is to use the existing EER-based tools for generating relational database specifications and queries from EER database specifications and queries. This is an example of using the object level translation technology to automatically support the application specific model.

The above object level tools have been used for a couple of other projects. One is the Comprehensive Epidemiological Data Resource (CEDR), whose goal is to collect all the information on low-level radiation in DoE sites and laboratories into a single repository, and to make it available for further analysis to researchers inside and outside of DoE. An interim system was developed using the current tools, and a future system will use the object level query translator mentioned above [7]. A second project using these tools is the SSCL magnet laboratory mainly for developing object level schemas and for generating Sybase database definitions.

Two additional projects currently pursued are concerned with managing massive amounts of data on tertiary storage. Current database management systems (both Relational and Object-Oriented) interface only to disks. Thus, the management of tertiary storage through a standard interface will be beneficial to both projects. One project is in collaboration with Argonne National Laboratory and the University of Illinois [10]. Its purpose is

to develop efficient access to high energy "event" data (data that results from collider experiments). The second project is in collaboration with Lawrence Livermore Laboratory. Its purpose is to develop efficient access to massive amounts of spatial and temporal data. Initially, it will concentrate on climate modeling data [11].

These two projects share the goal of organizing the data in such a way as to optimize its access for subsequent analysis. The main idea is that the datasets will be partitioned as to reflect their most likely access. The partitioning of event data is described by using an object model (EER), and thus our tools are already being used for the relational definition of the object structure. Similarly, we use the object level tools to describe the partitioning information as well as the metadata for the climate datasets.

References (partial list)

[1] V.M. Markowitz and A. Shoshani, Representing Extended Entity-Relationship Structures in Relational Databases: A Modular Approach, ACM Trans. on Database Systems, 17, 3 (September 1992), pp. 423-464.

[2] Shoshani, A., and Rafanelli, M., A Model for Representing Statistical Objects, International Conference on Management of Data, COMAD '91, India, December 1991.

[3] Segev, A. and Shoshani, A., Logical Modeling of Temporal Data, Proceedings of the International Conference on Management of Data (SIGMOD), May 1987.

[4] Li, P., Sequence Modeling and an Extensible Object Data Model for Genomic Databases, Ph.D. thesis, Lawrence Berkeley Laboratory Technical Report LBL-31935, December 1991.

[5] Rotem D., Clustered Multiattribute Hashing, ACM Symp. on Principles of Database Systems, 1989.

[6] Rotem, D., Spatial Join Indices, 7th International Conference on Data Engineering, Japan, 1991.

[7] V.M. Markowitz and A. Shoshani, Object Queries over Relational Databases: Language, Implementation, and Applications, 9th International Conference on Data Engineering, April 1993.

[8] V.M. Markowitz, S. Lewis, J. McCarthy, F. Olken, and M. Zorn, Data Management for Molecular Biology Applications: A Case Study, in Proceedings of the 6th International Conference on Scientific and Statistical Database Management, June 1992.

[9] Chen, I.A., and Markowitz, V.M., The Object-Protocol Model: Design, Implementation, and Scientific Applications, submitted for publication.

[10] A. Baden, L. Cormell, C. T. Day, R. Grossman, P. Leibold, D. Lifka, D. Liu, S. Loken, E. Lusk, J. F. MacFarlane, E. May, U. Nixdorf, L. E. Price, X. Qin, B. Scipioni, and T. Song, Database Computing in HEP—Progress Report, Computing in High Energy Physics 1992.

[11] P. Bogdanovich, S. Cann, R. Drach, S. Louis, G. Potter, G. Richmond, D. Rotem, H. Samet, A. Segev, S. Seshardi, A. Shoshani, Optimizing Mass Storage Organization and Access for Multidimensional Scientific Data, 12th IEEE Symposium on Mass Storage Systems, April 1993.

Connecting Scientific Programs and Data Using Object Databases

Judith Bayard Cushing *David Hansen* *David Maier* *Calton Pu*
Dept. of Computer Science and Engineering
Oregon Graduate Institute
Beaverton, Oregon 97006-1999
{cushing, dhansen, maier, calton}@cse.ogi.edu

Abstract

Scientific applications and databases rarely interoperate easily. That is, scientific researchers who use computers expend significant time and effort writing special procedures to use their program with someone else's data, or their data with someone else's programs. These problems are exacerbated in modern computing environments, which consist of multiple computers of possibly different types. Database researchers at the Scientific Database Laboratory at the Oregon Graduate Institute are using object-oriented databases to address problems of program and data interoperability. For the domain of computational chemistry, we are extending an existing object database system to facilitate the invocation, monitoring, and output capture of a variety of independently developed programs (aka legacy applications). A complementary project in materials science explores providing application programs with a common interface to a variety of separately published datasets. We are also developing an object-oriented toolbox to access the contents of a database of protein structures. We describe these three projects, then discuss their status and our future directions.

1 The Interoperability Problem for Scientific Computing

In a perfect world, data from one program could be transparently used as input to another. The world of scientific computation unfortunately is far from perfect, and its rich legacy of data and programs carries a major disadvantage: a plethora of data formats and input conventions. Business data processing has worked to solve this problem through common data models and shared databases, but current record-oriented database technology (such as relational database systems) does not match scientific applications well. Scientific data types such as multi-dimensional matrices or crystal structures cannot be implemented efficiently and directly using record-oriented models. We believe that object-oriented systems avoid such shortcomings, and we have identified computational chemistry, materials science, and protein structure analysis as areas in which to explore object-oriented systems that integrate diverse programs and data. Our approach constructs, for each domain, a unifying data model that encompasses a range of programs and data sources, creating a "plug-and-play" environment.

2 Diverse Computational Chemistry Programs

In the realm of computational chemistry, programs implementing quantum mechanics algorithms compute molecular properties given basic molecular structure data. These computationally intensive applications require the storing, viewing, and sharing of large amounts of specialized quantitative information, and could benefit from

using database systems. The environment of the computational chemist is further complicated in that these standalone applications run on different kinds of computers, and data must be transferred between them. The computational chemistry database project, a joint effort with Battelle Pacific Northwest Laboratories, aims to provide a database of past experiments and to render results computed by different codes comparable[CMR+92b, CMR92a].

Capturing both inputs and outputs in common formats, however, requires connecting computational chemistry programs directly to the database. To that end, we are currently defining and implementing a mechanism, dubbed "computational proxy", that relies on a common computational model along with descriptions of the applications' input and output files to provide the required interfaces. A computational proxy object "stands-in", within the database, for a computational experiment in preparation, currently in process, or recently completed. Using the proxy mechanism and the information in the proxy, a user is able to start up and control ongoing computational processes, and capture information about a given computational experiment. When the user schedules a run, a proxy uses a description of the application to automatically transform experimental attributes held in the database into textual inputs appropriate for a given application. If necessary, the input files are transferred to the computer on which that application is to run[CMR93]. Figure 1 illustrates the computational proxy encapsulation of syntactic detail of different programs and computer types.

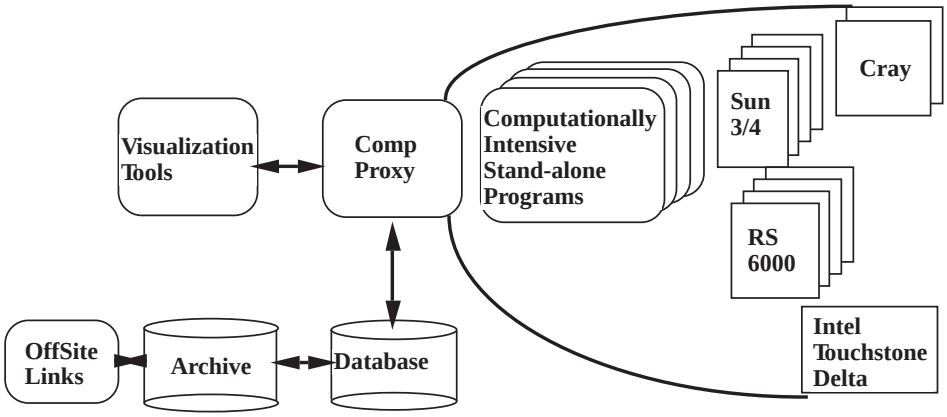


Figure 1: Computational proxies encapsulate computational applications.

Computational proxies aid in interfacing standalone, heterogeneous applications to a common database, simplifying the computing environment for domain scientists and masking syntactic differences among scientific applications. Proxies also help in the capture of inputs, intermediate results, and outputs of computational experiments, as well as associated descriptive data. Since the proxy mechanism renders these data into a common format, data from different applications can be compared, and the output of one program can be more easily used as the input to another. Computational proxies have been implemented in C++ and the object-oriented database system ObjectStore for the General Atomic and Molecular Electronic Structure System (GAMESS)[Rao93]. The project will later support additional application programs such as Gaussian. We believe proxies are generalizable to other computational programs, different domain sciences, and any object-oriented database system.

3 Integrating Materials Science Data Sources

Materials scientists are prolific users of computers. Modeling techniques and algorithms are well known, and refined and widely available computer-readable factual databases abound. Unfortunately, any given materials science application is typically developed in isolation, using a specifically tailored data model. Furthermore, scientists typically access available computerized databases manually, in an off-line fashion. Thus, researchers

repeatedly construct and populate new custom databases for each application. Our materials science database research bridges the gulf between applications and multiple sources of data by providing a uniform object interface to datasets in diverse formats.

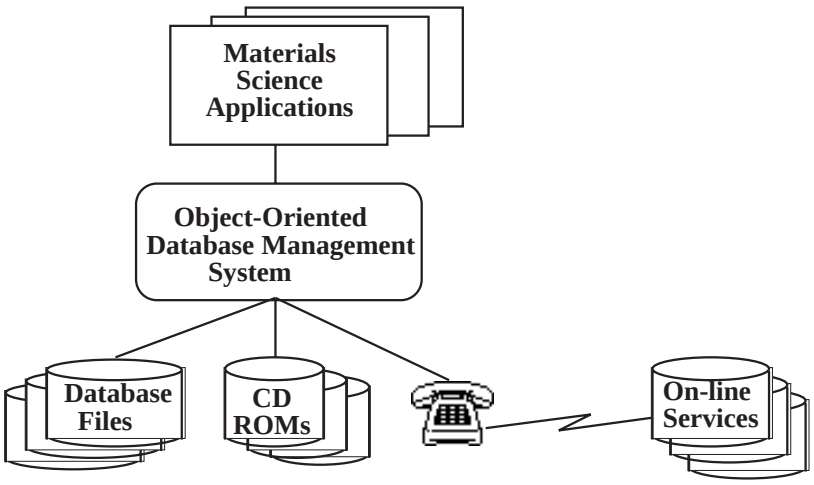


Figure 2: A single interface between programs and sources of data.

We have developed a unifying object-oriented data model to meet the needs of several materials science applications. This data model captures the essence of molecular and crystalline structure from a materials science perspective. We have implemented this data model in an object-oriented materials science database using the GemStone object-oriented database management system[HMSW93].

The database stores materials science data generated by users and user applications and provides transparent access to heterogeneous, commercial data sources. (See Figure 2.) The database currently provides access to the Electron Diffraction Database, distributed by the National Institute of Standards (NIST) on CD-ROM, as well as to files generated by the CAChe computer-aided chemistry system and the Desktop Microscopist. External data is cast into objects of the data model, providing users of the database with a single, object-oriented model of both internal and external data.

4 The PDB Toolbox

The Protein Data Bank (PDB) is a depository for the atomic structure of protein (and other) macro molecules. Currently with about 900 molecules occupying about 270 megabytes, the PDB is expected to grow to more than 6000 molecules by year 2000. Because of the complex structure of protein molecules, attempts to use relational DBMS's have not been totally successful. Jointly with Columbia University and Brookhaven National Laboratories, we are developing an object-oriented toolbox for PDB that will use modern software engineering principles and object-oriented DBMS technology[PSO+92]. The toolbox consists of software instruments to access the contents of PDB and will have the reliability, performance, usability, and extensibility of modern computer software. Now under development, the first demonstration program, PDBTool, will be used to verify molecular structures.

Because of the complex structure of PDB data, we have chosen an object-oriented approach and have found object identities, encapsulation, data abstraction, and inheritance to be critical features. Requirements of the PDB Toolbox include: (1) inspecting protein structures at different abstraction levels, i.e., chain, secondary structure, residue, and atom; (2) graphically presenting raw data and derived data; and (3) interfacing with other data sources such as the original PDB file format, relational databases such as SESAM, and standard formats such as the Crystallographic Interchange Format.

We have developed a three-level general architecture for the management of scientific data, applicable to many domains. To build the *user interface*, we used SUIT software developed at the University of Virginia, which provides graphical widgets to represent data in a windowed environment. We have implemented *verification algorithms* in C and C^{++} that calculate information such as torsion angles and Ramachandran plots. To implement the *storage manager* we relied on PDB files, the current standard storage system for crystallographic data.

The prototype PDBTool runs on any Sun platform and has been remotely run on an IBM RISC 6000 and an SGI Indigo workstation. The currently implemented verification tools accessible through PDBTool include the Ramachandran plot tool, which calculates and displays the ϕ and ψ angles of each residue, a 3-D graphical display of molecular structure, and histogram displays of geometric factors such as chain bond length, torsion angles, and chiral volumes. New tools as well as a protein query language are being actively developed and implemented.

The PDB Toolbox project has a scope larger than scientific data management, and a critical component is data storage in backends and efficient access to them. We have designed an abstract interface to backends, which consists of sequential access and direct access. Sequential access is implemented in the PDBTool as C^{++} iterators. We have implemented C^{++} iterators for the original PDB file format, in ObjectStore (using DML), and in the SESAM relational database.

5 Status and Future Work.

The current implementation of the proxy mechanism for GAMESS involves considerable custom code in the database. The main task currently is to develop a means to register new programs without the custom coding, and to explain how to adapt computational proxies to other program types and other domains. Among other things, the registration process involves specifying how inputs are to be formatted and how outputs are to be extracted. Some of this work is already being transferred to PNL and incorporated in the design of their laboratory support database for the Environmental and Molecular Sciences Laboratory.

In the materials science area, we have designed an initial domain model for crystallographic data, implemented that model in an object database, and constructed connections to external data sources. The connection to external sources uses a layered software architecture to hide format differences and give control over caching policies. We have been experimenting with the performance available with this architecture, and have been comparing that to a reimplementation in a different OODBMS. The next step is to look at approaches to connect the database to the Desktop Microscopist application. We are looking at extending our model and database support to accommodate information needed to calculate phase-structure diagrams.

The PDBTool architecture has been validated by the success of PDBTool development, demonstrated in the 1992 ACA meeting. Adding successive tools has been reasonably smooth. We are on schedule for the beta release of PDBTool as a software instrument for molecular verification in 1993.

Our long-range goal is a *Hybrid Data Manager* that contains generic facilities for connecting scientific programs and datasets in a variety of domains. However, our experience indicates that such database support is effective only after careful construction of a conceptual model to give a well-defined semantic basis to underlie the datasets and programs in use[MCPH93].

6 Acknowledgements

The authors acknowledge the collaboration of Dr. David Feller, Dr. Mark Thompson and D. Michael DeVaney of PNL in computational chemistry; Prof. James Stanley and Ramachandran Venkatesh of OGI in materials science; and Phil Bourne, Edwardo Aleesio and others of Columbia University, as well as members of the Brookhaven National Laboratory, in protein structure. Meenakshi Rao and Donald Abel, of OGI and Portland State, programmed

components of the computational chemistry database. Prof. Jonathan Walpole and Prof. Michael Wolfe of OGI are working with us on the design of the Hybrid Data Manager.

This work is supported by NSF grants IRI-9117008 and IRI-9116798, additional grants from the Oregon Advanced Computing Institute (OACIS) and PNL, and software grants from Object Design, Inc., and Servio Corporation. CAChe is a registered trademark of CAChe Scientific; Gaussian of Gaussian, Inc.; GemStone of Servio Corporation; and ObjectStore of Object Design, Inc. GAMESS is distributed by North Dakota State University and the USDOE Ames Laboratory; and the Desktop Microscopist by Virtual Laboratories.

References

[CMR92a] J. B. Cushing, D. Maier, and M. Rao. Computational chemistry database prototype: ObjectStore. Technical Report CS/E-92-002, OGI, Beaverton, OR, 1992.

[CMR+92b] J. B. Cushing, D. Maier, M. Rao, D. M. DeVaney, and D. Feller. Object-oriented database support for computational chemistry. *Sixth International Working Conference on Statistical and Scientific Database Management (SSDBM)*, June 9-12 1992.

[CMR93] J. B. Cushing, D. Maier, and M. Rao. Computational proxies: Modeling scientific applications in object databases. Technical Report CS/E-92-020, OGI, Beaverton, OR, 1993.

[HMSW93] D. M. Hansen, D. Maier, J. Stanley, and J. Walpole. An object oriented heterogeneous database for materials science. *Scientific Programming*, to appear 1993.

[Ken92] B. Kennedy. Architectural alternatives for connecting a persistent programming language and an object-oriented database. Master's thesis, OGI, Beaverton, OR, 1992.

[MCPH93] D. Maier, J. B. Cushing, G. Purvis, and D. Hansen. Object data models for shared molecular structures. *to be presented at the First International Symposium on Computerized Chemical Data Standards: Databases, Data Interchange, and Information Systems, Atlanta GA*, May 5-7 1993.

[Ohk93] H. Ohkawa. *Object-Oriented Database Support for Scientific Data Management: a System for Experimentation*. PhD thesis, OGI, Beaverton, OR, 1993.

[PSO+92] C. Pu, K.P. Sheka, J. Ong, L. Chang, A. Chang, E. Alessio, I.N. Shindyalov, W. Chang, and P.E. Bourne. A prototype object-oriented toolkit for protein structure verification. Technical Report CUCS-048-92, Dept. of Computer Science, Columbia University, 1992.

[Rao93] M. Rao. Computational proxies for computational chemistry: A proof of concept. Master's thesis, OGI, Beaverton, OR, expected: June, 1993.

Scientific Database Management with ADAMS

John L. Pfaltz *James C. French* *
Department of Computer Science
Thornton Hall, University of Virginia
Charlottesville, Virginia 22903
{pfaltz,french}@virginia.edu

1 Introduction

In direct response to the needs of scientific database management, we have developed ADAMS, a new language for describing, finding, and accessing persistent data by applications programs. One focus of the ADAMS project has been to provide database support for high performance scientific applications. This has led us to consider parallel databases as a means of achieving the desired level of performance. ADAMS follows an object-based approach to database management that we have implemented over a portable parallel programming environment. In our model, attributes and relationships are represented as functions.

Four of the really useful features of ADAMS to a scientist are:

1. every user has direct access (without an intervening database administrator) to a large, partitioned persistent data space in which data elements can be either private or shared;
2. ADAMS statements can be embedded in any procedural host language (C and Fortran have been implemented) so different applications can access and share data in a single persistent data space;
3. it supports very fast range search over type invariant numeric data with arbitrary precision; and
4. it provides the ability to easily modify the schema which helps support the process of model building and hypothesis testing.

2 Goals of ADAMS

ADAMS has not been developed with any specific database model, e.g., relational, semantic, or functional, in mind. Neither was it developed with respect to a particular programming paradigm, e.g., object-oriented. To understand ADAMS one must first understand the four database goals that it seeks to address.

First and foremost, its purpose has been to interface many different computing environments to a common, persistent data space. This arose from an experience several years ago in a NASA research division where we found that three groups, engineers coding in Fortran, CAD designers using C, and mathematicians doing finite element analysis in assembly language, could not share data regarding their common project — the design and analysis of airfoils. Consequently, a conscious decision was made that the applications code should be that of the programmers' choice, and that ADAMS would have to interface to it. This in turn dictated that ADAMS be

*This work supported in part by Dept. of Energy Grant no. DE-FG05-88EER25063 and JPL Contract no. 957721.

invoked by procedure calls; however, we wanted to present the programmer with a cleaner interface consisting of embedded statements [1], which the system itself would convert to the appropriate procedure calls.

Second, there should be different levels of data sharing and data privacy within the persistent data space. Some data might be widely shared throughout the system, some data might be common only to selected groups of users, while still other data might be completely private to a particular user.

Third, we wanted this persistent data space to be an extension of, or even a replacement for, the individual programmer's file system. Except for archival purposes, scientific data needs are not satisfied by systems managed by a database administrator. Not only does a scientist want to update existing data sets, he also wants to be able to: (1) create and name completely new persistent data sets; (2) dynamically reconfigure existing data sets to include new attributes or delete unnecessary attributes; and (3) dynamically create or destroy relationships between data sets. In short, individual programmers should be able to completely manage their own, and certain portions of the shared, persistent data space.

Finally, we envisioned a situation where the kinds of application code that now access persistent data within this data space could equally well access a variety of metadata. For example, one might have a program that could search through the entire data space for instances of data sets that relate measurements of dissolved oxygen to stream flow.

From the outset, we recognized that ADAMS would have to be a distributed, parallel database system, in order to achieve its design goals. Some of these design goals have not been fully met, but we believe that our existing prototype realizes many of them, and will serve as a stepping stone to realizing others.

3 The ADAMS Namespace

Not all the elements and sets in a data space need be named. But, in all databases, some sets, e.g., base relations, must have symbolic names. If any user is to have the capability of creating new data elements in his private, or in a shared, data space, he must also be able to symbolically name these elements. This requires dynamic management of a name space.

ADAMS offers the user a hierarchical name space consisting of four scope levels, *system*, *task*, *user*, and *local*. The *task* scope level is further partitioned into specific tasks, any of which may or may not be visible to a particular user. We use name space visibility to implement data privacy and data sharing. Names at *system* scope are visible and accessible to all users. Those at *task* scope are visible only to users for whom the specific task is visible. The *user* and *local* scopes are private; the former has private persistent element names, the latter names elements that exist only during process execution.

Our name space has been implemented within the ADAMS data space itself, so ADAMS code can search and dynamically modify its own name space. Name space management is tough; and there still remain unresolved issues, e.g. When can names be erased? Under what conditions can names be moved from one scope to another?

4 Implementation

Because of our intended application domain, the design goals stated above emphasize flexibility rather than performance. To achieve these goals, ADAMS has taken an implementation approach that is somewhat different from most database systems. First, in common with most object-based implementations, each element of the persistent data space is assigned a unique identifier, or *uid*. These *uid*'s never change; they are invariant for the life of the element. They are purely symbolic and not associated with any particular storage location. Second, although a user may declare and create logical structures in the persistent data space, such as tuples or objects, they are not actually represented as structures. Instead, they are represented functionally. For example, if t denotes a tuple with attributes a, b, and c, then to access the entire tuple, each attribute must be accessed individually by separate calls to the functions a, b, and c, each with the same symbolic argument, uid_t. If one views the database

in relational terms, ADAMS *vertically* partitions all relations by attribute. While performance was not an initial goal, this functional approach opens interesting possibilities for parallelism in a distributed environment. Many systems find it easier to distribute processes that to distribute data. This functional approach has been described in [2, 3]; here we will only touch on the basic highlights.

Functional evaluation, where the function can be dynamic as is the case with attribute functions, requires a very effective lookup mechanism. We have chosen to use O-trees [4], although mechanisms such as B-trees, or extensible hashing might also be appropriate. O-trees offer two advantages. First, they exhibit B-tree behavior, but yield much more compact indexes. An O-tree index is typically 30-50% the size of a corresponding B-tree [5]. Second, and possibly more important in database applications, they access on the basis of bit strings alone in a way that is both key type and key length invariant. Consequently, in ADAMS, both numeric and character data are accessed with precisely the same code. This opens up the possibility of indexing nontraditional data types, such as images or other arbitrary bit strings [6]. However, while we have stubs for image retrieval in our system, the current version has not fully implemented this capability.

One consequence of using O-trees and only bit string access has been our representation of numeric data. By means of a fast, simple transformation we are able to represent all numeric data (integer, real, or double precision) in persistent storage using a common format that ensures (almost) arbitrary precision [7]. This eliminates the need to know host language data types for retrieval, and eliminates the need to coordinate multiple indexes over the same attribute function for different computing environments (e.g., string types in C, Pascal, or Fortran).

The parallel implementation of ADAMS runs on top of Mentat [8, 9], a parallel, data flow, extension of C++. Very loosely speaking, Mentat separates its objects into two categories, computationally intensive objects which it calls Mentat objects, and other light-weight objects which are treated simply as C++ objects. Objects of the former category may be instantiated on any available processor in the system, in which case the Mentat runtime system uses dynamic data flow analysis to ensure synchronization. The availability of Mentat has greatly simplified the implementation of parallel ADAMS. ADAMS attribute functions are implemented as Mentat objects, consequently it is likely that the attribute functions a, b, and c mentioned above will be executing on different processors so that the cost of making multiple functional accesses to obtain a single logical structure may be little more than the cost of a single access.

Sets are important in any database, and the ADAMS language makes heavy use of the standard set operators, union, intersection, and relative complement. An ADAMS set is a set of *uid*'s, e.g., a relation consists of the set of *uid*'s denoting its constituent tuples. ADAMS *horizontally* partitions its sets into N subsets, where N is fixed by the system, independent of the number of available processors. A logical set s is actually implemented as N Mentat O-tree objects, $s[0]$, $s[1]$,..., $s[N-1]$. When an element (actually its *uid*) is inserted into s, the low-order bits of the *uid*, which we call its *parity*, are used to determine in which subset $s[k]$ it will actually reside. Now, to perform a set operation, say the intersection of r and s we can spawn N independent low level intersection operators, each of which only intersects $r[k]$ with $s[k]$. These N operators are independent, because if the same element (*uid*) exists in both r and s, and its parity is k, then it must be in just these two sets. We should note that element equality in ADAMS is based on *uid* equality, not on attribute values. This kind of horizontal distribution is not new; for example, it is the basis of hash joins. But, because ADAMS *uid*'s are persistent, symbolic identifiers rather than storage addresses, they never need to be redistributed. All sets, whether persistent sets, temporary retrieval sets, or the results of concatenated set operations, always exist in the same distributed form.

ADAMS tends to be slower with sequential I/O operations, i.e., where entire objects or tuples must be assembled for display. But, by simultaneously accessing multiple attribute streams, this sequentiality can be minimized[3].

Finally, we observe that, like many other database implementations, our ADAMS implementation is strictly layered. Except at the very lowest storage manager level, only symbolic *uid*'s are passed between procedures. Data values are represented as bit strings in preallocated buffers, and only in a single relatively high level, just below the application's host code are they converted to the format expected by that environment. This cleanly separates the ADAMS runtime system from the operational environments that it supports.

5 Research Applications

From the outset many of the research directions of the ADAMS project have been driven by the needs of real scientific applications. Some of these applications are described below.

Several important applications of interest have large sets of remotely sensed data. This data is in some sense an estimate of reality and has a measure of uncertainty associated with it. Part of the routine processing of this data involves running data correlation and fusion algorithms that require spatial retrieval of uncertain data. To support data correlation and fusion algorithms efficiently, we need to be able to query this data by, say, location. But, the location of the data points is not known precisely. Moreover, in some applications, the location of a query point is also uncertain. The usual method of obtaining the candidate retrieval set is to scan the data sequentially making individual chi-square (or some other) tests to determine if a record should be considered. Given an uncertain query point, we have developed techniques for estimating a "small" region of interest around point locations in the data sets. On the basis of this estimation, we are able to transform the problem into one for which point access methods are appropriate and use a two-dimensional range query to fetch the points in a rectangular region. Thus, efficient range querying has proved to be extremely important. Our work on O-trees has been directed toward this end.

We have also been collaborating with colleagues in the Department of Environmental Sciences to develop a database to support investigation into coastal wetland change. Our involvement in this work has been in two areas. First, we have been looking at techniques to capture the notion of environmental change. A number of interesting scientific applications are concerned with the detection of change with respect to variables other than time and we are looking at techniques to represent such change directly within a DBMS. Second, the process of scientific investigation is one of positing models and subsequently examining how well the models reflect reality. This leads to a feedback cycle of model refinement and new hypotheses. We have found that this effort requires a significant degree of freedom in database restructuring and, moreover, that the scientists must be able to make these modifications themselves. This has taken advantage of ADAMS's capability for flexible schema evolution.

It has been said that supercomputers change compute bound problems into I/O bound problems. Many of the scientific applications that we have been working with fall into this category. Therefore, it has been necessary from the outset to consider efficient solution to I/O problems on parallel hardware. This has greatly influenced our decision to support vertical as well as horizontal data decompositions. Because we are interested in high performance scientific computing, we have been measuring the performance of parallel I/O systems [10, 11] and looking for ways to improve their performance. This work has also led us to develop new metrics for I/O measurement. It has also uncovered anomalous behavior in parallel I/O subsystems.

Although ADAMS is a prototype research system, it has evolved into a rather stable robust one. Because of its layered design, we are able to experiment with different low-level implementations without changing the higher-level application interface. In addition to the applications described above, ADAMS is being used by students and faculty to develop a number of test applications of varying degrees of complexity. It is also used as a classroom teaching vehicle.

Acknowledgement. We would like to thank Yannis Ioannidis for his careful reading and comments that materially improved the presentation of this paper.

References

[1] J. L. Pfaltz, S. H. Son, and J. C. French, "The ADAMS Interface Language," *Proc. 3th Conf. on Hypercube Concurrent Computers and Applications*, Pasadena, CA (Jan. 1988), 1382-1389.

[2] J. L. Pfaltz and J. C. French, "Implementing Subscripted Identifiers in Scientific Databases," *Proc. 5th Inter. Conf. on Statistical and Scientific Database Management*, (Z. Michalewicz, ed.), Lecture Notes in Computer Science, 420, Springer-Verlag, April 1990, 80-91.

[3] J. L. Pfaltz, J. C. French, A. S. Grimshaw and R. D. McElrath, "Functional Data Representation in Scientific Information Systems," *Inter. Space Year Conf. on Earth and Space Science Information Systems (ESSIS)*, Pasadena, CA, Feb. 1992.

[4] R. Orlandic and J. L. Pfaltz, "Compact 0-Complete Trees," *Proc. 14th VLDB*, Long Beach, CA, Aug. 1988, 372-381.

[5] R. Orlandic and J. L. Pfaltz, "Analysis of Compact 0-Complete Trees: A New Access Method to Large Databases," *Proc. 7th FCT Conf.*, Szeged, Hungary Lecture Notes in Computer Science, 380, Springer-Verlag, Aug. 1989, 362-371.

[6] R. Orlandic and J. L. Pfaltz, "Q_0-trees: A Dynamic Structure for Accessing Spatial Objects with Arbitrary Shapes," Tech. Rpt. IPC-91-10, Institute for Parallel Computation, Univ. of Virginia, Dec. 1991.

[7] R. Haddleton, "Representing Numeric Data," Tech. Rpt. IPC-92-05, Institute for Parallel Computation, Univ. of Virginia, Aug. 1992. (Submitted to *Software Practice and Experience*.)

[8] A. S. Grimshaw, "Easy-to-use Object-Oriented Parallel Processing with Mentat," *Computer*, May 1993 (to appear).

[9] A. S. Grimshaw, J. L. Pfaltz, J. C. French and S. H. Son, "Exploiting Coarse Grained Parallelism in Database Applications," *PARBASE-90 International Conf. on Databases, Parallel Architectures and their Applications*, Miami Beach, FL (March 1990), 510-512.

[10] J. C. French, T. W. Pratt and M. Das, "Performance Measurement of the Concurrent File System of the Intel iPSC/2 Hypercube," *Journal of Parallel and Distributed Computing*, vol. 17, (1993), 115-121.

[11] J. C. French, "Characterizing the Balance of Parallel I/O Systems," *Proc. 6th Distributed Memory Computing Conf.*, Portland OR, April 1991, 724-727.

Desktop Experiment Management

Y. Ioannidis [*] *M. Livny* *E. Haber* *R. Miller* *O. Tsatalos* *J. Wiener*
Computer Sciences Dept., Univ. of Wisconsin
1210 W. Dayton St., Madison, WI 53706
{yannis,miron,haber,rmiller,odysseas,wiener }@cs.wisc.edu

1 Motivation

Traditionally, the scale and scope of an experimental study was determined by the ability of the research team to generate data. Over the past few years, we have been experiencing an unprecedented increase in the ability of small teams of experimental scientists to generate data. This has led to a shift in the balance between the different components of an experimental study. Today, it is not uncommon to find a study whose scale and scope have been determined by the ability of the team to manage the data rather than to generate it. Whether the discipline is experimental computer science [4], genetics, earth and space sciences, soil sciences, or high-energy physics, scientists are faced in their daily work with an experiment and data management bottleneck. Unfortunately, an experimental scientist can not find ready off-the-shelf management tools that offer both the functionality required by a scientific environment and an interface that feels natural and intuitive to the non-expert. While no special expertise is needed to manage a collection of images stored as files on a PC or as pictures in a paper notebook, existing database systems (DBMSs) that offer the desired functionality require expertise that most teams of experimental scientists do not have and can not afford to pay for. This poses a challenge to the database community to develop management tools that can be tailored by a typical scientific team to effectively manage their unique experimental environment.

To address this challenge, we have undertaken an effort to develop a desktop *Experiment Management System* (*EMS*) [2]. We view the desktop EMS as the sole interface between the experimental scientist and the data. Throughout the life cycle of a study, the system will support the scientist in a range activities; it will be used to design the study, to order experiments, to manage the data, and to analyze the results. Before such a system can be implemented and placed on the desk of typical experimental scientists, answers to a range of traditional and non-traditional database management problems must be obtained. In this paper, we give an overview of the activities performed by scientists throughout the course of an experimental study and present the overall architecture of the EMS under development. We then discuss the issues that we have addressed so far and outline some of the solutions that we plan to incorporate in the system.

2 Experiment Life Cycle

To achieve its goals, an EMS will use conceptual schemas for various activities that are important throughout the course of an experimental study. From discussions with scientists from different disciplines, we have concluded that these activities are common to most experimental studies. We use the term *experiment life-cycle* to denote the entire set of these activities together with the way scientists iterate over them during such a study. A pictorial abstraction of the experiment life-cycle is shown in Figure 1. It essentially consists of multiple loops traversed by the researcher multiple times in the course of a study. In the figure, the following stages can be identified. **Experiment Design:** The structure of each experiment is defined determining the input and the output of the experiments. **Data Collection:** Experiments are actually conducted. The researcher specifies the experiment

[*] Partially supported by the National Science Foundation under Grants IRI-9113736 and IRI-9157368 (PYI Award) and by grants from DEC, IBM, HP, and AT&T.

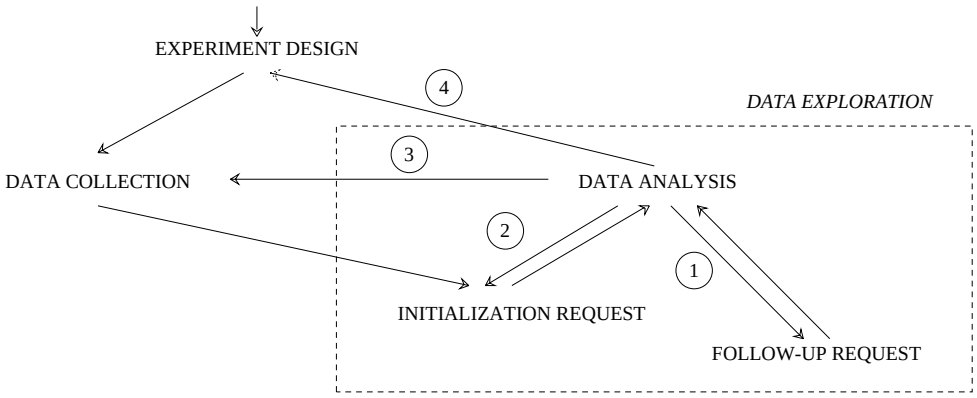


Figure 1: Life cycle of an experimental study.

set-up and the precise values of all the input parameters to the experiment, and the relevant output data is then collected. **Data Exploration:** The researcher studies the collected data to draw conclusions about the subject of the experiment. As shown in Figure 1, there are three types of actions that the scientist may perform on the data, which are described separately. *Initialization requests:* Whenever scientists start to explore a new vain of thought in an experimental study, their first request on the collected data must reference all properties of the phenomenon or system under study that are expected to remain unchanged throughout the exploration of the new idea. Thus, such a request needs to deal with a large portion of the experiment schema. *Data Analysis:* After receiving the requested data, scientists analyze it based on domain-specific knowledge that is relevant to the studied phenomenon. *Follow-up Requests:* Based on the results of the analysis of some obtained data, quite often scientists pose new requests that are very similar to the previous ones, having the answers of the latter as a reference point. This is due to the predominantly exploratory nature of experimental science. Follow-up requests represent the most common form of interaction during a study.

3 An EMS Architecture

A fundamental premise of our effort to develop a desktop EMS has been to provide the typical scientist with a single interface for all stages of the experiment life-cycle. Figure 2 presents the architecture of the EMS under development at the Univ. of Wisconsin, which is based on this premise. On the front end, the user interacts with the system via intuitive language and graphical interfaces (*User Interfaces*). At the heart of the system lies a database system (*Core DBMS*), which provides the traditional query and data storage services. On the back end, the EMS is coupled via *Data Translators* to a variety of data sources (*experimentation environments* where experiments can be run, other EMSs, and DBMSs). Finally, the EMS has an active component, which coordinates the interaction between the system and the external data sources (*Experimentation Manager*).

In the following sections we outline the main properties of these components. Before we do so, we would like to point out that the *User Interfaces* component is based on a 'schema-centric' approach. Since the most important piece of information that is necessary throughout the life-cycle of the experiment is the conceptual schema of the scientific data, we use it as the common foundation for all interactions between the scientist and the EMS. Whereas in a conventional DBMS, the schema captures the structure of the data in a database, in an EMS environment, the schema also captures the structure of the experiment itself. This is a side-effect of the effort to describe the structure of the data: in order to organize the data in a meaningful way, the design of the experiment is essentially represented as well. Therefore, the schema of an experiment is called to play two new major roles: (1) To serve as a formal document describing the experiment; (2) To serve as the template for specifying data and experiments. Conceptual schemas undertake these new roles not only within the EMS, but also in interactions between collaborating scientists.

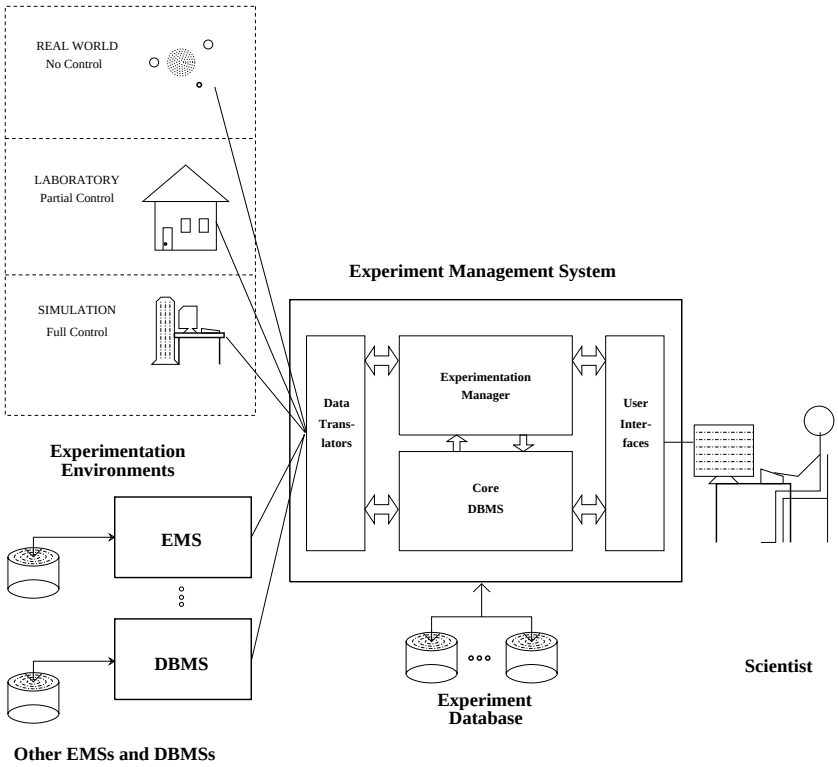


Figure 2: Architecture of an Experiment Management System.

While in a traditional DBMS environment, schemas are manipulated by professional database administrators, in a desktop EMS environment, the scientists themselves are the ones who define and modify schemas. Also, by the nature of scientific studies, ad-hoc queries are the main form of interaction with the system. This requires that scientists can easily obtain the necessary details of schemas. Given the complexity and size of typical schemas of experimental studies, the above implies that the EMS itself should be constantly prompting scientists with the appropriate schemas.

4 The Core DBMS

The Core DBMS of the EMS under development is based on object-oriented (OO) technology. Due to the special needs of many experimental sciences, we have developed our own OO data model, Moose (Modeling Objects Of Scientific Experiments) and query language, Fox (Finding Objects of eXperiments). Detailed descriptions of these may be found elsewhere [1, 2, 9].

The development of Moose and Fox was based on several characteristics of the data found in scientific experiments and of the way scientists are expected to interact with an EMS. The main features of Moose are: (1) Treatment of collections (e.g., sets) as first-class objects, which may be associated with other information possibly independent of the objects in the collection. (2) Distinction between the structural components of a complex object (*part-of* relationships) from any other objects with which it may be associated. (3) Explicit representation of collections indexed by some other, arbitrary, collection (i.e., generalized forms of arrays). (4) Support for *virtual* (i.e., *derived*) relationships (resp. classes), whose values (resp. membership) is defined through rules that are based on the Fox query language. In addition, any Moose schema has a straightforward graph representation, where classes are represented as nodes and relationships as edges.

The main features of Fox are: (1) Access to individual elements of collections indexed by arbitrary sets. (2) Query path expressions traversing arbitrary types of relationships in arbitrary ways. (3) Periodic data, e.g., time

series or spatial domains, concisely described and virtually defined using the virtual relationship mechanism of Moose.

Efficient processing of Fox queries requires extensive index support on the part of the DBMS due to the expected complexity of Moose schemas for experiments. Feature (2) of Fox above renders all existing path indexing schemes inapplicable. We have designed a novel indexing scheme, *generalized multi-level access paths*, that removes all such restrictions [8]. In addition, it allows partial indexing (e.g., indexing intentionally-specified collections of objects) and indexing that associates two groups of objects instead of the traditional pair of individual objects.

5 The Graphical User Interface

The design of the User Interfaces has been shaped by two goals: to provide an integrated tool to be used in all stages of the experiment life cycle, and to allow scientists to use the system in a manner that is natural to them [3]. For schema (i.e., experiment) design, the interface provides a *schema editor*, which uses the services of a *graph editor* due to the graph representation of Moose schemas. This component is already implemented and has been used in the context of real experiments in soil sciences [7]. Work is underway on examining if the graph representation is the right metaphor for Moose schemas at all times, what other alternatives exist, and how they can be supported simultaneously.

For querying (and possibly data display), the interface will use the graphical representation of a schema as a template. Because of the large size of experiment schemas, path expressions in queries will tend to be very long. We are currently working on developing techniques to allow the specification of incomplete paths (e.g., specifying only the two end-points) that will be completed internally by the system based on the schema structure and the semantics of the relationships involved in the candidate complete paths. In addition, based on the exploration state of the experiment life-cycle, follow-up queries are expected to be most common. Our efforts focus on allowing scientists to use the results of previous requests as the basis for future ones.

6 The Experimentation Manager

An important feature of an EMS is that it will be capable of hiding the distinction between the data collection and data exploration stages of the experiment life-cycle. The scientist will be given the freedom to request data without any knowledge of whether it has already been measured and recorded or not. The EMS will decide whether to simply retrieve the data from its database, or initiate some action outside of the system. The Experimentation Manager is the module responsible for making this decision. In the case where experiments need to be invoked, the Experimentation Manager must identify the appropriate experimentation environment, collect all the information necessary to run that experiment, and feed it to the translator. Much of that information will not be part of the scientist's request. We have been investigating techniques that can be used to infer the necessary information by using the virtual relationships mechanism of Moose.

7 The Translators

By the nature of experimental studies, an EMS should provide a cohesive interface to a range of experimentation environments, which may have been independently developed, and should be able to communicate with other EMSs and DBMSs that manage data of interest already collected as part of other studies, so that duplication of effort is avoided (Figure 2). Since these data sources will most likely not use Moose as their data model, Data Translators should be incorporated to the EMS to translate between the various schemas.

We have focused on the problem of translating schemas and their instances from different data models into Moose. In addition, for the case where multiple experimentation environments are generating data for the same experiment, special emphasis is given on integrating multiple schemas. We have developed a formal correctness criterion (based on information capacity of schemas) for schema translation and integration, which is motivated

by the practical requirements of the schema translation task [6, 5]. We are currently using this work in developing prototype Data Translators.

8 Status

Our work on the EMS has benefited immensely from our collaboration with scientists from soil sciences and molecular biology. The schema editor that has been developed is already being used to capture experiments in these disciplines. Part of the Core DBMS has been implemented, supporting a portion of the Fox language. We expect to have a first prototype of the EMS by the summer of 1994.

References

[1] Y. Ioannidis and M. Livny. MOOSE: Modeling Objects in a Simulation Environment. In G. X. Ritter, editor, *Information Processing 89*, pages 821–826. North Holland, August 1989.

[2] Y. Ioannidis and M. Livny. Conceptual Schemas: Multi-Faceted Tools for Desktop Scientific Experiment Management. *Journal of Intelligent and Cooperative Information Systems*, 1(3), December 1992.

[3] Y. Ioannidis, M. Livny, and E. Haber. Graphical User Interfaces for the Management of Scientific Experiments and Data. *ACM-Sigmod Record*, 20(1):47–53, March 1992.

[4] M. Livny. DELAB - A Simulation Laboratory. In *Proc. of the 1987 Winter Simulation Conference*, Atlanta, GA, December 1987.

[5] R. Miller, Y. Ioannidis, and R. Ramakrishnan. The Use of Information Capacity in Schema Integration and Translation. Submitted for publication.

[6] R. Miller, Y. Ioannidis, and R. Ramakrishnan. Understanding Schemas. In *International Workshop on Research Issues in Data Engineering: Interoperability in Multidatabase Systems*, Vienna, Austria, April 1993.

[7] L. M. Murdock. Developing an Object-Oriented Experiment Data Management System for the Cupid Plant-Environment Model. Master's thesis, University of Wisconsin, Madison, June 1992.

[8] O. Tsatalos and Y. Ioannidis. A Unifying Scheme for Multi-Level Access Paths and Materialized Views, February 1993. Submitted for publication.

[9] J. Wiener and Y. Ioannidis. A Moose and a Fox Can Aid Scientists with Data Management Problems, 1993. Submitted for publication.

The SEQUOIA 2000 Project *

Michael Stonebraker
EECS Dept., University of California at Berkeley
Berkeley, California

1 Introduction

The purpose of the SEQUOIA 2000 project is to build a better computing environment for global change researchers, hereinafter referred to as SEQUOIA 2000 "clients." Global change researchers investigate issues of global warming, the Earth's radiation balance, the oceans' role in climate, ozone depletion and its effect on ocean productivity, snow hydrology and hydrochemistry, environmental toxification, species extinction, vegetation distribution, etc., and are members of Earth science departments at universities and national laboratories. A cooperative project among five campuses of the University of California, government agencies, and industry, SEQUOIA 2000 is Digital Equipment Corporation's flagship research project for the 1990s, succeeding Project Athena. It is an example of the close relationship that must exist between technology and applications to foster the computing environment of the future [6].

There are four categories of investigators participating in SEQUOIA 2000: (a) Computer science researchers are affiliated with the Computer Science Division at UC Berkeley, the Computer Science Department at UC San Diego, the School of Library and Information Studies at UC Berkeley, and the San Diego Supercomputer Center (SDSC). Their charge is to build a prototype environment that better serves the needs of the clients. (b) Earth science researchers are affiliated with the Department of Geography at UC Santa Barbara, the Atmospheric Science Department at UCLA, the Climate Research Division at the Scripps Institution of Oceanography, and the Department of Land, Air and Water Resources at UC Davis. Their charge is to explain their needs to the computer science researchers and to use the resulting prototype environment to do better Earth science. (c) Government agencies include the State of California Department of Water Resources (DWR), the Construction Engineering Research Laboratory (CERL) of the U.S. Army Corps of Engineers, the National Aeronautics and Space Administration, and the United States Geological Survey. Their charge is to steer SEQUOIA 2000 research in a direction that is applicable to their problems. (d) Industrial participants include DEC, Epoch, Hewlett-Packard, Hughes, MCI, Metrum Corp., PictureTel Corp., Research Systems Inc., Science Applications International Corp. (SAIC), Siemens, and TRW. Their charge is to use the SEQUOIA 2000 technology and offer guidance and research directions. They are also a source of free or discounted computing equipment.

The purpose of this document is to give an overview of SEQUOIA 2000 project directions. For more detailed information, the reader should consult our strategic plan [16]. Section 2 first motivates the computer science objectives of SEQUOIA 2000. Then, Section 3 continues with a discussion of certain specific projects. Section 4 then explores four themes that cross most elements of the SEQUOIA 2000 architecture. Lastly, Section 5 discusses the longer-term agenda for research and prototyping.

2 SEQUOIA 2000 Motivation

The SEQUOIA 2000 architecture is motivated by four fundamental computer science objectives, namely big fast storage, an all-embracing DBMS, integrated visualization tools, and high-speed networking. We now discuss these points in turn.

*This research was sponsored by Digital Equipment Corporation under Research Grant 1243, DARPA Contract DABT63-92-C-007, NSF Grant RI-91-07455, and ARO Grant DAAL03-91-6-0183.

Our clients are frustrated by current computing environments because they cannot effectively manage, store, and access the massive amounts of data that their research requires. They would like high-performance system software that would effectively support assorted tertiary storage devices. Collectively, our Earth science clients would like to store about 100 terabytes of data now. Many of these are common data sets, used by multiple investigators. Unlike some other applications, much of our clients' I/O activity is random access.

Our clients agree on the merits of moving all their data to a database management system. In this way, the metadata that describe their data sets can be maintained, assisting them with the ability to retrieve needed information. A more important benefit is the sharing of information it will allow, thus enabling intercampus, interdisciplinary research. Because a DBMS will insist on a common schema for shared information, it will allow the researchers to define this schema; then all must use a common notation for shared data. This will improve the current confused state, whereby every data set exists in a different format and must be converted by any researcher who wishes to use it.

Our clients use visualization tools such as AVS, IDL, Khoros, and Explorer. They are frustrated by aspects of these products and are anxious for a next-generation visualization toolkit that allows better manipulation of large data sets, provides better interactive data analysis tools, and fully exploits the capabilities of a distributed, heterogeneous computing environment.

Our clients realize that 100 terabyte storage servers will not be located on their desktops; instead, they are likely to be at the far end of a wide-area network (WAN). Their visualization scenarios often make heavy use of animation, (e.g., "playing" the last 10 years of ozone hole imagery as frames of a movie), which requires ultra-high-speed networking.

3 SEQUOIA 2000 Technical Projects

To address these needs, SEQUOIA 2000 is pursuing six interrelated projects in the areas of massive storage, file systems for a deep store, DBMS, networking, visualization tools and electronic repositories. In this section we briefly discuss these projects.

Our environment is DECstation 5000's for both servers and client machines, moving to Alphas later this year. All clients are connected to FDDI local area networking, and the SEQUOIA 2000 sites are joined by a private T1 (soon to be T3) network. Deep storage is a collection of 6 robotic devices at Berkeley with a current aggregate capacity of 10 Tbytes.

The Storage Project: The focus of the hardware group is on extending RAID ideas [8] to tertiary memory. We are considering striping and redundancy over media in a jukebox, robot arms in a jukebox, whole jukeboxes and even whole systems. Also, we are concerned with the issue of backup and recovery in deep storage. For example, taking a dump of a 10 Tbyte storage system requires several months, and cannot be reasonably contemplated.

The File System Projects: We are building two file systems for deep storage, and plan to run three additional commercial systems. The first file system is **Highlight** [5]. It is an extension of the Log-structured File System (LFS) pioneered for disk devices by Rosenblum and Ousterhout [9]. LFS treats a disk device as a single continuous **log** onto which newly-written disk blocks are appended. Blocks are never overwritten, so a disk device can always be written sequentially. In particular problem areas, this may lead to much higher performance [11]. LFS also has the advantage of rapid recovery from a system crash: potentially damaged blocks in an LFS are easily found, because the last few blocks that were written prior to a crash are always at the end of the log. Conventional file systems require much more laborious checking to ascertain their integrity.

Highlight extends LFS to support tertiary storage by adding a second log-structured file system, plus migration and bookkeeping code that treats the disk LFS as a cache for the tertiary storage one. Highlight should give excellent performance on a workload that is "write-mostly." This should be an excellent match to the SEQUOIA 2000 environment, whose clients want to archive vast amounts of data.

The second file system is **Inversion** [7, 17], which is built on top of the POSTGRES DBMS. Like most

DBMSs, POSTGRES supports binary large objects (blobs), which can contain an arbitrary number of variable-length byte strings. These large objects are stored in a customized storage system directly on a **raw** (i.e., non-file-structure) storage device. It is a straightforward exercise to have the DBMS make these large objects appear to be conventional files. Every read or write is turned by the DBMS front end into a query or update, which is processed directly by the DBMS.

Simulating files on top of DBMS large objects has several advantages. First, DBMS services such as transaction management and security are automatically supported for files. In addition, novel characteristics of POSTGRES, including **time travel** and an extensible type system for all DBMS objects [12], are automatically available for files. Of course, the possible disadvantage of files on top of a DBMS is poor performance, but our experiments show that Inversion performance is exceedingly good when large amounts of data are read and written [7], a characteristic of the SEQUOIA 2000 workload.

The DBMS Project: Some users will simply run application programs against the file system, and will have no use for DBMS technology. Others will store their data in a DBMS. To have any chance of meeting SEQUOIA 2000 client needs, a DBMS must support spatial data structures such as points, lines, polygons, and large multidimensional arrays (e.g., satellite images). Currently these data are not supported by popular general-purpose relational and object-oriented DBMSs [13]. The best fit to SEQUOIA 2000 client needs would be either a special-purpose Geographic Information System (GIS) or a next-generation prototype DBMS. Since we have one such next-generation system within the project, we have elected to focus our DBMS work on this system, POSTGRES [12, 14].

To make POSTGRES suitable for SEQUOIA 2000 use, we require a **schema** for all SEQUOIA 2000 data. This database design process is evolving as a cooperative exercise between various database experts at Berkeley, SDSC, CERL, and SAIC. As we develop the schema, we are loading it with several terabytes of client data; we expect this load process to continue for the duration of the project. As the schema evolves, some of the already-loaded data will need to be reformatted. How to reformat a multi-terabyte database in finite time is an open question that is troubling us.

In addition to schema development, we are tuning POSTGRES to meet the needs of our clients. The interface to POSTGRES arrays is being improved, and a novel **chunking** strategy [10] is being prototyped. The R-tree access method in POSTGRES is also being extended to support the full range of SEQUOIA 2000 spatial objects. Moreover, our clients typically use pattern classification functions in POSTQUEL queries that are very expensive to compute. We have been working on the POSTGRES optimizer to deal intelligently with such queries [4].

To focus the attention of the DBMS research community on the needs of our clients, we have designed the SEQUOIA 2000 Storage benchmark and run it on several software platforms [18]. We are also working on an "end-to-end" benchmark, that would include explicit visualization and networking operations.

The Network Project: The networking project uses the SEQUOIA network as a prototype for our ideas. Specifically, we have avoided running "custom iron" as routers, instead believing that Alphas are fast enough to route T3 packets. In addition, we are trying to lower the number of copies of each byte made by the operating system on the way from storage to the network. Furthermore, we are exploring optimizing multicast protocols, required for successful video teleconferencing by SEQUOIA 2000 participants. Lastly, we are exploring guaranteed delivery protocols that will allow a client to specify an animation sequence which will be delivered to his workstation with a service guarantee. This will allow him to display it smoothly without local buffering. For a description of these algorithms, consult [2].

The Visualization Project: To improve on the limitations of visualization tools such as AVS, and IDL, we have designed **Tioga**, a new boxes-and-arrows programming environment that is "DBMS-centric," i.e., the environment's type system is the same as the DBMS type system. The user interface presents a "flight simulator" paradigm for browsing the output of a boxes-and-arrows network, allowing the user to "navigate" around his data and then zoom in to obtain additional data on items of particular interest. Tioga [15] is a joint project between Berkeley and SDSC, and a prototype "early Tioga" [1] is currently running.

The Repository Project: The final project entails viewing the entire 10 Tbyte storage system as a large elec-

tronic library, containing some text but mostly raw satellite data, "cooked" images, simulation output, computer programs, computational sequences ("recipes"), and polygonal data. This project is focused on providing indexing for such objects, and an ability for clients to browse the repository, without knowing exactly what they are looking for. In addition, a natural language understanding query tool is currently under development.

We are also loading a sizeable collection of text, including all Berkeley Computer Science technical reports, a collection of DWR publications, the Berkeley Cognitive Science technical reports, and the technical reports from the UC Santa Barbara Center for Remote Sensing and Environmental Optics.

4 Common Concerns

Four concerns of SEQUOIA researchers cannot be isolated to a single layer in the architecture; namely guaranteed delivery, abstracts, compression, and integration with other software.

Guaranteed delivery must be an **end-to-end contract**, agreed to by the visualization system (which puts information on the screen), the network (which transports data between machines), the DBMS (which satisfies an underlying query) and the storage system (which retrieves blocks of storage). One approach to this issue is discussed in [15].

Our clients want to **browse** information at low resolution. Then, if something of interest is found, they would like to **zoom** in and increase the resolution, usually to the maximum available in the original data. This ability to change the amount of resolution in an image dynamically has been termed **abstracts** [3], and we are exploring providing them in the visualization package and in the file system.

The SEQUOIA 2000 clients are open to any compression scheme to save storage capacity and network bandwidth, as long as it is lossless. In addition, they are not willing to throw any data away, since its future relevance is unknown. We are exploring the concept of **just in time** decompression. For example, if the storage manager compresses data as they are written and then decompresses them on a read, then the network manager may then recompress the data for transmission over a WAN to a remote site where they will be decompressed again. Obviously, data should be moved in compressed form and only decompressed when necessary. All software modules in the SEQUOIA 2000 architecture must co-operate to decompress just-in-time and compress as-early-as-possible. Like guaranteed delivery, compression is a task where every element must cooperate.

SEQUOIA 2000 researchers will always need access to other commercial and public-domain software packages. It would be a serious mistake for the project to develop every tool the researcher needs, or to add a needed function to our architecture when it can be provided by integration with another package. SEQUOIA 2000 thus needs "grease and glue," so that interface modules to other packages, e.g., S, are easily written.

5 Longer Term Efforts

Phase 1 of the SEQUOIA 2000 project started in July 1991 and will end in June 1994. We hope to continue with a second phase of SEQUOIA 2000 that will start in July 1994, and are embarked on several projects that will come to fruition only in Phase 2. These include an on-the-wire transfer protocol, a hardware storage manager, a distributed file system and a distributed DBMS.

References

[1] J. Chen, et. al., "The SEQUOIA 2000 Object Browser," University of California, Berkeley, SEQUOIA 2000 Technical Report 91/4, December 1991.

[2] D. Ferrari, "Client Requirements for Real-time Communication Services," IEEE Communications Magazine, November 1990.

[3] J. Fine, "Abstracts: A Latency-Hiding Technique for High-Capacity Mass-Storage Systems," University of California, Berkeley, SEQUOIA 2000 Technical Report 92/11, June 1992.

[4] J. Hellerstein and M. Stonebraker, “Predicate Migration: Optimizing Queries with Expensive Predicates,” Proc. 1993 ACM-SIGMOD International Conference on Management of Data, Philadelphia, Pa., May 1993.

[5] J. Kohl, et. al., “Highlight: Using a Log-structured File System for Tertiary Storage Management,” USENIX Association Winter 1993 Conference Proceedings, San Diego, January 1993.

[6] National Research Council, Computer Science and Telecommunications Board, “Computing the Future: A Broader Agenda for Computer Science and Engineering,” National Academy Press, Washington, D.C., 1992.

[7] M. Olson, “The Design and Implementation of the Inversion File System,” USENIX Association Winter 1993 Conference Proceedings, San Diego, CA., January 1993.

[8] D. Patterson, et. al., “RAID: Redundant Arrays of Inexpensive Disks,” Proc. 1988 ACM-SIGMOD International Conference on Management of Data, Chicago, Ill, June 1988.

[9] M. Rosenblum and J. Ousterhout, “The Design and Implementation of a Log-structured File System,” ACM Transactions on Computer Systems, February 1992.

[10] S. Sarawagi, “Improving Array Access Through Chunking, Reordering, and Replication,” (in preparation).

[11] M. Seltzer, et. al., “An Implementation of a Log-structured File System for UNIX,” USENIX Association Winter 1993 Conference Proceedings, San Diego, January 1993.

[12] M. Stonebraker, et. al., “The Implementation of POSTGRES,” IEEE Transactions on Knowledge and Data Engineering, March 1990.

[13] M. Stonebraker and J. Dozier, “SEQUOIA 2000: Large Capacity Object Servers to Support Global Change Research,” University of California, Berkeley, SEQUOIA 2000 Technical Report 91/1, July 1991.

[14] M. Stonebraker and G. Kemnitz, “The POSTGRES Next Generation Database Management System,” CACM, October 1991.

[15] M. Stonebraker, “Tioga: Providing Data Management Support for Scientific Visualization Applications,” University of California, Berkeley, SEQUOIA 2000 Technical Report 92/20, December 1992.

[16] M. Stonebraker, et. al., “The SEQUOIA 2000 Architecture and Implementation Plan,” University of California, Berkeley, SEQUOIA 2000 Technical Report 93/5, March 1993.

[17] M. Stonebraker and M. Olson, “Large Object Support in POSTGRES,” Proc. of the 1993 International Conference on Data Engineering, Vienna, Austria, April 1993.

[18] M. Stonebraker, et. al., “The Sequoia 2000 Storage Benchmark,” Proc. 1993 ACM-SIGMOD International Conference on Management of Data, Philadelphia, Pa., May 1993.

An Overview of the Gaea Project *

Nabil I. Hachem *Michael A. Gennert* *Matthew O. Ward*
Computer Science Department
Worcester Polytechnic Institute
Worcester, MA.
{hachem,michaelg,matt}@cs.wpi.edu

1 Introduction

The long-term goal of the Gaea project is to develop an extensible, object-oriented data management and analysis system to be used by researchers in the field of global change [1].

The current goal is to develop a prototype, which can be used by geographers in a user-friendly manner, yet permits integration of heterogeneous and complex data types, and interactive development of sophisticated methods for data analysis, prediction, and display. Focus is on the object manipulation and analysis aspects of the system, and the management of "meta-data," that is, data about the data.

The general abilities of the system will include: 1) deriving information about spatio-temporal objects, 2) maintaining information on the evolution of data objects, 3) providing derivation semantics for data objects, 4) the management of user-specified experiments, 5) providing a visual environment for browsing, querying, and analysis, and 6) providing user-extensible data and operator types.

2 System Architecture of Gaea

Gaea's overall architecture is divided along three levels (Figure 1): 1) the Gaea Kernel, core of the prototype, provides the essential meta-data management capabilities; 2) the Visual Environment provides visual facilities for scientific experiment design, data definition and manipulation, including querying, and semantic browsing; and 3) the Postgres [3] 3rd Generation DBMS serves as the backend.

2.1 Managing Derived Data in the Gaea Kernel

The most important aspect of the Gaea Kernel is the meta-data manager. It provides a framework for capturing and managing scientific data derivation histories [2]. Meta-data are viewed by the system at three semantic levels (Figure 2): 1) the *high level semantics* view which will provide the user with means to design and develop logical views of experiments, 2) the *derivation semantics* view which provides for the management of (scientific) derivations of data, and 3) the *system level semantics* which are essentially the abstract data type (ADT) view of the system.

2.1.1 High Level Semantics, or the Experiment Level

This level records the information that is necessary for understanding a specific experiment. In global change research, it is difficult to agree on carefully designed experiments. The Gaea kernel supports experiments through the experiment manager module of the meta-data manager. This module is capable of manipulating conventional semantic modeling constructs. In addition, we introduce the notion of *concepts*. A **concept** is a representation of a spatio-temporal entity set, extended with an imprecise definition. Concepts are very common in scientific databases.

*This work is supported by the National Science Foundation under Contract IRI-9116988.

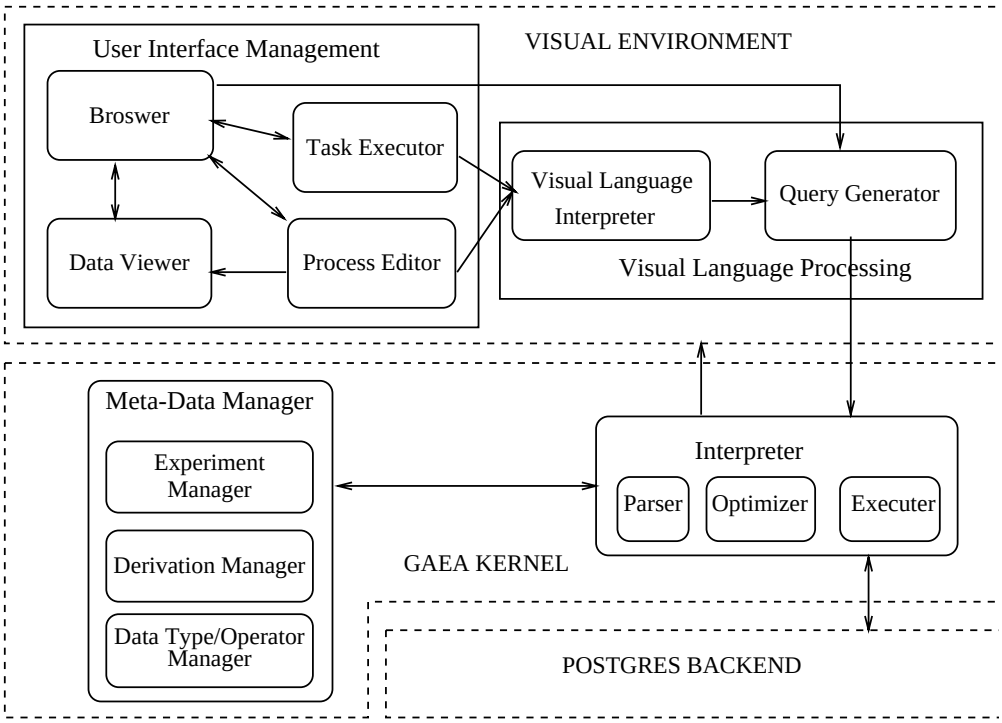


Figure 1: Architecture of the Gaea system prototype.

For example, PERSON is an entity set with a well understood definition. It may be considered as a concept with a well defined and agreed upon meaning. In GIS, a DESERTIC REGION is an entity set whose definition may differ from one user to another. An acceptable definition of a desert must include consideration of the amount of precipitation received, its distribution over a calendar year, the amount of evaporation, the mean temperature during the designated period, and the amount and utilization of the radiation received. Furthermore, every one of those factors may have different metrics.

At this high level of abstraction, we model deserts with a specialization hierarchy (Figure 2). This hierarchy does not capture the relationships between other concepts involved in the definitions of deserts. While general relationships can be provided using well-proven semantic modeling technology, new semantics for data derivation are necessary.

2.1.2 Derivation Semantics Level

The leaves of the concept hierarchy in the high level semantic layer map to a set of non-primitive classes in the derivation semantics layer. For example, in Figure 2, "hot trade-wind desert" maps to the set of (non-primitive) classes {C2, C3, C4, C5}. "Vegetation change" can be characterized using a derivation process (P7) based on principle component analysis (PCA), the outcome of which is class C7. Alternatively P8, based on standardized PCA, can be used and results into class C8. Class C5 could be the result of an interpolation on C2 using P5.

The derivation semantics layer records the derivation relationship among classes of data. Such relationships can also be used for the generation of new classes of data. Typically, when data are not stored in the database, we may generate the needed data with the help of such derivation relationships. The basic constructs used are: 1) A **Process** which captures the description of a scientific procedure used for the generation of new concepts from other concepts. 2) A **Task** which is the instantiation of a process with input data objects. Every task will generate a set of objects for the output class.

Formally, a process defines a mapping between a set of input object classes and an output object class. Es-

sentially, the outcome of a process is a unique class which is a member of a concept. Thus, object classes which do not represent base data are solely defined by their derivation processes. In this way a process captures the semantics of data derivations.

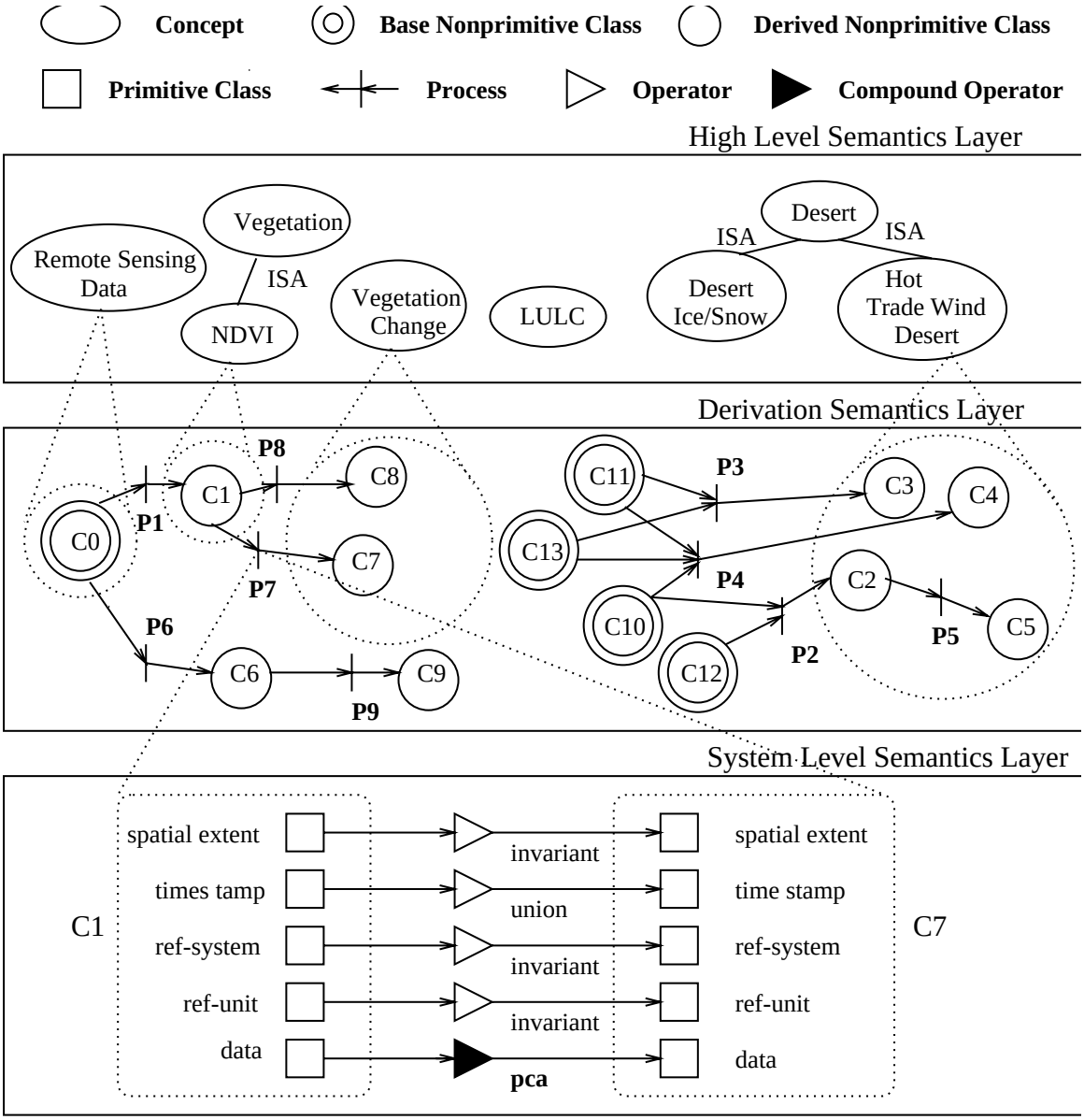


Figure 2: The three semantic layers in Gaea.

2.1.3 System Level or Low Level Semantics

The system level semantics of Gaea is responsible for the management of abstract data types (ADTs). Following the object-oriented paradigm, ADTs in Gaea are primitive classes encapsulated with the methods or functions applicable to them.

The mapping between the derivation semantics layer and the system layer consists of the mapping of a process as a transformation of a set of input classes to an output class using operators that are applied to primitive classes. For example, process P7 transforms class C1 into class C7. The mapping between input and output attributes is shown in the lower portion of Figure 2. It is observed that `pca` is a compound operator. It is composed of a

network of intercommunicating operators, whose structure is discussed in [2]. This network can be considered as a data flow network of functional operators that are applied on primitive classes, such as spatial coordinates, temporal attributes, and raster images.

2.2 Gaea Visual Environment

The Gaea Visual Environment provides the user interface to Gaea [4](Figure 1). The interface generates commands in the visual language, which are converted to queries for interpretation by the Gaea Kernel. The user interface functionality can be decomposed into four interrelated activities.

The **Browser** permits the perusal of the contents of the database and its meta-data. Users specify actions in the browser by interacting with visual representations such as images, graphs, lists and icons to indicate constraints on searches. Thus, to determine for which years there exists rainfall data for Australia, the user would draw a box around the section of the world map which contains Australia, browse all concepts available for Australia, choose rainfall from the result list, and specify the temporal resolution to be yearly. The particular years for which data exists would then be shown on a time line.

The **Data Viewer** is an interactive data visualization toolkit. Data of various types, such as tabular data, image data, and vector data, can be visualized using a set of visualization operators. Results of a visualization, such as a map of vector data, can be used for specifying spatial context in the Browser.

The **Process Editor** permits the user to interactively specify a process via a visual language, whereby data and operators are linked together in a network. Users may choose to edit an existing process, or create a new one from scratch and associate it with a new or existing concept. The system performs consistency checking on the connections between nodes, and the Browser may be invoked to search the database for compatible operator or data classes. Processes can be tested in the Process Editor by specifying input data objects. Results can be visualized by the Data Viewer.

The **Task Executor** allows the user to associate specific data objects of specific concepts to a process or a series of processes and execute the result. The Browser may also be used to locate and specify the desired objects. A fully specified task may be saved as an experiment. Meta-data, extracted from the nodes and supplemented by the user, may be stored with it.

3 Summary

In the first phase of this project, we have synthesized the needs of the global change research community in terms of the management of data, operators, and experiments. This led to a preliminary design for the Gaea system architecture, including the spatio-temporal data and operator models, data language, and the visual environment. By the end of the summer of 1993 we will have completed the implementation of a prototype system with a significant percentage of the desired functionality, including the visual browser, simplified object editors, translator/interpreter, data, operator, and derivation managers, and the interface to the Postgres backend. The prototype will be tested with a small set of geographical data types and some simple analysis and display operators used in global change studies.

References

[1] N.I. Hachem, M.A. Gennert and M.O. Ward, "A DBMS Architecture for Global Change Research," Proceedings of ISY Conference on Earth and Space Science, Pasadena, CA, February 1992.

[2] N.I. Hachem, K. Qiu, M.A. Gennert, and M.O. Ward, "Managing Derived Data in the Gaea Scientific DBMS," WPI-CS-TR 92-08, December 1992.

[3] M. Stonebraker, L.A. Rowe, and M. Hirohima, "The Implementation of Postgres," IEEE Trans. Knowledge and Data Eng. **2**-1, pp. 125–142, 1990.

[4] Y. Zhang, M.O. Ward, N.I. Hachem, and M.A. Gennert, "A Visual Programming Environment for Supporting Scientific Data Analysis," WPI-CS-TR 93-01, March 1993.

Database and Modeling Systems for the Earth Sciences *

Terence R. Smith, Jianwen Su, Divyakant Agrawal, Amr El Abbadi
Department of Computer Science
UC Santa Barbara
Santa Barbara, CA 93106 †

1 Problem Statement and Goals

We are involved in a cooperative study with a group of EOS investigators at the University of Washington whose long-term goals include the development and application of spatially-distributed models of water, sediment and solute transport in the Amazon basin. From a detailed examination of the computational activities and "requirements" of these investigators, we have found that they lack adequate computational support for the *iterative* construction and testing of their models. In particular, they are overwhelmed by the large size and processing requirements of these models. This problem is further compounded by the requirement to develop the models using a large and heterogeneous collection of datasets and the need to couple several complex models. They also lack genuine *database support* for managing their growing collection of datasets, which contain information in the form of satellite images and a large collection of digital datasets for elevation, vegetation, geology, soils, meteorology and hydrology. In large part, these data-handling difficulties result from the use of existing file systems as a repository for the datasets. The varied contents, formats, lineage, and large size of datasets result in an unmanageable collection of files scattered over a network. The absence of database modeling and management further complicates the task of maintaining these datasets, which undergo continued growth and evolution. In addition, efficient access to the contents of these datasets is severely restricted in the current environment. Additional difficulties arise from the fact that while modeling and database activities are closely related from a scientific point of view, they are artificially separated by inadequate computational support. Furthermore, there is no computational support for coordinating the modeling and database activities of researchers involved in joint projects.

The overall goal of our scientific database research project is to design and develop computational support that will permit these and other investigators to achieve their scientific goals more efficiently. Since our investigations of several earth science research projects have indicated that *many scientific teams focus up to 50% of their attention on computational issues that are irrelevant to their scientific research,* we believe that significant increases in efficiency are possible for applications in which computation is a major activity. It is critical, however, to provide support that will be adopted and used by scientists. Hence a first step in our research has been to understand the nature of the earth science investigations and the computational issues involved in such investigations. A second step involves designing and investigating *modeling and database systems* (MDBS) that provide explicit, high-level support for: (1) iterative model development; (2) database construction, maintenance and access; (3) multi-investigator research projects.

*This research was supported under NSF IRI91-17094.
†We gratefully acknowledge the hard work of our graduate students Gustavo Alonso, Yongmao Chen, Keith Park and Amitabh Saran.

2 General Approach

Our general approach to achieving these goals involves: (1) close, collaborative research with the scientists of a specific and representative project, in this case the Amazon project; (2) the design, investigation and prototyping of an MDBS whose functionality is based on an *appropriate model of scientific activity in which modeling and database activities are closely linked*; (3) a top-down design for an MDBS that is based on a specification for a high-level *modeling and database language* (MDBL). Such a language should be capable of expressing, in simple and natural terms, the greater proportion of the scientists' requirements with respect to modeling and database activities.

We believe it essential to base the design of an MDBS on an appropriate model of the goals and activities of scientists. We may view scientific investigators as organizing much of their knowledge in terms of large numbers of domains of entities and transformations between such domains. It is convenient to differentiate between *conceptual domains* (C-domains), which relate to *abstract views* of entities and transformations, and corresponding *representational domains* (R-domains), which relate to symbolic representations of the entities and transformations. For example, corresponding to the notion of a C-domain of *Polygons* and the associated transformations that provide semantics to the concept of polygons, there are a variety of R-domains whose elements may be interpreted as concrete representations of polygons and associated transformations, including representations based on sequences of *Points*, sequences of *Line-segments* and sets of *Half-planes*. Clearly, the discovery and investigation of symbolic representations for many entities and transformations and the evolutionary organization of this knowledge into a structured set of domains is a central theme underlying scientific modeling activities, since useful symbolic models of phenomena may be viewed as expressions involving representations of a set of domain elements and transformations.

We may therefore model scientific investigations as activities in which scientists (1) construct, evaluate and employ a large set of R-domains for representing conceptual entities; (2) construct, evaluate and employ representations of many transformations between these R-domains; (3) construct instances of R-domain elements; (4) apply sequences of specific transformations to specific sets of R-domain elements. In particular, we view scientists as employing *lattices* of domains, in which relatively "high-level" C- and R-domains are inductively defined in terms of relatively "low-level" C- and R-domains, as a basis for modeling complex phenomena, and we view a *model* as consisting of a set of domains, a set of associated transformations, a set of domain instances and sequences of applications of transformations to domain instances. For example, the modeling activities of investigators whom we have observed involve many low-level R-domains, including R-domains containing representations of *Polygons* and *Areas-of-Polygons* (which may be interpreted as measures of polygon area), and many "high-level" R-domains, including R-domains containing representations of *Digital Elevation Models* (*DEMs*) and *Drainage Basins* (*DBs*). Associated with such domains are transformations that include, for example, **area**: *Polygons* $\Rightarrow$ *Areas-of-Polygons* and **demtodb**: *DEMs* $\Rightarrow$ *DBs*. During the process of modeling some phenomenon, scientists may create, for example, instances of *Polygons*, *DEM*s and *DB*s (which reside in some database) and apply transformation to these instances, such as **area** to instances of *Polygons* or **demtodb** to instances of *DEMs*.

We are designing an MDBS that adheres to this view of scientific activity and, in particular, that supports the four general classes of activity listed above. Central to the MDBS is a high-level, largely declarative modeling and database language (MDBL). In relation to the iterative development of scientific models of phenomena and the associated activities of database creation and access, MDBL may be employed by investigators to construct and apply an extensible and potentially *very* large collection of R-domains of elements and associated transformations. Such a "lattice" of domains may be viewed as representing a relatively "complete" set of concepts that a scientist needs for modeling some set of phenomena, from "low-level" datasets to "high-level" models. MDBL supports the declarative specification and efficient construction of transformations associated with the domains; the creation of domain instances; and the application of specific domain transformations to instances, or sets of instances, of domain elements. MDBL also supports the representation of database activities in which scientists may view any dataset as an element of some domain, rather than as a collection of files that may be distributed

in the current computational environment, and access datasets by content.

The database support underlying MDBL incorporates the following general features: (1) uniform structuring and organization of the data that is independent of the physical implementation of the system and the distributed nature of the database (in particular, users see data in terms of named "virtual" datasets rather than in terms of files); (2) simple access to datasets that avoids problems related to I/O (for example, access to portions of datasets may be made in terms of domain names and declaratively expressed constraints on the values of the elements sought); (3) construction of, and access to, transformations on the domains; (4) support for concepts such as *project* and *database view*, including support for concurrent access to datasets within a project, that are independent of the system implementation; and (5) automatic updating of the database.

3 The MDBS Research Project

3.1 Representational Domains

The data model for MDBS is formulated in terms of R-domains. In MDBL, we represent any C-domain in terms of two sets of R-domains: first, in terms of a single *abstract* R-domain that corresponds to the C-domain and its associated transformations; second, as a *set of concrete, equivalent R-domains*, each of which implements the abstract R-domain and its transformations in terms of a specific, but distinct, representation. In computational terms, this permits us to define new concrete R-domains independently of the representation of the domain elements employed in its definition and facilitates the construction and use of an inheritance structure over the set of domains. We allow several equivalent concrete R-domains to represent a single abstract R-domain (as in the case of *Polygons* mentioned above), since it is frequently advantageous for scientific investigators to employ different representations of a C-domain. The defining characteristics of any concrete R-domain include: (1) the name of the domain; (2) the structure (or type, or representation) of the domain elements; (3) constraints on the values of domain elements; (4) sets of transformations on the domain elements that provide, in part, the semantics of the domains; (5) sets of transformations relating both to equivalence classes of elements within a concrete R-domain and to homomorphisms between equivalent R-domains.

An important mechanism for the inductive construction of new R-domains involves the application of constructors (including *set, tuple, sequence, array*) to elements from previously defined domains, which may also have constraints placed on their values. The placing of such constraints induces an *inheritance structure* on the R-domains with respect to both the representation of domain elements and the transformations applicable to elements. We have focussed attention on defining domains of elements that possess non-trivial *spatial* or *spatio-temporal projections* and particularly on domains of pointsets and domains of mappings from points and pointsets into other domains. For such domains, the concept of *finite representability* is of central importance, permitting the definition and application of domains having elements of infinite extent. We have a complete theory of such domains. Examples of typical R-domains are $Bools$, *Computable_Reals* (primitive domains); *Points, Finite_lattice_pointsets, Polygonsets* (purely spatial domains); *Elevations, Rainfalls* (non-spatial domains); *Digital_elevation_models, Drainage_basins* (geographic domains); *Hydrographs, Rainfall_sequences* (temporal domains).

A special subset of abstract and concrete R-domains correspond to the recently-established Federal *Spatial Data Transfer Standard* (SDTS). In this case, specific instances of concrete R-domains correspond to data structures for the electronic exchange of datasets.

3.2 The Functionality and Nature of MDBL

MDBL is designed to allow investigators to express easily and naturally most of the operations that are employed in the iterative development of their models, while hiding computational issues that are irrelevant to the scientific goals of the investigators. Organizing MDBL in terms of the core concept of domains of elements and associated

transformations is intended to make such a goal achievable. In MDBL, both abstract and concrete R-domains as well as the elements of domains and the transformations defined on domains, are treated as *first class entities*. With respect to such entities, one may express in a simple and declarative manner in MDBL: (1) their creation, storage, modification, and deletion; (2) queries concerning their existence, characteristics, and properties satisfying simple constraints (for domains and transformations) and complex constraints (for domain elements); (3) the retrieval and restructuring (reformatting) of domain elements; (4) the retrieval of transformations and their applications to instances of elements, including "datasets". Since MDBL is designed to permit scientists to express most of their computational activities in a simple and natural syntax, the language is not computationally complete. Hence we permit easy access to other languages from MDBL.

The creation, storage, modification and deletion of domains is accomplished by a subset of MDBL which may be viewed as a database schema definition language. The creation, storage, modification and deletion of transformations is accomplished through another subset of MDBL which is designed to take advantage of any hierarchical structure inherent in the lattice of domains, so that transformations on new domains may be efficiently expressed in terms of transformations on previously defined domains with the use of an algebra defined over the transformations. To exemplify the simplicity of MDBL, we note that expressions to access particular instances from the domains of *DEMs* and *DBs* may take the form: *access* e_1 *from DEMs where* $C_1, ..., C_n$ and *access* e_2 *from DBs where* $C_{n+1}, ..., C_N$, where the constraints $C_1, ..., C_N$ on the values of the elements e_1, e_2 are expressed in terms of other domain elements and any transformations of those elements that are employed in defining the element. The application of the transformation **demtodb** to a domain element may be expressed as: *apply* **demtodb** *to input* e_i *to obtain output* e_o.

Since scientific investigations are increasingly performed by groups of scientists, it is important that database systems support the notion of *project*, which may be viewed as a collection of interrelated operations carried out by a group of investigators on a subset of domains. At a coarse level, a project may be viewed as transforming elements from input domains to output domains. At a fine level, the organization of computation may be complicated, since non-trivial computing tasks, such as data access, may be divided among researchers. Moreover, individuals in the same project may want to have different *views* of the same (project) database schema. Conventional techniques for supporting database views are useful in this context. With respect to projects and views, such functionality includes the creation, modification or deletion of projects as well as the creation, modification or deletion of views.

3.3 Database Support

In scientific MDBS, it is important to provide storage mechanisms for domains that may contain large and complex elements. In particular, such mechanisms should not only provide efficient storage and retrieval of elements in these domains but must also provide means to store, retrieve, and modify transformations associated with the domains. Another degree of complexity arises as a result of the physical distribution of these domains. For example, the domain corresponding to rainfall-data in the Amazon basin may be structured in terms of a logically centralized abstract R-domain but the actual physical representation of this R-domain may be distributed. We are developing techniques to provide a logically centralized view of a domain that may be distributed spatially. In particular, as a first step towards prototyping MDBS, we will use existing file-systems as an underlying store for the domains in MDBS. We are using the Prospero file-system (from University of Washington) to provide an integrated view of an R-domain that may have been represented in terms of multiple files distributed over a network. Although file-systems impose a maximum size for a single file, the size of a domain is not restricted since it is represented in terms of a (theoretically) unlimited number of files. Furthermore, the Prospero system allows files corresponding to a domain to be organized in a variety of ways that include temporal, spatial, spatio-temporal and content-based indexing.

In large, scientific investigations, the sharing of information among scientists involved in various projects appears to be important. Concurrent sharing of data when used primarily in read-only mode does not give rise to

consistency problems. However, since we are permitting regular operations on the transformations associated with the domains in MDBS, concurrent sharing of domains and transformations may give rise to consistency problems. For example, a project may produce a "derived" domain as a result of applying a transformation to some set of domains and the new domain may be used in another project. If, however, the former group modifies their transformation to produce a new "derived" domain, it will make useless the results seen by the latter. Traditional database systems control concurrent executions by requiring that such interactions to the databases be atomic. However, serializability is very restrictive in scientific databases since interactions in the database may have very long durations. Maintaining atomicity of long interactions imposes long delays on users. We have instead developed a model based on the notion of "relative atomicity" of interactions. In this model, the users (in our case the scientists) explicitly specify the ways in which their interactions to the database should be interleaved with the interactions of others. For example, there may be no atomicity restrictions on interactions that are initiated within a project whereas these interactions must be atomic with respect to interactions of other projects. We believe that such a model will be of significant importance in capturing the notion of "collaboration" that is common in scientific investigations.

3.4 Experimental Studies and Applications

The prototyping and experimental components of our research are in part intended to capture the nature of computation in scientific research and in part to investigate and test the conceptual MDBS that we are designing in order to support such research. A first component of our experimental research involves the development of MDBL and its appropriateness for scientific database and modeling applications. In order to capture the computational "requirements" of the scientific investigators on the Amazon project, we have expressed complete sequences of activities relating to model development and database access in MDBL, and in other more complete languages where necessary. This activity has involved the iterative design and representation in MDBL of an appropriate lattice of abstract and concrete R-domains as well as expressing, in MDBL, several examples of specific scientific investigations. For situations in which we require a more complete language than MDBL, we have chosen to use the deductive database language CORAL, which combines the general logic programming paradigm and database manipulation, and other programming languages. We have designed a large number of domains and associated transformations and we have implemented an important subset of these domains. We have written several application examples in MDBL and CORAL. These examples had been previously expressed in FORTRAN by the applications scientists. One example involves the construction of a drainage network from a digital elevation model, and the application of a spatially-distributed, time-sliced hydrological model in order to compute the discharge of runoff from the network. Expressing spatial pointsets in the relational representations of CORAL currently presents some problems of efficiency since, for example, the *topological neighbors* of any point cannot be easily accessed. A second component of our experimental research relates to providing a network transparent file system layer to support this database design. For this purpose, we are using the PROSPERO system to provide a *virtual* file system over the network.

QBISM: A Prototype 3-D Medical Image Database System

Manish Arya, William Cody, Christos Faloutsos[*]
IBM Almaden Research Center
San Jose, California

Joel Richardson
The Jackson Laboratory
Bar Harbor, Maine

Arthur Toga
UCLA
Medical School

1 Introduction

The goal of the QBISM[1] project is to study extensions of database technology that enable efficient, interactive exploration of numerous large spatial data sets from within a visualization environment. In our work we are currently focussing on the logical and physical database design issues to handle 3-dimensional spatial data sets created from 2-dimensional medical images.

Our specific application is the Functional Brain Mapping project at the Laboratory of Neuro Imaging, UCLA School of Medicine [12]. The goal of the brain mapping research is to discover spatial correlations between activity in the brain and functional behavior, e.g. speaking or arm movement. Such activity in the brain is frequently characterized by localized, non-uniform intensity distributions involving sections or layers of brain structures, rather than uniform distributions across complete structures. Discovering the precise locations of brain activity, correlating it with anatomy, and constructing functional brain atlases is the goal of an ongoing major medical research initiative. Ultimately, this understanding has clinical applications in diagnosis and treatment planning, as well as scientific and educational value.

To support the requirements of exploratory queries on multiple 3-D images, we have built an experimental prototype using the extensibility features of the Starburst DBMS developed at IBM's Almaden Research Center. The prototype has a client/server architecture, with IBM's Data Explorer/6000 visualization package serving as the foundation for an interactive, query front-end.

Section 2 describes the particular medical research problem we are studying and its query and data characteristics. Section 3 presents the status of our prototype. Section 4 summarizes the overall project and describes its future directions.

2 The Medical Application

2.1 Problem Definition

As mentioned, the purpose of QBISM is to support the data manipulation and visualization needs of the brain mapping project. The system we envision must support queries across multiple medical image studies in a very investigative, interactive, and iterative fashion. A study is actually a "billing" term referring to a set of medical images collected for a single purpose on a single patient, such as a 50 slice MRI[2] study or three x-rays of a fractured elbow. Querying and visualizing collections of studies [3] will enable the return of statistical and comparative responses. Such capabilities will extend the power of medical visualization environments which today

[*]On sabbatical from the University of Maryland at College Park. His work was partially supported by SRC and by the National Science Foundation (IRI-8958546), with matching funds from Empress Software Inc. and Thinking Machines Inc.

[1]Query By Interactive, Spatial Multimedia

[2]Magnetic Resonance Imaging. MRI images show soft-tissue structural information.

typically deal with a single study at a time. The following scenario, which is representative of the queries that medical researchers (i.e., those at the UCLA Laboratory of Neuro Imaging) would like to ask, illustrates a sample session with such a system in which each step generates a database query:

- The medical researcher may start by specifying and rendering a set of brain structures, for example those supporting the visual system, from a standard atlas [10].
- After repositioning the scene to a desired viewing angle, structures may be colored with a patient's PET[3] study data to highlight activity along their surfaces (see Figure 1(c)).
- The range of intensities in these structures may be histogram segmented, and then other regions in this PET study may be identified that have matching distributions.
- An arbitrary region in the study may be specified for visual comparison of its intensity pattern with the same or nearby region from a previous PET study.
- Paths for targeting electrodes or radiation beams that focus on an arbitrary region of interest may be calculated or simulated to permit the visualization of anatomical structures spatially intersected.
- An individual PET (or other study) may be statistically compared with data from a comparable subpopulation of the same demographic group to assess abnormality.

2.2 Data Characteristics

The database consists of a large collection of 3D studies and a set of anatomic atlases of the human brain.

Each study is a 3-dimensional scalar field (a 3-dimensional array of scalar values) representing some measured quantity, such as glucose consumption as an indicator of physiological activity, at each point in space. Studies are collected via an assortment of medical imaging modalities used to capture structural (e.g. MRI, CT[4], histology[5]) and functional / physiological (e.g. PET, SPECT[6]) information about the human brain. These studies typically consume about 1 - 30 megabytes, using current spatial resolutions and image depths, and could potentially consume over a hundred megabytes with increasing resolutions and depths. As a reference point, at The University of Virginia, which has a large medical center, the number of such tomographic studies can range from 5,000 to 15,000 per year, depending on modality. The total number of all types of imaging studies at the same medical center is 181,000 per year, resulting in 3 terabytes of data per year. Table 1 contains a breakdown for the tomographic studies. Modern hospitals are beginning to store all this imagery in systems known as PACS (Picture Archival and Communication Systems [13]). Using this data for advanced medical applications further increases the storage requirements due to the need to save derived data. Such data is a result of transformations to align and register the raw data, to create models suitable for volume and surface rendering of the data, and to build database representations that enable exploratory query.

As mentioned above, the database also contains atlases of reference brains, one for each demographic group. Each atlas models the exact shapes and positions of anatomical structures in the corresponding reference brain. A study itself does not identify the structure to which each voxel (3-D pixel) belongs, but an atlas can provide this information when overlaid on top of the study. Such use of an atlas is illustrated in the previous scenario by the second step, in which the brain structures of the visual system are used to retrieve parts of a particular patient's PET data.

[3] Positron Emission Tomography. PET images show physiological activity.

[4] Computed Tomography. CT images show hard-tissue structural information (e.g., bones).

[5] Histology images are acquired by physically slicing and photographing tissue, one thin layer at a time.

[6] Single Photon Emission Tomography. SPECT images, like PET images, show physiological activity.

Modality	Studies/Year	Images/Study	Image Size (bits)
CT	14810	30	512x512x12
MRI	5418	50	256x256x12
PET	6134	26	256x256x8

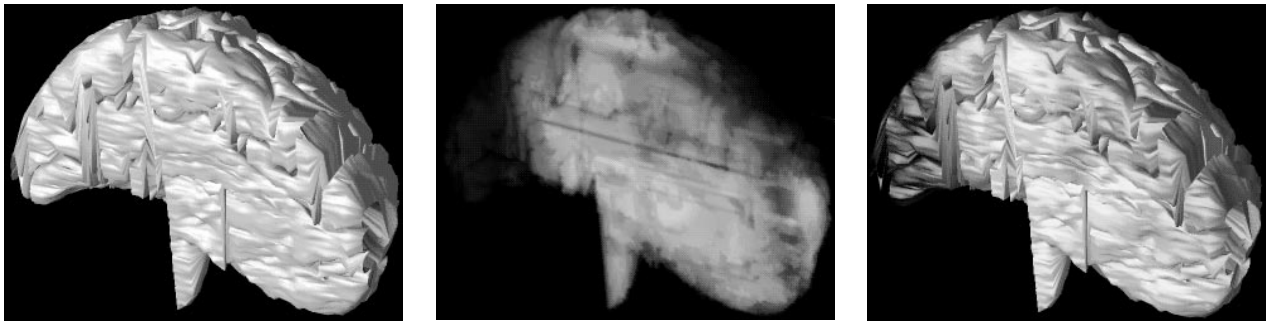
Table 1: Yearly tomographic study statistics for the University of Virginia Medical Center.

Note that an acquired radiological study of a patient is not perfectly aligned with the corresponding atlas. Warping techniques [11] are used to derive affine transformations that allow a study to be registered to one or more appropriate atlases. In QBISM, we store the original study, the warped ones, and the warping transformations. The details of the warping are outside the scope of this paper. However, these automatic or semi-automatic warping algorithms are extremely important for this application. It is precisely this technology that permits anatomic structure-based access to acquired medical images as well as comparisons among studies, even of different patients, as long as they have been warped to the same atlas. Furthermore, it enables the database to grow, and be queryable, without time-consuming manual segmentation of the data.

3 A Prototype Implementation

We built a prototype that runs on IBM RISC System/6000 workstations. It integrates and utilizes the extensibility features of the Starburst relational DBMS [8] and the IBM Data Explorer/6000 (DX) scientific visualization product.

The user specifies a query by choosing a modality, a study, anatomic structures of interest, and intensity ranges of interest. The system then renders the selected data in 3D in one of several ways (see Figure 1). The user can manipulate the result in DX by changing the viewpoint, adding a cutting plane, or generating an isosurface, for example, or refine the query itself to select data from a different patient/study or part of space.



(a) (b) (c)

Figure 1: Sample query results. (a) One brain hemisphere from the atlas. (b) The intensity data from a PET study inside the hemisphere. (c) The same PET data mapped onto the surface of the hemisphere. Note the difference in shading between a and c, which is more prominent in color.

The system stores all large objects (e.g., studies and atlas structures) in Starburst long fields [4] and the associated attributes in relations. New SQL functions we added to Starburst perform the necessary spatial operations, such as “intersection()”. A new processing module we added to DX accepts the user’s query and communicates with Starburst through a network connection to retrieve the spatially-restricted answer. Details on the data types and their representations, the operations, and performance experiments are in [2].

4 Summary and Future Directions

We have discussed the requirements and the initial implementation of QBISM, a prototype system for managing and visualizing 3D medical images. In order to allow convenient querying over multiple studies, we believe that such a system should be built on top of an extensible DBMS engine, appropriately extended to handle spatial data, and combined with a high-quality visualization tool as the user interface. The challenges in the project are to define and implement the database extensions that support medical researchers' ad-hoc queries over populations of studies with interactive response times, despite the large size of even a single study.

Future goals of QBISM include:

- Development of multi-study indexing techniques for anatomic atlases and patient studies to accelerate queries over large patient populations.
- Addition of approximate spatial representations and the associated filter/refinement query processing strategies to further optimize large-scale queries [7].
- Incorporation of data mining and hypothesis testing techniques to support investigative queries. An example of a hypothesis testing query is *"is it true that people with dyslexia show this intensity range in this region of their PET studies?"* An example of a data-mining/rule-discovery query is *"find PET study intensity patterns that are associated with any known neurological condition in any subpopulation"*. Data mining algorithms for relational databases are presented in [1].
- Integration of support for query by image content. We would like to support similarity queries like *"find all the PET studies of 40-year old females with intensities inside the cerebellum similar to Ms. Smith's latest PET study"*, or sub-pattern matching queries, like *"find patients whose MRI studies show a hippocampus with shape similar to that of this patient who has a particular neurological disorder"*. Clearly, we need feature extraction and similarity measures. Research in machine vision has yielded several good features for 2-D images, e.g. the "QBIC" project at IBM ARC [6] proposed and experimented with some color, shape and texture features. Our challenge is to discover appropriate features for 3-D medical images.
- Development of natural, spatial interaction mechanisms to help pose queries with 3-D regions of interest and to help manipulate the results. Effective solutions may require special hardware, such as a 3-D mouse or a "data glove".

Acknowledgments: We would like to thank Walid Aref and Brian Scassellati for helping initiate this work; the Starburst developers for providing advice and help with Starburst; and the UCLA LONI Lab staff for providing and helping interpret the human brain data.

References

[1] R. Agrawal, T. Imielinski, A. Swami, "Mining Association Rules between Sets of Items in Large Databases", ACM SIGMOD, 1993 (to appear).

[2] M. Arya, W. Cody, C. Faloutsos, J. Richardson and A. Toga, "QBISM: Extending a DBMS to Support 3D Medical Images", submitted for publication, 1993.

[3] H. Fuchs, M. Levoy and S. M. Pizer, "Interactive Visualization of 3D Medical Data", IEEE Computer, 22, 8, pp. 46-51, Aug. 1989.

[4] T.J. Lehman, B.G. Lindsay, "The Starburst Long Field Manager", Procedings of the 15th International Conference on Very Large Data Bases, Amsterdam, pp. 375-383, Aug. 1989.

[5] A.D. Narasimhalu and S. Christodoulakis “Multimedia Information Systems: The Unfolding of a Reality”, IEEE Computer, 24, 10, pp. 6-8, Oct. 1991.

[6] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, and P. Yanker, “The QBIC Project: Querying Images By Content Using Color, Texture, and Shape”, SPIE 1993 International Symposium on Electronic Imaging: Science & Technology, Conference 1908, Storage and Retrieval for Image and Video Databases, February 1993.

[7] J. Orenstein “Redundancy in Spatial Databases” Proc. of ACM SIGMOD conf., Portland, Oregon, May 1989.

[8] P. Schwarz, W. Chang, J.C. Freytag, G. Lohman, J. McPherson, C. Mohan, and H. Pirahesh, “Extensibility in the Starburst Database System,” Proc. 1986 Int’l Workshop on Object-Oriented Database Systems, Pacific Grove, September 1986, pp. 85-92.

[9] A. Silberschatz, M. Stonebraker, J. Ullman (eds.), “Database Systems: Achievements and Opportunities”, Communications of the ACM, Vol. 34, No. 10, pp. 110-120, Oct. 1991.

[10] J. Talairach and P. Tournoux, “Co-planar stereotactic atlas of the human brain”, Thieme, Stuttgart, 1988.

[11] A.W. Toga, P. Banerjee and B.A. Payne, “Brain warping and averaging”, Int. Symp. on Cereb. Blood Flow and Metab., Miami, FL 1991.

[12] A.W. Toga, “A digital three-dimensional atlas of structure/function relationships”, J. Chem. Neuroanat., 4(5):313-318, 1991.

[13] A.W.K. Wong, R.K. Taira and H.K. Huang “Digital Archive Center: Implementation for a Radiology Department”, AJR 159, pp. 1101-1105, November 1992.

Integration and Interoperability of a Multimedia Medical Distributed Database System**

Alfonso F. Cardenas‡ *Ricky K. Taira*† *Wesley W. Chu*‡ *Claudine M. Breant*†

‡ Computer Science Department
and
† Department of Radiological Sciences
University of California, Los Angeles

1 INTRODUCTION

Medical practice and research now rely heavily on the use of information stored in computer systems. Due to the many specialized branches of medicine, these database systems have developed independently, their design reflecting the innovativeness of the database implementors, the scope of data required for their operation, and the culture of their department. Each user group chooses its own database management system (DBMS) according to its needs. Example medical databases include: 1) hospital information systems (HIS), 2) radiology information systems (RIS), 3) picture archiving and communication systems (PACS), and 4) various research systems. These independent database systems provide only a first order solution to the management of the large repository of image, text, and scientific information generated by academic medical research centers. Research involving data stored in multiple databases often is hindered by the fact that these databases operate under different operating systems, have different data access and manipulation languages, have different data semantics, and have different communication protocols. Furthermore, researchers investigating similar areas of study often have no clue as to existence of databases which contain information that may accelerate or substantiate their own research. Figure 1 shows various medical data repositories present at our institution.

In addition to providing the researcher with more data, scientific medical databases require better query answering capabilities. Epidemiology and biological modeling studies (e.g., human genome mapping, functional brain mapping) are concerned with correlating image features (radiographic, pathology, etc.), patient symptoms, and diagnostic performance measures with underlying disease states. These correlations will vary with respect to patient profile (sex, race, diet, genetic composition, etc.). Furthermore, these mappings change with time and may evolve with respect to patient maturation processes, disease progression, and therapeutic events. This type of medical research requires the ability to access images by content rather than by artificial keys such as patient hospital identification number. Furthermore, access to a sequence of images demonstrating the evolutionary transformation of medical objects (e.g., organ development, disease progression, etc.) is required to understand the dynamics of the medical object under investigation.

Finally, multimedia medical database systems have advanced data analysis and visualization requirements. Several institutions are developing specialized algorithms for computer-aided diagnosis of digital images (e.g., tumor detection in chest x-rays) as well as new image visualization techniques (e.g., 3D imaging, target tissue enhancement, functional mapping, dual energy reconstruction) to improve medical diagnoses. These image analysis techniques generate other types of data characterizing diagnostic image patterns and/or image features related to a disease state. Some examples include cardio-thoracic ratio, fractal shape parameters for brain contours,

**This work is supported in part by a grant from the National Science Foundation.

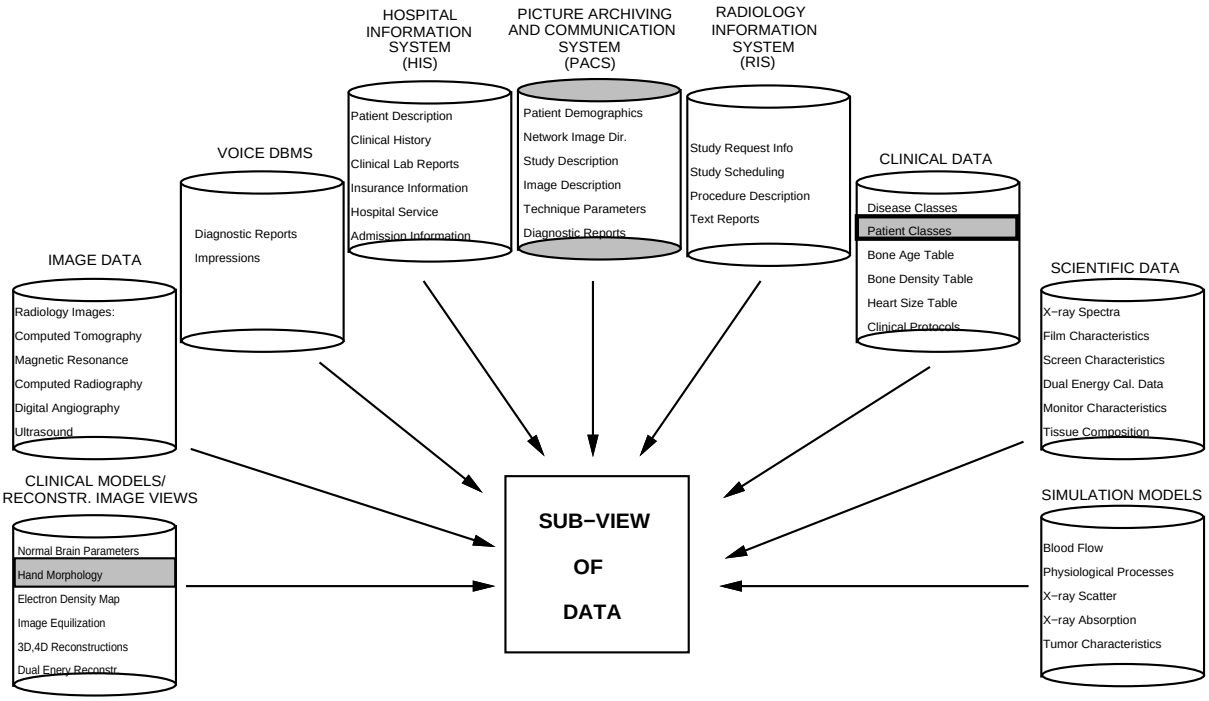


Figure 1: *UCLA Medical Center multimedia database federation. Each of the database systems shown were implemented independently.*

tumor/organ volumetric data, spatial Fourier spectral information, morphological shape parameters for various bones in the hand, segmentation information for various organ/structural boundaries, and critical gray level breakpoints used to optimize tissue visualization. It is desirable to incorporate these multimedia data analysis and visualization techniques as an integral part of the query processing operation.

2 PROJECT GOALS

Our first goal is to integrate the major medical information systems within our Medical Center: HIS, RIS, and PACS. This goal is driven by the development of a national scientific database system for breast cancer research. The research requires patient history, family history, pathology, and medical workup records from the Medical Center's HIS system, diagnostic reports from the RIS system, and mammographic images from the PACS system. Figure 2 shows the network configuration of these systems.

The goal to access images by content and collections of images by evolutionary processes is being addressed by the UCLA Knowledge-based Multimedia Medical Distributed Database System (KMeD) project. KMeD addresses the data modeling of both pictorial and non-pictorial data in both the spatial and temporal domains and the development of a high-level user query interface. We have initially focused on developing a database system for musculo-skeletal research (with future extensions to brain modeling research). Our goals are to develop an intelligent database system from which researchers can specify: a) query predicates involving bone structure, growth, and evolution, b) analysis methods for feature extraction and characterization, c) visualization methods for query results.

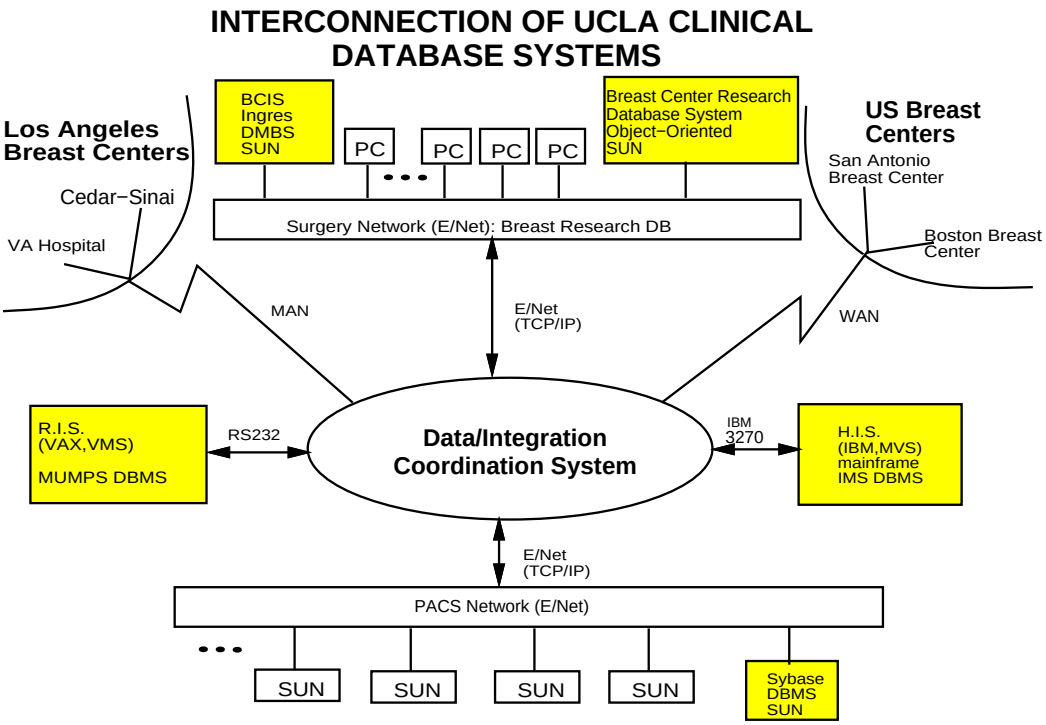


Figure 2: *Network diagram of various databases required for breast cancer research.*

3 DATA INTEGRATION, MODELING, AND QUERYING

The integration of the conventional HIS/RIS/PACS databases required for the breast cancer research project is being performed using the commercially available product Integration Works (Data Integration Solutions Corp.). The system provides high-level CASE tools (collectively called the "Integration Consultant") for designing, modeling, and testing the integration specifications. The CASE tools execute in an object-oriented development environment and include graphical interfaces for global schema design, definition of data integrity constraints, setup of communication protocols, and definitions for source and target machine data translations. The results of the integration consultant are a collection of communication scripts and data translators that are compiled and embedded into the run-time data integration server process (called the "Integration Coordinator"). Applications execute job managers running under an X-Windows environment. Job managers run the automatically generated communication and translation scripts.

The development of improved data models and advanced query processing capabilities for multimedia medical databases is addressed by the KMeD project. Accurate data models are of great interest in medical databases due to the complexity of object features and the complexity of object relationships. The requirement for storing, analyzing, and visualizing image data lead to KMeD's stacked image data model. The need to model the evolution and dynamics of objects in the body lead to KMeD's temporal evolutionary data model. A brief description of the image stack data model and of the temporal evolutionary data model follows.

Image data is modeled using a dynamic stacked logical data structure [Josep88]. Stacks consist of two-dimensional variables or pictures registered to the same gridded coordinate system. Multiple stacks of pictures can provide, for example, additional temporal, spectral, and/or spatial information. Stacks may consist of a hierarchy of other stacks. Furthermore, pictures composed of objects with pictorial as well as non-pictorial attributes and relationships can be represented. Thus, objects at different levels of abstraction can be modeled including time varying image stacks. KMeD users may view an image stack at a high level while still maintaining the

flexibility to manipulate images with pixel-level granularity.

The extension of traditional object-oriented data models into the temporal domain is an important requirement for accurately representing the data stored in medical image databases. This is because structures in the human body are not static and often change their characteristics and/or existence over time. Our temporal evolutionary data model (TEDM) handles complex queries involving the comparison of the lifespan of two objects and/or events, the creation of objects, the fusion of objects, the fission (splitting) of an object, and the gross transformation of object properties [Chu92a]. Formal constructs for the temporal relationships between objects, the specification of temporal constraints, and the inheritance characteristics passed between objects have been developed. TEDM uses evolutionary networks for modeling various object transformation processes and includes inheritance rules between objects that evolve through various stages in the time domain. For example, the human hand goes through nine distinct skeletal developmental stages (some bones may fuse into one bone and others may split into multiple bones). At birth, the hand inherits characteristics of the *HumanHandStageA* class. Over time, the hand inherits properties of the corresponding, successive stages, at the same time losing the particular characteristics of the previous stage. We have developed TEDM models capturing the development phases of the maturing human hand. This model allows us to store and retrieve image collections based on temporal and evolutionary processes.

The high-level domain-independent query language PICQUERY+ has been designed, and a subset of it is being implemented for the KMeD project [Carde93]. Queries are specified using either a tabular user interface or in a more application specific user friendly graphical interface. Both interfaces allow the user to specify high level operations including query definition (including temporal and evolutionary predicates), data analysis methods, and visualization methods for results. PICQUERY+ also provides menu options for visualizing knowledge including various type abstraction hierarchies [Chu92b] and rules as the user defines the query. Using PICQUERY+ we are able to conveniently express, for example, the following high level medical imaging queries:

- *Show brain cases which demonstrate abnormally shaped ventricles.*
- *Find an image of the proximal phalanx of the fifth finger for patient A.J. Smith and obtain the length of the major axis of this bone.*
- *Retrieve objects in image P2 similar in shape to brain objects considered abnormal in image P1.*

4 CURRENT PROJECT STATUS

The first phase of the integration of the RIS system to a general-purpose networked computing environment is complete. Access to RIS records is accomplished using a more friendly graphical X-Windows user interface. No modifications and/or disruptions to the RIS system were required. The integration server process appears as simply another user to the RIS system. Development time using the integration CASE tools was relatively short after user requirements and schema definitions were provided. Integration of the HIS and PACS systems are currently in progress.

We are applying the KMeD concepts discussed herein (TEDM, stacked data model, PICQUERY+) to a musculoskeletal database containing digitized hand radiographs. Custom PICQUERY+ graphical user interface screens are being developed for musculo-skeletal researchers to allow investigators to specify patient demographic profiles, disease pathology constraints, image feature profiles, and evolutionary event constraints. Our preliminary experience suggests that our approach is feasible for retrieving images by content in pictorial databases.

5 REFERENCES

[Carde93] A.F. Cardenas, I.T. Ieong, R.K. Taira, R. Barker, C.M. Breant, "The Knowledge-Based Object-Oriented PICQUERY+ Language", *IEEE Transactions on Knowledge and Data Engineering*, to be published 1993.

[Chu92a] W.W. Chu, I.T. Ieong, R.K. Taira, C.M. Breant, "A Temporal Evolutionary Object-Oriented Data Model and Its Query Language for Medical Image Management", *Proceedings of the International Conference on Very Large Databases (VLDB)*, Vancouver, Canada, 1992.

[Chu92b] W.W. Chu and Q. Chen, "Neighborhood and Associative Query Answering", *J of Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies*, pp. 355–382, Dec. 1992.

[DataI93] Integration Works: Integration Coordinator User's Guide, Data Integration Solutions Corp., 1993.

[Josep88] T. Joseph and A.F. Cardenas, "PICQUERY: A High Level Query Language for Pictorial Database Management", *IEEE Transactions on Software Engineering*, vol. 14, no. 5, pp. 630–638, 1988.

Algebraic Optimization of Computations over Scientific Databases

Richard Wolniewicz
University of Colorado

Goetz Graefe
Portland State University

Abstract

Using the extensible Volcano optimizer generator, we are exploring algebraic query optimization techniques for computations that combine database retrieval with numerical operations on sets, time series, and spectra. We are currently transferring our software to CU's Space Grant College.

1 Introduction

Since many scientific applications manipulate massive amounts of data, database systems are being considered to replace the file systems currently in wide use for scientific applications. In order to counteract the performance penalty of additional software layers (i.e., the database management system), we are investigating the use of traditional database techniques to enhance the performance of computations over scientific databases. Our two focus areas are automatic optimization and parallelization of processing plans that include both numeric and database operations.

The focus of this research is to extend the concept of a database query to include numerical computations over scientific databases by defining an integrated algebra with both traditional database operators for pattern matching and numerical operators that perform scientific calculations. This integration gives us the ability to perform automatic optimization of entire computations.

Since they are among the most common data types in scientific analysis, we started our research by adding time series and spectra to the Volcano extensible query processing system [1,2]. (Intuitively, spectra represent repetitive phenomena in time series as sine-cosine pairs of amplitudes or as amplitude-phase shift pairs for a series of frequencies.) The variety of alternative computational techniques between spectral and time series methods offer an opportunity to examine complex transformations of numerical computations by an algebraic optimizer. This prototype serves as the basis for a more complete system to perform scientific data management and analysis at CU's Space Grant College. The results learned from the examination of time series will be used to extend this research to transformation of computations on other scientific data types, particularly multi-dimensional arrays.

2 Algebraic Query Optimization

In our algebraic model, computations over a scientific database are specified as expressions consisting of logical operators on bulk data types. The bulk types considered in our prototype are sets (relations), time series, and spectra. Each operator in the computation takes one or more instances of a bulk type as its input, and produces a new instance of a bulk type as its output.

Transformation rules translate a portion of an expression into a different but equivalent form. Examples include relational join associativity and numerical filtering in either time- or Fourier-space in scientific databases. Unique problems may have to be considered, such as the equivalence of two numeric expressions with respect to numerical accuracy and stability. We are still working on a suitable definition of equivalence, which we presume will have to be amenable by the user-scientist.

Once applicable transformation rules have been applied to generate an equivalent logical expression, the optimizer must find a set of physical algorithms that can implement or execute the expression. This process is controlled by implementation rules. For instance, interpolation can be implemented by a variety of curve fitting algorithms. In the process of choosing physical algorithms, the optimizer considers anticipated executions costs as well as physical issues such as sortedness of data sequences.

The bulk types explored in this research are sets, time series, and spectra. Sets are viewed as relations in the relational model. Time series are also viewed as sets, but each record is tagged with a time value. Although time series are often treated as if they were implicitly sorted by time, such sorting is not necessary on the logical level. Spectra are treated similarly to time series, as sets of records tagged with a frequency value.

3 Operators

The operators in our current prototype were chosen for their common use in scientific processing and for their illustration of special issues that must be addressed by an optimizer for scientific databases. In addition to relational operators (select, project, join) and a random sampling operator, we are supporting operations on time series and spectra.

For time series, there are three new logical operators. The first is a digital filter operator, which replaces each time step in the time series with a weighted average of its neighbors; the weights are selected to pass desired frequencies in the data and suppress others [3]. Computations based on nearby records are a frequent operation in scientific analysis, and digital filtering is a common instance of this type of operation. Operators for interpolation and extrapolation of time series are also included in the prototype, as they are instances of operations that can generate new data records in a time series. Finally, an operator for merging two time series is included. This is essentially a join by time, although it is necessary that the two time series have corresponding time tags.

Two operations are provided for manipulating spectra. First, a spectral filter may be applied to a spectrum, i.e., a frequency filter performed in spectral space. This corresponds to the digital filter available for time series; filtering is a typical example of an operation that can be performed in either time-space or Fourier-space. Second, merging two spectra is supported by a merge operation similar to that for time series.

There are some operations which can be applied to all data types. The primary such operation considered in our prototype is a simple math function application, which supports many common scientific operations such as correlation, convolution and deconvolution.

Type conversion between bulk types is managed using explicit type conversion operators. In some cases, these operators do not require any physical manipulation. For example, converting either a time series or a spectrum to a set (when the appropriate record fields for time or frequency are available) requires no computation. When converting between time series and spectra, on the other hand, a Fourier transform or its inverse must be applied. As this is generally an expensive operation to perform, the decision on when to move between real and Fourier space is an important optimization.

Physical operators in the Volcano system are implemented in the form of iterators [1]. Volcano iterators already existed for relational operations as well as sampling, merging, sorting and mathematical function application. Additional, scientific operators were implemented as part of this research.

A new operator has been added to pass a window over a sorted data set, providing an implementation of interpolation, extrapolation, and digital filtering. A second new operator is the Fast Fourier Transform (FFT).

4 Logical Transformations

Logical transformations modify the order of computations. Identifying logical transformations for scientific operators is vital to the application of optimization in scientific databases, and constitutes a significant portion of our research.

The central issue in identifying logical transformations is equivalence. This is straightforward in relational transformations, where a transformation results in a logical expression that generates an identical output relation. Due to the effects of numerical accuracy and stability, transformations which include numerical operators frequently do not produce exactly identical results, even though the transformation is considered valid mathematically. This requires a broadening of the concept of equivalence in scientific analysis queries.

Multiple math operators (which apply mathematical functions to each record in a bulk type, possibly modifying the records or generating new result records) can be reordered using the standard mathematical rules for manipulating expressions. Since digital filtering and interpolation operations consider a window of nearby records when deriving an output record, these operators must not be commuted with operators that do not add or remove records or modify the portions of the records used by these operators.

In general, equivalence of two scientific computations depends on requirements of numeric accuracy by the application or the user. Thus, it is impossible to specify all transformation and all their conditions without knowledge of the application, leading to the need for an extensible optimizer such as the Volcano optimizer generator [2].

5 Implementation Rules

An implementation rule defines an execution strategy for a sub-expression of one or more logical operators. As a relational example, implementation rules indicate that a join operator can be implemented by either a hash- or a sort-based algorithm.

Once the implementation rules have been applied to generate an execution plan, the cost of that plan must be estimated to allow comparison of alternative plans. Cost is treated as an abstract data type in the optimizer generator, and may include factors such as resource utilization or response time. In our prototype, we have chosen to use time to completion as the basis for optimization. I/O and CPU costs are maintained separately, and we assume that we can achieve asynchronous I/O. The greater of I/O and CPU costs for the full computation is used as the time to completion.

6 Example

This example demonstrates the application of our prototype optimizer to a scientific computation, which includes a number of data preparation and cleaning steps in both time- and Fourier-space. The computation compares, in Fourier-space, high-energy ultraviolet light from the Sun to ozone concentrations in the atmosphere.

The initial computation is shown in Figure 1. It consists of merging two pre-processed data sets, one containing the high-energy ultraviolet (EUV) data and the other containing the ozone (O_3) concentrations. The EUV pre-processing restricts data to the wavelength under consideration, interpolates the data to a regular sampling frequency, applies a filter function, and convolutes the data with a response function to remove known noise sources. The last step is performed in Fourier-space. The O_3 processing calibrates raw data to account for measurement altitudes, selects the relevant longitude and latitude, interpolates to a regular sampling frequency, applies a filter function, and convolutes the data with the noise response function.

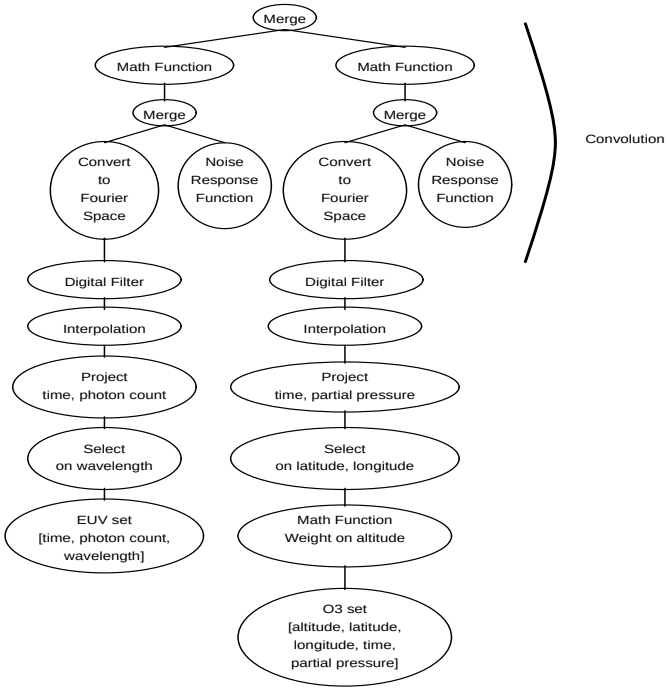


Figure 1: Example Logical Computation

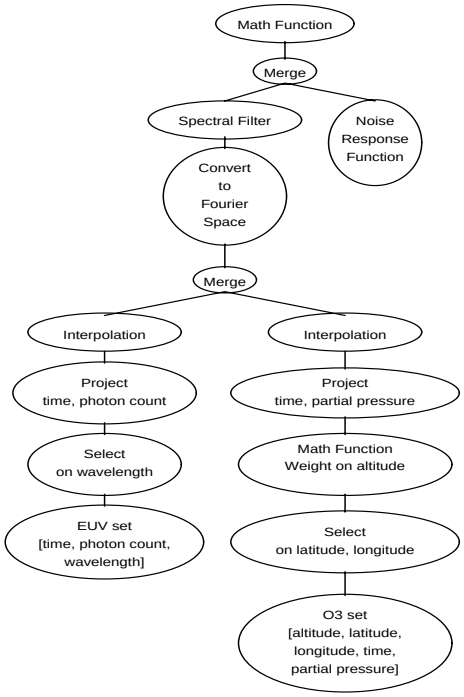


Figure 2: Optimized Computation

The optimized computation is in Figure 2. Pre-processing steps identical for both data sets in the initial computation are brought together, and some reordering is performed. Compared to a direct implementation of the original computation without using any algebraic transformation rules, these optimizations realize a speed improvement greater than a factor of two under our cost model. This improvement arises from three sources, namely the two-part Fast Fourier Transform algorithm, the placement of selections at earlier stages of the computation, and the reduction of data copying.

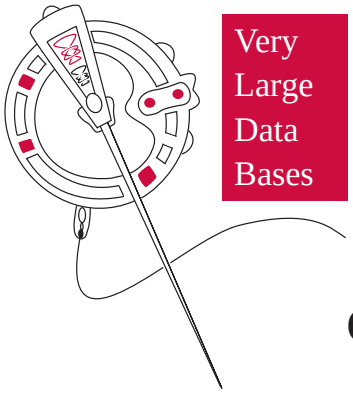
This example illustrates the applicability of algebraic query optimization to real scientific computations, and shows that significant performance improvements can be achieved as a result of this optimization. This example was taken from an analysis application at CU's Space Grant College, where we are currently implementing the techniques explored in our prototype in a scientific analysis environment. As part of our work on database support for computational models of turbulence, we are developing a more complete system based on these concepts and built on the Volcano extensible query processing system.

This research has been performed with the Space Grant College at the University of Colorado with support by NSF awards IRI-9116547 and ASC-9217394.

1. G. Graefe, "Volcano, An Extensible and Parallel Dataflow Query Processing System", *to appear in IEEE Trans. on Knowledge and Data Eng.*, 1993.

2. G. Graefe and W. J. McKenna, "The Volcano Optimizer Generator: Extensibility and Efficient Search", *Proc. IEEE Conf. on Data Eng.*, Vienna, Austria, 1993.

3. R. W. Hamming, "Digital Filters", Englewood Cliffs, NJ: 1977, Prentice-Hall.



19th International Conference
on Very Large Data Bases
August 24 - 27 1993
Dublin, Ireland

Call for Participation
VLDB 93

The Conference

The annual VLDB Conference is one of the major international conferences on the database calendar. It brings together researchers, developers and users of database management systems from academia, business and industry. In addition to a full scientific programme, VLDB 93 will include an exhibition, and special sessions and panels on speculative and futuristic topics and novel applications. A full tutorial programme, which covers a wide range of relevant topics, has been arranged and will be given by leading experts from the database community.

Registration	Early (before June 24th)	Full (after June 24th)
General	IR£266	IR£326
Student	IR£160	IR£160

The Location

VLDB 93 will be held in the modern Conference Centre of historic Trinity College, the oldest university in Ireland. Trinity is in the heart of the city of Dublin, famous for its architecture and literary heritage and world-renowned for its warm welcome. A wide choice of accommodation will be available to delegates including student rooms on campus, guest houses and hotels.
There are direct flights to Dublin from most major European cities and from several locations in North America. Excellent car ferry services are available between both the UK and France. London is just a short trip from Dublin and acts as a gateway to the rest of the world

For further information contact:

VLDB 93 Secretariat,
International Conference Organisers,
9 Western Parkway Business Centre,
Ballymount Road,
Dublin 12, Ireland.

Phone: +353 - 1 - 502422
Fax: +353 - 1 - 502954
email: vldbinfo@vldb93.ie

CALL FOR PAPERS

DATA ENGINEERING

Tenth International Conference on Data Engineering

February 14-18, 1994
Doubletree Hotel, Houston, Texas
Sponsored by the IEEE Computer Society

IEEE

SCOPE

Data Engineering deals with the modeling and structuring of data in the development and use of information systems, as well as with relevant aspects of computer systems and architecture. The Tenth Data Engineering Conference will provide a forum for the sharing of original research results and engineering experiences among researchers and practitioners interested in automated data and knowledge management. The purpose of the conference is to examine problems facing the developers of future information systems, the applicability of existing research solutions and the directions for new research.

An important part of the conference will be the Technology and Application Track in which we especially welcome practice-oriented papers discussing applications of database technology, performance studies, evaluation of new or existing technologies, new emerging applications, and multi-disciplinary issues. In the evaluation of papers submitted to this track, the committee will consider both their technical quality and the value of reported results to researchers and developers of information systems.

TOPICS OF INTEREST

The topics of interest include but are not limited to:

- AI and Knowledge-Based Systems
- Data Modeling and Database Design
- Data Consistency, Integrity and Security
- Database Programming and Query Languages
- Distributed Data Structures and Access Methods
- Engineering, Scientific and Design Databases
- Heterogeneous Database Systems
- Incomplete Information and Uncertainty
- Knowledge Representation & Discovery in DB.
- Multimedia Database Systems
- Object-Oriented Database Systems
- Optimization, Tuning & Performance Evaluation
- Parallel Databases
- Temporal, Spatial and Extensible Databases
- Transaction Management

PAPER SUBMISSION

Six copies of original papers not exceeding 6000 words (25 double spaced pages) should be submitted by May 31, 1993 to:

Marek Rusinkiewicz, ICDE
Department of Computer Science
University of Houston
4800 Calhoun Street
Houston, Tx 77204-3475
E-mail: icde94@cs.uh.edu
Tel. (713) 743-3350, FAX: (713) 743-3335

Tutorial and Panel proposals should be submitted to the same address.

ORGANIZING COMMITTEE

General Chairpersons: Ahmed Elmagarmid, Purdue **U.** and Erich Neuhold, GMD-IPSI Darmstadt
Program Chair: Marek Rusinkiewicz, University of Houston
Tutorial Program: Omran Bukhres, Purdue University and Peter Muth, GMD-IPSI Darmstadt
Panel Program: Witold Litwin, University of California at Berkeley

PROGRAM COMMITTEE

- Rafael Alonso, MITL
- B . R. Badrinath, Rutgers University
- Elisa Bertino, U. of Genoa (V. Chair, OO systems)
- Yuri Breitbart, University of Kentucky
- Alex Buchmann, TH Darmstad
- Phil Canatta, MCC
- John Carlis, University of Minnesota (publications)
- Arbee Chen, National Tsing Hua University
- Peter Dadam, Ulm University
- Amr El Abbadi, Univ. of California at Santa Barbara
- Ramez Elmasri, University of Texas at Arlington (V. Chair, data modeling)
- George Gardarin, INRIA
- Dimitrios Georgakopoulos, GTE Labs
- Goetz Graefe, Portland State University
- Ralf Hartmut Gueting, University of Hagen
- Theo Haerder, University of Kaiserslautern
- Tomasz Imielinski, Rutgers University. (V. Chair, database theory)
- Bala Iyer, IBM DB Technology Lab, San Jose
- H. V. Jagadish, AT&T Bell Labs
- Matthias Jarke, Aachen University
- Christian Jensen, Aalborg University
- Jie-Yong Juang, National Taiwan University
- Leonid Kalinichenko, Russian Academy of Sciences
- Yahiko Kambayashi, Kyoto University
- Larry Kerschberg, George Mason University
- Masaru Kitsuregawa (V. Chair, parallel databases)
- Yasushi Kiyoki, University of Tsukuba
- Wolfgang Klas, GMD-IPSI Darmstadtq
- Eva Kuehn, University of Technology Vienna
- Paul Larson, Waterloo U. (V. Chair, access methods)
- Dik Lee, Ohio State University
- Witold Litwin, University of California at Berkeley
- Akifumi Makinouchi, Kyushu University
- Yutaka Matsushita, Keio University
- James McKenna, Bellcore
- Dennis McLeod, USC (V. Chair, heterogeneous DB)
- Alberto Mendelzon, University of Toronto
- Michele Missikoff, IASI-CNR, Rome
- C. Mohan, IBM Almaden (V. Chair, transaction management)
- John Mylopoulos, University of Toronto
- Mike Papazoglou, Queensland Univ. of Technology
- Calton Pu, Columbia University
- Raghu Ramakrishnan, University of Wisconsin
- Krithi Ramamritham, University of Massachusetts
- Andreas Reuter, University of Stuttgart
- Nick Roussopoulos, University of Maryland
- Hans Schek, ETH Zurich (V. Chair, extensible and spatial databases)
- Peter Scheuermann, Northwestern University
- Gunter Schlageter, University of Hagen
- Arie Segev, University of California at Berkely
- Ming Shan, HP Labs
- Amit Sheth (V. Chair, Technology & Application track)
- Anoop Singhal, AT&T Labs
- Richard Snodgrass, U. of Arizona (V. Chair, temp. DB)
- Mike Stonebraker, U. of California - Berkeley (Awards)
- Alex Thomassian, IBM Watson
- Susan Urban, Arizona State University
- Gerhard Weikum, ETH Zurich
- Antoni Wolski, Research Center of Finland
- Gene Wuu, Bellcore

PUBLICATIONS & AWARDS

All accepted papers will appear in the Proceedings published by IEEE Computer Society. The authors of selected papers will be invited to submit an extended version for possible publication in the IEEE CS "Transactions on Data and Knowledge Engineering" and in the "Journal of Distributed and Parallel Databases." An award will be given to the best paper. A separate award honoring K.S. Fu will be given to the best student paper (authored solely by students). A limited number of travel grants to help defray travel costs of student authors may be available.

IMPORTANT DATES

- Paper, Panel, and Tutorial submissions: May 31, 1993
- Notification of acceptance: September 15, 1993
- Tutorials: February 14-15, 1994
- Conference: February 16-18, 1994

EUROPEAN COORDINATORS

- Eva Kuehn, University of Technology Vienna
- Gunter Schlageter, University of Hagen

FAR EAST COORDINATORS

- Yutaka Matsushita, Keio University
- Mike Papazoglou, Queensland U. of Tech.

IND. PROGRAM & EXHIBITS

Daniel Barbara, MITL

PUBLICITY

Abdelsalam Helal, Univ. of Texas at Arlington

FINANCIAL CHAIR

Ernst Leiss, University of Houston

LOCAL ARRANGEMENTS

- J.F. Paris (Chair), University of Houston
- C. Eick, University of Houston
- A. Cheng (Registration), Univ. of Houston

IEEE Computer Society
1730 Massachusetts Ave, NW
Washington, D.C. 20036-1903

Non-profit Org.
U.S. Postage
PAID
Silver Spring, MD
Permit 1398