MDBS: A Modeling and Database System to Support Research in the Earth Sciences [1]

Terence R. Smith, Jianwen Su, Divyakant Agrawal, and Amr El Abbadi [2]

Department of Computer Science,
Universtiy of California at Santa Barbara,
Santa Barbara, CA 93106
email: {smithtr, su, agrawal, amr}@ucsb.edu

Abstract

An interdisciplinary team of computer scientists and EOS earth-science investigators is jointly investigating the concept of a *modeling and database system* (MDBS). MDBS is intended to support large-scale earth science investigations by complementing EOS-DIS in terms of support for high-level modeling and data management. In particular, MDBS is intended to facilitate the iterative development of scientific models in data-intensive and computationally intensive applications. It is based on a simple characterization of scientific activity and on an advanced data model that together capture the essence of scientific modeling and database activities in terms of large lattices of *representational domains* and associated transformations that are employed to organize scientific knowledge, data and algorithms. A high-level, declarative *modeling and database language* (MDBL) that is based on this data model and on four fundamental sets of scientific operations provides support for scientific modeling activities and for easy access to distributed databases of datasets and algorithms. It permits the efficient resuse of code for building new algorithms and supports key aspects of scientific modeling activities, such as *lineage* and *error tracking* in datasets, and for the concepts of *model* and *project*. We are applying a version of MDBS in order to solve several earth science research problems that have arisen in EOS investigations.

1 Computational Support for Scientific Investigations

1.1 Goals and Strategy

The overall goal of our scientific database research project is to investigate, design and develop computational support that permits scientific investigators to achieve their scientific goals *more efficiently.* 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.* Hence 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 scientists will adopt and use. In particular, our support takes the form of a Modeling and Database System (MDBS) which incorporates support for:

1. the scientific process of iterative model development;

[1]This research was supported under NSF IRI91-17094.
[2]We gratefully acknowledge the work of our graduate students Ana Paula Dutra de Aguiar, Gustavo Alonso, Yongmao Chen, Keith Park, Amitabh Saran and Anuradha Sastri.

2. the construction, maintenance and access of scientific databases upon which such modeling activities are based;

3. the concepts of *models* and *multi-investigator projects*;

Some of our criteria for an MDBS are that it be based on a conceptually solid foundation; that it possess a realization as *scientist workbench*; and that it provide computationally transparent and high-level support. One of our criteria for success is that a prototype of our MDBS permit the successful solution of previously "intractable" scientific problems.

Our goals were derived in part from a general analysis of earth science investigations and in part from a detailed analysis of the computational activities and "requirements" of a group of EOS investigators at the University of Washington. The long-term goals of these investigators include the development and application of spatially-distributed models of water, sediment and solute transport in the Amazon basin. In particular, we have found that they lack adequate computational support for the *iterative* construction and testing of their models. Furthermore, the generation of research results in this project is heavily and negatively impacted 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. The researchers 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 the 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.

Our strategy for designing and providing computational support to these researchers involves:

1. basing the design of our MDBS upon a simple model of scientific activity and upon a detailed investigation of the requirements of scientific investigators;

2. a top-down design approach that is based on a high-level language that permits simple expression of *much* of the functionality required by investigators;

3. employing a natural conceptual basis for organizing data and knowledge in a uniform manner that is tied, in particular, to our model of scientific activity;

4. providing a simple view of data in terms of datasets and not in terms of a distributed set of files;

5. providing a framework in which many scientific modeling tools can be easily integrated;

6. providing efficient support for:

    (a) exploratory data analysis

    (b) rapid prototyping and testing of models

    (c) production runs of models

A final but critical component of our strategy involves the testing of our ideas in terms of applying a prototype MDBS to solve *real* scientific problems whose solutions lie significantly beyond the computational support currently available to most earth scientists.

1.2 A Model of Scientific Activity

We believe it essential to base the design of an MDBS upon an appropriate model of the goals and activities of scientists. Hence we take a viewpoint in which we conceptualize scientific investigators as organizing much of their knowledge about the world 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 *symbolic representations* of these abstractions, which we term *representational domains* (R-domains). In general, we may view a set of R-domains as being necessary for any *computations* that involve domain elements 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 a 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* →*Areas-of-Polygons* and **demtodb** : **DEMs** → **DBs**. During the process of modeling some phenomenon, scientists may, for example, create 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*.

1.3 MDBS and the Model of Scientific Activity

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 a manner that is in accordance with our model of scientific activity. For example, 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 may also access datasets by content. In general, the database support underlying MDBL incorporates:

1. a uniform structuring and organization of the data that is independent of both 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. The construction of, and access to, transformations on the domains.

4. Support for concepts such as *model* and *project* including support for concurrent access to datasets within a project, that are independent of the system implementation.

2 The Modeling and Database System

2.1 A Data Model for R-domains

The data model is formulated in terms of R-domains. Each C-domain is represented in terms of a single *abstract* R-domain with a set of associated transformations, and a set of *concrete, equivalent R-domains*. Abstract R-domains serve as the specification of the domains, including the domain names and transformation names. Each concrete R-domain implements the abstract R-domain: the structure of the elements and its transformations in terms of a specific 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. Different concrete R-domains of the same abstract R-domain may have distinct representations but they should be equivalent. The purpose of abstract R-domains is then to define a "uniform" interface for all concrete R-domains. The defining characteristics of any concrete R-domain include:

1. the domain name;

2. the *structure* (or *type*, or *representation*) of the domain elements, which is described using *tuple, set, sequence, array, finite function*, etc. constructors;

3. constraints on the values of domain elements, which are expressed declaratively and naturally in a logic for the data structures;

4. implementation 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.

In general, abstract R-domains form a lattice on which inheritance of both structures and transformations occurs naturally.

An important mechanism for the inductive construction of new R-domains involves the application of constructors (e.g., set, tuple, sequence, function) 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.

2.2 A Model of Metadata

The metadata of a database is also represented in the same database and allows simple queries. The model of metadata possesses the following characteristics:

1. it is based upon our concepts of concrete and abstract R-domains;

2. it has a simple representation using relations;

3. it permits simple queries concerning the existence and properties of abstract and concrete R-domains; the various concrete R-domains associated with a given abstract R-domain; the existence and properties of the transformations associated with R-domains and elements of the various R-domains; and the interrelations between the domains.

More specifically, the metadata is represented by four sets of relations: the *abstract relation*, which contains all abstract R-domains; the *concrete relation*, which describes all concrete R-domains for each abstract domain; the *homomorphisms relation*, which has the transformations for converting elements between different concrete R-domains; the set of *element relations* (one for each abstract R-domain), each of which includes the element identifiers (or addresses) and the descriptions of the elements.

2.3 MDBL: A High-level Modeling and Database Language

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. MDBL permits system access at a variety of levels for different classes of users and tasks and is a declarative and expressive high-level language providing simple support for about 80-90% of scientific activities in terms of a very simple functionality. It supports the construction of transformations using previously defined transformations and provides access to "lower-level" language for more powerful modeling facilities and for the construction of algorithms; MDBL is also designed to permit a simple translation of its commands in terms of an easy-to-use visual language that may be used as the basis for constructing a graphical interface.

In MDBL, both abstract and concrete R-domains as well as the elements of domains and the transformations defined on domains, are treated equally. As one may note from Figure I, the language supports four sets of high-level commands and provides a persistent workspace for each user.

$$\left\{\begin{array}{l}\texttt{ACCESS}\\ \texttt{CREATE}\\ \texttt{STORE}\\ \texttt{DELETE}\end{array}\right\}\left\{\begin{array}{l}\texttt{DOMAIN name}\\ \texttt{TRANSFORMATION name}\\ \texttt{ELEMENT name IN domain} - \texttt{name}\end{array}\right\}\texttt{WHERE conditions}$$

Figure 1: MDBL Commands Overview

The first set of commands permits *accessing of information on domains, transformations and domain elements* from persistent storage. The second set of commands permits the *creation of new domains, new transformations and new domain elements.* In particular new transformations may be created by a variety of approaches, including algebraic expressions involving previously defined transformations while new elements may be constructed by *applying* transformations to previously accessed or constructed elements. The third set of commands permit the *storage of newly constructed domains, transformations and elements* in persistent storage. The final set of commands permit *deletion of domains, transformations and elements.* The WHERE conditions in each of the commands are specified declaratively in terms of a simple syntax.

MDBL combines the conventional ideas of "data definition" and "data manipulation" languages. To exemplify the simplicity of MDBL, an expression to access a particular instance from the domains of *DEMs* is written as:

$$\texttt{ACCESS}\ e_1\ \texttt{from DEMs WHERE}\ C_1, ..., C_n$$

where the conditions $C_1, ..., C_n$ on e_1 are expressed in terms of other domain elements and any transformations of those elements that are employed in defining the element. The creation of a new element by the application of the transformation *demtodb* to a domain element may be expressed as:

$$\texttt{CREATE}\ e_2\ \texttt{in DBs WHERE}\ e_2\ =\ \texttt{apply}\ \textbf{demtodb}\ \texttt{to}\ e_1.$$

2.4 Database Support

2.4.1 Storage Support for MDBL

In order to cope with the size of individual data objects and the volume of scientific data, it becomes necessary to design and develop operating systems support for efficient retrieval, storage, and organization of scientific data. The attribute-based naming scheme used in conventional databases is not suitable for scientific data since it assumes a flat name-space. Hierarchical naming, which is widely prevalent for file-systems, is useful for efficient management of large number of data objects. Clearly, due to the large volume of data in scientific databases and due to the need for attribute-based browsing in such an environment, a fusion of the above two approaches becomes necessary. Furthermore, the naming scheme for scientific databases must allow customization of the name-space for individual scientists. This is because, in general, a scientific database will be shared by a variety of scientists and each of them will have different needs resulting in conflicting requirements for name organization. Recently, such a naming scheme, which provides customization, has been developed for the *Prospero* project [4]. Although this scheme is designed for filesystems, it provides mechanisms to integrate attribute based naming scheme necessary for databases.

In order to implement our ideas concerning database support, we are currently investigating the integration of several available database tools, including the declarative database language CORAL [5], the data management system POSTGRES [6] and the virtual file system PROSPERO, as well as software packages that support scientific modeling and data processing operations. Some of the main aspects of our current design include:

1. the integration of the Prospero file system with MDBL and CORAL;
2. a file-based interface to access database objects from CORAL;
3. the integration of database objects that are distributed over a network via a virtual file-system based on Prospero;
4. the tailoring of the virtual system according to a user's needs.

In the context of storage support, we are also investigating the following issues:

1. using POSTGRES to provide database storage support for very large objects in MDBS/CORAL;
2. developing a new storage architecture that will use the most recent version of CORAL and would eliminate the requirement for in-memory objects;
3. extending the notion of Prospero filters, which are used with file directories, in the context of databases.

The select filter mechanism allows the users to selectively view data objects in a given directory. The derived filter mechanism permits the users to organize objects in a directory based on the selected attributes of the objects (i.e., it creates virtual levels in the name space). Finally, the union filter mechanism allows the users to seamlessly combine multiple directories stored at different places into a single directory. Of course, these mechanisms can be combined to provide a rich and powerful naming mechanism for scientific data. In order to extend these notions in the context of databases, we need to apply the filters to the files instead of restricting them to directories. These filters can be used to implement efficient indexing for scientific data.

2.4.2 Database Support for Geographic Data

Geographic Information Systems (GIS), or database technology applied to geographic data, have several characteristics that distinguish them from traditional database systems. In particular, geographic data is often encountered in many different formats (paper, text files, raster files, vector files, relations, numbers, etc.), under many different representations (graphic, numeric, textual, etc.) and in large quantities (exceeding traditional database applications by orders of magnitude). Furthermore, geographic data is not only used for retrieval and consultation purposes but also as input to complex transformations, as part of the ongoing scientific research in the area. In many cases, geographic information is highly interrelated and isolated pieces of data may not be easily interpreted without external references to other data. The fact that the meaning of the data is exogenous, i.e. not contained in the data itself, adds another layer of complexity to the design of these systems.

Since a database can not be properly designed without a clear understanding of the data it will contain, a first goal in our project [3] was to formalize the contents and meaning of geographic data. This was done in the three steps of:

1. incorporating into our model the recently adopted Spatial Data Transfer Standard (SDTS), which defines the additional information required to provide geographic data with meaning;

2. providing means to associate the additional information specified by the standard to the actual data by means of an object oriented approach in the representation of geographic data;

3. providing accounting mechanisms for the transformations performed on the data.

When geographic data is not the product of direct observation (field measures, radar images, satellite photos, etc.) it is derived data, i.e. the product of a series of transformations applied to one or more sets of data. Most of the research in this field is based on the interpretation of this derived data, hence it is crucial to provide the means to recreate those transformations and store the relations created between the input and the output data. One of the most interesting results of our work has been to show that several problems encountered in these databases, usually regarded as independent and treated as such, are in fact only different aspects of the same problem, to keep track of object interdependencies. The object oriented model we propose provides the data structures necessary to store the information relevant to those interrelations. A geographic object that contains its meaning and the information necessary to recreate its genesis has proven to be a very powerful abstraction. To prove the flexibility of the model we have developed several concise and efficient algorithms for:

1. *lineage tracking*: describing the sources and methods used to create the data;

2. *coherency checking*: in a database with multiple versions of each object the system must provide support to detect inconsistencies in the retrieval of data;

3. *change propagation*: the updating of derived data when the sources are replaced by newer versions;

4. *version maintenance and storage optimization.*

2.4.3 A Model of Collaboration and Cooperation

We have also developed a more advanced model of collaboration and cooperation in scientific databases [2]. Several scientists may wish to collaborate together to correctly execute their algorithms concurrently. In particular, a scientist may start working on some raw data and then start using derived data that resulted from another scientist's computations. Hence, conflicting accesses to the same set of data may occur, and the need for concurrency and data flow control is necessary. The traditional database approach for concurrency control requires the complete isolation of the various users as long as their computation (transaction) has not terminated. We have recently developed a relaxed correctness criterion for such an environment referred to as *relative atomicity*. Using this model, a scientist explicitly states when and where other scientists may observe data resulting from incomplete computations. Hence, the traditional "isolation" criterion for transactions is relaxed in an organized manner. In particular, we have proposed a model that can efficiently recognize the correct execution of a set of concurrently executing processes, and have proposed protocols for ensuring such correct executions. In the course of this project, we plan to further relax atomicity requirements for scientific environments where teams of researchers access a common scientific database. Often, in such cases, groups of scientists may be divided into teams, where within each team, the scientists observe each others results, while across teams, results are only made visible at the end of well defined computations. In such cases, a team of researchers collectively execute a "collaborative transaction", while between teams of scientists, either standard or relative atomicity criteria are used to control and restrict information flow.

2.4.4 Detection of Corrupted Data

In large geographic databases, files are often replicated at several sites in a distributed system for purposes of efficient and fast access. If a copy of the file is available locally, the cost of read operations is negligible. This is especially important since many data files are very large and are often used as a source for deriving data, i.e., most operations are read operations. One prominent example of such systems involve image databases where many large (e.g. gigabyte-sized) images, often referred to as raw data, are transmitted to a central repository of data, and scientists across the globe acquire copies of this raw data in order to derive more structured images. Due to failures, some pages may get corrupted. Since the files are quite large and the portion of corrupted pages is expected to be small, exchanging all copies and comparing them is unacceptable. We have developed an approach, based on coding theoretic techniques, which provides an efficient and optimal solution for a coordinator-based model, in which a single site acts as the coordinator and identifies all corrupted pages in all the copies [1]. In particular, the solution is a decentralized solution with less communication overhead. The coordinator model is especially appropriate in the satellite-based image database where the scientists with copies of the raw data may only be known to the central coordinator site.

3 Prototyping Efforts

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. The main 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 of the EOS Amazon project at the University of Washington, 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 several examples of specific scientific investigations. We have focused on typical sequences of operations that appear to us to be "representative" of the modeling and database access operations of many scientists working in an interactive and iterative mode. In one application, for example, digital elevation models (DEMs) of part of the Amazon river basin are first selected and merged into one single DEM; point rainfall data are then chosen for the area that the new DEM covers and interpolated over the area; a new domain *Storms* is created and elements are created from the interpolated rainfall data; finally, a specific water routing algorithm is then applied to predict hydrographs for the area.

We have implemented complete examples in an environment in which MDBL commands are typed in by the user. MDBL uses a "virtual" tree of files to manage the datasets, the metadata, and specific algorithms, such as interpolation and water routing, for scientific modeling; it also applies display utilities from a mathematical software package. The centralized virtual view of the tree structure of files is provided by the PROSPERO file system. The operations in the example include creating domains and elements; accessing elements; executing algorithms stored in binary files; and applying display functions. The metadata can also be accessed through CORAL. We are currently in the process of building a more complete implementation of MDBS which manages and accesses the datasets and metadata, the application algorithms and which integrates other tools, such as Mathematica, as well as providing network transparency. We have also designed a large number of general and application-specific 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 previously been expressed in FORTRAN by the applications scientists. One example involves the construction of a drainage network from a DEM and the application of a spatially-distributed, time-sliced hydrological model in order to compute the discharge of runoff from the network.

References

[1] K. A. S. Abdel-Ghaffar and A. El Abbadi. Efficient Detection of Corrupted Pages in a Replicated File. In *Proceedings of the Twelfth ACM Symposium on Principles of Distributed Computing*, August 1993. To appear.

[2] D. Agrawal, J. Bruno, A. El Abbadi, and V. Krishnaswamy. Relative Serializability: A Theory for Relaxing the Atomicity of Transactions. Technical Report, 1993.

[3] G. Alonso and A. El Abbadi. A Versatile Modeling Approach for Geographic Databases. Technical Report, 1993.

[4] B. Clifford Neuman. Prospero: A Tool for Organizing Internet Resources. *Electronic Networking: Research, Applications, and Policy*, 2(1), 1992.

[5] R. Ramakrishnan, D. Srivastava, S. Sudarshan, and P. Seshadri. Implementation of the CORAL Deductive Database System. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 167–176, June 1993.

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