Managing Derived Data in the Gaea Scientific DBMS*

Nabil I. Hachem, Ke Qiu, Michael Gennert, Matthew Ward
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01609, USA
e-mail: hachem,qiu,michaelg,matt@cs.wpi.edu

Abstract

One important aspect of scientific data management is metadata management. Metadata is information about data (e.g., content, source, processing applied, precision). One kind of metadata which needs special attention is the data derivation information, i.e., how data are generated.

In our application domain of geographical information systems (GIS) and global change research, we view scientific objects according to three different extents: spatial, temporal, and derivation. While the spatial and temporal extents have been studied and formal semantics to those extents proposed, derivation semantics have been ignored.

This paper presents a framework for capturing and managing scientific data derivation histories as implemented in the Gaea scientific database management system. We focus on how Gaea handles metadata and propose to extend current semantic modeling and object-oriented technology with special constructs: concepts, processes, and tasks. Concepts are used to capture entity sets with imprecise definitions. A process captures the derivation procedure of a specific scientific object class, while a task is the instance representing the derivation of a scientific data object. We believe that this framework, useful for GIS and global change studies, generalizes well to other scientific fields.

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

1 Introduction

There are several issues in scientific databases which make conventional database techniques insufficient to achieve the goals of data integration and data sharing [8, 14, 41]. One of these issues is metadata management. Metadata is information about data (e.g., content, source, processing applied, and precision). One kind of metadata which needs special attention is the data derivation information, i.e., how data are generated.

In scientific databases, data may be classified into two categories: base data and derived data. By base data, we mean those data obtained from well known sources outside the system. Base data are well understood and accepted by most scientists. Base data may be provided by a variety of standard agencies, government departments, research institutions, or generated by the scientists themselves. By derived data, we mean data obtained by scientists in their research by applying some algorithms on base data[1]. Unlike base data, derived data are not well understood. One important objective for the efficient management of scientific information is to be able to build on pre-existing knowledge, by sharing both base and derived data.

The potential for management appears in many aspects of scientific investigations. The first and most intuitive aspect is data management, stemming from the increasing volume and types of data. Second, there is a growing need to manage the algorithms to be applied to the data. As there are standard mathematics and statistics libraries available to the general scientific community, so too there should be common and consistent algorithms for all components of data analysis. To accomplish this requires the development of methods to manage the development, evolution, verification, and dissemination of algorithms. A third focus of management is in the scientific experiments themselves. The view of some types of investigation as iterative refinement dictates a need to monitor the progression of experiments to best identify future directions of highest potential. Experiment management also helps avoid unnecessary duplication of experiments and may encourage the reuse of aspects of previously performed experiments in the design of new ones. Finally, to facilitate the dissemination of results, external confirmation, and verification, some form of management is needed. Some branches of science have already identified this need, with standard formats for distributing data and reporting experimental results.

[1] One user's derived data may be another's base data. For example, cloud cover maps may be derived data for a satellite imagery scientist, but base data for a climatologist.

In geographical information systems (GIS) and global change research, studies involve gathering many forms of scientific data. This diversity ranges from tabular rainfall or census data to satellite imagery, such as Advanced Very High Resolution Radiometer (AVHRR), LANDSAT Thematic Mapper (TM) or SPOT, and vector based cartographic data. In these investigations, scientists may evaluate many classification schemes (principle components, maximum likelihood, linear mixture modeling), and perform experiments over diverse regions at different periods of time. Comparison of regions with similar climatic, socio-economic, or geographic characteristics may reveal heretofore undiscovered relationships or trends. However, inconsistencies between different classification methods may prompt the development of entirely different techniques based on different types of data.

Different scientists may employ different methodologies or apply different algorithms to reach the same objective. In order to make use of the results or data obtained by other scientists, we must have a full understanding of the data derivation history — how they are produced. It is only when such metadata are available that shared data can be meaningfully utilized and interpreted.

Consider the following simple scenario: two scientists are working on detecting the changes in vegetation index in Africa between 1988 and 1989. One may subtract the NDVI[2] of 1988 from that of 1989, while another divides the NDVI of 1989 by that of 1988. In this case, if only the resultant images are stored (as in common GIS such as IDRISI and GRASS [11, 36]), there is no way to share and compare the produced data unless the derivation procedures are known to both scientists.

It should be observed that the above problem does not exist in business databases. Data stored in a business database are based on descriptions about an existing enterprise, which are commonly accepted by all the users of the database. This is reflected by the global schema in a business database. In a university administration database, for example, the data stored for a student object is not affected by the way the information is obtained or who has put the information into the database. In scientific environments, individual researchers may share some information but manipulate it using different algorithms or ad hoc experiments to derive new data, which are added to the knowledge pool. Therefore, it is of absolute necessity to manage the data derivation history in scientific databases.

[2]NDVI is the normalized difference vegetation index. It is a qualitative measure of vegetation derived from AVHRR satellite imagery data.

In this paper, we investigate this problem and propose a framework for the management of derived data. This framework is being implemented in the Gaea kernel, a spatio-temporal DBMS for global change research [18]. We focus on how Gaea handles metadata, and provide a general framework for the management of scientific experiments and procedures. Our contribution parallels other efforts such as [4, 7, 32], while addressing limitations of current systems such as [11, 36]. We propose to extend current semantic modeling and object-oriented technology with special constructs: concepts, processes[3], and tasks. Concepts are used to capture entity sets with imprecise definitions. A process captures the derivation procedure of a specific object class, while a task is the instance representing the derivation of a specific scientific data object. We believe that this framework, useful for GIS and global change studies, generalizes well to other scientific fields.

Section 2 contains an overview of the architecture of the Gaea data analysis and management system under development at Worcester Polytechnic Institute. We concentrate on the relevant metadata management portion of the Gaea kernel. Section 3 discusses the relationships of our work to previous or current research projects. In Section 4 we provide a critique of our approach, while we summarize this work and discuss future directions in Section 5.

2 Metadata Management in the Gaea System

Gaea is a prototype spatio-temporal DBMS currently under implementation at WPI. We have chosen to implement the first prototype on top of the Postgres 3rd Generation DBMS [38]. We picked Postgres mainly because it provides us with a flexible ADT facility and the capability of dynamically extending our system with new analysis functions and data types. The goals of the Gaea system are to provide:

1. an integrated environment for the manipulation of complex spatio-temporal objects,
2. a visual programming environment and user interface specifically designed for global change studies,
3. an encapsulation of the global change research process which may be generalizable to other scientific disciplines, and

[3]Here we use the term process to refer to its general definition as understood in the scientific community and not necessarily as perceived in the field of Computer Science.

4. most importantly, a metadata manager for the management of scientific experiments and procedures, providing the capabilities of data sharing, reproducibility of experiments and capturing the semantics of derived data.

The architecture of the Gaea prototype is shown in Figure 1. The system is divided into two parts: the visual environment and the Gaea kernel which interfaces with the Postgres DBMS. The visual environment is presented and discussed in [40]. We concentrate on the description of the Gaea kernel, specifically the portion of the metadata manager that involves the management of the semantics of derived data.

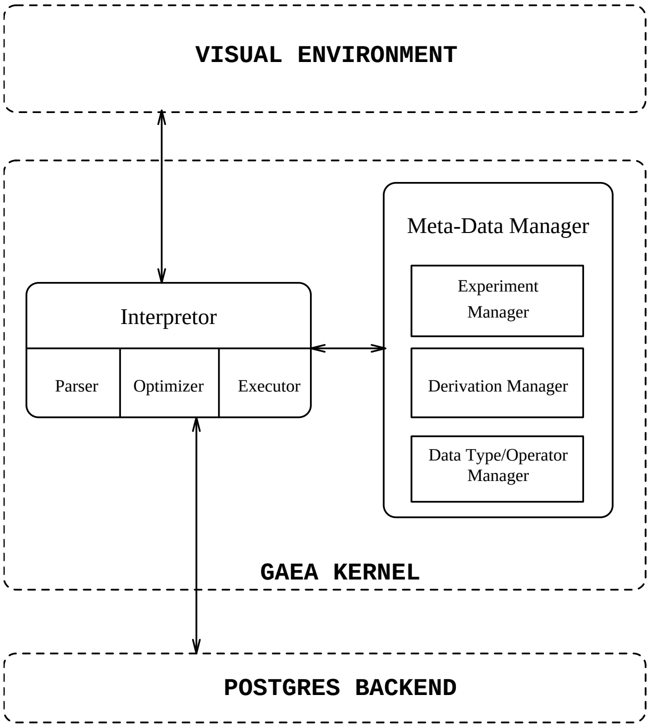


Figure 1: Gaea System Architecture

2.1 The Three Semantic Layers

We view scientists as manipulating objects following orthogonal extents. For example, in global change studies, objects have spatial as well as temporal extents. Although these two extents may be correlated, scientists retrieve and manipulate "scientific objects" by viewing those extents as orthogonal. The semantics of the spatial dimension [12, 13, 16, 27] as well as those of time [1, 24, 33, 35, 37] have been the subject of much research over the last decade. A third extent that has not received much attention so far is the data derivation extent, which deals with the derivation procedure followed in the generation of new or existing complex objects.

Metadata management in Gaea is based on the extension of the object-oriented technology with the following constructs: concept, process, and task. Those concepts are introduced within our discussion of the meta-data manager of Gaea, which consists of three layers: the low level data types/operators manager, the high level experiments manager, and the "liaison layer," which is the derivation semantics manager (Figures 1 and 2).

2.1.1 High Level Semantics, or the Experiment Level

This level records the information that is necessary for the understanding of 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 metadata manager. The experiment manager is capable of manipulating conventional semantic modeling constructs [20]. In addition, we introduce the notion of *concepts*, which may either be base data or data derived from other data according to any of several well-defined algorithms.

A general definition of a **concept** is a representation of a spatio-temporal entity set, extended with an imprecise definition. Concepts are very common in scientific databases. In the context of geographical information systems and global change research, one can effectively cite many examples of concepts.

PERSON is an entity set as defined by the ER model [6], and may be considered a concept with a well defined and agreed upon meaning. But can we define what a DESERT or DESERTIC REGION is? According to [5], an acceptable definition of a desert must include consideration of the following factors: the amount of precipitation received, the distribution of this precipitation 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: for example dryness, related to precipitation, can be measured by the Aridity Index, a Quotient of Dryness or the Radiational Index of Dryness [5]. So a DESERTIC REGION is an entity set whose definition may differ from one user to another.

In the above example, definitions of concepts inherently cover the spatial as well as temporal dimensions. Other similar examples can be drawn on the concepts of CLOUDS or CLOUD COVERAGE. Another well known imprecise entity is the concept of watersheds [39].

In semantic modeling [20, 28], concepts are modeled as derived entities where the derivation rules are uniquely specified and agreed upon. That is not the case in our application domain. We regard concepts as entities that "inherently" mean the same to any user when considered at the highest level of abstraction, but whose derivations may (and probably will) differ from one user to another.

At this high level of abstraction, we model deserts with a specialization hierarchy[4] as in 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 the well proven semantic modeling technology, semantics for data derivation are necessary. For example, hot trade-wind deserts refer to areas of high pressure with rainfall less than 250 mm/year, while ice or snow deserts refer to polar lands such as Greenland and Antarctica [26]. Essentially, class/subclass hierarchies are not sufficient to provide a clear view of the interrelationships of these "concepts." While inheritance can be used, specialization attributes are derived according to a specific scientific procedure, referred to as a "process," which provides the basis for the derivation semantics layer described next.

Although we have been treating "concepts" informally at this point, we note that it is possible to formalize this notion. Jumping ahead a bit, each type of base data and each process for deriving data defines a unique class; a concept is simply a set of classes. It is possible to create silly concepts, such as the union of the CLOUD and CENSUS classes, but we leave it to the user to avoid such.

2.1.2 Derivation semantics level

Once a full concept structure is developed within the high level semantic layer, the leaves of such a structure are mapped to a set of non-primitive classes in the derivation semantics layer. This is shown in Figure 2 as the dashed lines expanding the concept of "hot trade-wind desert" to the set

[4]Hierarchies can be general directed acyclic graph structures.

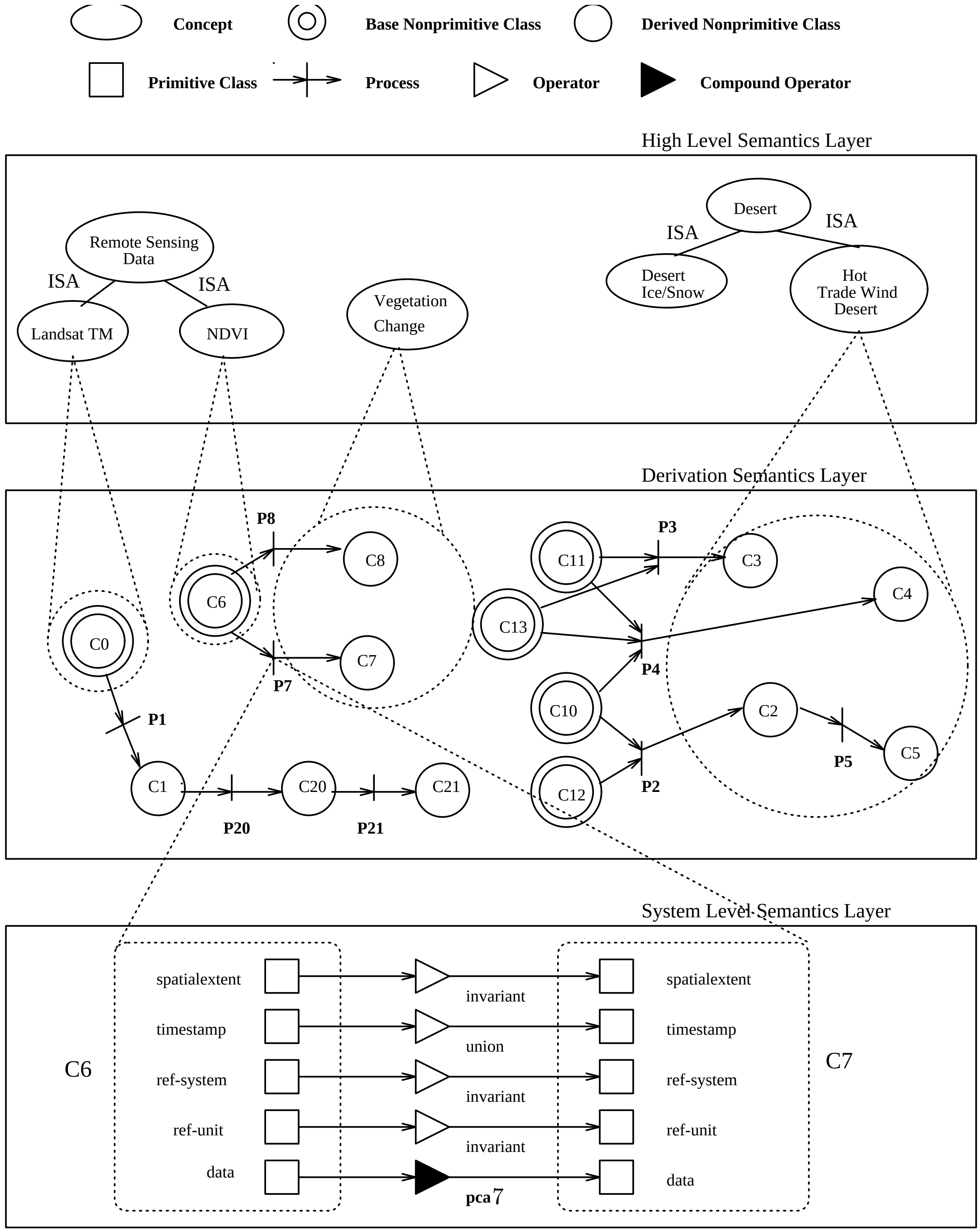


Figure 2: The three semantic layers in Gaea

of (non-primitive) classes {C2, C3, C4, C5}. Another example is shown as the concept NDVI mapping to the class set {C6} and Vegetation Change Mapping to the set of classes {C7,C8}.

In our implementation, concepts are represented by a set of non-primitive classes encapsulated with automatically defined (retrieval) functions on their attributes. For example, the structure of a nonprimitive class `landcover` is defined in Gaea as:

```
CLASS landcover (          // Land cover
  ATTRIBUTES:
    area = char16;         // area name
    ref_system = char16; // long/lat, UTM ...
    ref_unit = char16;     // meter, degree ...
    cell_x = float4;       // pixel size in x
    cell_y = float4;       // and y directions
    resolution = float4;
    data = image;          // image data type
  SPATIAL EXTENT:
    spatialextent = box; // bounding box
  TEMPORAL EXTENT:
    timestamp = abstime; // absolute time
  DERIVED BY: unsupervised-classification
)
```

The retrieval functions such as `area(landcover)` and `timestamp(landcover)` are automatically defined. (In most relational systems, such functions are specified with dot notation, such as `area.landcover`).

The derivation semantics layer records the derivation relationship among classes of data. Such relationships can also be used for the generation of new data objects in a class. 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. **Process**: represents the description of a scientific procedure used for the generation of new concepts from other concepts.

2. **Task**: The instantiation of a process with input data objects is called a task. Every task will generate a set of objects (most of the time just one) for the output class.

Formally, a process defines a mapping between a set of input object classes and an output object class. Essentially, 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 process. In this way a process captures the semantics of data derivations.

One can specify a process to be primitive or compound. A compound process is a network of intercommunicating processes. A primitive process is one that cannot be decomposed into a structure of other intercommunicating processes. It is composed of a network of basic operators that define the transformation from the input classes into the output class. Operators are part of the ADT facility in the low level semantics layer and operate on individual primitive classes (refer to Section 2.1.3).

In Figure 2, C2, which is a member of the concept "hot trade-wind desert," is derived using process P2, while process P5 derives C5. P5 might be an interpolation process which derives the same concept from itself, but using class C2.

Let us consider a database consisting of Landsat TM satellite imagery of the earth as base data. One can define the derivation of land-cover classification as a grouping of the remotely sensed data into land cover classes based on their similarity. The input data class could be remotely sensed and rectified Landsat TM data and the output class is LAND_COVER. This process is illustrated in Figure 3 and represents P20 in Figure 2. This example shows a definition of a process as consisting of:

1. A process name to identify the process.

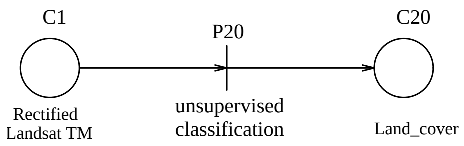
2. An output (non-primitive) class: a derived non-primitive class is defined uniquely by the outcome of a process. The user needs to supply the output class name. The structure of the ouput class is captured in the class definition from the mapping provided in the TEMPLATE section of the process definition.

3. Arguments: the set of input classes from which the output class is derived. The attributes of these classes are part of the mapping definitions in the TEMPLATE.

4. TEMPLATE: this is the part that defines the input to output mapping between the attributes of the classes involved in the process. It consists of a set of ASSERTIONS and the actual MAPPINGS. Assertions are conditions on the input classes. They correspond to constraints and

guard rules which need to hold before a process can be applied. Mappings are the transfer functions that are used to derive the attributes of the output class from the attributes of the input classes.

In the example of Figure 3, class C20 has four attributes: the spatial extent `C20.spatialextent`, the temporal extent `C20.timestamp`, the number of land cover classes `C20.numclass`, and raster image data `C20.data`. The extents are invariantly transferred from the input classes, while the image data is derived using the functional application of the image operators: `unsuperclassify()` and `composite()`[10]. The assertions using the rule `common()` make sure that the spatio-temporal extents of the input classes are the same or overlap.




```
DEFINE PROCESS    P20
OUTPUT   C20
ARGUMENT (  bands  SETOF   C1  )
TEMPLATE {
  ASSERTIONS:
        card ( bands ) = 3;       // need three bands
        common ( bands.timestamp );
        common ( bands.spatialextent );
  MAPPINGS:
        C20.spatialextent = ANYOF bands.spatialextent;
        C20.timestamp = ANYOF bands.timestamp;
        C20.numclass = 12;
        C20.data = unsuperclassify ( composite ( bands ),  12 );
}
```

Figure 3: Derivation Process for Unsupervised Classification

In this process the significant operation is applied just on the image itself. Other attributes such as `timestamp` and `spatialextent` are transferred invariantly. This is typical of most processes defined in Gaea. For most cases, the spatial extent will be the same. An example of a concept whose spatio-temporal extents do not map invariantly can be built around cloud coverage, hurricane or storm movements, or principle component analysis for change detection as described in Section 2.1.3.

A simple example of a task is the derivation of the land use classification for January 1986 for Africa. This involves a query on the LAND_COVER class, which translates into a conventional retrieval if the data have been precomputed; or into the retrieval of the proper Landsat TM spatio-temporal objects, followed by the application of the unsupervised classification process (P20). The actual mechanism used for such a query is briefly discussed in Section 2.1.5

It should be noticed that different users may use the same derivation method but with different parameters. For example, one scientist may choose to derive a desertic region based on rainfall less than 250mm, while another one choses 200mm for the same parameter. We make the assumption that the same derivation method with different parameters represents different processes.

2.1.3 System Level or Low Level Semantics

The system level semantics of Gaea is responsible for the management of abstract data types (ADT). Following the object-oriented paradigm [2], ADTs in Gaea are primitive classes encapsulated with the methods or functions applicable to them. In primitive classes, data objects are value identified, i.e., the object identifier for a data object is its value. Examples of primitive classes are the integer, float, string and boolean class. Changing the value of an object in a primitive class will always lead to another object. In our prototype, the low level semantics are handled by the ADT facility in Postgres [38]. For example, the primitive class `image` is defined in Gaea as:

```
  External Representation:
     : "(nrows, nclos, pixtype, filepath)"

Internal Representation:
  struct {
    int  nrow, ncol;    // # of rows & columns
    char pixtype[16];   // pixel data type
    char filepath[128]; // full path name
            }
```

Where `nrows` represents the numbers of rows and `ncols` represents the numbers of columns in the image, `pixtype` is “char,” int2,” “int4,” “float4,” "float8;" `filepath` is the absolute path of the file that stores the actual image data.

Following Postgres, functions on primitive classes are called operators. They specify the different methods that are applicable to the primitive classes. Example operators on the image primitive class are:

```
img_nrow(image); // return # of rows
img_ncol(image); // return # of columns
img_type(image); // return a pixel's data type
img_filepath(image); // return the file name
                     // which stores the data
img_size_eq(img1, img2);// check if 2 image
                        // sizes are equal
```

Other, more specialized, analysis operators are shown within the description of process P7 in Figure 2.

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, "vegetation change" can be derived as either class C7 or C8. Consider C7 as derived using principle component analysis (PCA) [31] which is part of process P7. The mapping between input and output attributes is shown in the lower portion of Figure 2. It is observed that the operator `pca()` is a compound operator. It is composed of a network of intercommunicating operators, whose structure is illustrated in Figure 4. This network can be seen as a data flow network of functional operators that are applied on primitive classes, such as spatial coordinates, temporal attributes, and raster images similar to the primitive image defined above. A variation of this procedure, called standardized PCA (SPCA), was described by Eastman [9]. In that paper, "vegetation change" was derived using SPCA and compared to the "same conceptual outcome" provided by PCA. This experiment was conducted using the IDRISI system through the manipulation of raster images. Using IDRISI, it is very difficult to duplicate the experiment unless the user specifically knows the procedure used and the operators applied. In the Gaea system, such an experiment can be reproduced once the derivation procedures are captured in the derivation semantics layer.

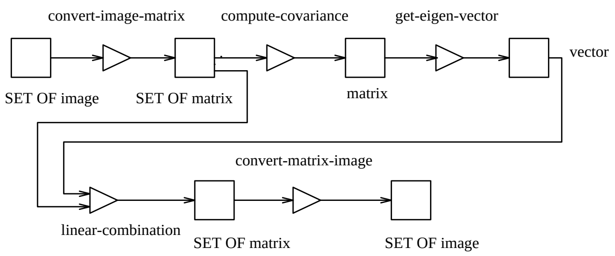


Figure 4: Principle component analysis

2.1.4 Observations and Clarifications

Several things need to be clarified here:

1. A process can be thought of as an aggregation of one or more operators. However, an operator itself is not a process (remember that a process is defined among non-primitive classes and an operator is defined on a primitive class).

2. A compound process can be defined with other processes. One such example might be land change detection, as shown in Figure 5. A compound process is merely an abstraction which can be used to simplify a derivation relationship between object classes. Thus a compound process cannot be directly applied, but must be expanded into its primitive processes before actual derivation takes place.

3. A new process may be defined by editing an old process by the addition, deletion, or modification of operators. In no case is the old process overwritten.

In summary, processes and operators are not at the same level. In the Gaea system, operators encapsulate primitive classes and are managed in the system level. The process level manages the derivation semantics of concepts represented by a set of non-primitive classes. Concepts themselves are considered part of the high level semantics layer, in which experiments are managed.

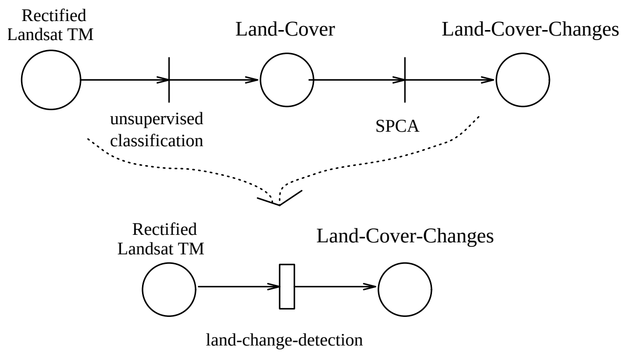


Figure 5: Compound Process: Land-Change Detection

2.1.5 Basic Functionality of the Metadata Manager

The three levels of the metadata manager are accessible to the user according to the needs and requirements of the application. For example:

1. Queries on concepts and, in the future, encompassing experiments, are handled through the high level semantics layer. At this level, the user will select and query reproducible or precomputed instances of experiments. In the current version of Gaea, only concepts are captured in the high level semantics layer.

2. Users may interact with the derivation level by choosing the processes applied to a set of concepts. This enables users to study the meaning and compare instances of concepts according to their derivation procedures.

3. The low level semantics layer provides for the extensibility of the DBMS. Currently, we rely on the features provided by Postgres through its ADT facility. Newly designed operators which apply to primitive classes can be used in the design of new derivation processes and the establishment of new experiments. Furthermore, as shown in Figure 4, operators can be combined into a self-contained compound operator that can be applied as a primitive mapping function between two primitive classes.

The availability of derivation relationships extends the functionality of database queries. The execution of a database query which involves the retrieval of a derived spatio-temporal concept is performed according to the following sequence:

1. Direct data retrieval from the non-primitive classes corresponding to the concept of interest.

2. Data interpolation (temporal or spatial). Interpolation can be used in many situations where data are missing. It is a generic derivation process which is applicable to many data types in many domains.

3. Data are computed, based on a derivation relationship.

Steps 2 and 3 are prioritized according to the user's needs.
The derivation relationship is expressed at two levels:

1. Class level: The class level derivation is expressed as a process. It is a template that shows how the new data will be generated. In some sense, a process plays a similar role to that of a view definition in relational databases.

2. Data object level: The data object level derivation will record the actual derivation relationship among data objects. It is represented as a task, which is recorded as a relationship among instances of non-primitive classes.

2.1.6 Modeling Class Derivation with Petri Nets

Petri Nets (PN)[29] have been used in various applications as a formalism for system modeling and analysis. Some of the application areas of PNs are in performance analysis [19], asynchronous systems modeling [30], and hardware modeling [3, 17]. In a different context, PNs have also been suggested as a tool to model a database system' s behavior [21, 25, 34].

PNs provide a useful framework for describing the derivation relationship among classes. Based on their inferencing capability, PNs may be used for the modeling of class derivation in scientific databases as follows: Every non-primitive class, which is a member of a concept, corresponds to a place in a PN, and every process corresponds to a transition. Tokens in every place represent the data objects needed for the instantiation of a process, i.e., the completion of a task.

Based on the PN representation, we can apply reachability analysis on the network to decide if a non-existing object could be derived from existing data. Some modifications are needed to adapt PNs to our application:

1. In a traditional PN, when a transition is fired, the tokens at the input places will be removed. In our system this is not the case, as tokens (data objects) used for derivation are permanent and can be reused if necessary. Although such a situation can be expressed by making sure that transitions output to the same input places, we simplify the PN by modifying the execution rules so that tokens are not removed from input places upon the firing of a transition.

2. The number of inputs to a transition denotes the minimum number of tokens needed to enable the transition. When a transition is fired, more tokens than the threshold may be used. For example, for principle components analysis, two input data images are enough, but more than two images are usually used.

3. In order to guarantee the integrity of data derivation, some form of relationship may be required among the input data objects (tokens). For example, the same or overlapping spatial coverage may be necessary. This can be expressed in the template of a process as constraint rules and assertions. Only when such relationships are satisfied, will the transition be enabled and fired.

We can apply the following recursive mechanism to retrieve the data needed:

1. Attempt to retrieve the data from the target class. If it exists, return;

2. Else back propagate the requirements through the derivation net and apply this procedure to the input class(es) of the derivation process. If input data are available, fire the process to generate the needed data; otherwise repeat this step.

3. The procedure is recursively applied until the needed data are generated or back propagation stops at some base class and we fail to find the needed data.

Using PNs, the above procedure can be formulated as: given a final marking, try to find the initial marking which can lead to this marking. This initial marking will identify the specific data objects that can be retrieved directly from the database.

3 Relationships to Other Work

In this Section, we review other proposed mechanisms that relate to our work, and make some comparisons.

3.1 Related Work in E-R Modeling

Markowitz [22] uses the extended E-R approach to model both the functional and structural components of an information system. The basic idea is to represent a process as a relationship and apply existential constraints to express the partial order implied in a process. However, we do not believe that the E-R approach is sufficient to represent derivation relationships among data classes for the following reasons:

1. An E-R diagram is basically a network structure, while the derivation relationship actually defines a hierarchical structure among data classes, which is not obvious in an E-R diagram representation.

2. Derivation relationships are different from other kinds of relationships in an E-R model. The input data classes and output class of a derivation relationship cannot be directly mapped into the E-R model. Furthermore, the constraints involved in a derivation relationship cannot be expressed in the E-R model.

3. Compared with the E-R diagram, the PN we propose to use expresses more semantics for a derivation relationship. It shows not only the input and output classes but also the constraints on a derivation procedure. Those constraints are in the form of guard rules that need to be satisfied for a derivation to be applicable. We have briefly shown how PNs can also be used to generate derived data automatically. Furthermore, PNs can be used to capture the control flow of the scientific computation on hand.

3.2 Derivation Management vs. Functional Modeling

One may find similarities between our work and functional modeling in the system analysis stage of business database applications. However they are different in their purpose and the methods used.

Usually an information system is described by two components: structure and function. In the structural component, entities and their relationships are identified. This is also called a static view of the database and forms the basis for schema definition. The dynamic view (behavior) of the database is described in the functional component, which forms the basis for application programs.

One popular method for functional modeling is Data Flow Analysis [23]. In data flow analysis, an information system is considered as a process that maps input data to output data, and can be represented as a data flow diagram. Then the transformation process is further decomposed into subprocesses until each is basic enough to be implemented with a piece of simple program.

Although functional analysis is also concerned with a process, the purpose is different from that of derivation management in scientific databases. A process in functional analysis is used to develop application programs, while a process in our work is used to define derivation relationships among data classes. In addition, a task, the instantiation of a process, is of no interest in functional analysis, while it plays an important role in data derivation management. It is an individual task that defines the derivation relationship among a set of data objects.

In summary, functional analysis is concerned with how to transform input data to output data, i.e., how to accomplish the task; while data derivation management is concerned with how the data were and will be generated, i.e., how the task was and will be accomplished.

3.3 Related Research in Scientific Databases

Experiment management is also the goal in [7]. The problem is to model experiments in computational chemistry, and the approach followed is based on the object-oriented paradigm. Cushing *et. al.* derived a model that captures the interrelationships between the data, its source, methods and instruments used, and other information relevant to the generation of the data. They provide a mechanism for managing the definition, preparation, monitoring and interpretation of computational experiments. We address the same problem, but identify differences between experiment management and data derivation management. By using different formalisms to model them, we have introduced more semantics into our system.

Semantic networks are an appropriate tool to capture the relationships among a set of data objects. This formalism has been used in the USD system [32]. Although their intention was to make use of the flexibility of semantic networks to represent unstructured data, it can also be adequately used to model an experiment. The problem with semantic networks is that they might become too complex with a large database system. In addition, data derivation relationships are not explicitly represented in the network.

4 A Critique of the Model

4.1 Limitations of Current Systems

IDRISI [11] and GRASS [36] are two GIS systems used for global change analysis. Both systems are primarily file-based, raster-oriented systems, although vector and scalar data can be manipulated. A typical working scenario for either system is to perform analysis with sequences of commands that read data from input files and store results into output files. The shortcomings of such a working environment are apparent:

1. A file name is the only identifier for stored data. As a result, a user has to name the file appropriately in order to recall it later, which is impractical when there are a lot of data. Many other problems also arise for such schemes, such as inadequacy for range retrieval, inadvertent file overwrite by other users, etc. Essentially, standard database management features are not provided.

2. Data sharing is almost impossible because there is not enough meta information to describe how the data are generated. (How can one deduce it from a file name?)

3. Scientists have to manage the analysis process on their own, including the commands used and intermediate data generated. This often takes the form of awkward transcript files.

4. It is hard to create abstractions of the analysis process. When a procedure needs to be applied to multiple data sets, the same steps have to be repeated manually.

4.2 Features in Gaea

The Gaea system overcomes the above problems by providing database support and metadata management. Specifically, all data in Gaea are stored in the database, thus data can be retrieved according to their descriptions. Furthermore, Gaea manages three levels of metadata; the experiment (concept), data derivation, and system levels. This liberates the user from the burden of managing the analysis process and makes data sharing possible. The analysis of data and its management are integrated into the same environment.

The main advantages of metadata management in Gaea are:

1. System level: All the primitive classes and their operators are managed in a hierarchical structure. Users can browse the hierarchy, look up appropriate operators for specific primitive classes, or find the primitive classes that have a specific operator.

   Users are allowed to define new primitive classes and/or new operators. This makes the Gaea system an extensible system.

2. Data Derivation Level: The use of a derivation process helps in understanding the semantics of derived data. Reuse of previously defined analysis processes is possible.Users can automatically derive data not stored in the database, either through interpolation or derivation.

3. Experiment Level: General concepts, such as desertic regions, can be described. Experiments can be reproduced, allowing rapid and reliable confirmation of results. Information exchange among scientists can be promoted.

4.3 Limitations

It is important to pinpoint the major limitations of our system.

1. At this time, non-primitive classes can only be composed of primitive classes as provided within POSTGRES. Although current scenarios for global change research do not require the support of non-primitive classes as attributes, future applications may require this feature.

2. Another problem is that interaction cannot be specified in the process definition. They are many situations in global change analysis that require the user to conduct the analysis process based on the intermediate result [11]. One can envision a procedure followed by a scientist which demands the specification or modification of input parameters based on some temporary result visualized on the screen. A typical example is supervised classification [11]. This process requires interaction with the scientist before a task completes the derivation of the output land cover classification data. We have not yet developed methods to express such interactions in a process.

We will address these limitations in future extensions of the Gaea system.

5 Conclusion and Future Work

We presented a framework for the management of data derivation relationships so that data can be shared in scientific databases. The main contributions include:

- A three layered view of the metadata manager of Gaea, specifically designed for global change studies. Those layers are accessible to the user according to the complexity of the study being performed.

- Three new semantic constructs: concept, process and task. Concepts are used to capture entity sets with imprecise definitions. A process captures the derivation procedure of a specific object class, while a task is the instance representing the derivation of a specific scientific object.

- Derivation diagrams based on Petri Nets to model and capture the semantics of data derivation in scientific databases. Derivation diagrams can be used to 1) browse data following their derivation relationships, 2) compare derivation procedures and their resulting data classes, and 3) derive data not stored in the database.

The proposed framework has many potential long term extensions: Derivation diagrams provide a knowledge acquisition environment that can be used for learning and automated derivation of scientific data. Data derivation is currently captured as a mapping which is composed of operators which can be applied locally. The need to deal with processes that are not locally available will be essential in the future. Furthermore, a process may be in general non-applicative, that is a process may consist of a mapping which is described by experimental procedures that do not follow a well known algorithm. We are currently planning long term research to deal with those requirements.

Acknowledgments

The authors wish to thank Robert Dugan, Jr., Nina Serrao, Yelena Yogneva-Himmelberger, Yuhong Zhang, and Yu Zhou for many helpful conversations.

References

1. J.F. Allen, "Maintaining Knowledge about Temporal Intervals," CACM, Vol. 26, No. 11, pp. 832-843, Nov. 1983.

2. M. Atkinson, F. Bancilhon, D. DeWitt, D. Maier, and S. Zdonik, "The Object-Oriented Database System Manifesto," Proc. Int. Conf. on Deductive and Object-Oriented Databases, pp. 40–57, 1989.

3. J.-L. Baer, "Modeling Architectural Features With Petri-Nets," Lecture Notes in Computer Science, Springer-Verlag, No. 255, pp. 258-277, 1986.

4. A. Beller, "Spatial/Temporal Events in GIS," GIS/LIS 91, V57, N4, pp. 407–411, 1991.

5. G. Bender, "Reference Handbook on the Deserts of North America," Greenwood Press, p. 594, 1982.

6. P.P. Chen, "The Entity-Relationship Model: Towards a Unified View of Data," ACM Trans. on Database Systems, Vol. 1, No. 1, pp. 9–36, 1976.

7. J.B. Cushing, et. al. "Object-Oriented Database Support for Computational Chemistry," Proc. SSDM'92, pp. 58–76, 1992.

8. J. Dozier, "Access to data in NASA's Earth Observing System," Keynote Address, Proc. ACM SIGMOD Intern. Conf. on Management of Data, San Diego, 1992.

9. J.R. Eastman, "Time Series Map Analysis Using Standard Principle Components," ASPRS/ACSM/RT 92, pp. 195–205, 1992.

10. J.R. Eastman and J.McKendry, "Change and Time Series Analysis in GIS," UNITAR, 1991.

11. J.R. Eastman, "IDRISI – A Grid-Based Geographic Analysis System (User's Manual)," Clark University, Worcester, MA, Nov. 1990.

12. M. Egenhofer and J. Herring, "A Mathematical Framework for the Definition of Topological Relationships," Proc. 4th Intl. Symp. on Spatial Data Handling, pp. 803-813, Zurich, Switzerland, 1990.

13. A.U. Frank, "Properties of Geometric Data: Requirements for Spatial Methods," In Advances in Spatial Databases, 2nd Symp., SSD'91, Spring-Verlag, Zurich, Switzerland, Aug. 1991.

14. J.C. French, A.K. Jones, and J.L. Pfaltz, "Summary of the Final Report of the NSF Workshop on Scientific Database Management," SIGMOD Rec. **19**-4, pp. 32–40, 1990.

15. H.J. Genrich "Predicate/Transition Nets," Lecture Notes in Computer Science **254**, pp. 207–247 , Springer-Verlag, 1986.

16. O. Guenther and A. Buchmann, "Research Issues in Spatial Databases," SIGMOD Rec. **19**-4, pp. 61–68, 1990.

17. N.I. Hachem, "Petri-Net Driven Knowledge Base System for Automated Microcode Generation in VLSI," in Proc. of the IASTED Int. Symp. MODELING and SIMULATION, Calgary, Canada, July 1991.

18. N.I. Hachem, M.A. Gennert, and M.O. Ward, "A DBMS Architecture for Global Change Research," Proc. ISY Conf. on Earth and Space Science, Pasadena, CA, pp. 186–194, Feb. 1992.

19. M.A. Holliday and M.K. Vernon, "A Generalized Timed Petri Net Model for Performance Analysis," IEEE Trans. on Soft. Eng., Vol. 13, Dec. 1987.

20. R. Hull and R. King, "Semantic Database Modeling: Survey, Applications, and Research Issues," ACM Computing Surveys, Vol. 19, No. 3, pp. 201–260, 1987.

21. G. Kappel and M. Schrefl, "Object/Behavior Diagrams," Proc. of Intl. Conf. on Data Engineering, pp. 530–539, 1991.

22. V.M. Markowitz, "Representing Processes in the Extended Entity-Relationship Model," Proc. Intl. Conf. Data Engineering, pp. 103–110, 1990.

23. J. Martin and C. McClure, "Diagraming Techniques for Analysis and Programming," Prentice-Hall, New Jersey, 1985.

24. L.E. McKenzie, Jr. and R. Snodgrass, "Evaluation of Relational Algebras Incorporating the Time Dimension in Databases," ACM Computing Surveys, Vol. 23, No. 4, pp. 501-543, Dec. 1991.

25. T. Muck and G. Vinek, "Modeling Dynamic Constraints Using Augmented Place Transition Nets," Information Systems, **14**-4, pp. 327–340, 1989.

26. F.J. Monkhouse and J. Small, "A Dictionary of the Natural Environment," Halsted Press, p. 320, 1978.

27. B.C. Ooi., "Efficient Query processing in Geographic Information Systems," In Lecture Note in Computer Science, Spring-Verlag, 1990.

28. J. Peckham and F. Maryanski, "Semantic Data Models," ACM Computing Surveys, Vol. 20, No. 3, pp. 153–189.

29. J.L. Peterson, "Petri Net Theory and the Modeling of Systems," Prentice Hall, 1981.

30. C.V Ramamoorthy and G.S. Ho, "Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets," IEEE Trans. on Soft. Eng., Vol. 6, Sep. 1980.

31. J.A. Richards, "Multispectral Transformations of Image Data," Chapter 6 in Remote Sensing Digital Image Analysis, Springer-Verlag, pp 127–145, 1986.

32. R. R. Johnson, M. Goldner, M. Lee, K. McKay, R. Shectman, and J. Woodruff, "USD - A Database Management System for Scientific Research," Video presentation at the ACM SIGMOD Int. Conf. on Management of Data, San Diego, 1992.

33. K. Qiu, N.I. Hachem, M.O. Ward, and M.A. Gennert, "Providing Temporal Support in Data Base Management Systems for Global Change Research," Proc. SSDM '92, Switzerland, 1992.

34. H. Sakai, "A Method for Entity-Relationship Behavior Modeling," Entity-Relationship Approach to Software Engineering, North-Holland, pp. 111–129, 1983.

35. A. Segev and A. Shoshani, "Logical Modeling of Temporal Data, " Proc. ACM SIGMOD Conf., pp. 454–466, 1987.

36. M. Shapiro, et. al., "GRASS 4.0 Programmer's Manual (Draft)," U.S. Army Construction Engineering Research Laboratory, April 1992.

37. R. Snodgrass, I. Ahn, "Temporal Databases," IEEE Computer, Vol. 19, No. 9, pp. 35–42, Sept. 1986.

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

39. L. Vincent and P. Soille, "Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations," IEEE Trans. on PAMI, Vol. 13, No. 6, pp. 583–598, 1991.

40. Y. Zhang, M.O. Ward, N.I. Hachem, and M.A. Gennert, "A Visual Programming Environment for Supporting Scientific Data Analysis," Proc. of the Int. Workshop on Visual Prog. Languages, August 1993. (extended version as WPI-CS-TR 93-01, March 1993)

41. Y. Zhou, M. Gennert, N. Hachem, and M. Ward, "Requirements of a Database Management System for Global Change Studies," ASPRS/ACSM/RT 92, pp. 186–194, 1992.