NCGIA

National Center for Geographic Information and Analysis

A Conceptual Framework for Integrated Metadata Management in Very Large Spatial Databases

By

Nehal Trivedi
Unisys Corporation

Terence R. Smith
University of California, Santa Barbara

Technical Report 91-2

February 1991

Simonett Center for Spatial Analysis
University of California
35 10 Phelps Hall
Santa Barbara, CA 93106-4060
Office (805) 893-8224
Fax (805) 893-8617
ncgia@ncgia.ucsb.edu

State University of New York
301 Wilkeson Quad, Box 610023
Buffalo NY 14261-0001
Office (716) 645-2545
Fax (716) 645-5957
ncgia@ubvms.cc.buffalo.edu

University of Maine
348 Boardman Hall
Orono ME 04469-5711
Office (207) 581-2149
Fax (207) 581-2206
ncgia@spatial.maine.edu

Abstract

A conceptual framework for integrated metadata management in large spatial databases is described. The primary function of this framework is to allow definition, location and control of metalevel information about the underlying database. The framework provides for a set of core metadata components and allows for addition of any auxiliary metadata that the user might want to define. The framework would support feature based retrieval as well as interactive browsing of metadata. The emphasis is on flexibility, extensibility and ease-of-use. The goal is integrated management of all kinds of metadata. The report gives an overview of semantic modelling of spatial data followed by a conceptual model for metadata. The basic tenet behind the conceptual model is classifying the database entities of interest into' data, process and environment entities. Corresponding to this, the metadata consists of metadata, metaprocess and metaenvironment entities. We then propose a forms mechanism to manage metadata. A set of basic operations for manipulating forms and catalogs is described. We present a case-study of metadata in a conventional Geographic Information System (GIS) environment. This is supported by the preliminary version of the schema for the Condor Database Project at the University of California at Santa Barbara.

1 Introduction

1.1 Very Large Spatial Databases (VLSDBs)

Very Large Spatial Databases (VLSDBs) axe systems for the storage, retrieval, display, manipulation and analysis of spatial data in very large quantities. Spatial data is typically data that can be referenced by spatial attributes such as geographic coordinates. VLSDBs have all the chaxacteristics of generic spatial databases. A detailed discussion of the properties and nature of spatial data and spatial data management systems can be found in spatial database and GIS literature [ARM90, BIL89, EGE87, EGE89, FRA88b, G0089, MCK84, ROE90]. A geographic information system (GIS) is a leading example of a spatial database. Spatial database operations typicaRy involve integration and management of spatial data acquired at different times, in different formats, with varying degrees of error from various sources.

We believe that following characteristics of VLSDBs justify the need for metadata for better data management facilities:

- Very large size in terms of sheer volume and number of objects, where very large means more than can be efficiently handled by a conventional system
- Spatial data in raster, vector or other forms
- Spatial (and perhaps temporal) indexing
- Complex schema: many different types of spatial and aspatial objects, relationships and attributes
- Multiple heterogeneous datasets
- Computationally intensive query and manipulation operations.
- Variety of input and output devices
- Large variety in data quality, source and authority information.
- Dynamic updating of the database (arrival of the new data, archiving of old data and relating new data to the old data)

1.2 The Data Management Problem

As shown in the figure 1, three dimensions to the problem of managing spatial data can be identified: spatial data management, representation and access and spatial manipulation and analysis. These three dimensions are not completely orthogonal. Most of the work in spatial databases has been in the areas of: 1) spatial data representation including spatial data structures, spatial indexes etc and 2) spatial data processing which covers manipulation and analysis of spatial data. As pointed out by the NSF workshop report on scientific database management [NSF90], major data management problems include handling increasing data volume, metadata management, integration of database facilities with applications, finding data, access policy and ease of use. Large spatial databases can benefit from the use of better data models, data administration facilities and tools for schema representation and manipulation. The NCGIA[1] workshop on VLSDBs held in Santa Barbara in 1988 identified the need for explicit metadata management in VLSDBs. This workshop also discussed the need for adoption of object-oriented techniques for representing and managing multicomponent spatial objects.

[1] National Center for Geographic Information and Analysis

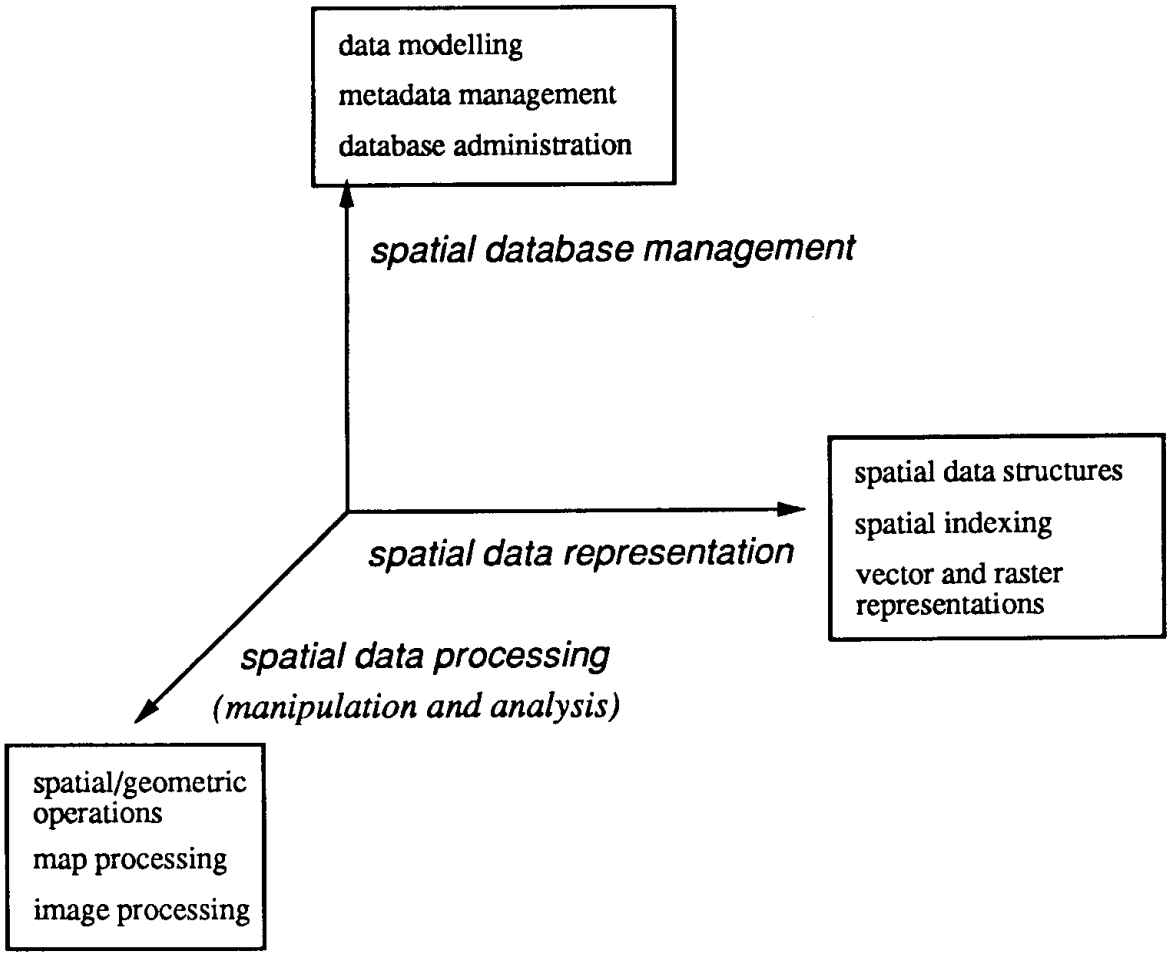


Figure 1: Spatial Data Handling

1.3 Problem Statement

Metadata is currently a salient and controversial topic in database research. There is no accepted definition and it may be too broad a concept to be useful. Given the current interest in the topic, however it is an important issue for investigation.

Conventionally metadata has been defined in a relatively narrow sense insofar as it is limited to the information in the data dictionary. In the context of VLSDB applications, metadata, is best viewed as a model of both the structure and contents of the database which may be used by the user and the system. Metadata can be used in all stages of query processing Le query formulation and validation, query plan formation and compilation, query optimization, data access and product (query-result) generation. In addition, metadata, can also be used to help in conventional data management functions such as requirements analysis, transaction analysis and monitoring physical storage e.g. data clustering and data formats.

This report studies the role of metadata in the management of VLSDBs. In the context of non- conventional database systems like VLSDBs, metadata is a vague term that can potentially include a wide range of information. There is no existing framework for uniformly describing different types of metadata, and for integrated metadata mangement. Hence we propose a framework to model and manage metadata. This framework is an extension of the data dictionary system as conventionally used for metadata management. The primary function of this framework is to allow the user to define, locate and control meta-level information about the underlying database.

We start by providing a framework for conceptual modelling of spatial data, followed by a description of a flexible and extensible metadata model. Such a framework provides the user with a set of core metadata components and allows the user to add

any kind of auxiliary metadata, as required. This framework also allows the user two forms of metadata, access: interactive browsing and feature based retrieval.

We then propose a forms based system for managing metadata. A forms based metadata management system would provide a flexible, extensible and easy to use environment for manipulating any kind of metalevel information by any category of end-user, including programmers, casual end-users and database administrators. This is in contrast to the rigid framework and limited functionality provided by commercial data dictionary systems. While describing the data modeling framework or metadata, management we have not concentrated on formal representations, completeness and minimality. Instead we have emphasized flexibility, extensibility and ease of understanding. The primary goal is to provide a conceptual basis for uniform and integrated management of all kinds of metadata.

1.4 Related Work

There is very limited literature that refers to metadata management in the context of VLSDBs. Most of the literature referring to metadata relates to the work done in the areas of relational data dictionaries and information resource dictionary systems (IRDS). The basic premise for most of such work involves the definition of multiple levels of abstraction with distinctions between data and metadata and schema and metaschema. The higher level is an intension of the level below while the level below is an extension of the level above.

1. Relational Data Dictionaries

Much of the work on metadata management in relational database systems is concerned with the use of data dictionaries to handle metadata. [NAR88] and [LE082] provide a useful insight into the implementation and use of data dictionaries in commercial database systems. [MAR86] describes metadata management in self-describing relational databases using active data dictionaries. For a detailed description of self-describing databases and associated metadata management, see [MAR85, MAR86]. The reference model for the ANSI/SPARC DBMS standardization [DAF85] contains a model for a data dictionary. [MAR85, MAR86, DAF85] all describe a model of data dictionary based on two orthogonal dimensions of data description: point-of-view dimension and intension-extension dimension. [DAV88a] describes an enhanced data dictionary model that uses the ANSI/SPARC DBMS model as a point of reference. [CAM86] presents an intelligent information dictionary system which concentrates on those aspects of data semantics that are not explicitly represented in the schema. Another example of an intelligent information dictionary system is provided by [RUS83]. [CUN83] describes the applications of the ER techniques to data administration including metadata management.

2. Information Resource Dictionary Systems (IRDS)

Many of the information resource dictionaries use the ER approach. For understanding the role of data dictionaries in information resource management see [NAV86]. [SIB86] proposes an active and extensible information dictionary with four levels of data abstraction. [GOL85] describes the organization and functioning of an IRDS based on the ANSI IRDS standards, as does [WIN88].

2 Metadata

Two basic questions concerning metadata are: what is metadata and why do VLSDBS need metadata? The primary motivation for metadata is better data mangement. To explain the need for metadata in processing of complex spatial data, we discuss the role of metadata in data management, query processing and helping the user.

2.1 What is Metadata?

Metadata is the data about the entities in a database. It is systematic and typically deductive information about the content, organization and use of the data in the underlying database. It acts as a repository for data from the logical and physical levels of the database. Metadata plays an important role while managing information as a resource that is shared by many users at all levels in an enterprise. It can be retrieved, manipulated and displayed in various ways. A collection of related data when managed and controlled as a unit forms a metadata database or meta-database [LE082]. Typically metadata users would include the system, all end-users from the very naive to the quite experienced and the database administrator.

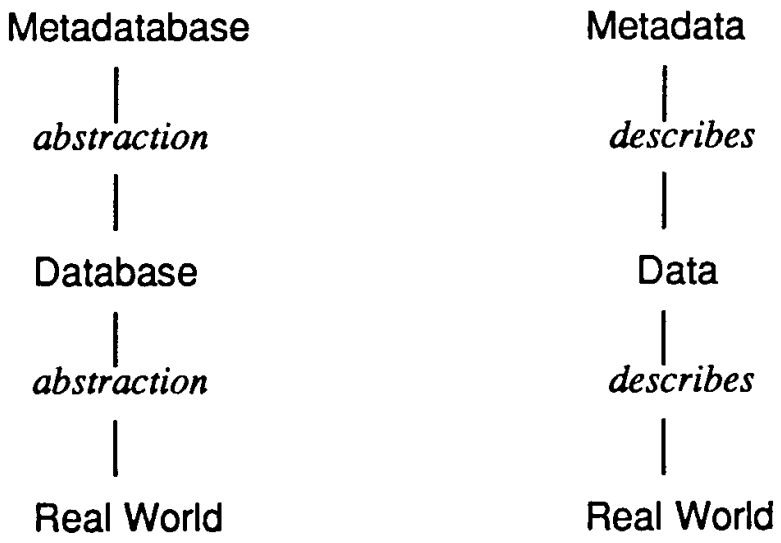
As illustrated in figure 2(a) the database provides an abstraction of the real world and the metadatabase provides an abstraction of the database Le at least one level of abstraction higher than the actual data used in the processing of the operational data. The database schema and the data model are a part of metadata since they provide metalevel. information about the data in the database. Figure 2(b) shows the basic functionality required from a metadata management system (MMS) for handling metadata [LE082].

There are two basic approaches to handling metadata:

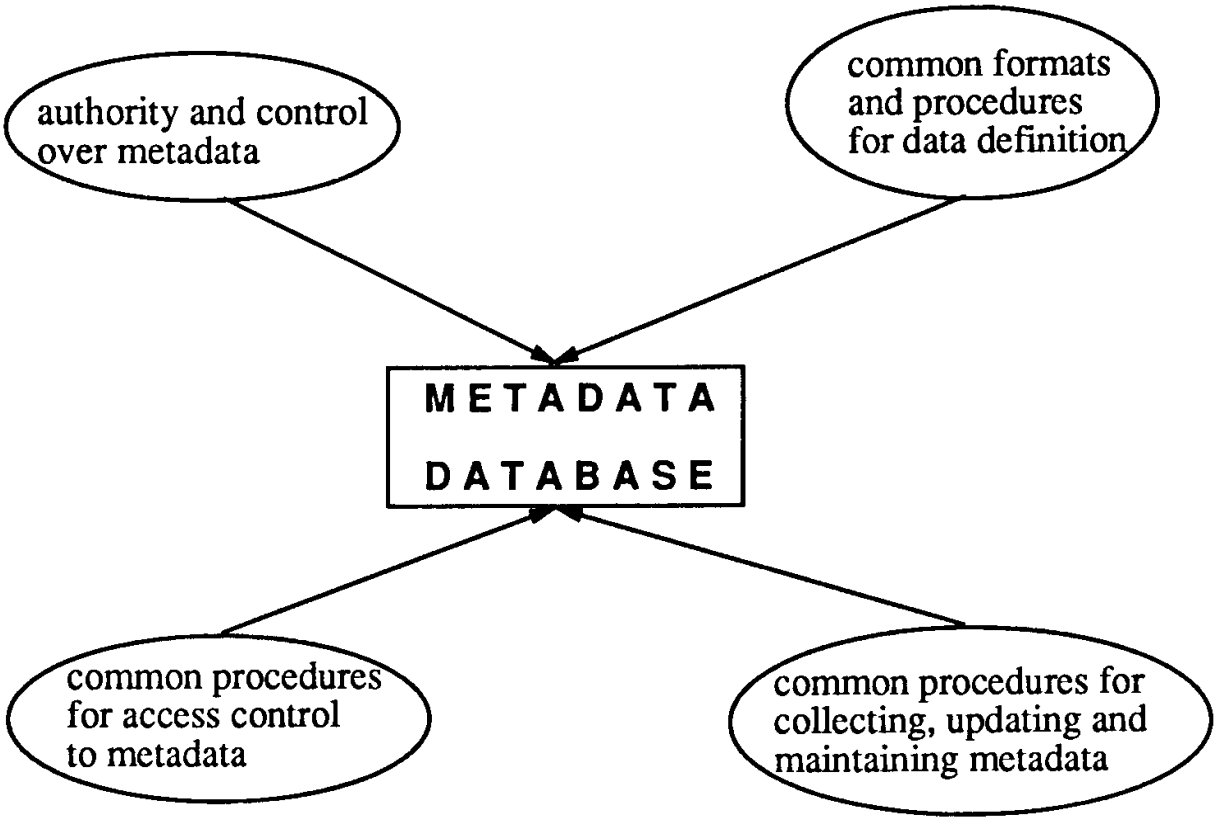
- *Self describing databases*: Metadata including the database schema, is represented using the data model used to represent the underlying data. Thus metadata is a part of the database.

- *Separate metadata layer*: Metadata resides in a deductive layer above the database and answers queries about the organization, structure and functionality of the underlying database.

The data dictionary and the data directory systems have been the conventional mechanisms to store and manage metadata. Much work has been done in the area of extending the ideas of what information can and should be handled by the data dictionary, how this can be done and how this information can be used.



a) Data, Metadata and The Real World



b) Functionality Required For Metadata Management

Figure 2: Metadata

2.2 Basic Metadata Queries

We classify the basic metadata queries into queries about entities relating to the data, processes and environment of a database management system.

The basic metadata queries about data entities include:
1) What is the data?
2) Where is the data?
3) What are the data characteristics?
4) Where did this data come from?
5) What is the quality (accuracy, lineage, resolution) of this data?

The basic metadata queries about process entities include:
1) What can one do with this data?
2) What happened when?
3) What types of spatial and geometric operators are available?
4) How can data be imported (acquired)?
5) How can entities like various data layers and spatial objects, be created, deleted and updated?
6) How can the results of analysis be presented?

The basic metadata, queries about environment entities include:
1) What does the primary and secondary storage profile look like?
2) What are the user profiles or who is who and who is doing what?
3) What are the characteristics of various physical devices?

2.3 Metadata Definition

Metadata is the data required to *describe, locate and control* data in a database. Metadata also includes data required to describe and locate process and environment entities in the database. It is the information that allows data identification and selection based on properties of data such as content, sources and quality.

Metadata maybe quite application independent and may also include any kind of descriptive information that the user wants to store. We divide metadata into core metadata and auxiliary metadata. The later would include any application- related or personal metadata that the end-user or the database administrator might need. Thus metadata, $M = M_c \cup Ma$ where M_c is the core metadata and M_a is the auxiliary metadata.

We can define the core metadata, as a 4-tuple:

M_c = (DD, DP, MP, ME)
where:
*DD = set of data defintions
*DP = set of spatial data properties
*MP = set of meta-process entities
*ME = set of meta-environment properties

The set of spatial data properties include:

1. **Theme**: descriptions, keywords, aliases
(What data is available? Is it useful?)

2. **Space**: geographic locations

3. **Time**: temporal validity and time related to:
data collection, data acquisition, major updates, product generation

4. **Location**: location in the search space, access paths
(Where is the data?)

5. **Quality**: thematic and positional accuracy
(How is the quality of data?)

6. **Sources**: source of data
(Where did the data come from?)

7. **Security**: authorization and access control
(Who is authorized to do what?)

8. **Miscellaneous**: scale, projection, format etc

The set of meta-process entities include:

1) pre-preprocessing history
2) data acquisition (digitizing or reformatting)
3) processing history
4) result presentation (report generation)
5) transactions

The set of meta-environment entities include:

1) System documentation
   (What are the operators/tools available?)
2) User profiles
   (Who is who and who is doing what?)
3) Physical device characteristics
   (What is the status of the physical devices)

2.4 Metadata and Data Management

There is a need for improved data management facilities in very large databases which handle large complex objects and involve heterogenous data. Very Large Spatial Databases (VLSDBs) fall in to this category. One of the main problems in such databases is schema manipulation and update as also in data modeling and schema manipulation. There are four categories of users whose data management requirements need to be satisfied: the system, end users, database administrators and the higher level managerial staff who rely on the end users and database administrators to satisfy their queries. Since we have a complex schema, typically large complex objects are stored, modified and manipulated. Attributes of these objects can have attributes. Semantic relationships between objects and their representations must be defined and maintained with respect to numerous intricate integrity constraints. Since the database schema is an intension into the underlying data, it is a part of metadata. Thus metadata will play an important role in data definition and description, conceptual data modeling and schema manipulation.

Navathe and Kerschberg [NAV86] point out the advantages of having metadata accessible to the user are almost self-evident. For example in a multi-database environment a user might simply want to know in which database or file a given lexical descriptor is meaningful. Comments are also important. The user should be able to refer to metadata for data definitions and descriptions. For an end-user a problematic situation is a large database, a complex schema and the need to formulate a set of complex queries. Frequently the users do not have a clear understanding of the data model, the application schema or even their own requirements. Under such circumstances formulating a query and trying to evaluate the end result can be difficult. A query based on a misconception of the data semantics may produce misleading results. In such cases error handling (diagnosis and correction) becomes complicated.

Metadata can also be important in processes such as preprocessing of spatial data and spatial data transfer between heterogeneous systems. The proposed standard for digital cartographic data is an example of how and why metadata is necessary for spatial data transfer between heterogeneous spatial database systems [DCD88]. For a database administrator metadata can be useful in keeping a track of the application (host-language) programs and long transactions as well as the physical devices, user profiles and usage information on various datasets.

2.5 Metadata and Query Processing

Metadata is required during various stages of query processing such as query formulation and validation, query plan generation and compilation, query optimization, data access and product (query result) generation. At the query formulation stage the

user need information about the database schema and about the different operations that are possible for manipulating the data. At the query optimization phase the system needs metadata in order to search efficiently for the query objects. Figure 3 illustrates the role of metadata in query processing.

For preparing reports, the report generator needs to use the stored data definition and stored report specification, both of which fall under metadata. For the process of constraint evaluation and validation, the system needs to know about constraint evaluators and validation routines and also needs to have access to integrity constraints. The metadata management system should be involved in such processes and also be active in the data access control process, since it has user profiles, authorization information and the access codes as well as the information about the data access control procedures.

3 Modeling Data

The data model can be defined as a mathematical formalism that includes a notation for describing data and a set of operations for manipulating the data [ULL88]. The data description includes the various data object types, their relationships and their attributes. A data model is an important part of an integrated metadata management system.

A variety of models are applicable for modeling spatial data including relaional, objectoriented and logic-based. It is widely believed that the relational model is insufficient for spatial applications [FRA88a, FRA88b, ROE90]. Object-oriented data models for spatial database applications can be found in current spatial database literature [EGE87, EGE89, LIP87, ROE90, ORE90]. In an object-oriented system, the world can be described as the user sees it conceptually, in the terms of high level objects and relationships between objects. The behavior of the objects is described in terms of the methods associated with the class to which the object belongs and the characteristics of the object are described as attributes that the object instance inherits from its class and that the object class partially or fully inherits from its superclass.

Here we present a conceptual framework for modeling spatial data. The model is based on the semantic data model and the idea is to build the background for a metadata model. The emphasis is on:

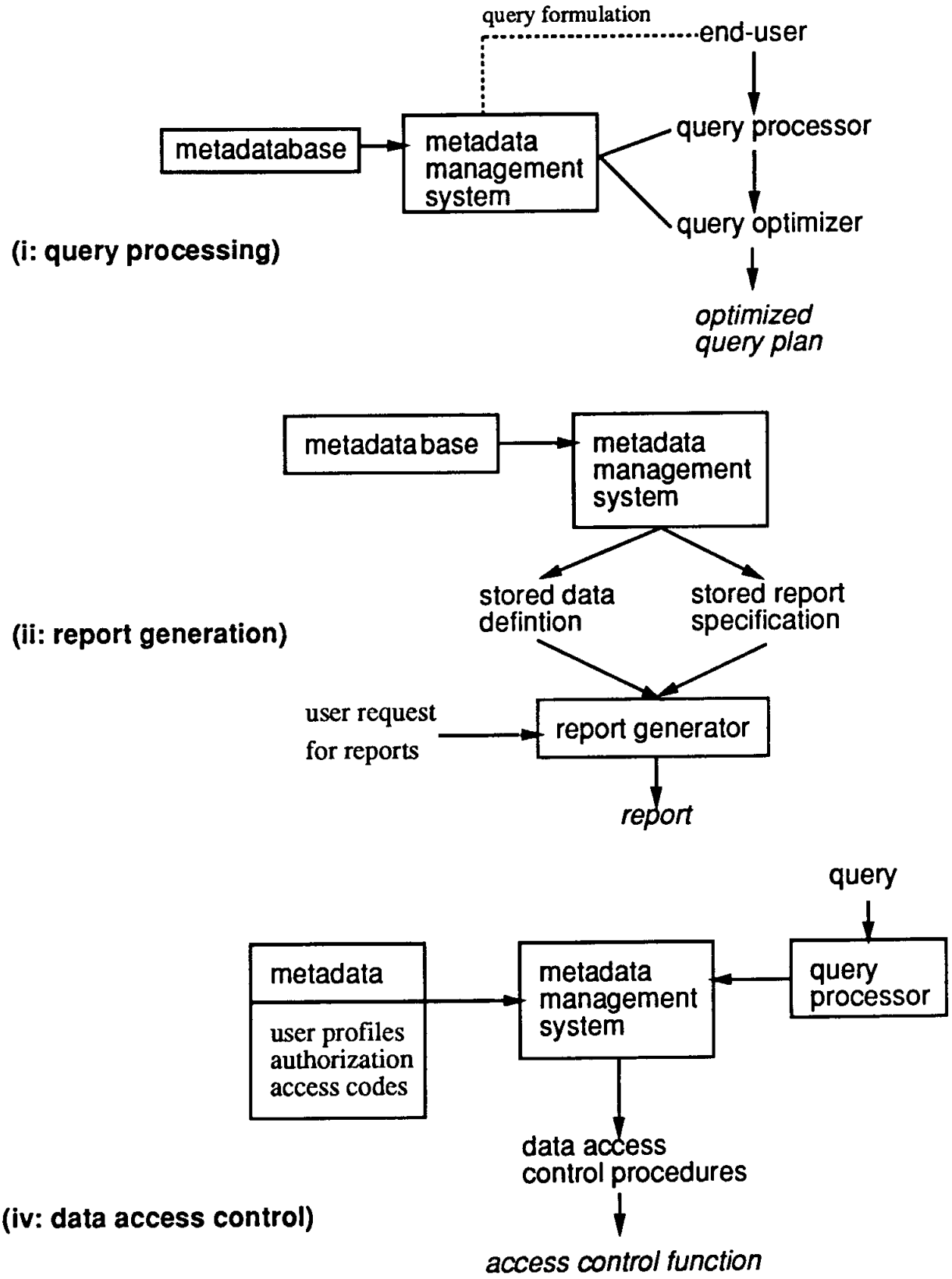


Figure 3: Using Metadata in Query Processing

- Explicit representation of objects, attributes and relationships
- Availability of abstraction mechanisms and modeling primitives to model generalization and aggregation (group) hierarchies.
- Increased separation between the conceptual and the physical schema levels.

3.1 Semantic Data Modeling

Semantic data models provide more powerful data abstractions and modeling constructs for schema specification and design than conventional (relational, hierarchical, network etc) models. Semantic models allow a natural and intuitive description of the real world because they allow the end users including database designers to think of data in a way which relates more closely to real world data.

A comprehensive survey of the area of semantic modeling may be found in [HUL87] and [PEC88]. [HUL87] contains a general semantic model referred to as GSM intended to be representative of a class of semantic models. The popular semantic models include the E-R model, the functional data model (FDM) and the semantic database model. Further details on semantic data models may be found in [HAM81, POT89] while related work can be found in the area of data modeling in object-oriented database systems [MA187, BAN87].

3.2 Framework

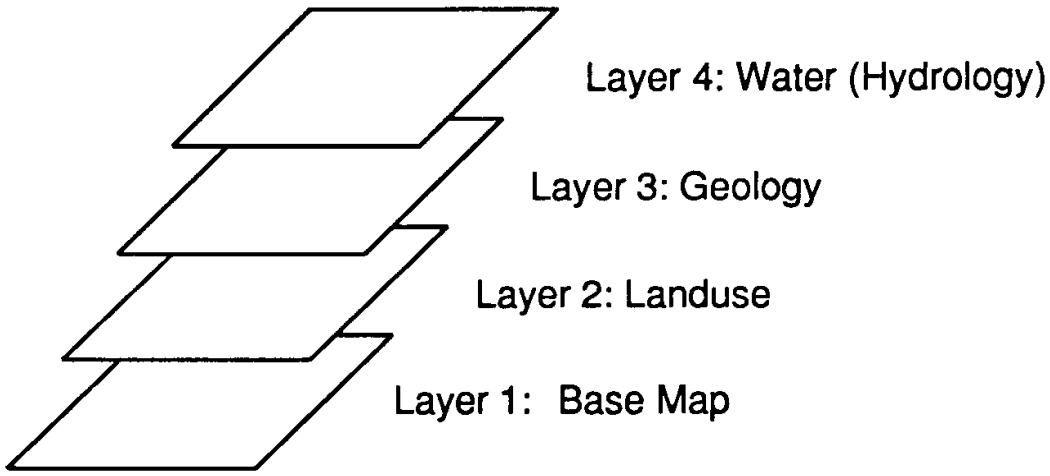
In this subsection we informally describe the framework for a database that can handle structured and unstructured data belonging to heterogenous datasets.

- A spatial database is a collection of spatial and aspatial *entities* or *objects*. These entities correspond to the conceptual entities or objects in the application schema.
- The data is represented by object instances. Each entity belongs to a *class* of entities. The structure and the behavior of these data objects is defined by this class.
- The core of the spatial database is a set of images/maps referred to as a *multimap*, which is composed of a set of "thematic" data layers.
- Each spatial object instance is associated with a data layer via a *location-set*. The location set of a multi-component object is a function of the location sets of its components.
- Thus the spatial data content is in the multimaps or in the attribute values of the spatial objects.
- The characteristics or features of the object classes is described by *attributes* associated with the class definition.
- The behavior of the object is described by the methods associated with the object class. These methods describe the creation, deletion, modification, display, storage and retrieval functions associated with the object.
- Different object classes can be related to each other via interclass relationships represented by a set of modeling primitives. Different attributes can also be similarly related
- Object instances inherit their attributes from the class that they belong to. Object classes inherit their attributes from their superclass.
- Attributes and relationships may have constraints on their values or their properties associated with them.
- The basic data entities are objects, attributes, relationships, and constraints. This set of data entities can be extended to get a sophisticated and rich model.

At the metadata level we are mainly interested in finding out the classification and description of the various data objects, their attributes and the relationships (inlcuding modeling constructs). Modeling constructs essentially provide a mechanism to externally define relationships between various object classes. The user should be able to find out: 1) What types of objects are available to

him and what is the description of a particular object class, 2) What types of attributes these objects possess and the description of a particular attribute and 3) What are the different types of relationships between various object classes and a description of a particular relationship type. The following discussion is in view of this interest.

As an example of the above concepts, consider a simple spatial database schema with four data layers which form the multi-map strata. The four data layers axe: hydrology, geology, landuse and the base map. Figure 4 illustrates this example. We have three superclasses: water, geology and landuse. In addition to this consider three other object classes, which are not part of the multi-maps (as shown here). One is rural town which is a group of three object classes: town, river and farm. The second is the airport which has the following component objects: runways, gates, terminal and parking lot. The third is also an aggregate object class - harbor which has docks, piers, cargo store and loading zone as component objects. Figure 5 illustrates these three multi-component objects.



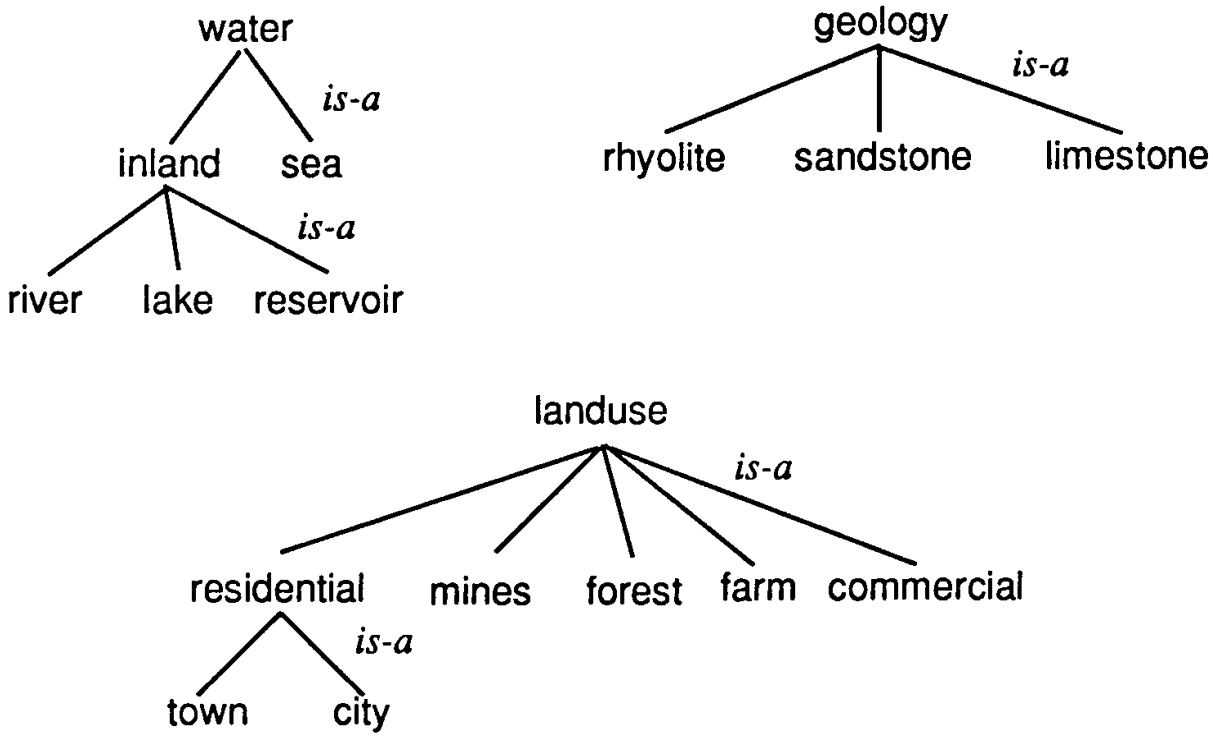


Figure 4: Example schema

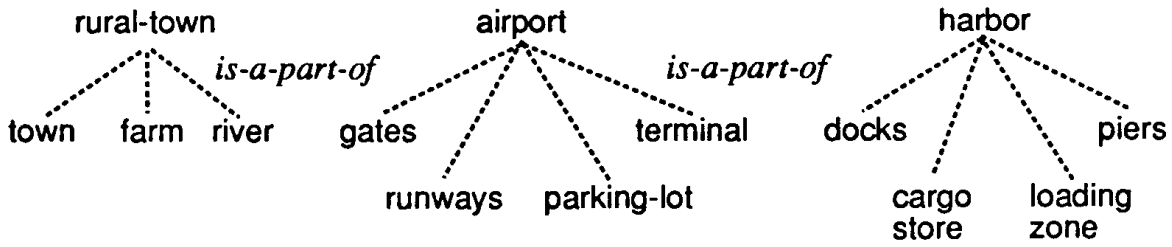


Figure 5: Examples: Complex Objects (is-a-part-of)

3.3 Objects

In this subsection we describe the term object and the metalevel information that may accompany it. The term object refers to two concepts - object instance and object class. An object instance represents a real world entity and stores actual data. An object class defines the structure and the behavior of the objects that are its instances. An object class is an intension into its instance while the instance is an extension of the class. Thus the object class provides metalevel information about the data that is in its instances.

Object Classification

An object class is an object in itself. It is defined as an instance of class object. Objects can be classified into five basic categories based on their nature and their functionality: atomic, simple or single- component, composite or multicomponent, procedural, class and metalevel. Additional categories can be added as required.

Atomic Objects: Atomic objects are the primitive objects that can serve as building blocks for simple objects. The attributes of simple objects are defined in terms of atomic objects. These objects are either displayable or executable and the system does not have to interpret them.

Examples: Examples of atomic objects are street number, street name, city name, zip code and so on. Alphanumeric strings; real, integer and complex numbers; boolean values (true/false); bitmaps/images and graphical primitives.

Single-component or Simple Objects: These objects axe at the bottom of the object hierarchy. They form one logical unit of spatial information that cannot be structurally divided further. The system interprets them in terms of atomic objects. Simple objects are mapped onto the layers in the multimap by their location set.

Examples: As shown in figure 4, town, farm, forest, mines are simple objects.

Multi-component or Composite Objects: Multicomponent or complex objects are abstract objects defined in terms of other composite or simple objects. Their data content can be understood meaningfully by their decomposition into their component objects. A multicomponent object can be a set or a list of objects. It could also be an aggregation of two or more objects. The subsection on modeling constructs explains how composite objects can be composed from simple objects.

Examples: See figure 5. An airport is composite object that is an aggregation of simple objects gate, runway, terminal and parking lot. Similarly a rural town is composed of a town, a farm and a river while a state beach park is an aggregation of a beach, a pier and a park.

Procedural Objects: Procedural objects represent executable procedures and functions that can be used to manipulate and operate on the data. Procedural objects can be classified into four basic categories: system defined, application specific, object methods and constraint evaluators.

Metalevel Objects: Metalevel objects handle metadata. They describe various concepts such as object classes, attributes, relationships and constraints. They form the basic metadata entities. Examples include various types of forms and catalogs etc.

Object Class Description

An object class can be described by a set of features. The basic set of features includes the class name, its type and its description.

Class Name The class name identifies the class. Multiple synonymous class names and class name aliases are allowed.

Class Type The class type indicates the nature of the class e.g whether it is an atomic object (if so what kind of) or a composite object.

Members If the object is a multicomponent object, then this defines the component members of the object.

Description describes the contents of the class and its significance and role in the schema.

Data Layer The data layer name identfies the data layer that the object belongs to. The object can be mapped onto that layer by its location set.

SubClasses are a special case of this class. They differ in some ways from the existing class. While they are classes in their own right, they inherit most of their attribute from their parent class [single inheritance] or from their parents [multiple inheritance].

Attributes describe properties of the class as a whole [class attributes] or those of the class members [member attributes].

3.4 Attributes

An attribute corresponds to a particular property or characteristic associated with an object. An attribute is also an object and is described by its attributes. Generally most of the attributes are inherited by an instance from its class and by a class from its superclass. Attribute types and values have constraints defined on them. The data content in the entity attribute is in the attribute value. The attribute definition provides the metalevel information about it. Thus the attribute defintion is an intension into the attribute values. Metalevel information about an attribute should also include:

1) rules that describe attribute inheritance 2) rules describing derivation of one attribute from another 3) rules describing applications of constraints.

Attribute Classification

There are a variety of ways in which attributes can be classified. One classification is into spatial, aspatial and temporal attributes. The spatial attributes are those characteristics that are location dependent. The aspatial attributes describe the thematic characteristics that are not location dependent. The temporal attributes describe characteristics which axe essentially time dependent. Another classification is based on source of the attributes. According to this criteria, attributes are divided into three basic categories as under:

Primitive Attributes: Primitive attributes are of atomic object type. They are inherited from the parent class. Primitive attributes include structured types such as arrays, files and subranges.

Relationship Attributes: These are the attributes that describe relationships between objects internally. These attributes have as their value a simple or composite object. We identify two types of relationship attributes: reference and set-of.

Derived Attributes: These are attributes whose values are derived from elsewhere. We identify three types of derived attributes: group, computed and inferred.

Attribute Description

As with any other object, an attribute is described by a set of basic features like name, description, applicability and the type of value.

Name: identifies the attribute. Again as with object names multiple synonymous names are permitted. Obviously the attribute names must be unique in the set of attribute names used in the class.

Description: describes the meaning and the purpose of the attribute. The nature of the attribute is described in terms of the classification described above.

Applicability: describes the domain of the attribute functionality. For example this may describe whether this attribute is a *class attribute* or a *member attribute*.

Value: is either a data entity (single valued), a set of such entities (set valued) or a list of values (list valued). Also the attribute value could be *structured* e.g an integer, real number or a character string or it could be *unstructured* e.g an image or an icon. Another property of the attribute value is that it could be *mandatory* which means that a null value is not allowed or it could be *unchangeable* i.e. once set to a non-null value it cannot be changed except to rectify a mistake.

Domain and Interval: provide more information on the attribute value type. Domain specifies the class of values to which the attribute belong. The domain could either be lexical or non-lexical. The interval specifies the possible range of the attribute value.

Existence Constraints: are conditions that must be satisfied for the attribute to be valid. Existence constraints can include integrity constraints, cardinality constraints and syntax constraints.

Calculation/Derivation Method: defines how the attribute value is computed or derived.

3.5 Relations

Relationships between objects can be classified along two dimensions. One dimension is the nature of the relationship which can be classified into three categories: spatial, aspatial and temporal. The second dimension is the way the relationship is specified which is classified as: external and internal. External relationships are specified outside the object definition whereas the internal relationships are specified within. Relations can be expressed as attributes or in terms of modeling primitives and operators. Relationships can be predefined and stored as a part of the object definition or they can be computed at query time.

Spatial relationships are usually expressed in the terms of operators. Finding a spatial relationship usually involves some kind of geometric computation. Various classifications for spatial relationships have been proposed in GIS literature. Three basic spatial relationship categories axe:

1) Topological Relations - describing neighborhood
2) Metric Relations - describing distances
3) Order Relations - describing inclusion and preference

The relationships mentioned above exist at the object level. The modeling primitives are useful in modeling these relationships. At the metalevel the following kinds of relationships exist:

- relationship between two object classes (generalization, aggregation etc.)
- relationship between a class and its instance (classification)
- relationship between a class and associated constraints.
- relationship between a class and associated procedures

3.6 Modeling Primitives

This subsection describes the various relationships between the different type of data objects and the modeling primitives that define these relationships. There are three abstraction mechanisms that deal with relationships between classes: generalization, aggregation and grouping. The generalization mechanism produces the generalization hierarchy, whereas the the aggregation and grouping mechanisms produce the aggregation hierarchy. Classification deals with relating instances and classes.

1. Generalization

Generalization is an abstraction of several classes with common properties to a more general superclass. Generalization involves bottom-up propagation of the common properties (inheritance), from the existing sub-classes to the newly formed superclass. Specialization is the inverse of generalization. Here the propagation is top-down from the existing superclass to the newly formed sub-classes. These two mechanisms model the sub class- superclass relations.

* Generalization: is-a-superclass-of(superclass, class)

Example: Consider figure 4. Assume three simple object classes for fresh water: river, lake and reservoir. Also we have an object class for sea water. Then we can generalize the classes: river, lake and reservoirs to a superclass inland-water. At the next level of generalization, we take inland-water and sea-water and generalize them to a superclass called waters. This can be specified in terms of modeling predicates as:

- is- a- superclass- of(inland-water, river)
- is- a- superclass-of(inland-water, lake)
- is-a-superclass-of(inland-water, reservoir)

2. Aggregation

Aggregation deals with collecting various component objects into one unified higher level object. There is a part-of relationship between an aggregate object (agg-object) and its components. Decomposition is the inverse of aggregation. Generally the component will have some relationship with the aggregate object, but this is not necessary. An aggregate object might also has an associated aggregation expression which specifies the relationships between the component objects in the form of constraints.

* Aggregation: is-a-part-of(component, agg-object, reln)

Example: Again consider fig. 5. The Object class airport is an aggregate of the object classes gates, runways, terminaJ and parking lot. Similarly the object class harbor is composed of docks, piers, cargo store and loading zone. Each component object can have a relationship with the parent object which usually indicates the component's role. For example in the airport example:

reln(airport, gates) = [embark- and-disembark- passengers] and
reln(airport, runways) = [takeoff- and- land- planes]

3. Sets

The set abstraction allows definition of abstract objects that are sets of objects. Thus we get a grouping class which is of *second order* in the sense that its members are themselves object classes. Usually the set of objects is defined by a grouping expression which expresses the relationships between the members of the group and also determines how the members are placed in the group. This expression is based on attribute values of instances of member object classes and the spatial relationships between them.

* Set: is-a-member-of(set, <list of object classes>, grouping-exp)

Example: We have three simple object classes: town, farm and river. One can define a group object class rural town which is a set of three objects: town, farm and river. The associated expression with the group object rural town might involve constraints like:

((DISTANCE town river) < 10 miles) and
((DISTANCE river farm)< 3miles) and
((DISTANCE town farm) < 4 miles)

4. Classification

Classification is an abstraction from individual instances with common properties to a class. Instantiation is the inverse of classification. These two constructs model the relationship between a class and its instance. The instance inherits all of the parent classes's attributes and methods.

* Classification: is-instance-of(instance, class)

Example: Santa-Barbara, Ventura and Goleta are three possible instances of the object class town. Similarly Lake Cachuma and Lake Casitas are examples of instances of object class lake.

5. Other Constructs

Constraint Construct: Contraints are associated with an object via the is-constrainton relationship. There are two basic classes of integrity constraints: implicit constraints and explicit constraints.

Heuristic Construct: Heuristics are associated with the objects via the is-heuristicon relationship. Heuristics can be defined on object class, object attributes or on relationships.

Temporal Construct: Temporal construct is provided by an abstraction mechanism based on the notion of synchronous and asynchronous objects or object types. This construct can help us express causal relationships between objects. Synchronous objects are related to other synchronous objects by either the predecessor or succesor relationship. Asynchronous objects are related to other asynchronous objects by a concurrent or parallel relationship. They are characterized by a trigger that initiates it, an action or a procedure which is performed on it and a post-condition that holds on completion of the action.

4 Metadata Management

In this section we describe a conceptual model for metadata. This is based on the metadata definition given in subsection 2.3. The main characteristics of this model are simplicity, flexibility and extensibility. We believe that this model covers much of the metadata domain.

4.1 Three Level Schema Structure

In many non-standard database applications such as VLSDBs, the metadatabase can be of a considerable size. Hence it becomes necessaxy to use standard data management techniques for metadata such as data independence and data sharing and particularly the separation of the conceptual and the internal levels. The conventional three level logical structure used for this purpose in database management also applies here.

The conceptual schema for the metadata management system (MMS) defines, describes and controls the concepts in the metadata model. It also contains the rules and laws for using these concepts to construct metalevel schema defintions. Like the dictionary schema in conventional data dictionary systems, the conceptual metadata schema must control all the operations on the metadatabase. The concepts described at this level include all the concepts from the database metaschema as well as concepts such as data quality, authorization and transactions.

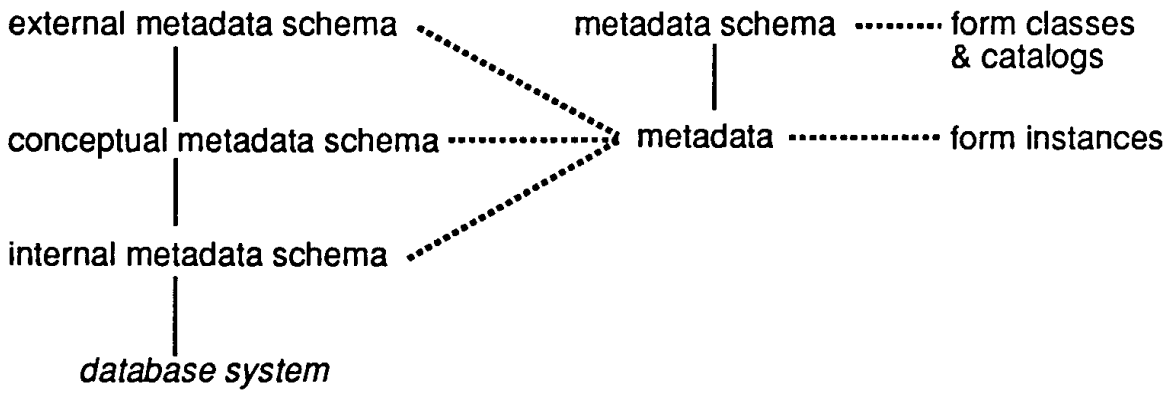


Figure 6: Three level schema structure

There can be one or more external schemas. These contain a subset of the conceptual schema for the use of the database administrators. Usually external metadata schemas are application specific. An internal metadata schema describes all the information at the conceptual schema level, in a form which can be easily and efficiently stored and retrieved. Figure 6 illustrates this three level logical structure.

4.2 Conceptual Meta-model

The metadata model essentially consists of metaentities that describe entities in the database. Metaentities are required for three kinds of entities: data entities, system or process entities and environment entities [LE082]. Meta-data entities describe objects or entities which are units or aggregates of data. Meta-process entities describe objects or entities that are processes or systems or their components. The query language process, the host language process and the report generator process are examples of the process entities. Other examples of process entities are modules, programs, transactions or methods to create, edit or delete objects. Meta-environment entities are used to describe system environment entities like physical devices and users.

While the process and environment entities are quite similiar for different data models, the data entities are quite dependent on the underlying data model. The data entities that that are independent of the data model include reports, databases, constraints, schema graphs and sub-graphs etc. In the case of the record based models, the basic metadata entities would be element, record, group and file; in the relational model the basic entities are relations, domains and attributes whereas in the case of the object oriented model the basic entities would be objects, attributes and relationships (Figure 8).

Metadata can thus be classified into metadata entities, metaprocess entities and metaenvironment entities, as shown in figure 7. The metadata entities can be stored under one of the following three categories: data dictionary, data directory and data quality. The data dictionary module stores the metainformation about the data content and the conceptual schema. The data directory module stores the metainformation about the location of the data and how it can be accessed. The data quality module stores metainformation related to data quality Eke lineage, positional accuracy and attribute accuracy. The metaprocess entities fall into five basic categories: modelling primitives, operators, transactions,procedures and functions, and programs. Each of these can be divided into further categories if required. Operators can be divided into spatial, geometric and aggregate operators and procedures can be divided into system defined and user-defined or application- specific. The metaenvironment entities are divided into physical devices, users, software maintanence and others.

4.3 Core Metadata Components

We indicate how some of these metadata components may be organized and discuss how these components can be grouped into modules and what metaentities they represent. Most of this information is present in conventional systems but it is not explicitly and clearly represented. Also it is scattered over manuals, on-line documentation, user guides and data dictionaries as well as personal notes made by individual users.

The data dictionary has data about the concepts in the data model of the underlying database. In addition to the basic data entities mentioned above and shown in figure 8, this list could include overview and example data and frequently used general superclasses like datasets, datalayers and data acquisition instruments. The data dictionary could also contain housekeeping data about individual applications running in the spatial database. This would help capture data Eke the starting date of the application, expected finish date, person-in- charge, datasets involved, special features and so on.

The data directory has data about where the data entities are located and how they can be accessed, including access paths, spatial indexing methods, security features e.g access rights and authorization, data derivation information and source citations and coverage indexes.

The data quality module contains information regarding the quality of spatial data in the database. This includes information about lineage, positional accuracy, attribute accuracy, consistency and completeness.

Schema graphs would be required for a graphical presentation of the conceptual schema along with an interactive browsing mechanism. If there is a need for browsing through multiple hierarchies (aggregation and classification) then multiple schema graphs should be provided with a uniform browsing mechanism.

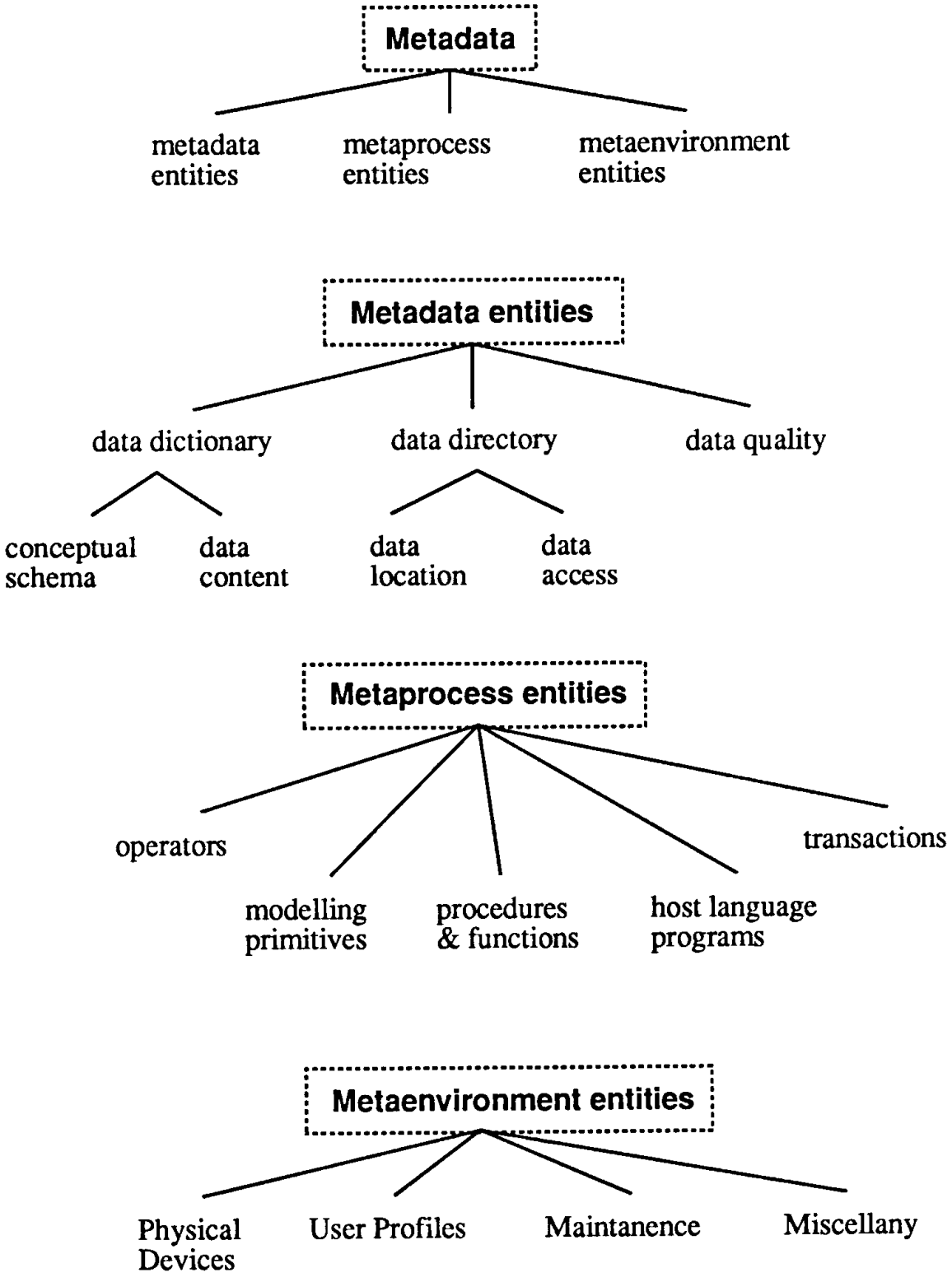
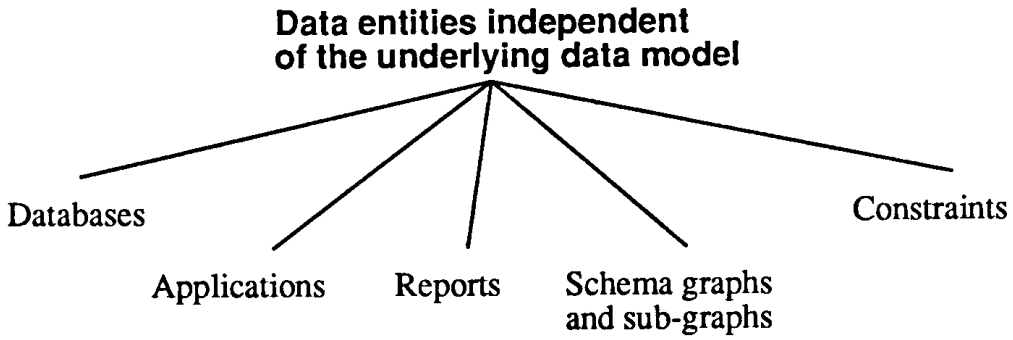


Figure 7: A Model For Metadata

Data entities dependent on the data model

Record based	Relational	Object-oriented	Deductive
element	relations	objects	facts
record	attributes	attributes	terms
group	domains	relationships	predicates
file			rules



Operators	Procedures/Functions	Physical Devices	Users
Spatial	System Defined	Storage and Memory	Individuals
Geometric	Application Specific	Input devices	Groups
Aggregate	Object Methods	Output devices	Organizations
	Constraint Evaluators	Terminals	

Figure 8: Metaentity Classification

Preprocessing History has information about the derivation and preprocessing history of each dataset. This describes the procedure used to convert the data in the dataset from a raw unstructured form to a structured manuscript ready for digitizing

Processing History has information about the major updates or edits done on a particular coverage (map layer) or spatial object. This is a generalized version of a log or an audit file.

System documentation module could be considered to be a part of an integrated metadata management system. The documentation would explain the global system functionality and also the functionality of the various system components. Documentation could include a usage guide for the database query language, a systems tools guide that shows what tools are available and what is their functionality as well as a guide to the various spatial, geometric and statistical operators available for analysis.

Physical device characteristics module describes the characteristics of the physical devices that affect the database system performance and usage. This includes input/output devices and storage devices. Physical storage device characteristics includes: a secondary storage profile, disk i/o statistics, information on available data compression procedures and paging mechanism characteristics like page distribution over multiple disks and so on.

4.4 The Forms Mechanism

We now discuss the conceptual framework of a forms mechanism to manage metadata. The word metadata is used very loosely to cover all the meta-level information that is application dependent as well as application independent. The user has a skeleton schema consisting of core metadata components. We need one uniform mechanism, which allows the user to document required whatever kinds of metadata, as long as it is within the specified framework. The forms mechanism should support interactive browsing as well as query-based retrieval. The forms mechanism allows metadata to be handled in a structured and modular fashion. The core building blocks include fields, forms, catalogs and modules whereas advanced features could include browsers and facilities for automating metadata.

There are four basic entities: fields, forms, catalogs and modules at the metaobject level. The other meta-entities are: tabular and transient forms, fileboxes and orphan catalogs. The primary purpose of these entities is to describe data about the entities in the underlying database and to assist in manipulating, querying and making inferences on the metadata which is stored in them. The oragnization and structure of these entities also allows them to be defined, stored and manipulated within the database management system thus allowing for a kind of "self describing" databases.

Forms are made of *Fields* and forms are filed in *Catalogs*. Each catalog has a set of similiar forms (with the same structure) and catalogs themselves might be filed in higher level catalogs. Thus effectively the form is a set of fields and a catalog is a set of forms or a set of catalogs. Forms describe a concept whereas a catalog defines concept sets.

A *Module* is a top-level catalog which lists catalogs storing metadata that is similiar in some way. Examples of modules would be data dictionaries, data directories etc. Each catalog is either filed in another catalog or it is a module. Each catalog or a module provides a conceptual indexing mechanism to access a required form or another catalog. Between them, forms fields, catalogs and modules can describe the concept hierarchy effectively and cleanly.

4.5 Defining Fields, Forms and Catalogs

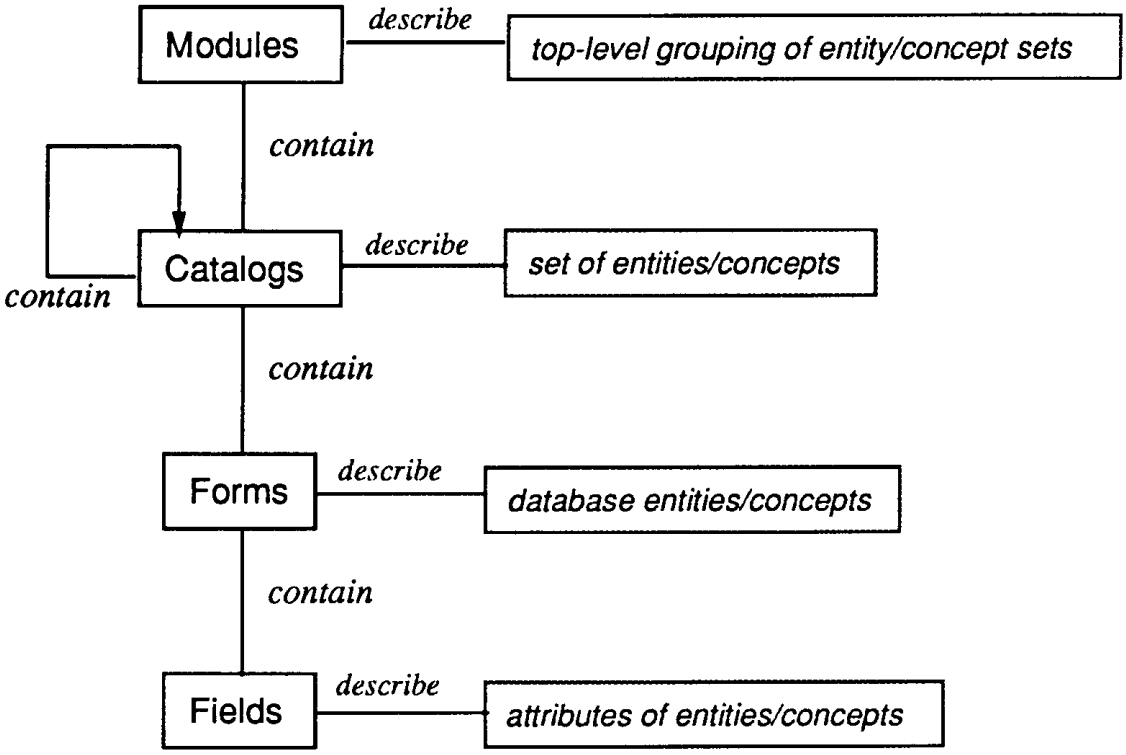
A field is an object with a name (unique within the form) that, in conjunction with the name of the parent form, forms its identifier (id). It also has a set of properties, a value-type and a value. Thus a field can be defined as under:

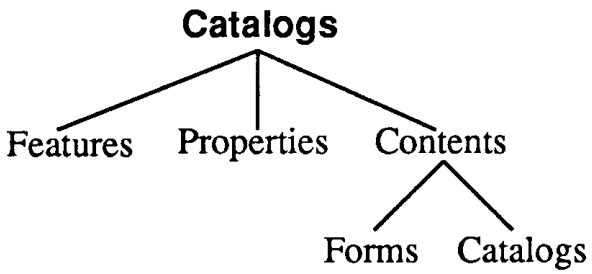
$$\mathrm{FI} = (P_{fi}, T_{fi}, V_{fi})$$

where:

P_{fi} = set of field properties
T_{fi} = field value type
V_{fi} = field value

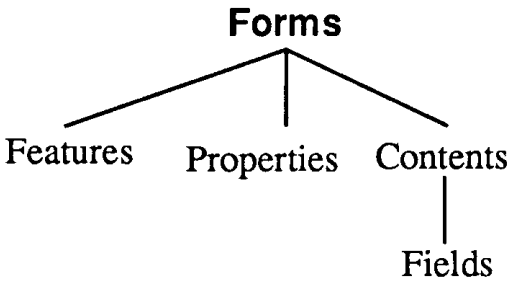
A form consists of a set of features (name, id, description, keywords etc), a set of properties and a list of fields. The fields in a form can be referred to by their name. Each field in a form can be selected by its name, its value or its value type. A form can be defined as:





CATALOG FEATURES

1) Name
2) Identifier
3) Type
4) Aliases
5) Description
6) Keywords



FORM FEATURES

1) Name
2) Identifier
3) Form Class
4) Aliases
5) description
6) Keywords

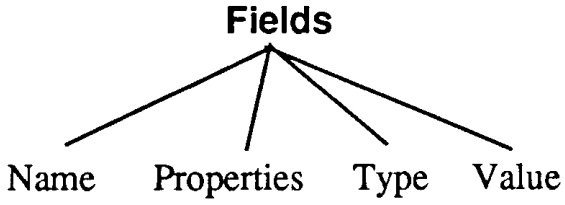


Figure 9: Metadata Entities

$$FO = (P_{fo}, F_{fo}, C_{f0})$$

where:

P_{fo} = set of form properties

F_{fo} = set of form features

C_{fo} = set of form contents

= set of Fields {FI}

A catalog has a set of features (name, id, description, class, keywords etc), a set of properties and a list of entries. Each entry can be a form or a catalog, that is filed in that catalog. The entry in the catalog is represented by its name, identifier (id) and description. This entry can be referred to by its name or the id. A catalog can be defined as:

$$C = (P_c, F_c, C_c)$$

where:

P_c = set of catalog properties

F_c = set of catalog features

C_c = set of catalog entries

= set of catalogs{C} or set of forms{FO}

Examples of catalog/module properties are:
created-by, date, status, etc

Examples of catalog/module features are:
type, identifier, aliases, keywords etc

Typically each form is an instance of a class of forms. The structural layout of the form is defined by its class definition. The forms describing form classes are stored in a separate catalog. Thus to a certain extent this system is self describing. Typically there will be a one-to-one association between a catalog and a form class, in the sense that a catalog will contain forms describing similiar information and thus having the same layout. There are exceptions to the form structure mentioned above in the case of temporary forms and tabular forms.

4.6 Special Cases

We consider two special cases for form structures: temporary (or transient) forms and tabular forms. It may be cumbersome to represent tabular information with the form and field structure described above. For such a case we can define a tabular form where the fields are arranged in rows and columns. In such cases, the forms still have a set of fields as their contents and each field still has one value and one set of properties. The only change is that the field is referred to by a combination of a row name and a column name. Assuming the dot convention for separating name, field name = row-name.colname. The form has the set of row and column names stored separately. When the field is referred to by name, both the row and the column name have to be specified.

If a form is not going to have more than one instance, one can just create the form as an instance of a general-purpose form class called Transient form. Transient forms also serve as a template for defining catalog structure. A *Filebox* is a catalog which contains heterogeneous form types. This can be used to store notes, drafts, memos etc. An *Orphan catalog* contains all the forms that haven't been filed in any catalog. These orphan forms stay in here till either they are eventually moved to an appropriate catalog or are deleted.

4.7 Operations

The following basic classes of operations are required for definition and manipulation of fields, forms and catalogs. If we assume a primitive for each of the operations then we have the following types of primitives:

1) Create
2) Delete
3) Edit

4) Search and Lookup
5) Transfer
6) Miscellaneous

Create primitives are required for creating form class, form instance, and catalog. Delete primitives are required to delete form classes, form instances, fields and catalogs. Edit primitives for forms include adding a field, editing form name and form properties. Edit primitives for fields include editing field names, field properties,and field value. Edit primitives for catalogs include adding and deleting an entry, editing catalog name, editing catalog description.

We can identify three kinds of search: Feature based, Content based and Property based. Feature based search includes getting the field by name; the form by name, form class, keywords, and the catalog by name, catalog type and keywords. Content based search includes getting the field by value, getting forms by fields, getting catalog by entry-name and description. Property based search gets fields by field properties, forms by form properties and catalogs by catalog properties. Lookup primitives allow the properties and features of fields, forms and catalogs to be queried. Transfer primitives allow transfer of information between metaentities and include two kinds of transfer commands - move and copy. Miscellaneous primitives include (1) Initialization primitives for intializing forms and catalogs and (2) Cataloging primitives for filing forms into catalog and catalogs into other catalogs or modules.

5 Case Study: Metadata in a traditional GIS environment

5.1 Introduction

In this subsection we describe a case study to illustrate the use of the metadata framework suggested in section 4. We consider an example case of a large spatial database in a "traditional" GIS environment exemplified by **Arc/Info**. Arc/Info is a vector GIS which consists of two parts: Arc which handles all the spatial data such as map layers and Info which handles all thematic data such as attributes. Arc/Info uses the toolbox approach for management of spatial data and its data model is based on a combination of the topological network model for the spatial data and the relational model for thematic data [MOR89]. Figure 10 shows the logical links between the various Arc/Info entities and also the spatial data properties that we are interested in.

In our conceptualization , the basic Arc/Info data entities are:

• Workspaces

• Coverages (data-layers or maps)

• Spatial objects

• Attributes

• Lookup tables

• Expansion files (to expand id codes)

The basic process entities are:

• Macro Language (AML) Code Files

• Transactions

• Host Language Programs

• Key Files

• Projection Files

• Info files

• Relate files

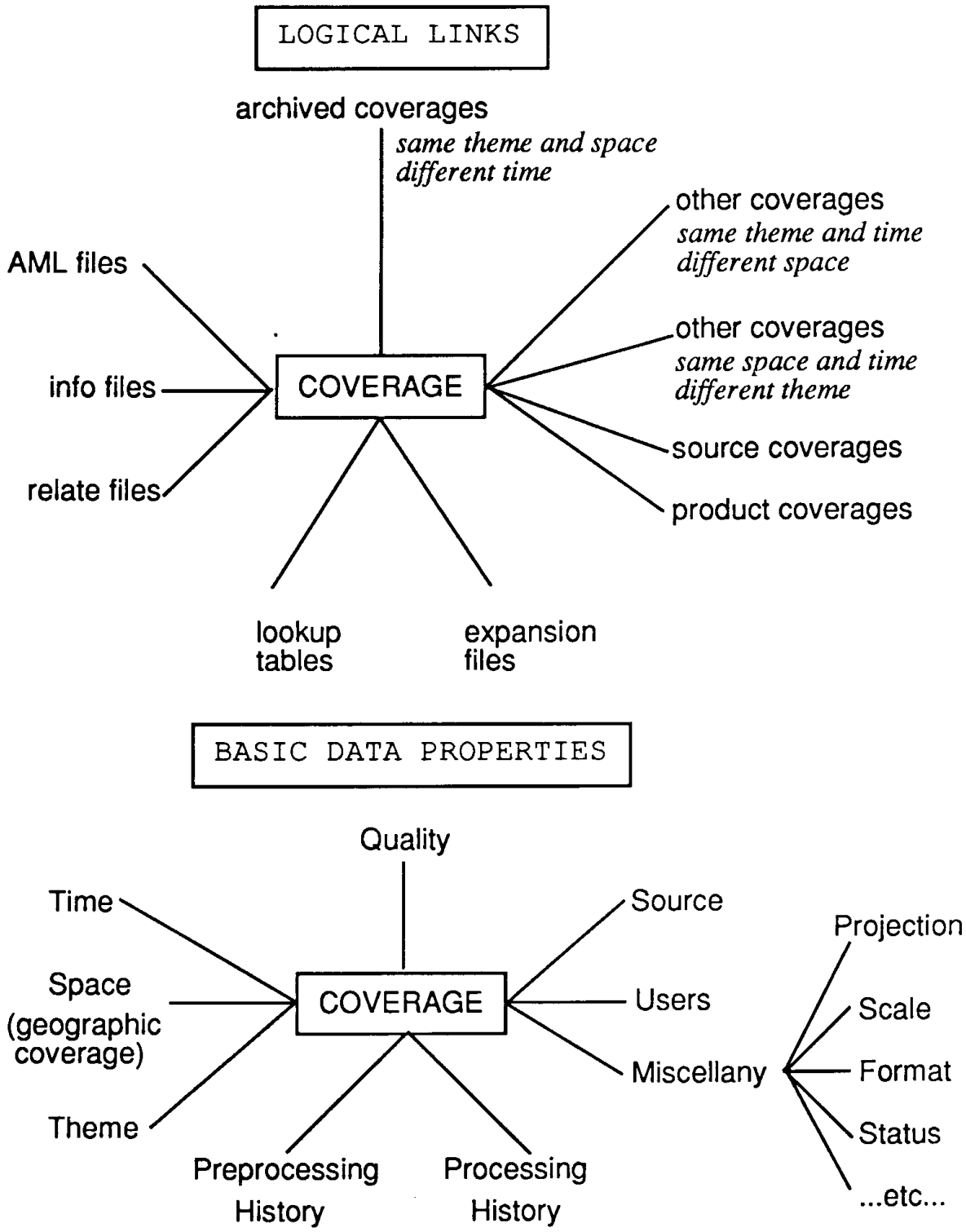


Figure 10: Spatial Data and Arc/Info

In addition to the data definitions and the meta- environment entities (documentation, user profiles and device characteristics), the metadata that we are interested in covers the following basic properties of spatial data:

Theme: descriptions, keywords, aliases

Space: geographic location, extent of spatial coverage

Time: data collection, data acquisition, major updates, product generation

Quality: thematic and positional accuracy, completeness

Location: location in the search space (access paths)

Source: source and derivation information

Miscellaneous: scale, projection, resolution etc

5.2 Environment

We assume an environment with multiple applications. An **application** represents a non-trivial project spanning a period of time and requireing: **a**) acquisition of spatial data in various formats, **b**) management of this data, **c**) spatial /geometric analysis and **d**) presentation of results of the analysis.

We assume the following characterization of a very large spatial database in such an environment in addition to that described in section 1.1:

- Multiple, long-term, non-trivial appli cations/ projects
- Multiple users of different types (individuals, groups and organizations)
- Query-based as well as Product-based operations
- Multiple, heterogenous datasets with one or more coverages per dataset organized according to space, time or theme.
- Various data properties like quality and source are important.
- Keeping a track of the major preprocessing and processing operations performed on the data.

Each application has one or more **workspaces** associated with it. Each workspace is a catalog of **coverages**. Each coverage is a map layer and is the basic unit of data management. Each coverage has a number of **spatial objects**. Spatial objects can be single-component (simple) or multi-component (complex). The **user** can be an individual, a group or an organization. An user can be concerned with more than one application at the same time.

Below we describe a typical classification for metadata queries and a general purpose metadata schema containing the core metadata components as well as some auxiliary components. Appendix A, describes the schema for the Condor Database which was based on this metadata query classification.

5.3 Emphasis

The main emphasis in using metadata in this environment is as follows:

1. To provide the user a high level description of the underlying database in terms of:

   (a) What data entities exist
   (b) Logical links (Dependencies) between data entities
   (c) Properties of coverages or data-layers
   (d) How the coverages are organized

2. To allow the user to locate the dataset of interest, by being able to locate the coverages representing that dataset. The user can locate the coverages by:

   (a) Coverage Name
   (b) Coverage Properties
   (c) Conceptual Dependencies

3. To help the user with data management by keeping a track of the various properties of data layers (including status and times), related data and process entities and associated people.

5.4 Metalevel Queries

Following the classification proposed in section 4.2, we classify the meta-level queries into:

- metadata queries: on data entities
- metaprocess queries: on process entities
- metaenvironment queries: on environmental entities

The metadata queries are further classified into queries on:

(a) Logical links
(b) Data source
(c) Data quality
(d) Data content
(e) Other data properties
(f) Products

The metaprocess queries are further classified into queries on:

(a) Processes e.g digitizing, format conversion and AML files
(b) Transactions

The metaenvironment queries are further classified into queries on:

(a) System documentation
(b) User profiles
(c) Device characteristics

In the following subsections, we describe in brief the nature of different types of metalevel queries, excluding those on device characteristics and system documentation.

5.5 Metadata Queries

1. Queries On Logical Links

Queries on logical links of a coverage include queries about all the coverages or other data and process entities that are related to a particular coverage. This covers the following logical links:

a) Source and Product coverages
b) Other coverages related in time, space or theme
c) Macro Language (AML) files
d) Info Files
e) Relate Files
f) Lookup Tables
g) Expansion Files

Example:

a) What are all the coverages related to coverage A?
b) If I modify/edit/revise coverage A, what other coverages were derived from A that might need to be revised also?
c) What AML, Info and Relate files relate to coverage A (so that if I change or delete coverage A then these links would also have to be updated).

2. Queries On Data Source

Queries on data source involve various data source attributes including:

a) Source Layer Name
b) Source Agency
c) Original Media
d) Feature Types
e) Responsible Agency
f) Scale
g) Projection
h) Source Accuracy (distinguish between positional thematic accuracy?)
j) Measuring Instrument(s)
k) Observation/ Measurement Criteria
1) Reliability/Consistency Example:

Find all < *DATASETS* > in < *WORKSPACELIST* > and the instruments used that satisfy the following:

a) source agency and responsible agency is USGS
b) original media is paper
c) scale = 1:250,000

3. Queries On Data Quality

Queries on data quality involve the following quality parameters:

a) Positional Accuracy
b) Thematic Accuracy
c) Horizontal Resolution
d) Vertical Resolution
e) Completeness (Complete/Incomplete/ Unknown)

Example:

Find all < *landuse/landcover* > and < *road* > coverages that:

a) are complete
b) have certain positional and thematic accuracy
c) have horizontal resolution greater than a certain Emit.

4. Queries On Data Content

Queries about underlying data content, cover the following parameters which describe the data.

a) Theme/description (keywords and aliases)
b) Dataset (landuse, vegetation, roads etc)
c) Attributes
d) Relations (names and purpose)

5. Queries On Other Data Properties

Other data properties that can be queried upon include:

• a) Format (ARC, SIF, MOSS etc.)

• b) Feature Type (Poly, Line, Point, TIN, Grid, Network)

• c) Status:
    i) newly-digitized
    ii) topology-ready
    iii) after- spatial- error- correction
    iv) final-version

• d) Times:
    i) Observation time, (when measured/ collected?)
    ii) Acquisition time, (when imported?)
    iii) Last Edit time, (last major update?)
    iv) Temporal Validity (Valid from when to when?) start time and end time

• e) Tolerances (Dangle, Fuzzy, etc.)

• f) Scale (if different from the scale of data source)

• g) location within search space

• h) Last updated by whom

Example:

Find all the < *polygontype* > coverages in MOSS format that:

a) are newly digitized (status)
b) polygon or point features (feature type)
c) collected /recorded after mm-dd-yy (observation time)
d) valid from mml-ddl-yyl to mm2-dd2-yy2 (temporal validity)

6. Queries On Products

Queries on products involve various product attributes including:

a) Product Name
b) User/Customer Name
c) Product Usage
d) Product Media
e) Product Quality (accuracy, reliability, resolution, etc.)
f) Responsible Person/Group
g) Responsible Agency
h) Date of Release

Example:

List all the coverages involved in products released after mm-dd-yy where:

a) product media is paper
b) usage is "county administration"
c) customer name is "SB county"

7. Queries On Workspaces

Queries on workspaces involve the following:

a) Content (coverages, workspaces, AML files, relate files)
b) Users
c) Authorization
d) Backup

5.6 Metaprocess Queries

1. Queries On Processes

Queries on processes can be classified into a) queries about preprocessing and derivation, b) queries about reading in the data, c) queries about processing history and d) queries about the macro language code files (AML files).

• **Preprocessing and Derivation**: This stage is concerned with converting raw unstructured source data to a structured format that is ready for digitizing. Queries would include questions about the preprocessing methods, agency responsible and the criteria involved.

• **Digitizing**: Queries about digitizing from manuscripts involve queries on: - Rectification:
* Coverage Name
* Source(s)
* Number of Tics
* Rectification Procedure
* Ground Control Points
* Scale
* Person Responsible
- Polygon Digitizing
- Polygon Labelling
- Additional Processing

• **Format Conversion**: Queries on format conversion for input from digital datasets include queries on:

a) Coverage Name
b) Source
c) Original Format
d) Format Conversion Procedure
e) Time
f) Person Responsible

• **Processing History**: This stage is concerned with the tracking edits and updates (probably similar to log / audit file except that you can ask questions here on the different aspects of processing history or locate coverages based on their processing history

• **AML (Macro Language) Code Files**: A user can have queries about the macro file attributes such as:

a) Name
b) Coverages (related to which coverages?)
c) Purpose (keywords)
d) functionality
e) Author (who wrote this?)

2. Queries On Transactions

Queries about transactions can cover the following information about each non-trivial transaction submitted by a user:

a) Type of Transaction (what is the length? is it computation or i/o intensive?)
b) Description

c) Volume
d) Start Time
e) Related Coverages (read/write set)
f) Author (who issued this transaction?)
g) Start Constraints
h) Finish Constraints
i) Degree of Importance (can this transaction be aborted?)

5.7 Metaenvironment Queries

Queries On User Profiles

Queries on user profiles include the following information about each user:

a) User Name and accounts
b) Projects and Role (concerned with what?)
c) Status (student f UG/ MS /PhD}, RA, faculty, non-academic)
d) Start Date (when started?)
e) End Date (how long is student going to stay?)
f) Job (what is this person doing?)

5.8 Metadatabase Schema

Based on the requirements from sections 5.1, 5.3, and 5.4 and the metadata model described in 4, the metadatabase consists of the following set of modules, which are described below:

(a) Data dictionary module
(b) Data directory module
(c) Process module
(d) Documentation Module
(e) Environment Module

5.8.1 Data Dictionary Module

1. Application Catalog: This has description of the different applications currently being managed. Each application is a project with sufficiently large number of coverages grouped into workspaces.

2. Workspace Catalog: This has description of the different workspaces. It is a catalog of catalogs since each workspace is itself a catalog.

3. Coverage Catalog: The Workspace is the coverage catalog. This provides a listing of the various entities like coverages, AML files and so on, and how they are grouped.

4. Object Definition Catalog: This stores object definitions including attributes and constraints for each object. Predefined spatial and geometric relationships can be stored as object attributes.

5. Attribute Definition Catalog: This stores attribute definitions for complex attributes like the type of attribute, the nature of attribute value, the source and the data quality information.

6. Data Quality Catalog: This module has lineage and accuracy information for various coverages /datasets.

5.8.2 Data Directory Module

1. Access path Catalog: This has information about access paths for various workspaces and coverages.

2. Data Source and Derivation Catalog: This has data source descriptions and derivation information.

3. Product Catalog: This has descriptions of the significant products released to outside agencies

4. Security and Authorization Catalog: This has access rights and authorization information.

5.8.3 Process Module

1. Transaction Catalog: This describes the various transactions of different granularity.

2. Preprocessing Catalog: This describes the different preprocessing methods available for incorporating data of different format from different sources.

3. Digitizer Catalog: This describes the different digitizing procedures associated with earch dataset (or if necessary with each coverage).

4. Format Conversion Catalog: This describes the format conversion procedure associated with a dataset or coverage which is derived from a previously digitized source.

5. Processing History Catalog: This contains the processing histories for each important coverage.

6. AML Catalog: This has descriptions of the various AML code files, their purpose and their functionality etc.

5.8.4 Documentation Module

1. System Tools Catalog: describes the different tools available like ARCPLOT, INFO, ARC, ARCSHELL, ARCEDIT etc

2. Query- Formulation- Guide: Has information about: the kind of queries that can be answered and how these queries can be formulated (what are the related commands?).

3. Spatial/ Geometric Operators: This module describes the various spatial and geometric operators available for spatial analysis.

5.8.5 Environment Module

1. User Profile Catalog: Contains information about individuals, groups and organization who are currently active users.

2. Physical Device Characteristics: Contains status and usage information (including maintainance) about various storage (tapes, floppies and hard disks) and i/o devices (printers, scanners, plotters etc).

References

[AD186] Adiba and Nguyen, Handling Constraints and Metadata on a Generalized Data Management System, Proceedings of the First International Workshop on Expert Database Systems, 1986.

[ARM90] M:P. Armstrong and P.J. Densharn, Database organization strategies for spatial decision support systems, International Journal of Geographical Information Systems, Vol. 4, No. 1, Jan-March 1990.

[BAN87] J. Banerjee et al, Data Model Issues for Object-oriented Applications, ACAI Transactions on Office Information Systems, 1987.

[BIL89] F.C. Billingsley et al., Facilitating Information Transfer in the EOS Era, IEEE Transactions on Geoscience and Remote Sensing, Vol. 27, No. 2, March 1989.

[BIS83] Bishop and Freedman, Classification of Metadata, Proceedings of the 2nd International Workshop on Statistical Database Management, 1983.

[CAM86] S. Cammarata and M. Melkanoff, An Interactive Data Dictionary Facility for CAD/CAM Databases, Proceedings of the First International Workshop on Expert Database Systems, 1986.

[CAM88] S. Cammarata, An Intelligent Dictionary for Semantic Manipulation of Relational Databases, Advances in Database Technology - EDBT '88, Springer Verlag Lecture notes in Computer Science, 1988

[CUN83] A. Cunha and T. Radhakrishnan, Applications of ER Concepts to Data Administration, Proceedings of the Third International Conference on the EntityRelationship approach, 1983.

[DAF85] Database Architecture Framework Task Group (DAFTG), Reference Model for DBMS Standardization, Report - NSBIR 85-3173, National Bureau of Standards, 1985.

[DAV88a] J.P. Davis and R.D. Bonnell, EDICT - An Enhanced Relational Data Dictionary: Architecture and Example, Proceedings of the Fourth International Conference on Data Engineering, 1988.

[DAV88b] F.W. Davis et al, California Condor Database Project, Final Report Year 1 Submitted to State of California Department of Fish and Game, 48 p.

[DAV89] F.W. Davis et al, California Condor Database Project, Final Report Year 2 Submitted to State of California Department of Fish and Game, 43 p.

[DCD88] Digital Cartographic Data Standard Task Force, Digital Cartographic Data Standard, American Cartographer, Vol. 15, No. 1, Jan 1988

[EGE87] M. Egenhofer and A. Frank, Object-oriented databases: database requirements for GIS, International Geographic Systems Symposium, 1987.

[EGE89] M. Egenhofer and A. Frank, Object-oriented Modelling in GIS: Inheritance and Propagation, Auto Carto 9, 1989.

[FRA88a] A. Frank and M. Egenhofer, Object Oriented Database Technology for GIS: Seminar Workbook, National Center for Geographic Information and Analysis.

[FRA88b] A. Frank, Requirements for a database management system for GIS, Photogrammetric Engg and Remote Sensing, Nov. 1988.

[GOL85] A. Goldfine, The Information Resource Dictionary System, Proceedings of the Fourth International Conference on the Entity-Relationship approach, 1985.

[G0089] M. Goodchild and D. Brusegard, Spatial Analysis Using GIS: Seminar Workbook, National Center for Geographic Information and Analysis.

[HAM81] M. Hammer and D. Mcleod, Database Description with SDM: A Semantic Database Model, A CM Transactions on Database Systems Vol. 6 No. 3, 1981

[HUL87] R. Hull and R. King, Semantic Database Modeling: Survey, Applications, and Research Issues, ACM Computing Surveys, Vol 19, Number 3, September 1987.

[LE082] Leong-Hong and Plagman, Data DictionarylDirectory Systems, Wiley Interscience Publication, 1982.

[LIP87] U.W. Lipeck and K. Neumann, Modelling and Manipulating Objects in Geoscientific Databases.

[MA187] D. Maier and J Stein, Development and Implementation of an Object-oriented DBMS, Research Directions in Object-oriented programming, 1987.

[MAR85] L. Mark, Self Describing Database Systems - Formalization and Realization, Technical Report TR #1484, University of Maryland, April 1986

[MAR86] L. Mark and N. Roussopoulos, Metadata Management, IEEE Computer, December 1986.

[MCC82] J.L. McCarthy, Metadata Management for Large Statistical Databases, Proceedings of Very Large Database Conference, 1982.

[MCK84] D. M. McKeown, Jr., Digital Cartography and Photo Interpretation From A Database Viewpoint, New Applications of Databases, Academic Press, 1984.

[MOR89] S. Morehouse, The Architecture of ARC/INFO, A UTO-CARTO 9, Proceedings of Ninth International Symposium on Computer Assisted Cartography, 1989.

[NAR88] R. Narayan, Data Dictionary, Implementation, Use and Maintanence, Prentice Hall, 1988.

[NAS01] NASA Working Group Report, Earth Observing System, Science and Mission Requirements Working Group Report, Vol 1.

[NAV86] S. Navathe and L. Kerschberg, Role of Data Dictionaries in Information Resource Management, Information and Management, 1986.

[NSF90] NSF Workshop report, Scientific Database Management, Technical report 9021, Aug 1990, Department of Computer Science, University of Virginia.

[ORE90] J. Orenstein, An Object-Oriented Approach to Spatial Data Processing, 1989.

[00189] Ooi, Davis and McDonell, Extending A DBMS For Geographic Applications, Proceedings, IEEE Fifth Conference on Data Engineering, 1989.

[PEC88] J. Peckham and F. Marynashi, Semantic Data Modeling, ACM Computing Surveys, Vol. 20, No. 3, Sep 1988.

[POT89] W. Potter et al, Hypersemantic Data Modeling, Data and Knowledge Engineering, Vol 4, No 1, July 1989.

[ROE90] L.H. Roelofs and W.J. Campbell, Applying Semantic Data Modelling Techniques To Large Mass Storage System Designs, Proceedings Tenth IEEE Symposium on Mass Storage Systems, 1990.

[RUS83] E. Ruspini and R. Fraley, ID: An Intelligent Information Dictionary System, Proceedings of the Third International Conference on the Entity-Relationship approach, 1983.

[SIB86] E.H. Sibley, An Expert Database System Architecture based on an Active and Extensible Dictionaxy System, Proceedings of the First International Workshop on Ezpert Database Systems, 1986.

[STA90] Star and Estes, Geographic Information Systems, Prentice Hall, 1990.

[ULL88] Jeff Ullman, Principles of Database and Knowledge-base Systems, Computer Science Press, 1988.

[WHA891 S.W. Wharton and J.A. Newcomer, Land Image Data Processing Requirements for the EOS Era, IEEE Transactions on Geoscience and Remote Sensing, Vol 27, No. 2, March 1989.

[WIL89] R.J. Williams, Geographic Information: Aspects of Phenomenology and Cognition, Auto-Carto 9, Proceedings of the Ninth International Symposium on Computer-Assited Cartography, 1989.

[WIN88] J. Winkler, The Entity- Relationship Approach and The Information Resource Dictionary System Standard, Proceedings of the seventh International Conference on the Entity - Relationship approach, 1988.

AN EXAMPLE: METADATA SCHEMA FOR THE CONDOR DB

A.1 Introduction

The Condor database is implemented in Arc/Info. Section 5 discusses metadata in traditional GIS environments like Arc/Info. The details of the Condor Database Project can be found in [DAV88b, DAV89].

Info offers a record based structure for data management which is relatively primitive compared to conventional relational DBMS. The datafile (DF) is the basic unit of data management and it consists of a number of records. A forms mechanism and a report generation facility is available to facilitate input and output of data. The data retrieval facilities are primitive compared to SQL or similiar relational query languages. Hence, the functionality and the scope of metadata is quite limited, eventhough we provide for metadata management as per the framework described in sections 4 and 5, .

The datafile acts as a relation template and each record can be compared to a tuple in a relational database. Data items (DI) are fields of the record or attributes in a relation. Commands are available for selection, projection, join and merge. In terms of the forms mechanism described earlier in this report, a datafile can be compared with a catalog, a record can be compared with a form and fields of the record can be compared with the fields of a form.

INFO provides for an alternative name for each data item and hence one can use that for aliases. Many of the catalogs store descriptions as well as a Est of keywords (max length 30), which describe the nature of the contents of the catalog. One approach would be to put the names of all important data items in the keyword list, whereby a user can select the data file by data item names. There are two basic commands that provide some metadata about the various data files and data items: DIR(ECTORY) and IT(EMS).

A.2 Schema Outline

According to the outline presented in the sections on metadata management and the case-study, we have organized the metadata in three modules: Data Dictionary Module, Process Module and Environment Module. The Data Directory Module will be incorporated at a later date. Since at this point, security and authorization is not a critical issue, this information isn't explicitly represented. Access paths and source descriptions are covered in the Data Dictionary module, but product descriptions are not required at this stage. The Data Dictionaxy module is self-describing in the sense that it has its own description (as well that of the other modules). The Root Module lists the three modules mentioned above. At this point, in time, Transactions, Processing and Preprocessing History and Products are not covered in the metadata described below.

Data Dictionary Module:

01) Source DF
02) Source-Link DF
03) Info-Link DF
04) Rel-Link DF
05) LUT-Link DF
06) AML-Link DF
07) Quality DF
08) Content DF
09) Properties DF
10) Times DF
11) Wksp-Catalog DF
12) Workspace DF
13) Data Dictionary Module
14) Process Module
15) Environment Module
16) Root Module

Process Module:

17) Rel-Files DF
18) INFO-Files DF
19) AML-Files DF
20) Rectification DF
21) Format-Convert DF

Environment Module:

22) User-Profile DF

One can easily define a input form (using the INPUT FORM command) corresponding to each datafile template. The output can be structured by writing INFO programs or by using the Report generation facility provided by INFO. Below we describe the structure of the various datafiles mentioned above. This forms the schema for handling basic metadata. The format used for this is:
FORMAT: Data Item Name: Type [Size] : Description

A.3 Basic Schema

1) SOURCE: Information about various data sources

Name:	Char String [10] Coverage Name
Agency:	Char String [10] Source Agency
Media:	Char String [20] Source Media
Feature:	Char String [10] Feature Type
Scale:	Char String [11] Scale of the source layer
Projection:	Char String [20] Projection

2) SOURCE-LINK: Coverage - Source Link

Name:	Char String [10] Coverage Name
Source:	Char String [10] Source Layer Name
Comments:	Char String [20] Comments about the relationship

3) INFO-LINK: Coverage - Info File Link

Name:	Char String [10] Coverage Name
INFO:	Char String [15] List of related INFO files
Comments:	Char String [30] Comments about the relationship

4) REL-LINK: Coverage - Relate File Link

Name:	Char String [10] Coverage Name
RELATE:	Char String [15] List of related RELATE files
Comments:	Char String [30] Comments about the relationship

5) LUT-LINK: Coverage - Look Up Table Link

Name:	Char String [10] Coverage Name
LUT:	Char String [15] List of related LUT files
Comments:	Char String [30] Comments about the relationship

6) AML-LINK: Coverage - Macro Language File Link

Name:	Char String [10] Coverage Name
AML:	Char String [15] List of related AML files
Comments:	Char String [30] Comments about the relationship

7) QUALITY: Information about quality of the data in the coverage

Cov-Name:	Char String [10] Coverage Name
Pos-Accuracy:	Floating Point [10] Positional Accuracy
Them-Accuracy:	Floating Point [5.3] Thematic Accuracy
Hor-Resolution:	Floating Point [5.3] Horizontal Resolution
Vert- Resolution:	Floating Point [5.3] Vertical Resolution
Completeness:	Char String [15] Completeness of the coverage data

8) CONTENT: Information about the data content

Cov-Name:	Char String [10] Coverage Name
Description:	Char String [30] Coverage Description
Dataset:	Char String [10] Dataset Name
Attributes:	Char String [30] List of Attribute Names

9) PROPERTIES: Miscellaneous Data Properties

Cov-Name:	Char String [10] Coverage Name
Format:	Char String [10] Coverage Data Format
Feature:	Char String [07] Feature Type
Dangle:	Floating Point [8.4] Dangle Tolerance
Fuzzy:	Floating Point [8.4] Fuzzy Tolerance
Location:	Char String [30] Access Path (Location)
Status:	Char String [15] Coverage Status

10) TIMES: Various Times Related to the Coverage

Cov-Name:	Char String [101 Coverage Name
Obs-Time:	Date [081 Observation Date
Input-Time:	Date [081 Data Input Date
Last-Edit:	Date [08] Last Edit Date
Val-Start:	Date [08] Valid From This Date
Val-End:	Date [08] Valid Up to This Date

11) FORMS-CATALOG: Catalog of Input Forms

Form-Name:	Char String [15] Form Name
Related-DF:	Char String [15] Related DataFile
Comments:	Char String [30] Comments

12) REPT-CATALOG: Catalog of Report Specs

Report:	Char String [15] Report Name
Related-DF:	Char String [15] Related DataFile
Comments:	Char String [15] Comments

11) WS-CATALOG: Catalog of Workspaces - List of Workspaces

Ws-Name:	Char String [15] Workspace Name
Description:	Char String [30] Brief Description
Keywords:	Char String [30] Upto 3 Keywords
Comments:	Char String [20] Brief Comments

12) WORKSPACE DF: Information about Various Files in Each Workspace.

File-Name:	Char String [151 File Name
File-Type:	Char String [10] File Type
Keywords:	Char String [30] List of Keywords (upto 3 keywords)
Comments:	Char String [30] Brief Comments

13) DICT-MODULE DF: The Data Dictionary Module

Cat-Name:	Char String [15] Catalog (Datafile) Name
Description:	Char String [30] Brief Description
Keywords:	Char String [30] List of upto 3 keywords

14) PROC-MODULE DF: The Process Module

Cat-Name:	Char String (15] Catalog (Datafile) Name
Description:	Char String [30] Brief Description
Keywords:	Char String [30] List of upto 3 keywords

15) ENV-MODULE: The Environmental Module

Cat-Name:	Char String [15] Catalog (Datafile) Name
Description:	Char String [30] Brief Description
Keywords:	Char String [30] List of upto 3 keywords

16) ROOT-MODULE: The Root Module

Module:	Char String [15] Module Name
Description:	Char String [30] Brief Description
Keywords:	Char String [30] List of upto 3 keywords

17) REL-CATALOG: Information about Relate Files

Name:	Char String [15] Relate File Name
Type:	Char String [10] File Type (e.g linear)
Src-Name:	Char String [10] Name of Source File
Src-Type:	Char String [10] Source File Type
Dest-Name:	Char String [10] Name of Destination File
Dest-Type:	Char String [10] Destination File Type

18) INFO-CATALOG: Information about Info Files

Name:	Char String [15] Info File Name
Author:	Char String [10] Name of the Author
Purpose:	Char String [201 P urp ose/ Functionality
Coverages:	Char String [301 List of Related Coverages
Path:	Char String [30] Access Path

19) AML-CATALOG: Information about AML Files

Name:	Char String [10] AML File Name
Author:	Char String [10] Name of the Author
Purpose:	Char String [20] Purpose/Functionality of the macro code
Path:	Char String [30] Access Path
Coverages:	Char String [30] List of Related Coverages

20) RECTIFICATION: Information about Rectification Process

Name:	Char String [101 Coverage Name
Source:	Char String [10] Source Layer Name
Tics:	Char String [03] Number of Tics
Procedure:	Char String [30] Rectification Procedure Name
GCP:	Char String [05] Number of Ground Control Points
GCP-Criteria:	Char String [30] Criteria for selecting GCPs
Scale:	Char String [111 Scale of the rectified file.

21) FMT-CONVERT: Information about the Format Conversion Process

Name:	Char String [10] Coverage Name
Source:	Char String [10] Source Layer Name
Format:	Char String [10] Source Format
Procedure:	Char String [30] Procedure used for format conversion

22) USERS: Information about the Various Users

Name:	Char String [10]	User Name
Account:	Char String [10]	Login Name
Project:	Char String [20]	Project Name
Supervisor:	Char String [10]	Supervisor/ Manager
Role:	Char String [101	Role of the Student
Status:	Char String [10]	Status of the Student
Start-Date:	Date [08]	Started Working from
End-Date:	Date [08]	Stops Working from
Job:	Char String [10]	Job of the user (if different from role)