

Mapping A Common Geoscientific Object Model to Heterogeneous Spatial Data Repositories*

Silvia Nittel
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90024
silvia@cs.ucla.edu

Jiong Yang
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90024
jyang@cs.ucla.edu

Richard R. Muntz
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90024
muntz@cs.ucla.edu

Abstract

Lately, a need to integrate specialized data management systems such as geographic information systems (GIS), or multimedia systems has gained importance [6]. A large variety of different data sets are available in various specialized repositories, and users would like to access and manipulate these data sets in a uniform way. Additionally, it is desirable to make the specialized functionality provided by the individual repositories available to the user application through a homogeneous interface. At UCLA Data Mining Laboratory, we are developing **geoPOM** (**geoscientific Persistent Object Manager**), a heterogeneous geoscientific object system which provides a homogeneous interface to heterogeneous spatial data repositories [20].

geoPOM provides an object-oriented spatial data model for the definition of user-defined spatial object types. Internally, *geoPOM* maps user-defined spatial object types to different specialized spatial data repositories, and employs their storage, search and spatial query capabilities. In this paper, we focus on the goals, problems and approach taken in *geoPOM* towards defining the spatial functionality of the heterogeneous geoscientific object system available to the user, as well as, the mapping of the spatial object model to the diverse semantic and functional characteristics of the heterogeneous spatial data repositories.

1 Motivation

During the past decade, distributed, heterogeneous data management has been a prominent database research area due to the existence of a diversity of independent databases and file systems, and the desire to provide easy and uniform access to integrated data. The focus has been to integrate data stored in heterogeneous, however, general-purpose data repositories such as relational DBMS or file systems, and to provide a relational or object-oriented DBMS-like interface for access, query and manipulation of the data. Lately, a need to integrate information in *specialized data management systems* such as geographic information systems (GIS) or multimedia systems has gained importance. Many different data sets are available in various specialized repositories such as spatial databases managed by spatial database management systems (DBMS), scientific data sets stored using domain-specific file format libraries, maps and raster data stored in file-based GIS systems, or pictures, video and sound managed by specialized multimedia storage

servers. In a scientific computing environment, the user would like to write applications which access and manipulate these data sets in a uniform way. In addition, it is desirable to make the specialized functionality provided by the individual repositories available to the user application.

However, the integration and abstraction from a collection of *specialized data repositories* introduces a set of novel problems. Of particular interest to our research efforts are information systems which support the collection, analysis and generation of geoscientific data. Our goal is to implement a *heterogeneous geoscientific object system* which provides a homogeneous interface to a set of heterogeneous spatial data repositories. Numerous spatial data repositories exist today that have syntactically and semantically heterogeneous data models describing basic spatial data or storage types. Furthermore, the definition and availability of spatial operators such as intersection, containment, union, etc. varies significantly from system to system, as does the availability of spatial query languages, support of spatial access paths, etc.. Thus, the semantic and functional gap between spatial data repositories is much larger than is found in traditional heterogeneous systems today. Therefore, the task of defining the functionality of a CDM, and the mapping of object types defined in the CDM to the various heterogeneous spatial data repositories is much harder. First, the spatial data repositories support different kinds of specialized spatial functionality; here, the question arises which of the functionality should be made available through the interface of the integrated system. Since the goal is to make a significant fraction of the functionality available, this leads to the question of how to design the internal architecture of the heterogeneous spatial object system in order to achieve plug-in semantics for spatial repositories without changing functionality available at the interface of the heterogeneous geoscientific object system.

At the UCLA Data Mining Laboratory, we are developing *geoPOM* (**geoscientific Persistent Object Manager**), a geoscientific object system capable of integrating heterogeneous spatial data repositories. *geoPOM* facilitates uniform access to spatial data in the repositories while maintaining the autonomy of the spatial repositories [20]. *geoPOM* provides an object-oriented data model extended with a set of predefined geoscientific object types based on the OGIS object model [4, 9]. Internally, user-defined temporal-spatial object types are mapped to a variety of low-level spatial data repositories like scientific file format systems (e.g. HDF, netCDF, etc.) and high-level repositories

*Fourth ACM Workshop on Advances in Geographic Information Systems (ACM-GIS96), Rockville, Maryland, Nov., 1996.

such as spatial DBMSs (e.g., ESRI’s SDE or Illustra). Spatial data repositories are used according to their capabilities in geoPOM, so that minimal amount of additional code has to be developed. geoPOM provides a high common denominator of the functionality available in the underlying spatial data repositories, thus, providing not only data integration but also *functionality integration* across different specialized spatial data stores. To achieve a semi-transparent handling of the repositories, we introduce a classification of spatial data repositories, and ensure that the repositories of one class can be used interchangeably. We claim that a mapping approach as described above results in the efficient utilization and integration of specialized data repositories, especially for spatial data repositories. In this paper, we describe a set of novel problems introduced by the integration of spatial data repositories, and the solutions implemented in geoPOM.

The remainder of this paper is organized as follows. Section 2 discusses the goals and main problems that arise when attempting to map a common geoscientific data model to the data models of heterogeneous spatial repositories. In Section 3, we present a short overview of the geoPOM data model and the spatial data repositories used for the first version of geoPOM. Section 4 describes the mapping approach utilized by geoPOM. We discuss related work in the area in Section 5, and conclude with a discussion of remaining problems and the status of geoPOM.

2 Mapping a Common Geoscientific Object Model to Spatial Data Repositories

A heterogeneous spatial object system provides applications with the illusion of a single spatial data repository, but without actually storing all the data in a single storage system. Instead, any (spatial) DBMS or file system available can be used to store the data. The heterogeneous spatial object system provides a spatial data model, the so-called common data model (CDM), which is used to define temporal-spatial object types. Internally, the common object types are mapped to a *spatial data repository*. Thus, temporal-spatial geoPOM types are translated to “types” defined in terms of the data model or API of the local data repositories. The mapping is performed by a *repository wrapper*. A repository wrapper translates between the CDM and the native data model of the repository, i.e. translates requests for object creation, manipulation, and queries to the native spatial repository.

In the following sections, we first describe our goals towards integrating and abstracting from the functionality of heterogeneous spatial data repositories. Second, we present a set of novel problems introduced by the fact that the interface and functionality of domain-specific, highly-specialized data repositories shows a much higher degree of functional and semantic differences than those found in traditional data management systems like relational or object-oriented DBMS.

2.1 Requirements

Facing the task of designing and implementing a heterogeneous spatial object system using a variety of different spatial repositories with varying capabilities, we must first define a set of goals and requirements. There are mainly three goals to achieve:

- **Highest common denominator:** The set of spatial repositories that have to be integrated vary significantly in their degree of functionality. Spatial repositories like file systems and disk storage archive format libraries provide only rudimentary functionality (basic storage types and simple operations) [13]. On the other hand, spatial DBMS such as the Illustra DBMS with its Spatial Blade [15], ESRI’s Spatial Database Engine [12] or Paradise [11] provide a high-level set of functionality such as abstract spatial data types, spatial-temporal operators, optimized spatial query processing, spatial access paths, etc.. When abstracting functionality from such a set of spatial data repositories, the goal is to preserve a high common denominator of the functionality provided by the individual spatial repositories thereby achieving a higher degree of functionality than using the systems on their own. Comparable to a heterogeneous multimedia system, the different components are used according to their capabilities, and a high common denominator of the component systems should be provided in the CDM.
- **Minimal additional implemented functionality:** Providing a common spatial data model with high-level functionality, the functionality has to be mapped to the available spatial repositories. Although, some spatial repositories provide only basic spatial functionality, it is not the goal to implement a large amount of additional code to “bridge” between the high-level, DBMS-like functionality of the CDM and the rudimentary functionality of a scientific file format system.
- **Transparency:** In a traditional approach to heterogeneous data repositories, all object types definable with the CDM can be mapped to each participating data repository; thus, the data repositories are interchangeable providing the same degree of functionality and therefore can be used transparently by the integrating system. However, this is not the case when specialized repositories are used, each providing a different set of specialized functionality, since we do not want to implement a lot of functionality on top of these systems. In this case, complete transparency of the repository used within the heterogeneous spatial object system can not be achieved. The integrating system somehow has to “know” which data types and their behavior are mapped to which repository. However, it appears to be cumbersome to compile a list detailing which data repository an object can or cannot be mapped to, as well as, the dependencies of object types on particular spatial repositories. Thus, to reduce the complexity of the heterogeneous spatial object system, some degree of transparency has to be provided.

These goals are typical for the integration of specialized data repositories. Most of the goals are known from the “traditional” area of heterogeneous data management; however, they take on a new meaning and lead to a different architectural approach when applied in the area of integrated, heterogeneous spatial object system. Here, spatial data repositories are used according to their specialized capabilities with the goal of not

only offering a rich set of functionality available on top of the repositories, but of also providing an architecture that makes the implementation of only a minimal amount of additional code necessary.

2.2 Problems in Integrating Heterogeneous Spatial Repositories

When mapping the concepts of a common geoscientific data model to the heterogeneous spatial data repositories, several problems are introduced. Some problems are already known from the more “traditional” field of heterogeneous data management, but a new set of problems arises due to the fact that we are dealing with semantically higher-level, specialized functionality. The mapping process can be divided into two parts: mapping the non-spatial part of the geoscientific CDM, and mapping of spatial object types to the spatial functionality. For the first part, a set of solutions has been proposed in the field of “traditional” heterogeneous data management systems. The second part, however, introduces a set of novel problems which are described below.

2.2.1 Inadequate Coordinate Systems

Spatial objects are specified with respect to a coordinate system, i.e. a n -dimensional vector space consisting of a basis O (the origin of the coordinate system), and a set of n linearly independent vectors x_1, \dots, x_n . Commonly used coordinate systems are the two- or three-dimensional Cartesian coordinate system representing a spatial object by (x, y, z) values, or the three-dimensional spherical coordinate system using latitude-longitude measures. Problems arise if the CDM supports a coordinate system that is not available in the underlying spatial repository, e.g. the CDM supports a three-dimensional spherical coordinate system while the underlying spatial repository only supports a two- and three-dimensional Cartesian coordinate system. Supporting a spatial model like the OGIS object model in the CDM makes the problem even harder since the OGIS model allows one to use and define many different coordinate systems, and the base geometry types are defined to be representable in and transformable to all the coordinate systems.

2.2.2 Difference in Supported Geometry Types

A geoscientific CDM provides a set of predefined spatial object types. In the OGIS model, geometries like *point*, *curve*, *linestring*, *ring*, *line*, *surface*, *polygon*, etc. are supported. Spatial repositories, actually used for the storage of data, might provide a different set of base geometry abstract data types such as *point*, *line segment*, *box*, *quad*, *circle*, *ellipse*, *path*, *polygon*, and *point set* in the Illustra Spatial Blade or *point*, *spaghetti*, *line*, *ring*, *polygon*, *doughnut* and *point cluster* features in ESRI’s Spatial Database Engine (SDE). When mapping a common spatial object model to a spatial repository, typically only a subset of the CDM’s geometry types can be mapped directly to supported spatial types of the underlying repository.

There are different solutions to deal with this problem. Assuming that the spatial repository is extensible (i.e., supports the introduction of new spatial

types), the corresponding spatial data type can be implemented and used directly. On the other hand, the spatial repository might not be extensible and/or we may not want to implement additional code. In this case, non-supported spatial data types have to be *approximated* using existing data types, e.g. the *line string* object type is supported via the *path* ADT in Illustra. In this context, we distinguish between two kinds of approximations: first, approximation by a data type that represents basically the same geometry, e.g. a *line* is represented by a *path* consisting of two *points* only, and second, approximation through a spatial data type with a different geometry, e.g. approximating a *curve* by a *path*.

2.2.3 Inaccuracy

In the section above, we discussed that a subset of the spatial data types of a CDM has to be approximated through other supported data types of the underlying spatial repository. The second kind of approximation we described, however, introduces a certain degree of *inaccuracy*. In this context, we define inaccuracy as the error introduced by approximating a geometric object by a simpler geometric object. E.g., in the OGIS object model, a *curve* is a one dimensional geometry that is stored as a sequence of *points*, while *line strings* are the simplest form of *curves* being linearly interpolated between the stored *points*. In the Illustra and SDE data models, however, the data type *curve* does not exist; thus, a *curve* has to be approximated through a *path* in Illustra and a *spaghetti feature* in SDE. Since, a *path* and a *spaghetti feature* have the semantics of a *line string*, i.e. linear interpolation between *points*, a representational error for *curves* is introduced. The degree of error, however, is not fixed. It can be minimized by e.g. representing the *curve* by a large number of *points*, or it can be magnified by a coarse approximation which uses a small number of *points*. The degree of error becomes apparent in the results of spatial operators applied on the data in the spatial repositories.

In summary, mapping a common spatial data model to a spatial repository introduces inaccuracy for the set of data types that have to be approximated. In the context of the integration of and abstraction from several heterogeneous spatial repositories, the approximation leads to a more severe problem since the set of geometry types that must be approximated varies from repository to repository. Thus, also the degree of inaccuracy varies from repository to repository, and is introduced for *different* spatial data types. However, a user of the integrated system expects for his/her applications the same degree of accuracy for all data types and all spatial operations independent of the spatial repository that is actually used for storage, i.e. spatial operators used in an application should return the same results independent of the fact that the data is mapped to repository A or B.

2.2.4 Different Degrees of Spatial Operators

A spatial repository supports a set of spatial data types and a set of corresponding spatial operators such as *Equal()*, *ContainedIn()*, *Intersect()*, etc. As was the case with different representations of geometry types mentioned above, the CDM might support a different

set of spatial operators than those provided by the underlying spatial repositories. *Missing spatial operators* are not easily simulated by a combination of available operators. Several options exist to deal with this problem. First, it might be possible to cast the spatial data type to another data type for which the spatial operator exists. E.g., Illustra supports the operator *Contains()* only for the type *box* which is a bounding box. Thus, it is possible to cast e.g. a *ring* to a *box*, and apply the *Contains()* operator to the *box* and the other passed geometry object. However, this is highly inaccurate. Second, the missing operator can be added to the spatial repository if the spatial repository is extensible. Third, if this is not the case, the data has to be retrieved from the repository so that the operator can be applied outside of the repository. This problem exists for most spatial repositories with the set of operators that is not supported varying from system to system.

2.2.5 Different Data Layout for Data in a File

In a low-level spatial repository like a scientific file format library, a data file can contain the same information, e.g. a raster, but the layout of the data within the file might be completely different than the layout of another raster of the same scientific data set. E.g., imagine a multidimensional data object consisting of the values for a measurement of certain parameter set represented in 4-dimensional space. The representation stays the same for all raster elements (each one stored in a file) of the data set, the layout of the 4-dimensional space, however, might vary from file to file. Thus, not all elements of a single scientific data set representing a spatial type of the CDM can be handled in the same way.

3 The geoPOM System

We faced many of these problems when designing and implementing the geoPOM system. In the following sections, we describe the geoPOM system, its data model, the spatial data repositories used in the first version of geoPOM, and the approach taken in geoPOM to implement an efficient integrated, heterogeneous spatial object system utilizing efficiently the capabilities of the heterogeneous spatial data repositories.

3.1 The geoPOM Data Model

geoPOM's data model is based on object-oriented data modeling techniques which have proven to be useful in geoscientific environments [10]. The geoPOM data model is based on the object-oriented data model standard proposed by ODMG-93 [7]. The basic modeling primitive in the ODMG-93 object model is the (persistent) object. An object can consist of a set of *attributes*, *relationships* with other objects, and a set of *methods*. The values of the attributes and relationships define the object's state. Values of attributes can be atomic, structured or unidirectional references to other objects. Relationships are bidirectional references to other objects, and can be 1:1, 1:n or n:m. An object can also contain collection-valued attributes such as bags, lists, sets, and arrays. The set of methods supported by an object type defines its *behavior*. Additionally, each object is uniquely identified by an object

identifier. Objects are strongly typed, and their characteristics are described by their *object interface*. The model also supports inheritance between types ([7]).

The ODMG-93 object model also defines a query language, i.e. the Object Query Language (OQL), a declarative query language usable to query objects based on their interfaces. OQL is a superset of SQL92, and based on select-from-where clauses. However, OQL also deals with complex objects, and allows method invocation and path expressions within select-from-where clauses. Using the ODMG-93 data model as a basis for a geoscientific data model, we need to support the ability to query collections of spatial objects. In order to provide fast access to subsets of these temporal-spatial object collections, we provide in addition to ODMG's basic access path types like hash functions, or B-trees, a set of *spatial indexes* like R-trees, Quadrees, KD-trees etc. These access paths can be used within the definition of spatial object collections, i.e. objects with at least one spatial attribute.

The second part of the geoPOM data model comprises the predefined spatial object types of the data model. These object types are based on the OpenGIS Specification (OGIS), a spatial object model proposed as a standard by the Open GIS Consortium (OGC) [4, 9]. The goal of OGIS is to provide well-known spatial types and common aggregates as basic building blocks for geoscientific applications. OGIS defines two basic geodata types, i.e. *features* and *coverages*. Features represent real world or abstract entities, and are associated with a spatial (or spatial-temporal) domain (e.g. roads in California). A coverage is an association of points, polygons or solids within a spatial-temporal domain with a value (of a certain type) (e.g. sea level pressure values over the area of Africa for a year). A *spatial domain* consists of a *geometric description* such as a description of points, curves, surfaces etc, and includes a mechanism for mapping the underlying geometry to a *temporal-spatial reference system*.

Considering a feature and a coverage as abstract objects, a spatial object is a concrete object and in general represents a spatial feature with both non-spatial and spatial attributes. A spatial object can also be an aggregation of features, i.e. a so-called *feature collection*. OGIS also defines operators on spatial objects based on their geometric properties. Since the OGIS standardization effort is an ongoing effort, we have taken a snapshot of the OGIS model, and extracted a subset that is significant for the first cut of the geoPOM data model. Figure 1 depicts the geoPOM object model.

3.2 Spatial Data Repositories Supported by geoPOM

In the first version of geoPOM, we are targeting a set of high- and low-level spatial repositories. In the first category, we are using two spatial DBMS, i.e. the Illustra DBMS and its 2-dimensional and 3-dimensional Spatial Data Blade [15], and ESRI's Spatial Database Engine SDE [12]. While both systems support spatial data types, their overall data model is significantly different. Illustra provides a set of predefined spatial abstract data types (ADTs) and a set of spatial functions. These ADTs can be used like base data types such as *int* or *char*, and are used as relational attributes. Spatial operators are applied on spatial attributes, and as predicates within queries. SDE employs a different

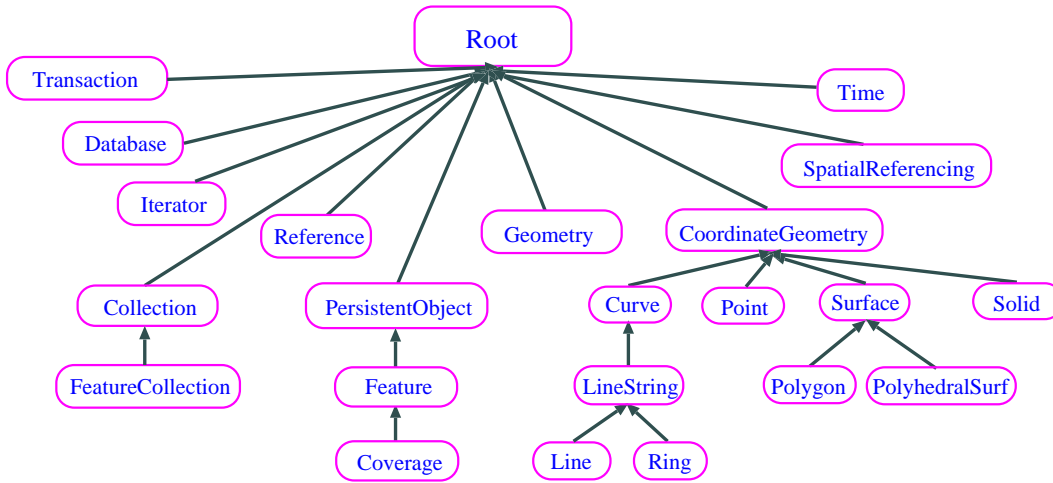


Figure 1: geoPOM interface hierarchy

strategy to supply spatial entities. In SDE, the entry point is a data set which consists of a set of map layers. All map layers of a data set share the same coordinate system. A map layer contains a set of features which are the spatial entities in SDE. All features of a map layer contain the same attributes, however they may have different geometries. Spatial operations are applied on features, and queries can only be executed within one map layer. Thus, Illustra and SDE vary significantly in the spatial base types and spatial operators they offer.

Low-level spatial data repositories are mainly used for the machine and operating system independent storage and retrieval of multidimensional scientific data sets such as raster data. They are, indeed, standard data interchange and archival formats built on top of a operating system’s file system. Many file formats have been developed and are used in different scientific domains such as meteorology, astronomy, for space mission data, etc. [13]. In first version of geoPOM, we employ the HDF file format [14] which has been chosen as the standard data format for EOSDIS, and the netCDF file format [19]. The HDF library contains interfaces for storing and retrieving compressed or uncompressed raster images with palettes, and an interface for storing and retrieving n-dimensional scientific data sets together with information about the data such as labels, units, formats, and scales for all dimensions.

4 Mapping Approach in geoPOM

In this section, we describe the approach that we employed for mapping object types defined in the geoPOM object model to spatial data types available provided by a set of commonly used spatial data repositories. The mapping process is divided into two parts: mapping the ODMG-part to the non-spatial part of the underlying spatial repository’s data model, and mapping the OGIS-part to its spatial functionality. Today, the non-spatial data model support provided by most spatial DBMS is based on relational DBMS technology; the mapping of an object-oriented data model to the relational data model is a known problem in DBMS research and different solutions have been developed during the past decade. We have based our mapping

strategy on the approach developed in [1, 16].

In this section, we focus on the approach we have taken to map the spatial part geoPOM’s data model to the spatial data repositories used within geoPOM. The main contributions are an architectural approach for the integration of different specialized data repositories, and solutions to problems found especially when integrating spatial data repositories.

4.1 General Mapping Approach

We decided early on in geoPOM’s design phase that geoPOM’s data model has to provide a high common denominator of the functionality supported in the individual spatial data repositories. Therefore, we used the OGIS object model [9] as a basis for the spatial part of the geoPOM data model. The OGIS model supports a rich set of functionality found today in spatial data repositories, and provides for representation of fine-grained vector data, for coarse-grained raster data, as well as, a set of spatial operators for both types of data. In addition, the OGIS model provides diverse types of coordinate and spatial-temporal referencing systems. To minimize the amount of code that had to be written, we decided to use a spatial data repository according to its *capability* in geoPOM, i.e. disk storage archive format libraries are primarily used for the storage of large multidimensional scientific data sets, and spatial DBMS are mainly employed for the storage and retrieval of fine-grained vector data, but also for large scientific data sets if this functionality is supported. However, achieving transparent use of spatial data repositories was harder to solve.

geoPOM’s predefined set of spatial object types is used to build complex, user-defined spatial object types. For each of the predefined spatial object types, a mapping to spatial functionality of available spatial data repositories used in geoPOM is defined. This strategy is then reused in the mapping of the user-defined spatial object types. However, only a subset of geoPOM spatial object types map naturally to the functionality provided by a given repository. For example, fine-grained vector data can be stored in disk storage format libraries, but since these systems do not provide sufficient query support or access paths, one would

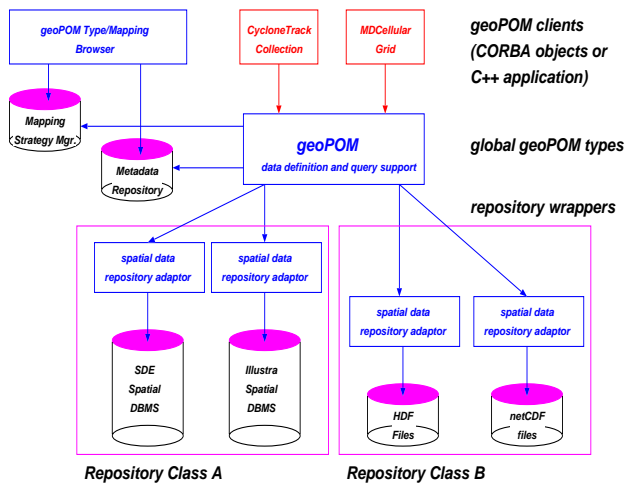


Figure 2: Architecture of geoPOM

need to implement the missing functionality outside of these systems. On the other hand, large-sized raster data can be stored in a spatial DBMS; however, if the spatial DBMS does not support storage of large data sets with internal mechanisms, mapping a raster data set to this repository type results in significant performance loss. To alleviate these problems, we introduce a *classification* of geoPOM’s *spatial object types* as well as of *spatial repository functionality*; we call a member of the first classification a *functionality class*, and a member of the second classification a *repository class*. For each *functionality class*, we define the *repository class* that can be used to map the spatial types of the *functionality class* to the functionality of the spatial repositories available in the *repository class*. For example, a *functionality class* \mathbf{F} contains the fine-grained geoPOM vector data types, and a *repository class* \mathbf{R} contains spatial DBMS, so that we guarantee that all object types of the *functionality class* \mathbf{F} are mapped “naturally” to all repositories of *repository class* \mathbf{R} .

A spatial data repository belongs to one or more *repository classes*. When a user defines a geoPOM object type, geoPOM’s mapping manager checks the spatial object types that are used in the user-defined object type, and determines the *repository class* to be used for mapping the object type. Within the repository class, all repositories can be used interchangeably, i.e. the use of a repository is *transparent* to geoPOM. If a user-defined geoPOM type contains spatial object types of two *functionality classes*, the mapping manager determines the *repository classes*, and checks if both classes contain a common member, i.e. a repository providing both sets of functionality. This repository is then used for the storage of the user-defined object type, i.e. its data. Since repositories of a *repository class* are interchangeable, each *repository class* has to contain at least one repository so that it can be guaranteed that the complete functionality of the geoPOM common data model is available to the user. The geoPOM architecture is depicted in Figure 2.

4.2 Solutions to Remaining Problems

Having determined this overall architecture for geoPOM, problems due to the goal of abstracting from

heterogeneous spatial data repositories, which were mentioned in Section 2.2, such as the non-availability of adequate coordinate systems, spatial types and operators, etc., still remain, and are addressed below.

Inadequate Coordinate Systems:

One of the first problems mentioned is the non-availability of adequate coordinate systems in the underlying spatial repositories. The OGIS object model used in the spatial part of the geoPOM CDM allows a large set of coordinate systems¹, thus, the problem described above exists in geoPOM when an underlying spatial repository only supports a limited set of coordinate systems.

In geoPOM, we solved the problem of inadequate or missing coordinate systems in the following way: the geometry of spatial objects represented e.g. in the three-dimensional spherical coordinate systems is *transformed* into an equivalent geometry represented in an available coordinate system of the spatial repository. The transformation is performed during the initial mapping process and at creation time of the object type within the repository. Thus, spatial objects are represented on two different levels, i.e. the coordinate system of the geoPOM data model, and the (potentially different) coordinate system used in the spatial repository. The transformation of geometry between two coordinate systems (and its inverse) is mathematically well-defined assuming that both coordinate systems have the same dimensionality, so that (in an ideal world) inaccuracy is not introduced. Also, all operations on the spatial object such as insertion and update of data, spatial operators and queries, are transformed to operators for the corresponding coordinate system used in the repository, and results are re-transformed back to the coordinate system used at the geoPOM object level. The transformation introduces a certain degree of performance trade-off, especially, if a large result set is converted to the coordinate system used on the geoPOM data model level.

Differing Supported Geometry Types and Inaccuracy:

The geoPOM data model supports a set of spatial data types that are not available in the underlying spatial repositories such as Illustra or SDE. We approximate a set of these data types by using corresponding geometry types of the spatial repositories, e.g. representing the geoPOM spatial type *line* by the Illustra spatial type *path*. In this case, inaccuracy is not introduced since a *path* subsumes a *line*.

However, there is a set of spatial data types of the geoPOM data model which is not supported by the underlying repositories, e.g. the geoPOM type *curve*. However, both Illustra and SDE support a spatial data type that can be used to approximate a *curve* (i.e. *path* for Illustra, and *spaghetti* for SDE). In both systems, the approximation introduces an error. However, since the data types used contain similar semantics, we apply the same type of approximation and are able to introduce the *same* degree of error for both repositories. Thus, the use of repositories is made transparent for the user application.

¹However, in the first release of geoPOM, we only provide predefined system-supported rather than user-defined coordinate systems.

The problem of inaccuracy has another aspect. For example, a repository in a *repository class* might support a spatial data type, while another repository of the same class does not support it. Since we would like to achieve transparency within a *repository class*, and introduce only a limited number of repository classes, the problem has to be solved within the class. As mentioned, the *degree* of error introduced by an approximation depends on the type of approximation used. In order to achieve transparency within a *repository class*, a close approximation is chosen so that the error introduced is small. However, complete transparency cannot be achieved in this case.

Different Degrees of Spatial Operators: Another problem is caused by missing spatial operators in the underlying spatial data repositories. When these operators are necessary to support the predefined behavior of geoPOM's spatial object types, the missing operators have to be implemented on top of the spatial data repositories. Here, we employ two strategies with respect to the functionality of the spatial repository. If a repository is extensible, i.e. allows user-defined spatial data types to be added, we implement the missing operator directly in the spatial repository, thus, achieving better performance since the operator is executed within the DBMS. On the other hand, if the spatial repository is not extensible, the operator is implemented outside the repository, thus, the data is first retrieved from the repository, and the operator is applied on the data outside of the DBMS. This is one of the cases, when additional code is implemented in geoPOM.

Different Data Layout for Data in File:

As mentioned, scientific data format libraries allow a flexible layout of data within files, so that not all semantically equivalent data files can be handled in the same way. We divided this problem into two parts: first, using scientific data format libraries for the creation of new files, and storing data into the files, and second, instantiating geoPOM objects to represent existing files containing scientific data sets. The first problem is easy to solve since we are able to impose a standard convention of how scientific data sets are stored in files. However, in the second case, we do not have influence over the data layout within the file since the data sets used will often be created outside of the local environment (provided by other scientists, sites or data archives). In this case, several solutions can be applied: first, tools are employed that automatically extract meta data information from the file, and convert the data file into the format used within geoPOM. However, this solution might not be efficient since scientific data sets are typically very large (tens of gigabytes), and often, just small subsets of a file are actually read. Thus, copying the data to another file might not be feasible. Thus, we employ a second solution which also uses tools to extract information of the file of the data layout within the file. The information describing the data layout in a particular file is stored within the data dictionary, and is used when the file is actually read.

5 Related Work

The problems and solutions introduced in this paper are related to the problem of mapping a geoscientific common data model to the interfaces of heteroge-

neous spatial data repositories within a heterogeneous geoscientific object system. However, to the best of our knowledge, related work specifically in the area of mapping a common specialized data model to the functionality of heterogeneous specialized data repositories does not exist today. In [17], a related problem is described, i.e. the definition of the topological equivalence of spatial databases. This work, however, focuses on the equivalence of the *content* of spatial DBMS rather than their data model. There are two other areas related to our research efforts, i.e. the mapping and its techniques in traditional heterogeneous data management systems, and querying techniques for heterogeneous semi-structured data repositories.

In the area of heterogeneous data management systems employing general-purpose DBMS like relational or object-oriented DBMS, two general approaches have been proposed to map the common data model to the data model(s) of underlying DBMS and file systems [23, 3, 22, 5], i.e. the structural mapping approach and the operational mapping approach. In the structural mapping approach, a schema defined in terms of the common data model, the so-called global schema, is mapped to an equivalent schema of the underlying DBMS. During query execution, queries on the global schema are transformed into queries on the local schema defined in terms of the native DBMS. Bertino [2] introduced the *operational mapping approach*. Here, instead of the correspondence between the data elements of the global and local schemas, the correspondence between *operations* of the different schemata is used for the mapping.

Recently, a lot of work has been done in the area of querying heterogeneous data repositories with varying capabilities, and exploiting the available functionality of these repositories during query processing ([8, 18, 21]). This work is mostly related to the integration of low-level repositories providing a limited set of functions and information such as repositories that are available via the World Wide Web providing data sets with minimal structure. However, this work mostly focuses on query processing; the problem of defining a high-level specialized common data model and the mapping of data types to the underlying repositories is not discussed. Furthermore, the repositories employed provide low-level, however, not necessarily specialized functionality as found in spatial data repositories.

6 Conclusions and Status of geoPOM

In this paper, we presented the problems encountered when attempting to integrate and abstract from a set of *specialized spatial data repositories*, and to provide a homogeneous interface to these repositories that makes a significant fraction of the underlying spatial data repositories' functionality available to the user application. We gave a short overview of the geoPOM system, a heterogeneous geoscientific object system developed at the UCLA Data Mining Laboratory, and described the approach taken in geoPOM to employ spatial data repositories according to their capabilities, and to provide a high-level set of their functionality at the geoPOM interface. In particular, we described the problems arising when mapping the spatial object types of the geoPOM data model to a set of spatial data repositories which spatial data models show significant semantically heterogeneity, and presented solutions im-

plemented in geoPOM.

Currently, we are implementing the mapping of geoPOM's data model to different spatial repositories. The initial prototype employs the Illustra DBMS, ESRI's Spatial Database Engine, and the HDF and netCDF libraries. The implementation of the mapping for the object-oriented data model part to the relational data model has completed as well as the mapping of the spatial part of geoPOM's data model to the spatial DBMS Illustra and SDE. At the moment, we are working on the extraction tools for file-based scientific data sets, and have started the implementation of the mapping to HDF and netCDF, which is expected to be running by the end of the year.

Acknowledgements

We sincerely acknowledge support from NASA EOS-DIS grant NAGW-4242, and we thank Edmond Mesrobian for his helpful comments on the paper.

References

- [1] T. Barsalou, N. Siambela, A.M. Keller, and G. Wiederhold. Updating Relational Databases through Object-Based Views. In *Proc. of the ACM SIGMOD Conference, Denver, Colorado*. ACM Press, 1991.
- [2] E. Bertino, M. Negri, G. Pelaggati, and L. Sbatella. Integration of heterogeneous database applications through an object-oriented interface. *Information System*, 14(5):407–420, 1989.
- [3] M.W. Bright, A.R. Hurson, and S.H. Pakzad. A Taxonomy and Current Research Issues in Multidatabase Systems. *Computer, IEEE Computer Society*, 1992.
- [4] K. Buehler and L. McKee. Introduction to Interoperable Geoprocessing. Technical Report OGIS TC Document 96-001, OGIS Project Technical Committee of the Open GIS Consortium, Inc, 1996.
- [5] O.A. Burkhres and A. Elmagarmid, editors. *Object-Oriented Multidatabase Systems*. Prentice Hall, Englewoods Cliffs, New Jersey, 1996.
- [6] M. Carey, W. Codey, L.M. Haas, P. Schwarz, M. Arya, W.F. Cody, R. Fagin, and et.al. Towards Heterogenous Multimedia Information Systems: The Garlic Approach. Technical Report RJ-9911 (87291), IBM Research Division, 1994.
- [7] R.G.G. Catell, T. Atwood, J. Duhl, G. Ferran, M. Loomis, and D. Wade. *Object Database Standard: ODMG-93, Release 1.2*. Morgan Kaufmann Publishers, San Mateo, California, 1996.
- [8] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *IPSI Conference, Tokyo*, 1994.
- [9] Open GIS Consortium. The OpenGIS Abstract Specification: An Object Model for Interoperable Geoprocessing, Revision 1. Technical Report OpenGIS Project Document 96-015R1, Open GIS Consortium, Inc, 1996.
- [10] M. Coyle, S. Shekhar, D.-R. Liu, and S. Sarkar. Experiences with Object Data Models in Geographic Information Systems. Technical Report TR 95-10, Department of Computer Science, University of Minnesota, 1995.
- [11] D. DeWitt, N. Kabra, J. Luo, J.M. Patel, and J.B. Yu. Client-Server Paradise. In *Proc. of 20th Intl. Conference on Very Large Databases (VLDB), Santiago, Chile*. Morgan Kaufmann, 1994.
- [12] SDE The Spatial Database Engine. *User's Guide, Version 2.0*. ESRI, 1995.
- [13] Scientific Data Format Information FAQ. <http://fits.cv.nrao.edu/traffic/scidataformats/faq.html>, 1996.
- [14] HDF. *Hierarchical Data Format, Software and Documentation*. <ftp://ftp.ncsa.uiuc.edu/HDF>, 1996.
- [15] Illustra. *User's Guide, Version 3.2*. Illustra, Inc., 1996.
- [16] M. A. Keller. Updating Relational Databases Through Views. Technical report, Dissertation, Computer Science Department, Stanford University, 1995.
- [17] B. Kuijpers, J. Paredaens, and J. Van den Bussche. On topological elementary equivalence of spatial databases. In *6th Int'l Conf. on Database Theory (ICDT'97), Delphi, Greece*. ACM/IEEE, 1997.
- [18] A. Levy, A. Rajaraman, and J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In *Proc. of the 22nd Int. Conference on Very Large Databases (VLDB), Bombay, India, September*. IEEE Computer Society, 1996.
- [19] netCDF. *Network Common Data Form*. <http://www.unidata.ucar.edu/packages/netcdf/>, 1996.
- [20] S. Nittel, R. Muntz, and E. Mesrobian. geoPOM: A Heterogeneous Geoscientific Object System. In *Submitted for publication*, 1996.
- [21] Y. Papakonstantinou, A. Gupta, and L. Haas. Capability-Based Query Rewriting in Mediator Systems. In *4th Int'l Conference on Parallel and Distributed Information Systems (PDIS96), Miami Beach, Florida*. IEEE Computer Society, ACM SIGMOD, 1996.
- [22] E. Pitoura, O. Bukhres, and A. Elmagarmid. Object Orientation in Multidatabase Systems. *Communications of the ACM*, 27(2):141–195, 1995.
- [23] A.P. Sheth and J.A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *Communications of the ACM*, 22(3):183–236, 1990.