Realizing a Spatial Topological Data Model in a Relational Database Management System

Ekow J. Otoo and M.M. Allam

SCS-TR-193, AUGUST 1991

School of Computer Science, Carleton University
Ottawa, Canada, KIS 5B6

Realizing a Spatial Topological Data Model in a Relational Database Management System

Ekow J. Otoo
School of Computer Science
Carleton University
Ottawa, Ontario, K1S 5B6

M. M. Allam
Geographic Information Systems Division
Energy Mines and Resources
Surveys Mapping and Remote Sensing
Ottawa, Ontario, K1A 0E9

August 12, 1991

Abstract

A Spatial Topological Data Model is presented for defining spatial entities and the relationships between them. The spatial entities are assumed to be vector encoded. The model is specialized for defining the geographic features in geographic information system. It is first described using an Entity Relationship diagram (ER-diagram). This is then translated, in a consistent manner, into a set of relational schemes. Although the paper focuses on the use of a relational database management system to represent a spatial topology model for vector data, the diagram can equally well be translated into a collection of object classes for an eventual implementation in an object oriented database. The model is simple, flexible, extendible and non-redundant. By maintaining sites of GIS databases that adhere to the proposed modeling technique, distributed processing in a network of heterogeneous GIS database can be easily achieved. Attribute information is easily associated to the defined spatial entities in a simple manner using the relational join operation.

Keywords and phrases

Data model, topology, spatial features, geographic information system, vector and raster encoding.

1 Introduction

A data model for defining the topological space in Geographic Information System (GIS) is a tool for defining an abstract model of the real world of topographic features. The information required for an application that makes use of spatial data can always be modeled using all or a subset of the model. The basic idea of a spatial data model is a definition of an abstract mechanism, consisting of a data structure and a set of operations, such that the information requirement of any application on the data can always be defined using such an abstraction mechanism [12, 51]. Specific model of an application is derived either as a subset of the general model or by specializing certain elements of the general model to fit the domain of application.

Spatial topographic data modeling has been the subject of discussion in the literature on GIS [41, 42, 44, 45, 8]. While the subject of data modeling has been extensively studied in the field of databases it has only recently been seriously addressed in geographic information system.

In examining the literature, particularly papers by Peuquet [45], it appears that the idea of a data model has been given a different interpretation from that in the field of databases. In GIS the sense of its usage suggests it is data structure for storing spatial data. In databases, a data model is a generic abstract representation of information with which every application - data, operations on the data and the results of applying the operations - can be defined. What is considered as a data model in several GIS literature is really what is typically considered as a data structure. A data structure defines the storage representations and the operations for manipulating it. The two ideas, a data model and a data structure, are completely different. The ARC/INFO [1, 31] topological definition of spatial data and the Intergraph TIGRIS [24] are examples of spatial data models. The quad-tree [14, 19, 47], the k-d-tree[6], the cell-tree [22] and R-trees [23] are examples of data structures. These data structures, interestingly enough are designed to facilitate the storage and manipulation of spatial data.

We take the view that a GIS is a specialized application tool on a database. Consequently, the problem of a GIS database design is equivalent to that of

- modeling the spatial geographic features so that they can be maintained in a database;
- defining access paths for the storage and retrieval of the spatial features and their associated attribute information; and
- defining specialized operators required by the GIS applications for manipulating the data,

using where appropriate, the data manipulation language of the underlining database.

Most of the additional functional requirements of a GIS database - concurrent access, security, data exchange between two database, etc., have been addressed by the database community and some satisfactory solutions have already been integrated in commercially available database technology. An immediate natural approach in designing a GIS database is to adopt a commercial database and extend its functionality for GIS applications by augmenting it with GIS specific operations [16, 36, 48]

A related problem currently being addressed in the GIS community is that of a standard GIS data interchange format. The issue of data exchange between database management systems have been addressed through specifications of pseudo-standard interface languages, such as the Open/SQL of Ingres [46]. Similarly most commercial relational database management systems (DBMS) can communicate and access data resident in other relational DBMSs. In this respect the database community is well advanced in providing mechanism for data sharing in a heterogeneous environment. GIS database implementation can and should benefit from the experience and advances made in database technology.

A related problem in GIS data usage is that of data exchange in a heterogeneous GIS environment and the provisions of methods for distributed processing. One immediate solution is to maintain a tight coupling of the GIS data, both the geometric entities and the attribute information of the spatial data, with an underlying database management system. In this manner, the problem of data exchanges between different GIS reduces to that of data exchanges between database management systems. This problem has already been addressed extensively in the database field. The solutions for data exchanges and distributed access in a heterogeneous DBMS's can be carried over to the GIS community.

The approach of delegating the management of both the spatial and non-spatial aspects of geographic information system to DBMS's has only recently been seriously considered. The reason is that most GISs maintain a database architecture where the spatial data and the attribute information are maintained separately. While the responsibility of managing the attribute data is kept under the control of a database management system (DBMS), quite often a relational database management system, the spatial features are maintained by custom designed storage schemes. The reason being that a GIS user interface is highly graphical. For performance reasons, the geometric entities used in the graphical display are decoupled from the attribute data that are of interest in analyses and responses to queries. We take the view that a GIS is a specialized application software built on top of a database management system and as such, both the spatial data and the attribute data can and should be maintained by an underlying DBMS. With this view we propose a framework for a GIS database architecture and define a GIS topological data model that can be used to facilitate standard specification for GIS data exchange. The approach we take also facilitates distributed processing in a heterogeneous GIS environment.

There are mainly five distinct classes of data involved in GIS databases:

1. vector data - data on spatial features defined by two or three dimensional coordinates of points, lines and polygons;

2. still raster data - digitized pictorial images of spatial features.

3. dynamic raster data - dynamic digitized pictorial images of spatial features, i.e., digital video images.

4. structured data - textual data that are structured into fields of a record. Each field defines some attribute value of the spatial feature.

5. unstructured data - unstructured textual data such as historical documents and narrative or descriptive prose on the spatial entities and geographic phenomenon.

Of these data classes, 1, 2, and 4 are the most widely integrated ones with current GIS technologies. Factors such as less storage overhead and fast access methods using indexes, have made the use of vector and structured attribute information the dominant data classes in GIS activities. Sophisticated compression methods and high storage media have eased the integration of both still and dynamic raster data in GIS. Unstructured data are slowly being admitted into geographic information systems.

Assuming that a common architectural framework for a GIS database is adopted, the problem of which data model can best serve as an underlying database management system arises. In general, the DBMS themselves are based on different data models: network, hierarchical, relational, object oriented, etc. Given the diversity of the data categories that must be maintained in a database system, GIS developers and researchers continue to explore the use of the Network, Relational and Hierarchical DBMS. More recently the use of object oriented and multimedia

DBMSs in GIS have roused serious consideration. Each DBMS maintains storage schemes best suited for a high performance processing of the class of queries it is well suited for. It is not clear which model of a DBMS best meets all the requirements of a GIS database. For reasons that we explain subsequently, a DBMS based on either the relation or an object oriented data model are favorable candidates.

Geographic information systems use various types of digital maps, e.g., topographic, choroplethic, etc., depending on the application. While some agencies, normally government departments, specialize in the production of such maps, most agencies are interested in simply using them. In this respect, we identify two groups involved with GIS data: producers and users. The question then is "To what extent are the data generated by data producing agencies easily integrated into a database for GIS applications?". Alternatively one could ask if there is a standard data format to which digital data generated must conform to in other that they can be easily integrated into a GIS database. This leads then to the question of standard specifications for GIS databases.

There are a number factors involved in specifying a standard for data exchange. These include: the degree of accuracy, the completeness of information being represented, the various data types and formats and ease of understanding in adopting the standard. When these considerations are coupled with the fact that i) there are a number of data models upon which DBMS are based and ii) that database management systems based on the object oriented or multimedia database management system are still in their evolutionary stages and have no formal characterization yet, a satisfactory solution is still far from being attained. Addressing the problem on data standards today is not totally hopeless. We believe a first step towards a solution to the problem of a GIS data standard for both representation as well as exchange, is to accept first, a common architectural framework and a topological data model for a GIS. This paper addresses these problems but concentrates more on the latter.

We briefly mention the issues of an architectural framework for a GIS database with respect to a project being conducted at National GIS Technology Centre, GIS Division, Energy Mines and Resources. We emphasize data modeling for vector data since the bulk of digital data for base maps currently generated at the sector are in this format. The model however can be extended to include raster data and 3-dimensional digital elevation modeling. The main objective in developing the model is to simplify querying and manipulation of elementary spatial entities - point, line and polygonal features. Briefly, the problem we address is a definition of a topological data model that can be mapped onto any representational model of an actual database management system so that the functionalities of a GIS can be realized.

In the next section, we present the various views and external conceptual model of the spatial information. Our proposed architectural framework is also presented. In section 3, the rationale behind our approach is discussed. Further, we discuss some of the navigational requirement in GIS and the manner of utilizing related data sets at varying degrees of resolution and precision. We term this the horizontal and vertical navigation of the database. In section 4, we present a spatial topological model for a GIS database using the entity relationship diagrams. The technique for translating our proposed model into a database is illustrated by mapping the model onto a set of relational schemes for an eventual storage in a relational DBMS. The details are discussed in section 5. An illustrative example is given in section 6. We compare our model with some well known GIS data models discussed in the literature in section 7 and we conclude in section 8

2 External vs. Internal Conceptual Data Models

In defining a spatial data model, we first consider the view of the real world from the user's perspective. Second, we consider how a model of this world is represented to allow for various analysis, and operations on the model. Maps are themselves abstractions of a 3-dimensional world onto 2-dimensional space. This concept of a map today should be seen as a historical legacy imposed by technological limitations of the past. It has been the most convenient form, in all practical purposes, for navigating the world around us. Ideally, utilizing current computer technology, one should be able to realize an actual model of the real world at varying degrees of precision and resolution. That is one should actually be modeling the world as a 3-dimensional object.

It is possible with the current technology at our disposal to depict scenery and terrains using digital elevation models. Realizing such models can be very demanding in main and secondary storage of a computer, particularly where colour scenes are involved. Further, we have been trained to conceptualize the world as a two dimensional space, leaving the exercise of projecting coordinates of 3-dimensional world onto 2-dimensional maps to the experts. As a result we would continue to live with maps as models of the real world projected onto 2-dimensional space for some time.

The GIS software is a tool for depicting exactly the 2-dimensional maps we are used to. The 3-dimensional depiction of the world as digital elevation model is considered an exception rather than the norm with 2-dimensional representations being the exception. For most parts, we are simple continuing with modeling an abstraction of the real in 2-dimensional form rather than directly modeling the 3-dimensional world around us. A futuristic GIS technology should directly model the real world in 3-dimensions as the norm. The eventual goal of GIS technology is to provide pseudo-realistic models of the world around us. The data required towards this goal are available and are already being captured as remote sensed data, airborne photographic images and vectorized digital elevation data. Unfortunately, the computational requirement for 3-D modeling, rendering, etc., is very high.

Even with the current state of the art of GIS technology, there is the lack of appropriate data organization schemes for realizing the total functionality achievable in a GIS. Interestingly enough, the two conceptual models of the world, i.e., 2-D maps and 3-D models, are maintained independently. A map is a 2-dimensional model representing features of a specified geographic region. The features are represented by specific geographic symbols, points, lines and polygons. Often the whole area of interest at a particular scale may not be represented conveniently on a single map sheet and therefore is segmented into smaller sub-areas.

The GIS user however should perceive a region of the world as a single continuum of space. At a session, the user maintains a view window that depicts only a subregion of the total area of interest to him. The total area of interest is defined within some geographic limits. We term this subset of the geographic extent of the world as the *Universe of Spatial Coverage (USC)*. This is normally defined by the largest extent of the area relevant to an application. Typically, the extent of the area relevant to an application is much larger than the area seen through the view window.

Definition 2.1 (Universe of Spatial Coverage) *The Universe of Spatial Coverage (USC), is the largest geographic region containing the totality of data relevant to an application and at the smallest available map scale of at which the data is represented in a database.*

Note that the above definition is with respect the data captured at the smallest scale that a user intends to work with. It does not exclude maps at larger scales that fall within the same geographic region. For example, a USC may correspond to map sheet at a scale of 1:250000.

This implies that all the maps at larger scales for the regions enclosed by the same referenced USC should be accessible. A user may define smaller view windows into the same USC.

Within each view window, the user's perception of features in this space should be either 2-dimensional or 3-dimensional as he or she wishes. We term the union of all users' conceptual model of the GIS database *an External Conceptual Model.* It is essential to note that the conceptual model at this level does not favour any specific application but maintains an abstraction of the information relevant to all possible applications that the database can be used for. An application only defines a subset of this external scheme, i.e., a subschema of the database relevant to the application.

The external conceptual model is mapped onto an *internal conceptual model.* The internal conceptual model is a union of subunits of views that the external conceptual models map into. The internal model is mapped subsequently into an actual representational model of a DBMS. More specifically, we use a relational data model as our representational data model. To handle other data categories such as unstructured textual data, special methods may be developed to enable the relational database to handle such information types. Alternatively, an information retrieval systems may be used to handle the unstructured textual information.

Our GIS database framework, does not preclude the use of other representational data models, e.g., Network, to store the spatial data and the related attribute information. The fact that a GIS database must maintain a mix of different data models within the same architecture has prompted serious consideration of the use of multimedia and object-oriented databases for GIS [17, 48].

Once the mapping from an external to an internal model is done and then mapped to a DBMS specific representation, control of access to the stored data, i.e., storage and access methods, concurrency control, query optimization, etc., becomes the responsibility of the DBMS. We believe the use of customized spatial indexing methods are necessary to augment the standard storage and access methods often found in conventional DBMS. The access methods in most commercial database management systems are predominantly, variants of $B^{+} - trees$ and conventional hashing. These are not appropriate for accessing spatial object, particularly where performance is of major concern.

Figure 1 shows a proposed database architecture for a GIS. There are four distinct classes of data that the architecture is projected to maintain. These are:

1. vector data that define the geometry of spatial features in vector format.

2. raster data that define the geometry of spatial features in raster format. We include in this category dynamic raster data that define the video images of dynamically changing scenery or events related to spatial features.

3. structured text data that define attribute information of spatial features.

4. unstructured textual information.

The topological model we present, assumes that the geometry of spatial features is defined in vector format. This does not preclude those features that are represented in raster format. Most of the spatial data used in our project is in vector format. Further discussions will assume a vector representation of the spatial data. The spatial topological data model we propose however may be adopted for both vector and raster representation of spatial features.

3 Rationale of the Topological Data Model

The rationale behind the topological model is primarily to provide the user with an abstraction of spatial information that is simple, consistent, non-redundant, flexible and extendible. The major spinoffs from adhering to the fundamental model are that distributed access and distributed processing can be achieved.

Historically, different agencies, at the different levels of government (federal, provincial and municipal) have been mandated to collect the data required for GIS functions. The problem is that, over the past several years, each agency has conducted its work exclusively and often independent of all others. In particular the format, structure and model for representing the data vary from one agency to another. The amount of data that have already been collected is very large, i.e., current estimates are in hundreds of terra-bytes. Unfortunately, almost all have to undergo extensive restructuring before they can be incorporated in a GIS. Further, there is the problem of respecting a common mode of geographic referencing to provide a user a consistent addressing mechanism for geographic features. This problem fortunately, is easily resolved by conversion mechanism that already exits between different projections.

To appreciate the modeling technique, we distinguish between 2- and 3-dimensional spatial topological models. Although the 3-dimensional topological model subsumes the 2-dimensional model, we concentrate on the 2-dimensional models in this paper. Extending the 2-dimensional model is straight forward once the significant modeling techniques are understood. To achieve the desirable characteristics of simplicity, consistency, non-redundancy, flexibility and extendibility, we distinguish between five components of the information contained in a unit of spatial coverage.

Geometry: The geometry of the space is defined by points, lines and areas or polygons.

View Modes: This is the mode in which a feature is depicted. The same feature may be depicted either by reconstruction from its vectorized data or simply by reproduction of its image from its raster data. The view mode allows for the display of a feature simple as a still image, a pseudo-realistic image, or as a dynamic or video image. In some cases a symbol may be used to represent a feature.

Interpretation The interpretation of the data deals with the semantic aspects of the feature being perceived. These semantic issues are addressed in conventional cartographic maps by layering. For example, given a polygonal feature the defined polygon can be interpreted as a water body, e.g., a lake, a soil type, a forest type or a park.

Attributes Attributes of a feature concerns the structured or unstructured textual information about it. This aspect of information deals with the specificity of information that uniquely identifies or is related to the feature. For example if a polygonal feature is a water body then structured attribute information will be such things as its name, average depth and salinity. Unstructured information will be historical documents and publication, legal treaties signed on issues such as fishing rights. A spatial object such as a place can be significant as a result of a historical event.

Behaviour Specific rules and constraints that govern the behavioral aspects of a feature. For example, a lake may have flooding potential at certain times of the year or when other conditions are satisfied or events occur. Such rules are necessary to describe the behaviour of features particularly when seasonal changes in the characteristics of a feature influence the behaviour.

Separating out these different aspects of a spatial coverage allows us to focus on the geometry of the space. All other categories of data types can be related to the features via some consistent referencing mechanism either locally or globally. The major advantage of our method is that it is independent of the actual representational form required by the database management system. The main characteristics of the model are briefly outlined below.

A) Simplicity

The success of most standard proposals depends, to a great extent, on how easy it is for a user to understand and how widely applicable it is for the design of different applications. The modeling technique we propose is simple in the sense that the user can focus only on those aspects of information that are of interest to his application. Further it simplifies the interpretation of the data and the structural relationships between spatial entities.

B) Consistency

The model provides a consistent way of relating attribute data to spatial features. By adhering to a common modeling technique and ensuring that the redundancy in the stored data is minimum, one can assume safely that the information contained in the database is consistent across all usage of it. Maintaining a consistent view of the data, as relational tables for example, eases the maintenance work of the data generating agencies since updates and accuracy verification can be done solely by one agency. This minimizes the duplication of efforts and potential source of errors.

C) Tiling and Referencing

A GIS data model should provide the user the illusion that he or she has access to the whole *domain* of the data set. Only a subset of this forms the universe of spatial coverage for an application. We will term this the *Domain of Spatial Coverage (DSC).*

Definition 3.1 *The Domain of Spatial Coverage is the largest extent of area contained in the entire database.*

The domain of spatial coverage within a country is often the region contained within national boundaries including offshore limits. We will term this the *National Domain of Coverage (NDC).*

Definition 3.2 *A national domain of coverage (NDC) is the entire region of a country defined by its national boundaries.*

The relationships between the areas covered by the three terms introduced are such that

$$USC \leq NDC \leq DSC.$$

The limiting bounds of navigation in any GIS session then should be the whole expanse of NDC. From any location, the data specific to an area at any scale and resolution should be accessible.

This implies that the user should be able inquire about any feature or access any information that fall within the defined area of coverage. We term this mode of browsing through the data space *horizontal and vertical navigation.*

To achieve this, a domain of spatial coverage is perceived as tessellated into regular regions called *tiles.* We will assume in the rest of the paper that the tiles are some regular shaped quadrilaterals. Each tile is composed of different layers of information. Typically, a tile corresponds to a map sheet at some specified scale. By arranging so that corresponding edges of tiles match accurately, a process called *edge matching*, a seamless covering of the Domain of Spatial Coverage is achieved.

A tile at some scale may be recursively tessellated further until the smallest unit of a tile is reached. This normally corresponds to the unit of map sheet at the largest scale in the database. Using an addressing mechanism, such as the quadcode method [27, 28], a tile can be associated with a unique address. The address generated in this manner facilitates spatial addressing of features. A tile constitutes a unit of coverage and its address may be mapped to the actual UTM zone and grid coordinates or the bounding longitudes and latitudes.

D) Flexibility

As indicated in the preceding section, each tile is composed of a number of themes or layers. While recognizing the different types of maps such as topographic, choroplethic, cadastral, etc., one objective is to be able extract tiles composed of any combination of layers of common themes. Equivalently, the data model is envisaged to be flexible enough to allow extraction of only the relevant information required for an application. This flexibility however must not be achieved at the expense of storage, i.e., storage of redundant information.

The model is also flexible in the sense that it responds to arbitrary queries on features, allows for the manipulation and extraction of all or a subset of the stored information and permits easy updating of the non-redundant stored data. The queries acceptable in the model include both direct and inverse queries. A direct query asks for information about a spatial feature. An inverse query requests the selection of spatial entities whose attributes satisfy certain criteria. Inverse queries have been called attribute based queries [52, 4]. The concept of flexibility of the model also implies that it is realizable in different computing environments and on different hardware platforms that support a database management system.

E) Extendibility

Certain relationships between spatial features may be represented either explicitly, e.g., as base relations, or computed from the given. The model we propose is extendible in the sense that, it can be augmented with more relationships or entities classes of other categories of data. The model can be realized in any DBMS irrespective of the underlying data model of the DBMS. That is, it can equally be supported in a relational, network and object oriented database management systems. In this paper, the relational database management system is assumed.

F) Non-redundancy

One major characteristic of a GIS data model is that while the concept of layering or separating out of different themes of information into layers is supported, it must be achieved with the minimum redundancy in the stored data. The idea is to associate multiply interpretations and attributes to point, line and polygon features. The geometry of the spatial entity is defined once. However, the identifier for the feature may be related to different attribute tables that associate different interpretations to the same geometric object. The same may be done with the tables that define relationships between spatial entities. This design philosophy is unique in our model in contrast to other modeling schemes described in ARC/INFO [31], CARIS [26], TIGRIS [24], GEOVIEW [53], etc.

G) Distributed Access

There are two major issues to be addressed when considering distributed access in GIS applications. First, since GIS databases are very large, care must be taken to ensure that only the relevant information are transferred in response to queries. In short, the volume of data transferred per request must be minimum. Second, each site must be perceived as a database that stores the information defined in a common data model. In this manner the database at any site sees itself as part-of or an extension of a common globally defined database.

Given that the GIS database at each site adheres to the same data modeling technique, the above two issues are implicitly addressed. Note that we do not advocate that the DBMS at each site be based on the same data model but that the spatial data model mapped onto the DBMS should be the same. From the flexibility and non-redundant properties alluded to in the preceding subsections a response to a query from any site transfers the minimal information requested with no extraneous information. Further, since each site stores a mapping of the same data model onto their respective DBMS format, one can still perceive the information in any database as a simple extension of or part of a globally defined GIS database.

3.1 External Conceptual Model

The external conceptual model is the total data of the Domain of Spatial Coverage as envisaged by a user. At this level, the database is perceived as a single component of defined spatial features. There is no concept of the space being tessellated into tiles or subregions. It is irrelevant whether this universe is perceived as a map sheet of vector data or a big raster image. The maim interesting idea is that a view window can be moved across the total expanse of space. The user generates different views of the space that falls within the view window. The view seen may have varying degrees of resolution with no visible boundaries of the tiles. For any spatial feature, the related attribute information, either structured or unstructured may be interrogated.

The size of the view window is defined by a user and is often no more than what can be conveniently examined within the size of a display device. The map displayed in a view window may be at any scale the user chooses. The external model allows for the definition of procedures and functions for a seamless navigation within the whole universe of spatial coverage in both horizontal and vertical directions. At any location, where the view window is focussed, the mode of viewing the database may be varied. The different modes of viewing the information within a view window are:

- as a conventional 2-dimensional map of varying scales and details. The details within a view window may be enhanced by loading and displaying data captured at larger scales. In the other direction the details may be generalized to reduce the details of the information being displayed.
- as a 3-dimensional model of the space, e.g., digital terrain models;
- as a static raster image of the region in the view window or as a dynamic video image of particular features.
- as a structured or unstructured text of information in response to queries.

The different modes of viewing the information within a domain of spatial coverage implies that the different categories of data exit in the database for the whole coverage. Since the universe of coverage is composed of a collection of regular tiles of smaller units, the repository of information for all categories of the data is also in units of these tiles. The tiles and their corresponding units of the various categories of data form the internal conceptual model. Specifically while the spatial data at the external conceptual level is not distinguished as a collection of tiles, the internal conceptual level is cognizant of the units of tiles. The database supporting the different modes of views is a collection of recursively defined units of tiles and subtiles.

3.2 Internal Conceptual Model

The internal conceptual model is a level of the database that is of interest to database designers and advanced users. It deals with collections of manageable units of spatial information. At this level the distinct data sets supporting the different modes of view are partitioned into smaller collections that correspond to the tile sizes. Each unit is further partitioned into smaller and smaller tiles until the smallest tile size is reached. Each level of a tile defines a different map scale. The smallest tile gives the largest map scale within the universe of coverage.

Each tile has an identifying name, say a map sheet name, and a unique reference code. We present an encoding scheme appropriate for supporting the vertical and horizontal integration subsequently. The different categories of information relating to each tile are clustered under different subdirectories. The directory naming observe a consistent hierarchical convention. The clusters of information for the spatial data include, vector, raster, structured attribute data, unstructured text, etc.

How the spatial information within each tile is represented is our main concern in the data model presented in this paper. At this level, the spatial data are classified into two distinct categories: vector and raster. We focus on the data model for the vector data of each tile. Strictly, in so far as storing the spatial entities is concerned, the model is a hybrid scheme. We have assumed that the content of each tile corresponds to the information on a map sheet. The information contained in a tile may be represented either in vector format or raster format. By representing the spatial information in vector format, as we do subsequently, we devise a hybrid scheme. Alternatively, the technique for generating reference codes for the tiles may be extended to encode the digital image of the tile itself. The modeling tool used at this level for the internal conceptual data model is the Entity Relationship data model [10, 51]. The spatial topological data model is depicted in the form of entity enhanced entity relationship diagram (EER-Diagram) [12].

Given the internal conceptual model as an EER-Diagram the model may be translated into a DBMS specific representation. In this paper the mapping of the enhanced entity relationship model to relational schemes is addressed. One can now appreciate how the structured attribute information of a tile represented as a collection of normalized relations can be related to the geometrically defined objects. The unstructured texts that must be related to features in a tile can similarly be associated with the spatial entities. These need not necessarily be maintained as relational tables. The rules that govern the behavior of features can be retained as expressions or functions in relational tables. Embedding the behaviour of a map feature has been addressed in GISs based on knowledge bases [43, 49]

The unstructured information may be stored in an information retrieval system. Note that it is possible to organize unstructured information using a relational database management system [18]. However, this requires further data modeling techniques beyond our current concern.

3.3 External to Internal Conceptual Mapping

The external to internal conceptual mapping bridges the gap between the external model and the internal model. The mapping is defined by a collection of functions and procedures by which edge-matching of adjacent tiles at the same scale is done. These functions also aid in the horizontal and vertical navigation of the spatial data. The external conceptual model of the universe is a single spatial entity. The internal conceptual model is a collection of disjoint and non-overlapping tiles. A tile is defined according to some regular tessellation of the universe of coverage. We will assume that the tessellation is into regular shaped quadrilateral or quadrangles. This enables us to adopt a quad-tree like partitioning of the universe. Often, the universe of spatial coverage will be a region defined by national boundaries.

The mechanism for mapping the external to the internal conceptual model is illustrated using the Figure 2. Suppose the region in Figure 2 is named ABC. Let ABC be partitioned along the X and Y axes by a 4×4 grid. A rectilinear cell of size $1/4 \times 1/4$ constitutes a first level of tiling. A tile corresponds to a map sheet at some scale.

Each tile may be further partitioned into a 4×4 grid giving a second level of tiles of size $1/4^2 \times 1/4^2$. This second level of tiles now corresponds to map sheets at a scale 4 times larger than the first level of tiles. This process may be repeated recursively to generate tiles that correspond to map sheet of larger and larger scales. The process stops when tiles corresponding to the largest map scale allowed in the database are generated. The assumption here is that within each tile, the spatial features are defined in a vector format. Alternatively, we could envisage each tile as a raster image. The resolution of the images varies as we move from one level to another in the tessellation process. Currently, the raster images (i.e., satellite images - landsat, spot, etc.) [4] are available only in fixed predefined degrees of resolution.

The spatial indexing technique developed is what allows for the seamless horizontal and vertical navigation of space. In this manner a smooth integration of spatial data at varying degrees of resolution of the map scales is achieved. Each tile is assigned a quadcode. In figure 2 the first level of tiles have the following quadcodes {00, 01, 02, 03, 10, ..., 20, 21, ..., 31, 32, 33}. If we concentrate on the tile 03, and consider the second level of grids within this tiles, the quadcodes generated are {0300, 0301,0302, 0303, ..., 0330, 0331,0332, 0333 }. This encoding scheme is exactly the linear quadtree codes described in [19]. So far it appears that this tile addressing is dense and may not warrant the use of quadtree encoding method. In practice this is definitely not the case. A GIS database will contain only a subset of the tiles available in the domain of spatial coverage.

The data of any particular tile may be present or absent. If it is present then this corresponds to a black region in the linear quadtree encoding. If it is absent it corresponds to a white region. The codes represented are for all black region or tiles whose corresponding data set are available in the database with one exception. If the data set of a tile is not present but that of one or more of its lower level tiles exit then the tile is designated as a black node. The encoding scheme of the tiles is actually, the quadcode method defined by Li and Loew [27, 28] and not the linear quadtree encoding of Gargantini [19]. A quadcode is a linear quadtree code with no don't care digits.

The quadcode is essentially used for spatial indexing of tiles in the database. In the next section we present a detailed 2-dimensional enhanced entity relationship model of the information of a tile. This may be represented as a collection of relations. The quadcode of each tile serves as the spatial index to the directory that holds all the information relating to a tile.

More specifically, each tile has a record consisting of a pair of values: the quadcode and its directory path. The records are indexed by using a $B^+ - tree$ index on the quadcodes. The $B^+ - tree$ forms the spatial index for the tiles. This method of indexing the tiles is similar to the approach proposed by Mark and Lazon [30]. A similar idea is shared by Libera and Gosen [29]. In place of a $B^+ - tree$, an extendible hashing method [50, 38, 11] may be used. There are two main advantages gained by storing the quadcodes in either a $B^+ - tree$ or an extendible hashing scheme.

1. The time to locate the information relating to the tile is independent of the level of the tiles formed by the recursive tessellation of the space.

2. The approach simplifies the definition of the mapping functions for the horizontal and vertical navigation of the domain of spatial coverage.

Figure 3 shows an example of the quadcodes of figure 2 stored as a $B^+ - tree$ of order 3. In the sequel we will assume that the pairs of values, i.e., the quadcode and directory path, are stored in a $B^+ - tree$. A $B^+ - tree$ index is used since the tessellation of the space can quickly grow into a large number of tiles and consequently must be retained on secondary storage.

With the quadcode address assignment for each tile, it is easy to see how the external-to-internal mapping function is performed. We should point out that the external-to-internal mapping transcends the conventional schema or subschema definition in database terminology. The subschema definitions provide the inverse mapping and restricts the information contained in the database to that required by a particular user. In a GIS database, such a view definition may be used also to limit the spatial extent of the database in which a user can navigate. The spatial extent defined by a user may span several tiles depending on the number of levels and details stored by the internal model.

Spatial navigation may be done by invoking one or more of the mapping functions described below. Normally a user may not invoke these functions explicitly. Once the coverage of interest is defined, movement of a cursor beyond a view window invokes these functions. The basic mapping functions are given in C-like style.

```
enum  dir  {N, S, E, W, NE, SE, SW, NW};
enum   qDigit  {0, 1, 2, 3};
typedef dir   Direction;
typedef qDigit  QuadDigit;
typedef unsigned long  Quadcode;
          .
          .
```

```
QuadCode NextAdjacent (QuadCode current, Direction dir);
QuadCode DownLevel (QuadCode current, QuadDigit  RelPos);
QuadCode UpLevel (QuadCode current);
```

The function *NextAdjacent* returns the quadcode adjacent to *current* in the direction given by *dir* but at the same level. If the adjacent tile exists its quadcode is returned otherwise it returns the value of *current.* The value of *dir* is specified as one of {N, S, E, W, NE, SE, SW, NW}. This function is defined similar to the adjacency function discussed in [28].

DownLevel returns the value of the lower level quadcode of the tile that is at the same relative position of tessellation as the value given by RelPos. The relative position of a tile at a particular level is given by the last 2n quaternary digits of the quadcode whenever the level is partitioned into $2^n \times 2^n$ cells. *Uplevel* is the inverse operation of *Downlevel.* It simply returns the parent tile of *current.* If such a parent tile does not exist it returns *current.* These last two functions provide the vertical navigational ability of the data model. The three functions above together with the standard view definitions on the base relations of the tiles define the external-to-internal mapping.

Once the quadcode of any required tile is computed, access to the spatial, attribute and other information related to the tile is obtained by locating the directory where all the information on the tile is stored. This is done, with the same time complexity for every tile, irrespective of its level, using the $B^+ - tree$ index on the quadcodes.

The different levels of models, i.e., views, external conceptual model, internal conceptual model, DBMS representational model in the GIS database correspond to the well known conventional level of database architecture. What we have done can be seen as having modified, in the layered architecture of a DBMS, *the conceptual-level* into two components of internal and external conceptual views. The *logical* and *physical* levels remain the same. The figure 4 shows the correspondence between the modeling concepts introduced for GIS database and that of a conventional database architecture.

4 A 2-Dimensional Spatial Topological Data Model

4.1 ER-Diagram of the Data Model

The model presented in this paper is said to be 2-dimensional since the Z-coordinate value is used only to specify height information. It does not influence the search process for a spatial object. In a three dimensional data models, where modeling of terrains and volumetric objects are of major concern, the Z-coordinate value becomes significant.

The topological model of a tile is first presented using the entity-relationship diagram. The tessellation of a universe of spatial coverage into tiles provides us with a simple but consistent approach to store and access the different categories of map information retained in a GIS database. A map sheet at a given scale corresponds to a tile. A tile is an entity and is related to another tile, possible at a different scale, by some relationship. Appendix A gives the diagrammatic symbols used in describing the spatial model. For further explanation on the ER-modeling technique used in this paper the reader may consult [12].

Essentially, a map sheet is defined as a tile entity. A universe of spatial coverage is made of several tiles that can be edge-matched. The relationship between a map sheet at one scale and another at another scale is strictly hierarchical. Each tile is separated into layers of common themes. We will assume that there at most 64 distinct base layers per tile. This number is more than adequate for constructing maps of different layers of information. From these distinct layers one can construct about $(2^{64} - 1)$ possible combinations of layers for a tile.

The relationship between a cartographic map sheet at a particular scale and its layers is 1:N (see figure 6). We avoid the explicit storage of horizontal adjacent relationships, since this can always be computed from the quadcode of the tile. The spatial topological model discussed next assumes that the data on a tile is vector coded. The spatial information on the tile can be raster coded but we do not elaborate on this in this paper.

4.2 Entities and Relationships

A layer of a tile contains a homogeneous theme of information. Examples of such themes are, transportation network, fishery resources, drainage basins, inland waters, wetlands, population density, minerals, vegetations, forest stands, etc. The annotations and labels on a map are secondary issues that are handled by the user interface subsystem.

Figure 5 shows a typical example of a thematic layer of a tile. The basic geometric objects on the map tile at any scale are nodes, lines (which are either straight line segments or arcs), polygons (which may be regular or irregular). Polygons may be triangulated into triangular facets particular where digital elevation model is important. Polygons or regions may be partitioned into rectilinear regions. There are three major geometric features on a map: *points, lines* and *area* features. These are also designated as nodes, lines and polygons respectively when they are accorded specific meanings. These concepts are explained below.

Point: This represents a location of a geographic phenomenon and is assumed to have no linear or area dimensions. Typical usage of it is in a sequence or chain of points that describe a line or vertices of a polygon.

Node: When a point is used as an isolated singular feature or has some special significance in the definition of the topology of the space, we call this a *Node*. Examples of such usage are as an oil field, a mine, a city on a map at a very small scale, the end or intersection points of line features. A *Node* is distinguished from *a point* by the fact that it has a significant role besides being a point feature. Both a node and a point are defined according to a particular projection, by three coordinate values $\langle X, Y, Z \rangle$.

Line: A linear feature that starts and ends at two nodes that are not necessarily distinct, is referred to as a *line*. A line has no width. Special cases of a line are:

i) *a line segment* and ii) an arc. A segment is the straight line that joins two points. A *line segment* is sometimes referred to as *an edge*. An arc is a line defined by a sequence of points called the *chain*. Two nodes are always distinguished as the end points of a line. One is called the *Start Node*, the other is designated the *End Node*. We associate a direction to a line. This is always from the *Start Node* to the *End Node*.

Region: A *region* is any enclosed simple area bounded by lines that intersect in at least three noncolinear points.

Polygon: A polygon is a special case of a region in which the bounding lines are straight line segments. If the points of a polygon are close enough, it can be used to approximate a region. This is normally the case when the bounding lines of a region are generated by digitization. We will use the term *polygon* as a generic term for any area feature.

Using the above definitions and the symbols for extended entity relationship model, we can represent the spatial model in an entity-relationship diagram. Figure 6 shows the ER-diagram of a 2-dimensional Topological Spatial Data Model. This conceptual model is easily translated into either a set of relations or a collection of object classes. For now we consider only the translation of the data model to relational schemes. A complete labeling of the diagram is avoided in other not to clutter it and subsequently loose the significant ideas on the diagram. The entities, relationships, attributes and cardinality ratios of the relationships are as follows. In the diagram, rectangles denote entity classes and diamonds denote relationships. The pairs of numbers in parentheses (N, M), on a line connecting an entity to a relationship is interpreted to mean that an instance of an entity class can participate in at least N and at most M relationships.

4.3 The Entities of the Model

The significant entities in the model are:

NODES A node is the smallest unit of a feature represented in this model. It is defined by a triplet of coordinate values. A node is assigned a unique identifying attribute, *Nid* and coordinate values (Xcord, Ycord, Zcord).

LINES Linear features that spans two nodes are collectively stored in the entity class *LINES*. A line can be a straight line segment or an *arc*. Hence, a line may be specialized into two distinct classes: *ARCS* and *SEGMENTS*. An *ARC* is described by a sequence or chain of points that approximately describe its path from the start node to its end node or a chain of line segments that join adjacent pairs of points. A line segment is completely defined by its two end points. Whenever a chain is defined, the line is approximated by a spline function passing through the chain of points. We would use the term *line* as a generic term for both a segment and an arc. A line has a unique identifier denoted by *Lid* and two special nodes, the start node, *Snode*, and the end node, *Enode*. The direction or orientation of a line is from the start node to the end node. The entity class *LINES* may include attributes specific to a line such as its length, etc.

POLYGONS The polygon entity class consists of area features enclosed by bounding lines. The attributes associated with a polygon entity include the unique polygon identifier *Pid*. Other attributes that may be associated are the coordinates of the centroid *PcntX, PcntY, PcntZ*, its area and a chain of bounding lines *Bndlns*. Strictly, the list of bounding lines for a polygon is not essential. This can always be generated from the relationship POLYLINES which defines the association of lines with polygons. We include this attribute simply for performance reasons.

LAYERS A layer represents all features of a common theme. Examples are vegetation, soil types, water bodies, land use, wetlands, etc. A number of layers may be combined by overlaying. A set of layers may be combined to form other defined layers. The entity class LAYER is a collection of defined layers in a *TILE.* Each layer has a unique identifying code, *LyCode.* A layer may be assigned a name, *LyName.* For display purposes, a layer has a display priority code given by *DispPriority.*

TXTSYMBS This entity class defines the characters and symbols that appear in a layer and are used for annotations, labels, legends, etc. These may be single characters, strings or special defined symbols. The TXTSYMBS entity class is a generalization of two entity classes: LBLTXTS and SYMBOLS.

The entity classes described above define the topology of a tile. The next two entity class are given for completeness. These show the required entities required to define the domain of spatial coverage that form one unit of a GIS database.

TILES The TILE entity class, is composed of all the tiles represented in the database. A tile has a unique identifier, Tid, that is the same as the assigned QuadCode for the tile. It has possible a Tile Name *TileName.* Several tiles define a coverage. This entity class form a catalogue of the tiles in the database. The attributes that characterize this class are described latter.

COVERAGE A coverage entity class is formed from the identifiable autonomous regions of the globe that are to be represented in the database. A coverage is equivalent to the Domain of Spatial Coverage discussed in section 2. It contains two key attributes which are: a unique identifier, *CvgKey*, and a name, *CvgName.*

4.4 Relationships of the Spatial Model

A number of relationships exists between the entities defined above. The EER-diagram of figure 6 defines about eight relationships. Depending on the application, only a subset of the entities and relationships may be used. It may appear, on first encounter, that a lot more relationships than the ones shown need to be defined. For example it may appear that a relationship between the NODES and POLYGONS entity classes is necessary. This means that pairs of node and polygon entities, that define the association of a node with a polygon, must be stored explicitly as a relation. However, a relationship can be computed using the relations already stated in the EER-diagram of Figure 6.

Usually, a design decision dictated by space/time tradeoff must be made on how significant a relationship is to warrant an explicit representation. The cost of deducing certain relationships on the fly is dependent on the underlying storage structures. We utilize quadcodes for five different purposes:

1. For global indexing of the tiles, i.e., determining the directory path of the location where all the information relating to the tile is stored.

2. For bucketing the spatial information of the tiles. This provides a second level of indexing scheme for accessing the stored information within a tile. Normally this is done by the indexing mechanism of the DBMS. For efficient processing of spatial queries, it is worth replacing this with a customized indexing scheme.

3. To facilitate horizontal and vertical navigation through the database.

4. For raster encoding of the images of the map.

5. For encoding data in digital elevation models.

The use of quadcode for the last two items are not discussed in this paper. The versatility of the quadcode methods as a storage scheme for GIS design is the subject of a Master's thesis in preparation. Quadcode encoding for spatial indexing allows a number of relationships to be computed from the current set of relations in an optimal way. For example in determining a node (interpreted say, as a hospital) that is within a fixed distance δ away from another node (say a mine), we execute a circular search operation for those nodes designated as hospitals that are within a radius δ from the mine. The search region is approximated by the grids that either cover or are in the neighborhood of the mine. The grid cells correspond to buckets into which points, or labeled chain of points, are stored. The quadcodes of the cell are maintained in a B-tree index for fast access to the buckets. This technique simplifies retrievals of spatial objects.

The following relationships are the basic set in our spatial model and is independent of the application. The model is still extendible in the sense that it can accommodate further relationships if a user wishes to add to them.

LINENDS The relationship LINENDS(Nid, Lid, StartEndFlag) gives the relationship between Nodes and Lines, i.e., those nodes that define the start and end points of line segments. The attributes are:

Nid The node identifier. This is the key of the NODE entity class.

Lid Line identifier - the key of the LINE entity class.

StartEndFlag A 0/1 flag that indicates whether the node is a start or an end of a line. A value of 1 specifies that Nid is the start node of the line identified by line Lid. A value of 0 indicates that it is an end node.

A node may be associated with at least 0 and at most N line segments. A line is associated with exactly 2 nodes.

LAYNDS The relationship LAYNDS(Nid, LyCode, RVMode, Fcode), gives the association of nodes with the layers they belong to. The nodes are points that are interpreted as a geographic feature, e.g., a mine or an oil well. The attributes are *Nid* from the NODE entity class and *LyCode* from the LAYER entity class. Other attributes assigned to the relationship are:

Fcode: A feature code that states what type of feature the node is designated as.

RVMode: A flag that indicates whether the node must be considered as real or virtual.

OVPASS This is a relationship class of the set of lines that pass over or under another without intersecting it. The attributes are:

OverLid: The identifier of the line segment that passes over another line.

UnderLid: The identifier of the line segment that passes under another line.

Xcord: The X coordinate value of the overpass point.

Ycord: The Y coordinate of the overpass point.

ZcordO: The Z-coordinate value of the line given by OverLid.

ZcordU: The Z-coordinate value of the line given by UnderLid.

A line can participate in at least 0 and at most N such relationships.

LINENCLS This is a relationship that specifies lines enclosing or containing other lines and what lines are enclosed or are contained by other lines. The attributes of the relationship are:

ExclsLid: The identifier of the line segment that encloses other lines.

InclsLid: The identifier of the line segment that is enclosed by another line.

IncExc: A flag indicating whether the enclosed line is an inclusion or an exclusion.

LyCode: The layer code in which this relationship between the two lines occur.

A line may enclose 0 or N other lines and it may be enclosed by 0 or N other lines.

POLYLINES The relationship POLYLINES(Pid, Lid, PolyLineOrient) associates a polygon with its bounding lines. The attributes are defined as follows:

Lid: The identifier of a line that bounds a polygon.

Pid: The identifier of the polygon that the line is a boundary of.

PolyLineOrient: Each line has a direction that goes from the start node to the end node. *PolyLineOrient* is an attribute that gives the orientation of the polygon with respect to the direction of the line (i.e., whether the polygon is to the left or to the right in the direction of the line). If the polygon is to the left *PolyLineOrient* takes the value 0 otherwise it takes the value 1.

POLYNCLS This is similar to the LINENCLS relationship. It is a relationship that associates polygons with their enclosed polygons. The attributes are:

LyCode: The layer code in which this relationship between the two polygons occurs

ExclsPid: The identifier of the polygon that encloses other polygon.

InclsPid: The identifier of the polygon enclosed.

IncExc: A 0/1 flag that indicates whether the enclosed polygon is to be considered as excluded from the enclosing polygon or included.

A polygon may enclose at least 0 and at most N other polygons. A polygon may be enclosed by at most 1 polygon. The relationship is hierarchical and can be used to define the hierarchical representation of spatial entities defined by such storage structures as the Quad-tree [14, 47, 19], the Grid-File [32], the R-tree [23], the Cell-tree [22] and the BM_HEX [37].

The hierarchical relationships embedded in the linear quad-tree encoding of raster data [19], hex-trees [3] can all be considered as special cases of the POLYNCLS relationship. Although each of these schemes can be defined as a hierarchical relationship between a cell and the immediate enclosed cell and consequently, stored in relational tables, we do not recommend this. The structures can be stored more efficiently by exploiting their hierarchical representation. Answering spatial queries directly using these structures is more efficient than using the relational equivalent representations.

LAYPOLYS This relationship defines the association between polygons and the layers they appear in. The attributes are:

Pid: The polygon identifier.

LyCode: The code associated with the layer.

Fcode: The feature code of the polygon.

RVMode: A flag that indicates whether the polygon must be considered as real or virtual within the layer.

LAYLNS In some special case, a line may not necessary form the boundary of a polygon in which case it can be associated directly with a layer. Essentially, those lines that do not form boundaries of polygons but simply lie within a layer establish this relationship. The attributes are;

Lid: The identifier of the line.

LyCode: The layer code of the layer.

Fcode: The feature code associated with the line.

ANNOTXT The text or symbols that appear on each layer of a tile are bound to their respective display via the ANNOTXT relationship. The attributes are:

SymbKey: The character or symbol to appear on the layer;

LyCode: The layer code and tile identifier of the layer the symbol appears on.

Xcord, Ycord: The X- and Y- coordinate values measured in *Map Reference Coordinates*, for the placement of the Symbol or text.

AbsRel: The mode of measurement in the map reference coordinate - either absolute or relative.

Size: The size of the character or symbol.

Angle: The angular orientation from the horizontal in which the symbol is to be placed.

Color: The color to display the symbol or text in for the given layer.

SymbFunct: The function or procedure for constructing the symbol.

TILECVGS This relationship is not defined for a tile. It simply defines the association between a coverage and the set of tiles that form the coverage at the smallest map scale in the database. The association is defined by the pair of a tile identifier, *Tid*, and the *CvgKey*.

There are other relationships that may be stored but these are not particularly essential to the basic model. These include:

1. **TILELAYER:** This is the relationship between a tile and a layer. Since the LAYER entity is a dependent entity on a TILE and the key of LAYER is composed of the *LyCode* and *Tid*, there is no need to represent this relationship explicitly. We also recognize that the relationship is embedded in the directory hierarchy of the tiles and need not be stored.

2. **TILEHIERARCHY:** This defines the hierarchical relationship between a TILE and its subtiles in the hierarchical decomposition of the coverage into quad cells. It is not necessary to store this explicitly since the hierarchical relationship is implicitly encoded in the quadcodes for each tile.

Notwithstanding the above discussion, the user may still extend the model to include other relationships that he or she may want to maintain. This aspect is what we express as the extendibility property of the model. The model is not only extendible from the fact that new relationship can be added, but the attributes that define the entities and relationships may be expanded as along as the relations they map to remain in third or Boyce-Codd normal forms [25, 12].

What we have established here is the basic framework of a consistent spatial data model for a GIS database. The decision to extend the model with further relationships should be weighed against the cost of processing the base tables of the proposed model to extract the needed information. It is not clear if the spatial model presented is minimal and complete in the sense that it generates the minimal set of relations required to answer any 2-dimensional spatial query optimally. This is still an open question.

4.5 Extensions to 3-dimensional Spatial Models

Although we do not address 3-dimensional modeling in this paper, we maintain that the spatial model presented can be easily extended to include digital terrain modeling(DTM) or digital elevation modeling (DEM). A 3-dimensional spatial data model is studied in detail in [40].

There are two main methods of maintaining the data for digital elevation modeling in GIS. These are either as a triangulated irregular networks (TIN) or as a grid-array of elevation points.

Data for digital elevation can be generated from contour (or iso-line). The main idea of the 3-d modeling is to assume that one layer of a tile consists only of iso-lines or alternatively a triangulated irregular network. The triangulated tile consists of straight line segments that form triangular facet. Each facet is bounded by exactly three straight line segments, and has attributes of slope and aspect. The spatial modeling of the geometric space of a triangulated polygonal region is similar to the 2-dimensional case discussed above. The major difference in that the Z-coordinate value of each point is now particularly significant. The triangular facets are then patched to form surfaces of solids or more specifically a terrain. Surfaces are can then be corner-stiched to form volumetric objects.

Some objections have been raised in the use of grid-array of elevation points for DEM [9]. The argument is that it generally requires more storage than TIN. Nevertheless it continues to be the popular method of collecting DEM data. In both cases, either TIN or grid-array of elevation points (GAEP), the amount of data manipulated, is normally limited to the size of available main memory. A third method that we are currently studying is the use of quadtree-like tessellation method to generate regions of approximate very small variation in the relief. A variation of this method is a hybrid-scheme where the quadtree-like partitioning generates a first level of rectangular cells. The terrain data in each cell may then be stored as a grid-array of points.

5 Mapping the Spatial Model onto Relations

5.1 Straight Translation from the Spatial Model

The figure 6 shows the entity relationship diagram of the 2-dimensional topological spatial data model of a tile. A straight forward translation from the EER-diagram gives the following relations:

1. NODES (Nid, Xcord, Ycord, Zcord);
2. LINES (Lid, NoPnts, Chain);
3. POLYGONS (Pid, PcntX, PcntY, PcntZ, MinElv, MaxElv, NoLns, BndLns);
4. LAYERS (LyCode, Tid, LyName, DispPriority);
5. LBLTXT (SymbKey, String, StrLen, Style, Font);
6. SYMBOLS (SymbKey, SymbName, Fcode, Scale, SymbMethod);
7. LINENDS (Lid, Nid, StartEndFlag);
8. OVPASS (OverLid, UnderLid, Xcord, Ycord, ZcordO, ZcordU);
9. LINENCLS (LyCode, ExclsLid, InclsLid, IncExc);
10. POLYLINES (Pid, Lid, PolyLineOrient);
11. POLYNCLS (LyCode, ExclsPid, InclsPid, IncExc);
12. LAYNDS (LyCode, Nid, RVMode, Fcode);
13. LAYLNS (LyCode, Lid, RVMode, Fcode);
14. LAYPOLYS (LyCode, Pid, RVMode, Fcode);
15. ANNOTXT(LyCode, SymbKey, Xcord, Ycord, AbsRel, Angle, Size, Color);

There are about fifteen relations generated from a direct mapping of the EER-diagran onto relational schemes. Each relation is at least in third normal form. There are three relations for which a slight modification to the meaning associated with the attributes will cause then to be combined. In the relation

LINES ($\underline{\text{Lid}}$, NoPnts, Chain);

the following functional dependency holds,

$$Lid \rightarrow NoPnts, Chain.$$

In the relation

LNNDS ($\underline{\text{Nid, Lid}}$, StartEndFlag);

we have the following functional $\{Nid, Lid \rightarrow \textit{StartEndFlag}\}$. If we define the pair (Nid, 1), as a start node (Snid) and (Nid, 0) as an end node (Enid) we have the following FD.

$$Lid \rightarrow Snid, Enid.$$

By similar reasoning we can define in the relationship

POLYLNS ($\underline{\text{Lid, Pid}}$, PolyLineOrient)

the pair (Pid, PolyLineOrient) as a right polygon (*RightPid*), whenever $PolyLineOrient = 1$, and as a left polygon (*LeftPid*), otherwise. The FD, $\{Lid \rightarrow RightPid, LeftPid.\ \}$, holds. With these changes, we can combine the three relations, LINES, LINENDS and POLYLNS into one giving

LINES ($\underline{\text{Lid}}$, Snid, Enid, RightPid, LeftPid, NoPnts, Chain)

Having defined the geometry of the space, we introduce attributes in some of the relational schemes to associate each feature to the tables of feature classes. These tables contain attributes that are specific to the features. The set of relations derived for each tile now reduces to those shown in Figure 7.

The first 12 relational schemes of figure 7 form the set of basic relations that define the spatial topological model of a tile in the database. The last three are added to show how symbols, labels and annotations can be placed on a map. It is important to note that we have not defined relational schemes for the feature classes. We leave the definitions of these feature class tables to the users and data generation agencies. The specific attributes in these feature class tables would be defined by the geomatic agencies responsible for collecting and maintaining the required information. For example, the feature class tables for soils, vegetation, crops would be defined by the department of agriculture. The feature class tables for lakes, rivers, streams, etc., would be provided by department of inland waters and similar tables for roads and highways would be defined by the department of transportation.

Given the model above, it is easy to see how to address the problem of data exchange between two GIS environments using vector data. Assuming then that every GIS retains the basic set of relations, defined by the spatial model, in a relational database management system. Then, as long as the underlying DBMS of one GIS software can access information from the relational tables of the other GIS the problem of data exchange in a heterogeneous GIS environment is implicitly resolved. The model so far has addressed only the spatial model with respect to vector data. Raster data can be similarly handled by encoding the image of a tile in raster format.

There are other relations that are essential to the description of the data for a GIS database beyond the spatial topological model for one tile. These define the global information and relationship between one tile and another. We describe these in the following subsections.

5.2 Other Ancillary Relations

The data model so far has described the details of how to represent the vector information of one tile. Each tile is related to a coverage as a sub-component of it. A coverage is composed of a set of tiles. A tile has a map scale associated with it and may have special labels, symbols, a legend. The tile and the relationships between them are formed into a catalogue of the GIS database. The catalogue consists of the three relational tables. There are tables that contain global information relevant to all tiles in the database. Others maintain information on how the digital data was compiled, the feature codes relevant to the domain of spatial coverage and the geographic names of named features.

Examples of the relational tables that maintain global information in our model include the EMR feature code table and the toponomy database. We represent these as the relations EMRFCODE and GEONAMES respectively. The relational schemes defined below give some of the relations of the model. The attributes assigned to the relation TILES should not be considered as universal. These should be considered as typical of what may be required for a tile. The illustration given here is relevant to the scheme used for compiling vector data for maps at Energy Mines and Resources, Canada. The meaning associated with the attributes in these relations are explained subsequently.

1. TILES (Tid, NtsCode, TileName, Scale, Utmz, Dim, RegPnts, MinMerid, MaxMerid, MinPar, MaxPar, TileDir, MaxNid, MaxPid, NaxLays);

2. COVERAGE (CvgKey, CvgName);

3. TILECVGS (Tid, CvgKey);

4. CMPINFO (Tid, NtsCode, Datum, Prjtion, CmpDate, SrcScale, Dpi, Device, CmpMethod, RevDate, Accuracy, Comment);

5. GEONAMES(Tid, ProvName,Fkey, GenCode, StatusCode, Long, lat, UTMGrid, GazetteMap, locate1, Locate2, LocDescript, NtsCode);

6. EMRFCODE (Fcode, ElemType, Weight, Level, Color, Style, Stdl, Sntl, AscCmd, FeatureName);

Some of the attributes used in the above tables are defined below:

Tid: The quadcode associated with the tile;

NtsCode: The National Topographic Series number with which the tile (map sheet) is referenced;

TileName: A name assigned to the tile;

Scale: The scale of the map, one of {250000, 50000, 25000, 20000, ... 1000 };

Utmz: The primary UTM zone in which the tile lies;

Dim: The dimensionality of the coordinates of points in the data. Valid values are 2 and 3;

RegPnts: Four triplets of values that define (X, Y, Z)-coordinates of four registration corner points of the tile.

TileDir: The directory path where all the information specific to the tile are stored;

MinMerid: The minimum meridian value;

MaxMerid: The maximum meridian of longitude of the tile;

MinPar: The minimum value of parallel;

MaxPar: The maximum value of parallel;

MaxNid: Maximum Node identifier value assigned to the tile;

MaxLid: Maximum Line identifier value assigned to the tile;

MaxPid: The maximum polygon identifier value assigned to the tile;

MaxLays: The maximum number of layers available for the tile

CvgKey: The unique identifying key of a coverage.

CvgName: A name assigned to the coverage.

⋮

5.3 Naming Convention

Consider a tile named "Sample" and assuming that the file system under which the GIS runs is hierarchical. Then using a UNIX-like approach to express the directory path, we associate a directory name "∼/Sample" to the location where the data for the tile named "∼Sample" will be stored. More precisely, "∼/Sample" has the following subdirectories: "∼/Sample/Vec", "∼/Sample/Att", "∼/Sample/Ras", "∼/Sample/Dem", "∼/Sample/Txt". These subdirectories respectively hold the vector data, attribute data, raster data, data for digital elevation models and free text information. The schematic diagram of the directory and its subdirectories is depicted as in figure 8.

A tile is located by specifying the quadcode, the NTS map sheet number or the bounding longitude and latitude values of the southwestern and northeastern corners. The result of the spatial index search is the directory name where all the subdirectories of the various categories of data sets are maintained. Besides identifying the locations that hold the various data sets the names of the relations under each subdirectory are predefined. For instance the names of the relations held under the "∼/Sample/Vec", are exactly the relations described in section 5. We do not elaborate further on the naming conventions since the specific details depend on other limitations imposed by the file system and the naming conventions of the database management system systems used. One important aspect of the model is that the meanings associated with the point, line and area features are defined in a set of independent attribute tables.

5.4 Associating Attribute Information to Spatial Features

The spatial model describes the geometric properties of the features. We should remark that the tables that describe the topology of the space has columns that are termed *attributes* in database parlance. This should no be confused with the relational tables that contain attribute information. The information in these tables are related to the set of relations that define the geometry of the space.

The attribute tables define the interpretation given to the spatial features. Associated with each feature are other attribute information that describe significant characteristics of the features. For example, in the schema definition of the polygon entity class POLYGONS, and the relationship LAYPOLYS, we have

POLYGONS (Pid, PcntX, PcntY, PcntZ, MaxElv, MinElv, NoLns, BndLns);
LAYPOLYS (LyCode, Pid, RVMode, Fcode, FClassName);

The polygon entity may represent a lake, a forest stand, wetland, vegetation type, a county, etc., depending on the scale of the map and the theme of the layer it belongs, Note that the same node, line or polygon feature may be accorded different attribute information depending on the layer. The relationships LAYNDS, LAYLNS and LAYPOLYS can map the same geometric object to different features in different layers of the same tile.

Suppose a polygon, identified by the Pid, is interpreted as a lake. This is obtained from the value of the feature code field (Fcode). The relation given by the value of FClassName = LAKE, will store specific tuples for lakes. The relation LAKE may be defined as

LAKE (Pid, LyCode, Fcode, NameKey, NameOfLake, AvgDepth, Salinity, Area, ...)

The specific information on a lake, whose Pid is given, can be obtained from the relation LAKE by selection on Pid. If the polygons are to be interpreted as forest stands, then the value of FClassName = "FORESTSTND". The types of trees, the average age of the trees, etc., may be of interest. Here the polygons are related to an attribute table of forest stands in the following manner. First a relationship FORESTSTND(Pid, LyCode, Fcode, StandNo) is created. This links each polygon with a stand number. Second, the table *STANDS(StandNo, Species, Age)* is created to define the characteristics of each forest stand. The relationship FORESTSTND, associates a tuple in LAYPOLYS to a tuple in the STANDS table.

The illustration just presented gives the general framework for associating attributes to spatial features. Given the relational tables that constitute the base maps, an application developer interested in using the base maps only has to augment these with the appropriate attribute information required for his or her application. Assuming then that the databases of feature class tables are already provided by the various agencies, this exercise is simply that of subschema definitions. The spatial features and their attribute data, maintained in separate tables, can always be appropriately related to each other by the relational JOIN ($\bowtie$) operation.

In practice, the exercise of associating attribute information to spatial entities is driven by the intended application. For example in an application using road network, information such as the number of lanes in a segment of a road, the average width of a road, the surfacing material, the road names etc., are examples of attributes that may be relevant to the application and must therefore be retained and related to the line features of roads by the line identifiers (Lid).

The approach of associating attribute information may be extended to other information categories such as free text, still and dynamic raster data. The same idea may be carried over to rules that trigger observable behaviour of features.

6 Illustrative Example

We illustrate the main ideas of the topological spatial model with a simple example. Suppose figure 5 is a thematic layer of some tile named *Sample*. The essential relations required to maintain the topology of the vector data are shown in the tables below. We will ignore the Z-coordinate values in this example.

In general a spatial queries can be answered by the use of the relational algebra on the schemes provided. While current relational query languages such as SQL and QUEL do not address some of the problems specific to spatial queries, two avenues my be followed to provide such capabilities:

1. One could utilize a standard query language embedded in a host language to support specific spatial queries in an application. This approach requires that we provide functions specific to processing spatial query that cannot be directly expressed in a relation algebra. The functions may make explicit use of embedded query language statement. For example, to answer a circular query such as the nearest hospital to an accident site, this can be formulated as a nearest neighbour search for a node feature whose associated feature code

defines a hospital. The nearest neighbour search for point features can be done by first constraining the search window to an orthogonal range search. This can be expressed using the relational algebra or calculus.

2. A second option is to augment a standard query language with special operators so that spatial queries can be expressed directly. This requires that the language be extendible in the sense that new operators can be defined.

A number of ongoing research projects [36, 53] have adopted the latter approach. We recommend the former at the present time for two main reasons:

1. There is such a broad range of spatial relationships and criteria for expressing queries that developing a language that is complete may be difficult if not impossible, let alone to achieve an implementation efficiency that can meet both normal data processing activities and GIS functionalities.

2. The problem of efficient processing of spatial queries is dependent also on the use of efficient spatial data structures for organizing and accessing the stored data. This has to be reconciled with the predominant simple indexing method, such as B-trees and Conventional hashing method, used in commercial database management systems.

The question of whether the basic set of relational scheme in our design is complete is not formally proven yet. Some further work is required to show this. On the other hand, we conjecture that the set of relations is complete. A thorough study is not expected to drastically modify the design philosophy used here either by the addition or deletions of other relational schemes. We have not yet encountered a query in 2-D space that cannot be efficiently processed with the basic set of relations stated. In the worst case, we augment the basic set with relationships that resolve the efficiency problem.

6.1 Answering Spatial Queries

It is difficult to anticipate all the various types of queries a user is likely to ask. Nevertheless, within any application the user can always augment the basic set with further relations that define feature classes or relationships between spatial entities. In the absence of attribute tables of the feature classes, queries that enquire only about characteristics of spatial entities and the defined topology of the space are easily answered. The following are some examples.

- *Determining the coordinates of the end points of line.* Given a line identifier Lid = "LX", the end points of this line can be determined using relational algebra statement below.

$$\Pi_{Nid,Xcord,Ycord,Zcord}(((\Pi_{Snid}(\sigma_{Lid=LX}(LINES)))$$
$$\cup(\Pi_{Enid}(\sigma_{Lid=LX}(LINES)))) \bowtie NODES)$$

  In words this is derived by first forming the union of two separate projections on *Snid* and *Enid* in turn on a selection on LINES where Lid = "LX". The result is a unary relation say ENDPOINTS with two tuples. The attribute is of the same type as a node identifier. The projection of the join of ENDPOINTS and NODES onto Nid, Xcord, Ycord and Zcord gives the required result.

- *Determining the bounding lines of a specified polygon.* This is similar to the above query except that now we first take a union of those lines where the given polygon identifier (Pid) occurs either as a right polygon (*RightPid*) or as a left polygon (*LeftPid*).

- *Determining the lines that intersect at a given node.* Given node identifier *Nid = "NX"*, the lines that intersect at that node "NX", can be derived as the union of those lines such that either Snid = "NX" or Enid = "NX", in LINES.

- *Determining the immediate enclosing polygon of a Node.* This is a much more difficult query to process efficiently with a relational algebra expression. The essential idea in formulating such a query is to determine any one point of the bounding lines of the enclosing polygon. This can be done by a nearest neighbour search of the node to find any point of a line closest to a given node. The plumb-line algorithm is then used to confirm enclosure. It should be noted that the space is partitioned into disjoint polygons. The method for processing such a query is discussed below.

A relational algebra formulation using a *Theta-Join* can be used in the nearest neighbour search. Once a closest line is found, the polygons, and their bounding lines, that are either to the left or to the right of this lines can be determined using selection and projection. The rest of the query computation is carried out using plumb-line test. This, unfortunately, cannot be done by relational algebra expressions. A special function is required for this computation.

The basic access methods (B-tree, or conventional hashing) do not provide efficient retrieval for this problem. This an example of the situation where the spatial access method is significant. Recall that in section 4, we alluded to the fact that the quadcode is utilized for bucketing, i.e., the storage of points in buckets or data pages. Within each bucket, a line passing through that bucket is stored as a point-set. Each point-set is labeled with the line identifier it belongs to. An isolated node has only its node label. A point-set is part of a line chain. The quadcode method is used as an index into the bucket that store nodes and point-sets. This simple technique gives an efficient means of processing queries such as the one stated above.

6.2 Some Operations

Besides being able to answer queries using the relational query language, there are other operations that require the use of combined retrieval functions of the database management system and other utilities and system modules. For example the graphical interface will be combined with the retrieval functions to carry out operations such as:

1. *Displaying all features from all layers*;

2. *Displaying features of a particular layer*;

3. *Overlaying two or more layers*;

4. *Deleting the partitioning line between two polygons*;

5. *Modifying a line chain*;

6. *Assigning attributes to a node, a line or a polygon*;

7. *Copying an arbitrary region of a map*;

8. ⋮

Current GIS provide different submodules with which a user can perform certain special functions such as editing, network analysis, displaying the results of map attributes, 3-D modeling, etc. While such modularity in the overall GIS architecture is a welcome idea, the switch from one module to another should be smooth. This implies that once a user invokes a data set for any module, the switch to other modules for the same geographic area should not require further explicit data loading for the module. Rather it should be done implicitly since the geographic region of interest is known, unless a different geographic region is required. Even then, one can either browse through the space until an area of interest is located or the bounding region is specified directly. Invoking an analysis function (or module) should automatically load the appropriate data set for the intended mode of operation.

7 Some Spatial Topological Data Models

The quest for a spatial topological data model has been a challenging task ever since the early development of computer cartography. Over the years, a number of spatial data models have been proposed. These include [2, 31, 13, 20, 7]. Talk of accepting a common standard data format for the exchange of digital map data is currently being addressed. We believe the acceptance of a standard topological data model must be a prelude towards such standardization efforts for digital data interchange. As a result of the recent advances in Computing Technology and the requirement of more functionality in GIS today, some of the proposed models have been revised to address data representations that cater best to new analysis techniques and functions. Notable among these proposed models are GBF/DIME, DLG-E [7], CCOGIF [2], MDIF [33], ARC/INFO-Model [31], DIGEST [20], VPF [13] and Intergraph's TIGRIS[24]. We discuss some of these and compare them with our proposed model in Delta-1. The Delta-1 model evolved with a study of some of these models particularly CCOGIF, previously called the CCSM format [34], DLG-E and ARC/INFO.

We distinguish between models designed for the representation of the spatial and attribute data in a GIS environment and those designed specifically for data exchange, i.e., storing of the data on a suitable media (e.g., tape or removable disk) for transfer purposes. For those data format design for data exchange purposes, these must be translated to a stored form of the actual model in a particular GIS environment before being processed. One of the principal objectives of this paper is to show that given an appropriate model for efficient processing in a GIS, the model can be exchanged using conventional ISO EDP file exchange formats which have been well established. In particular the stored data can be easily accessed over a network or involved in distributed processing. This should overcome the problem of continuously utilizing valuable computing resources for data conversion in GIS installations. We briefly discuss some of the early works on the subject of data models. For more details in the area, the reader is encouraged to consult the references.

7.1 The GBF/DIME data Model

One of early topological structured representation of digital map data is the GBF/DIME system. The acronym GBF/DIME stands for Geographic Base Files using Dual Independent Map Encoding. The system, developed in the early seventies by the US Bureau of Census, was used for map data that included street networks, addresses and political boundaries. The representation was not accurate enough to represent curved lines or arcs. About 1988 a new model, called the TIGER (Topological integrated Geographic Encoding and Referencing) was proposed to replace the GBF/DIME model.

Beside being topologically structured like its predecessor, the TIGER file is able to merge GBF/DIME and USGS DLG-3 (described subsequently) as seamless data sets. Attributes in the TIGER files include feature names, political and statistical geographic area codes, census tracts, block numbers, ZIP codes, etc. Location of a point is measured in geographic coordinates (Longitude and latitude). There are about six record format in the TIGER model and these correspond, respectively, to the record formats of six separate files of the model. The transfer format is simple a collection of these file.

7.2 The DLG-E data model

The DLG-E[7] is an enhancement of topological structured digital map data put out by the National Cartographic Information Center of the United States Geological Survey (USGS) called DLG. There are two DLG formats, DLG-S (standard or 144 byte record format) and DLG-O (optional or 80 byte record format).

The DLG model defines three principal feature elements: nodes, lines and areas. In Delta-1 an area feature is termed a polygon. Attributes are encoded as numeric data. These are used to establish the links between the actual records that hold the attribute information and the spatial objects. An attribute code is defined as a pair of major and minor codes. The major code defines the category of the feature, i.e., road, stream, etc., while the minor code defines attributes specific to the feature. A feature may have two or more attribute codes. Originally, the DLG format was designed as a collection of files for the purposes of digital data transfer.

Its successor, DLG-E [7] however, has been designed with ample considerations to the form for retaining the model under a database management systems as well as the format for data exchange within the U.S. National Mapping Divisions. The design of DlG-E is described from its conceptual model, through its logical and physical models to an implementation.

Digital Line Graph Enhanced has the same design philosophy as the architectural framework in Delta-1. That is, that cartographic data must be centered about a spatial database which is itself a multifaceted model of geographic reality. At the conceptual level, the model identifies the following distinct entities and relationships:

Entities Data set, theme group, theme, feature object, spatial objects - points, lines and area.

Relationships Composition, Network Membership, Interactions, Bounding and Bounding Entities

At the implementation level, the model is translated into physical file structures, called DLG-O+ files. DLG-O+ extends the DLG-O files with modules. There are about eight base relational tables that define both the spatial features and non-spatial attributes of features. Unfortunately, techniques for spatial indexing to achieve efficient processing at the implementation level is not addressed. The transfer format for DLG-E is expected to be a collection of sequential files with strict adherence to the U.S. Spatial Data Transfer Format (SDTF).

7.3 The CCOGIF data model

A model of a topological structured data is being proposed by the Canadian Council on Geomatics for the purposes of digital data interchange between the geomatics agencies and institutions. CCOGIF is an acronym for Canadian Council on Geomatics Interchange Format [15]. This format was previously known as the CCSM format [34]. As the name implies, it is intended purely for data transfers. Canadian Council on Geomatics, CCOG, puts out a set of software modules for the reading and writing of data intended to be transferred in this format. Current status of the development is a preliminary implementation. This version of the software package is available in ANSI Fortran'77 for a VAX machine. The data is assumed to be resident on a 9-track tape media with a blocksize of either 2048 or 9126 bytes. Given the current predominance of workstations, particularly Unix based workstations the present release is too restrictive for the majority of end users.

The conceptual model in Delta-1 was formulated with the view to subsume the information contained in an earlier specification, CCSM format [35] which was the forerunner to CCOGIF.

The relationship between Delta-1 model and CCOGIF is analogous to that between DLG-E and DLG. Like DLG, CCOGIF defines a topological structure of features. The elements of spatial features are points, lines and polygons. The format defines information in units of volumes. A volume contains a number of *data sets*. Each data set consists of a collection of *data groups.* Each data group is comprised of a number of *data themes.* A data group is preceded either by a header file or a header record. A digital map data of defined geographic extent may span more than one physical tape volume.

In many respects the Delta-1 (Δ-1) model is to CCOGIF as DLG-E is to DLG. In this paper, the conceptual levels, through to the logical level definitions are exhaustively covered. We stop at the logical level where this aligns with the data model of the underlying DBMS. A prototype implementation using the INGRES as the base relational DBMS is currently in progress. A translation of the model into an object oriented database implementation is discussed in [39]. The conceptual level of the Δ-1 model is described for 2-dimensional data using an EER-Diagram. The set of relations to be stored as the entities and relationships are also defined. In particular, the mechanism for associating the spatial features and their defined topological to the non-spatial attribute information is also explained. The model is sufficiently open-ended to allow for the inclusion of further relationships and linkage to other geomatic database that maintain attribute information on geographic features. Viewed in this way, the Δ-1 model is an enhancement to CCOGIF that allows it to be used as a model for retaining the digital data in a GIS environment.

To maintain the current status of CCOGIF implies that for direct online distributed access, the software modules provided must serve as interfaces at both the communicating and the receiving sites. The sending site uses the interface module to generate the ASCII equivalent of the data resident at that site. The receiving site uses the interface modules to interpret the data element received in binary format and communicates this to the user's running program. The assumptions made at the time CCOGIF was conceived and designed is currently outdated. It will require major revisions to attain the high level of modeling capabilities embodied DLG-E if it is to be considered serious candidate as a standard interchange format for GIS applications. Fortunately, this is where the Delta-1 data model provides an answer.

7.4 The ARC/INFO data model

The ARC/INFO model [5, 31] closely resembles the Delta-1 model. In fact this model together with CCOGIF and DIGEST influenced, greatly the development of the Delta-1 data model. The ARC/INFO data model defines four primary features of a coverage. Namely, arcs, nodes, polygons and label-points. A coverage is stored as a directory under the same name. A coverage has a set of files within the directory. A number of coverages within the same geographic extent define what is termed *a Workspace*. Some typical files associated with a coverage are:

ARC: A file that describes the arc coordinates and the topology.

AAT: The arc attribute table.

LAB: Label point coordinates and topology.

PAL: Polygon topology.

PAT: Polygon/Point attribute tables.

⋮

A *Workspace* has a set of relational tables for the attributes of the features under the coverages. The relational tables are maintained by the INFO portion of the system. INFO is the a relational DBMS that maintains the attribute information. While no formal theoretical justification is reported on why the content of the tables are structured the way they are, it has been reported that the design is topologically sound. It appears that there is some degree of redundancy in the content of the files that hold topology of features. However, this is justified on practical grounds. They help to achieve efficient processing of the spatial elements which may otherwise be difficult.

The model does not fit our architectural framework as suggested in our earlier discussions. It is a form of a hybrid scheme in which the spatial elements are maintained as a collection of files while the non-spatial data or attributes information are maintained by a DBMS. This separation can be attributed to the fact that it is one of the early GISs long before efficient databases were developed for use with spatial data. This makes it difficult to easily migrate to a non-homogeneous distributed processing environment, that has ARC/INFO GIS as a node in the network. Such a network of heterogeneous GIS is likely to be the configuration of the future where different geomatic institutions maintain their databases with different database management systems. Another problem created by the ARC/INFO independent duel modeling of the spatial and attribute information is that syntax of the relational queries on the attribute data do not carry over onto the spatial data. Typically one would like to use the same syntax for expressing queries on both the spatial and non-spatial data. Although the architecture maintains a map library with access to the coverages via a name index, it lacks the indexing scheme for the horizontal and vertical navigation of the coverages.

7.5 The DIGEST data model

The DIGEST model is a one of the most detailed described and documented data model for geographical data encoding, storage and exchange. DIGEST stands for DIgital Geographic Information Exchange Standards. It defines both a representational model in a GIS environment as well as an exchange format. Spurned by the efforts of some members of NATO to provide a standard digital data exchange format, the model attempts to accommodate the standards already existing in the participating countries. The model, as we perceive it currently, is an amalgamation of a vector based topological structured and non-topological structured (or Spaghetti) data, free text attribute and structured text data of the spatial features and a matrix or raster formatted data (i.e., still and video images).

One unique characteristic of the model is that it associates level of security to spatial features as well. Unfortunately, it does not go far to recommend digital data encryption standard. A rather more serious omission in the model is lack of the ability to model the behaviour of spatial entities. Given the degree of details in the specification of the preliminary report for geo-referenced information and conformance to ISO standards, it appears that a recommendation for an adoption of some graphics standards should be made. The suggested exchange media for DIGEST include magnetic tapes and CD-ROMs.

The format is projected to be implemented either as a relational database or an object oriented database. A prototype implementation of the DIGEST model by ESRI is a relational database where this is mapped onto the ARC/INFO Vector Product Format (VPF) data model [13] designed for the Digital Chart of the World. Given the high volume of data generated for topological structured and raster data, standard format for data transfer using high capacity storage media such as Digital Audio Tapes (DAT), CD-ROMs, etc., should also be addressed.

Further details of the DIGEST model, design and implementation are discussed in [21, 20].

7.6 The TIGRIS data model

The TIGRIS model is a topological structured data model that relies on object oriented programming approach for its implementation. The term TIGRIS stands for Topologically Integrated Geographic Information System. It is the first model that has alluded to fact that in modeling spatial features in GIS databases behavioral information must also be stored [24].

The model supports the storage and manipulation of complex objects, features (composed of points, lines and area), multiple representation of features and arbitrary feature to feature relationships. It uses techniques of algebraic topology for efficient computation of some spatial queries. The defined objects in TIGRIS are maintained by an Object Manager that combines database management functions and process control in an object oriented design and implementation environment.

The building blocks in the topological definitions consist of points (i.e., nodes or 0-cells), nonintersecting curves (edges or 1-cells) between points and connected two dimensional areas (faces or 2-cells) bounded by curves. The relations between these topological objects are based on the notion of *boundary*. The boundary of a node is empty; that of an edge consists of the two end points of the edge and that of a face is the set of all nodes and edges bounding the face.

The basic spatial feature in TIGRIS corresponds to a fundamental geometric entity of a node, an edge or a face. These are composed in a hierarchical manner into more complex and abstract feature objects. The objects fall into well defined groupings or classifications. The classes provide the data structures with which schemas are built. A the schema definition, a cartographic entity is defined by three components of information: the name of the base class, a list of attributes and links to other topologically connected objects. The modeling techniques used provide efficient processing of spatial queries that identify relationships such as intersection, connectivity, containment, etc., between objects.

Although the use of object oriented programming techniques provide a better secure, reliable and easy modifiable software module for the implementation of TIGRIS, the claim that the implementation uses an object oriented database to manage the objects is questionable. Given the high volume of data normally generated with topological structuring of GIS data, it is not clear how efficient persistent storage has been used to manage the large number of objects in a map database to achieve a high level of processing efficiency. Further, the processing of attribute based queries to locate objects that satisfy certain criteria may have to rely on the use of inverted indexes.

The lack of consensus in the database community as to how an object oriented database system should be characterization makes the use of object oriented database systems a dicey exercise. Current systems are too specialized and often custom tailored to address specific applications. Although they provide a semantically rich modeling tools, the claim for performance improvement over the relational model is not yet satisfactorily justified. The TIGRIS model raises other questions of data integration in a heterogeneous GIS environment.

8 Conclusion

Given the topological modeling technique outlined, it is easy to see why an acceptance of a common spatial topological data model must be a prelude towards an acceptable standard for a GIS data exchanges. By ensuring first that the spatial data models used by all GIS data collection agencies conform to some common standard, any agency can take the liberty to map the model onto any implementation form that best serves its role in the geomatic community. The problem of exchanging data reduces to that of intercommunication between the Database Management Systems. The use of concepts such as the Open/SQL idea in the INGRES DBMS is worthy of serious consideration.

It is implicitly accepted then that once the data model is adhered to, the information content of any category of data in the database, will be the same irrespective of the DBMS model. In this manner a coherent and consistent view of the data for any GIS application running at any site is achieved. The main idea resulting from this exercise then is that the Delta-1 model, under its architectural framework, facilitates distributed processing and distributed access of GIS data in a heterogeneous GIS network environment.

This papers has addressed the issues of spatial topological data modeling for 2-dimensional vector data. Its extensions to 3-dimensional vector data is easy to realize. The details of the 3-dimensional model warrants that it be treated in an independent paper. For the same reasons we address how other data categories are integrated into the model in a series of follow-up papers. More specifically, we examine details of incorporating raster data and free text into the Delta-1 GIS database later.

A relational data base management system has been used to illustrate how the EER-model can be mapped onto a DBMS. This is done for two reason; simplicity, and the use of well understood standard DBMS tools. The same mapping exercise may be done for an object oriented DBMS.

The major contributions of this papers are:

- It formalizes a spatial topological data model and presents how it can be realized in a relational DBMS.
- It presents an encoding scheme for horizontal integration of tiles at the same scales.
- It presents an encoding scheme for vertical integrations of tiles at different scales.
- It shows how behaviour can be incorporated in GIS databases.

The model is simple, extendible and establishes a common framework for distributed access in the different databases required in a heterogeneous GIS environment. It forms the underlying model for the GIS database, code named Delta-1, currently being implemented at the National GIS Technology Centre, Ottawa, Canada.

The commitment to utilize a GIS is a major decision that must be carefully made. There are a number of controlling factors that must be understood clearly. First, it should be recognized that the data involved are very large. Once a commitment is made to utilize a specific GIS, the price must be paid to undergo the expensive data conversion process to store all the required data in the model dictated by the GIS. This investment in data conversion cost cannot be recovered if in the future a new GIS is adopted. The *COBOL syndrome* whereby a department is forced to conduct its activities with outdated methods because of the large overhead cost to change, should be avoided. The data component of a GIS is the critical resource for a GIS application development. A careful study of the manner of structuring the data so that it can be utilized without major expensive reorganization when the technology changes, is essential. This is what this paper has attempted to address.

References

[1] *ARC/INFO Users Manual (April 1985), ARC/INFO Programers Manual (October 1985), Librarian Users Manual (July 1987), ESRI.* Environmental Systems Research Institute, Redlands, California.

[2] Technical Sub-Committe Working Group No 1. *Proposed standards for a digital topographic information model.* Revision to Draft Report No 2 0.0, Canadian Council of Surveying and Mapping, Surveys and Mapping Branch, Energy Mines and Resources, Ottawa, Ontario, Oct. 1986.

[3] Narendra Ahuja. On approach to polygonal decomposition for hierarchical image representation. *Computer Vision Graphics and Image Processing*, 24:200 – 214, 1983.

[4] S. Aronoff. *Geographic Information Systems: A Mangement Perspective.* WDL Publ., Ottawa, Canada, 1989.

[5] P. Aronson and S. Morehouse. The arc/info map library: a design for a digital geographic database. In *Proc. Auto-Carto Six*, pages 372 – 382, Ottawa, Canada, 1983.

[6] J. L. Bentley. Mutidimensional binary search tree used for associative searching. *Comm. ACM*, 18(9):509 – 517, Sept. 1975.

[7] K. J. Boyko, M. A. Domaratz, R. G. Fegas, H. J. Rossmeisll, and L. Usery. *An enhanced digital line graph design.* Technical Report, US Geological Surveys Circular 1048, Federal Center, Box 25286, Denver, CO 80225, 1990.

[8] M. L. Brodie. *On the development of data models*, pages 19 – 48. Springer-Verlag, Nov. 1982.

[9] P. A. Burrough. *Principles of Geographical Information Systems for Land Resouces Assessment.* Oxford University Press, Oxford, United Kingdom, 1986.

[10] P. Chen. The entity relationship model- towards a unified view of of data. *ACM Trans. on Database Syst.*, 1(1), Mar. 1976.

[11] W. A. Davis. Hybrid use of hashing techniques for spatial data. In *Proc. Auto Carto London*, pages 127 – 135, London, England, 1986.

[12] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems.* Benjamin/Cummings, Redwood, CA., 1989.

[13] Inc. Environmental Systems Research Institute. *Digital chart of the world (DCW) - Draft vector product standard.* Contract Report No DMA600-89-0023 CDRL B018, Defense

Mapping Agency Systems Cemter, 12100 Sunset Hills Road, Ste. 200, Reston, Virginia, 22090-3207, Nov. 1990.

[14] R. A. Finkel and J. L. Bentley. Quad trees: a data structure for retrieval on composite keys. *Acta Infomatica*, 4(1):1 – 9, 1974.

[15] Canada Centre for Geomatic. *A guide to the CCSM software package.* Technical Document 2.0, Canada Centre for Goematics, Department of Energy, Mines and Resources, Sherbrooke, Quebec, Jan. 1989.

[16] A. U. Frank. Application of dbms to land information systems. In *Proc. 7th International Conference on Very Large Data Bases*, pages 448 – 453, Cannes, France, 1981.

[17] M. N. Gahegan and S. A. Roberts. An intelligent object-oriented geographic information system. *Int'l. J. Geographic Information Systems*, 2(2):101 – 10, 1988.

[18] G. Garderin and E. Gelembe. *New Applications of Databases.* Academic Press Inc., Orlando, Florida, 32887, 1984.

[19] I. Gargantini. An effective way to represent quad-trees. *Comm. ACM*, 25(12):905 – 910, Dec. 1982.

[20] Digital Geographic Information Working Group. *DIGEST: Digital Geogaphic Information Exchange Standards.* Draft Report, Enviromental System Research Institute, 380 New York Street, Calfornia, Sept. 1990.

[21] Digital Geographic Information Working Group. *DIGEST: Digital Geogaphic Information Exchange Standards Appendices.* Technical Report, Enviromental System Research Institute, 380 New York Street, Calfornia, Dec. 1989.

[22] O. Gunther. The design of the cell tree: an object oriented index structure for geometric database. In *5th Int'l Conf on Data Engineering*, pages 598 – 605, Feb. 1989.

[23] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proc. ACM-SIGMOD Conf.*, pages 47 – 57, ACM, Boston, Mass., 1984.

[24] J. R. Herring. Tigris: topologically integrated geographic information system. In *Proc. 8th Int'l. Symp. on Comp. Assisted Carto*, Boltimore, Maryland, 1987.

[25] H. F. Korth and A. Silberschatz. *Database system concepts.* McGraw-Hill, Inc., New York, 2 edition, 1991.

[26] Y. C. Lee. A data structure for resource mapping with caris. In *Proc. Auto-Carto Six*, pages 151 – 160, 1983.

[27] S. X. Li and M. H. Loew. Adjacency detection using quadcodes. *Comm. ACM.*, 30(7):627 – 631, Jul. 1987.

[28] S. X. Li and M. H. Loew. The quadcode and its arithmetic. *Comm. ACM.*, 30(7):621 – 626, Jul. 1987.

[29] F. D. Libera and F. Gosen. Using b-trees to solve geographic range queries. *Comput, Journal*, 29(2):176 – 181, 1986.

[30] D. M. Mark and J. P. Lazon. Linear quadtrees for geographic information systems. In *Proc. Int'l. Symp. on Spatial Data hasndling*, Zurich, Switzerland, Aug. 1984.

[31] S. Morehouse. Arc/info: a geo-relational model for spatial information. In *Proc. Auto-Carto 7*, pages 388 – 397, Washington, D.C., 1985.

[32] J. Nievergelt, H. Hinterberger, and K. C. Sevcik. The grid file: an adaptable symmetric multikey file structure. *ACM Trans. on Database Syst.*, 9(1):38 – 71, Mar. 1984.

[33] Government of Ontario - Canada. *MDIF: Techncal specification of the mapping data interchange format, Vs 3.0.* Technical Report, Ministry of Natural Resources, Surveys, Mapping and Remote Sensing Branch, Nov. 1989.

[34] Canadian Council of Surveying and Mapping. *National Standard for the Exchange of Digital Topographic Data - Standard EDP File Exchange Format Ver 0.0.* Revision to 2nd Draft Report 0.0, Canadian Council of Surveying and Mapping, Surveys and Mapping Branch, Energy Mines and Resources, Ottawa, Ontario, Oct. 1986.

[35] Canadian Council of Surveying and Mapping. *National standard for the exchange of digital topographic data - Data classification, quality evaluation and EDP file format.* 2nd Draft Report, Canadian Council of Surveying and Mapping, Topographic Survey Division, Surveys and Mapping Branch, Energy Mines and Resources, Ottawa, Ontario, Jul. 1984.

[36] B. C. Ooi, R. Sacks-Davis, and K. J. McDonell. Extending a dbms for geographic applications. In *Proc. 5th Int'l Conf. on Data Engineering*, pages 590 – 597, IEEE, Los Angeles, California, Feb. 1989.

[37] E. J. Otoo. Balanced multidimensional extendible hash tree. In *5th Int'l Symp. Principles of Database Syst.*, pages 100 – 113, ACM SIGACT-SIGMOD, 1986. Cambridge,MA.

[38] E. J. Otoo. Linearizing the directory growth of of extendible hashing. In *Proc. Int'l Conf. on Data Engineering*, IEEE, Los Angeles California, 1988.

[39] E. J. Otoo. Realizing a spatial topological data maodel in an object management system. Aug. 1991. In print.

[40] E. J. Otoo and L Dong. Quadcodes and their application to digital elevation modeling. Aug. 1991. Manuscript.

[41] D. Peuquet. An examination of techniques for reformatting digital cartographic data/part 1: raster-to-vector process. *Cartographica*, 18:34 – 48, 1980.

[42] D. Peuquet. An examination of techniques for reformatting digital cartographic data/part 2: vector-to-raster process. *Cartographica*, 18(3):21 – 33, 1981.

[43] D. J. Peuquet. Data structures for a knowledge-based geographic information system. In *Proc. International Symposium on Spatial Data Handling*, pages 372 – 391, 1984.

[44] D. J. Peuquet. A hybrid structure for the storage and manipulation of very large spatial data sets. *Computer Vision, Graphics, and image Processing*, 24(1):14 – 27, 1983.

[45] Donna Peuquet. A concepetual framwork and comaprison of spatial data models. *Cartographica*, 21(4):66–113, 1984.

[46] Inc Relational Technology. *INGRES/OpenSQL Reference Manual.* Relational Technology, Inc, 6.3 edition, Nov. 1989.

[47] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2):187 – 260, 1984.

[48] H.-J Schek and W. Waterfield. A database kernel system for geoscientific applications. In *Proc. 2nd Int'l Symp. on Spatial Data Handling*, pages 273 – 288, 1986. Seatle, Washington.

[49] T. R. Smith and M. Pazner. Knowledge-based control of search and learning in a large scale gis. In *Proc. International Symposium of Spatial Data Handling*, pages 498 – 519, 1984.

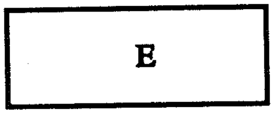
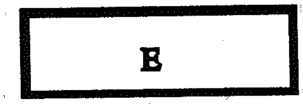
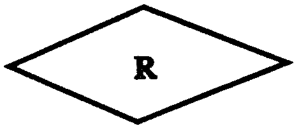
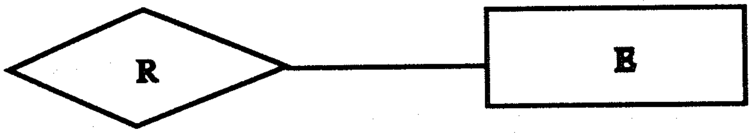
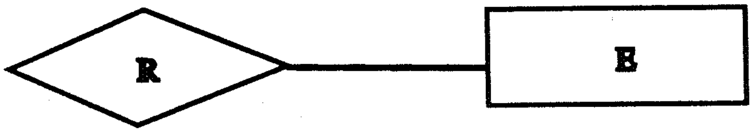
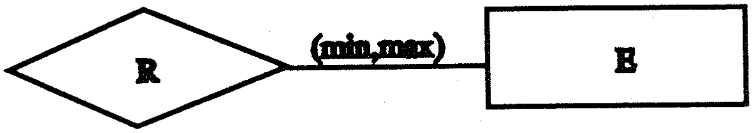
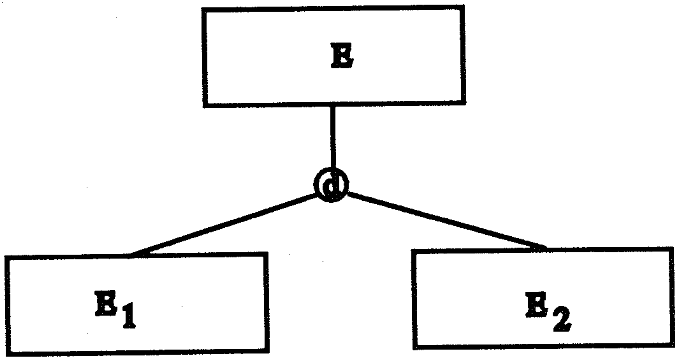
[50] M. Tamminen. Order preserving extendible hashing and bucket tries. *BIT*, 21:419 – 435, 1981.

[51] D. C. Tsichritzis and F. H. Lochovsky. *Data Models.* Prentice Hall, Englewood, Cliffs, NJ, 1982.

[52] L. Vanzella. *Classification of data structures fro thematic data.* Technical Report 88-14, Dept. of Computer Science, University of Alberta, Edmonton, Alberta, Canada, 1988.

[53] T. C. Waugh and Healey. The geoview design: a relational database approach to geographic data handling. In *Proc. Int'l. Symp. of Spatial Data handling*, pages 193 – 212, Int'l. Geo. Union, Commiss. on Geo. Data, Seatle, Washington, Jul. 1986.

A Symbols for EER-Diagrams

Symbols	Meaning
	Entity Type
	Weak entity type or dependent entity type
	**Relationship type**
	Partial participation of E in R
	Total participation of E in R
	Structural constraint (min,max) of participation of E in R
	E_1 and E_2 are disjoint specializations of E

NODES		
Nid	Xcord	Ycord
N1	0.0	5.0
N2	4.0	5.0
N3	7.0	9.0
N4	9.0	4.0
N5	7.0	0.0
N6	5.0	0.0
N7	2.0	0.0
N8	1.0	1.0
N9	3.0	4.0
N10	2.0	7.0

Table 1: The relation NODES

LINES						
Lid	Snid	Enid	LeftPid	RightPid	NoPnts	Chain
L1	N7	N1	P0	P1	3	2.0, 0.0, 0.0, 0.0, 0.0, 5.0
L2	N1	N3	P0	P2	3	0.0, 5.0, 0.0, 9.0, 7.0, 9.0
L3	N3	N4	P0	P3	3	7.0, 9.0, 9.0, 9.0, 9.0, 4.0
L4	N4	N5	P0	P4	3	9.0, 4.0, 9.0, 0.0, 7.0, 0.0
L5	N5	N6	P0	P5	2	7.0, 0.0, 5.0, 0.0
L6	N6	N7	P0	P4	2	5.0, 0.0, 2.0, 0.0
L7	N8	N9	P1	P1	5	1.0, 1.0, 1.0, 3.0, 2.0, 3.0, 2.0, 4.0, 3.0, 4.0
L8	N7	N2	P1	P4	4	2.0, 0.0, 2.0, 2.0, 4.0, 2.0, 4.0, 5.0
L9	N2	N3	P2	P3	4	4.0, 5.0, 4.0, 7.0, 7.0, 7.0, 7.0, 9.0
L10	N2	N4	P3	P4	4	4.0, 5.0, 6.0, 5.0, 6.0, 3.0, 9.0, 4.0
L11	N6	N5	P4	P5	4	5.0, 0.0, 5.0, 2.0, 7.0, 2.0, 7.0, 0.0
L12	N1	N2	P2	P1	2	0.0, 5.0, 4.0, 5.0

Table 2: The relation LINES

POLYGONS

Pid	PcntX	PcntY	MinElv	MaxElv	NoLns	BndLns
P0	∞	∞	Null	Null	6	L1, L2, L3, L4, L5, L6
P1	1.5	2.0	0.0	10.0	3	L1, L8, L12
P2	2.5	7.5	0.0	25.0	3	L2, L9, L12
P3	7.5	5.5	0.0	0.0	3	L3, L9, L10
P4	5.0	3.0	0.0	35.0	5	L4, L6, L8, L10, L11
P5	6.0	1.0	0.0	0.0	2	L5, L11

Table 3: The relation POLYGONS

LAYERS

LyCode	Tid	LyName	DisPriority
3	231	Forest	1
5	231	Water Body	1

Table 4: The relation LAYERS

LAYNDS

LyCode	Nid	RVMode	Fcode	FClassName
5	N10	1	5740	Swamp

Table 5: The Set of Special Nodes in a Layer

LAYLNS

LyCode	Lid	RVMode	Fcode	FClassName
5	L7	1	5040	STREAM

Table 6: The special line features in a layer

LAYPOLYS

LyCode	Pid	Fcode	FClassName
5	P3	5010	LAKE
5	P5	5010	LAKE
3	P1	6330	FOREST_AREA
3	P2	6340	ORCHARD
3	P4	6330	FOREST_AREA

Table 7: The Class polygon features in the layer

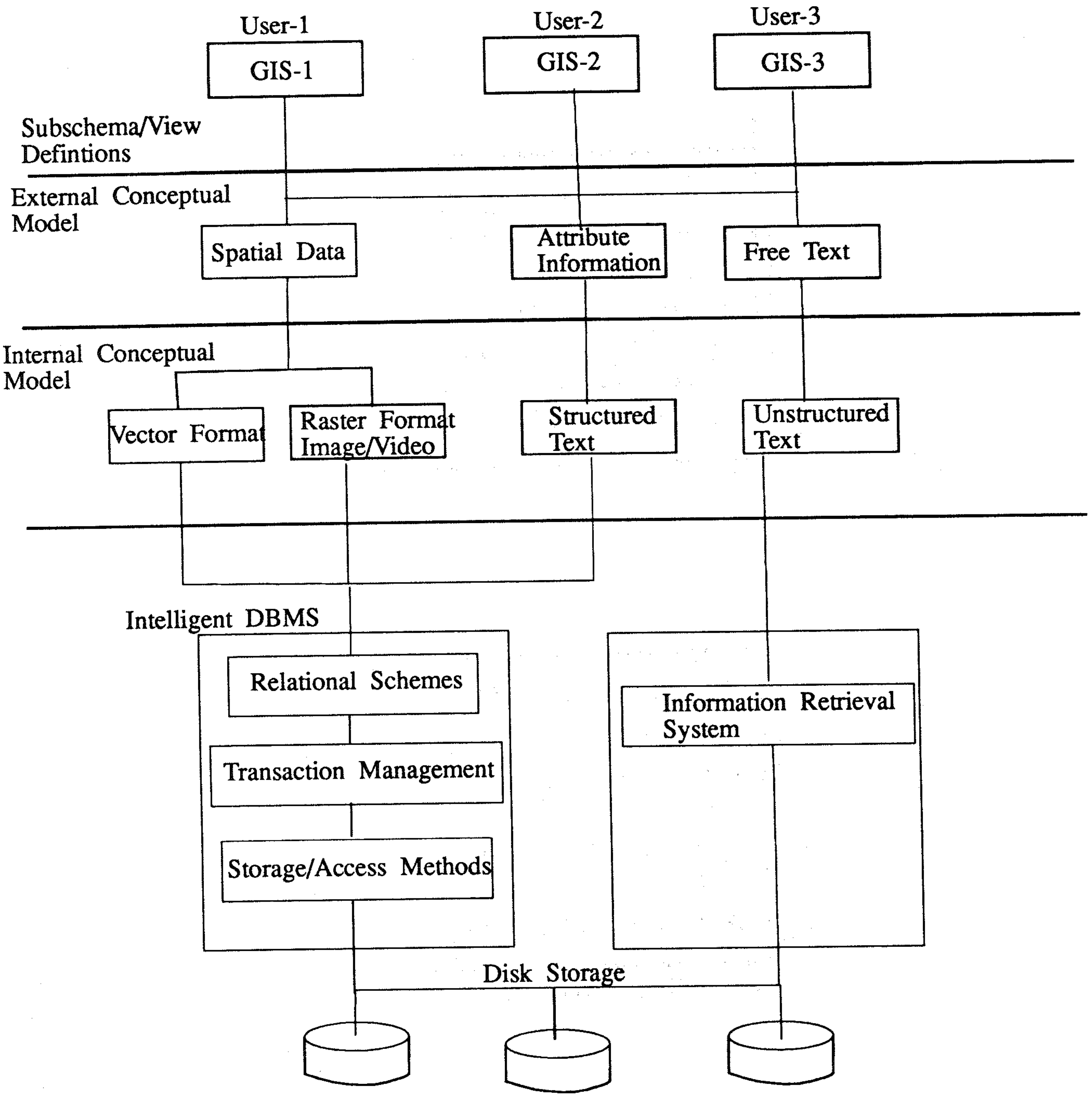


Figure 1: A typical GIS database architecture

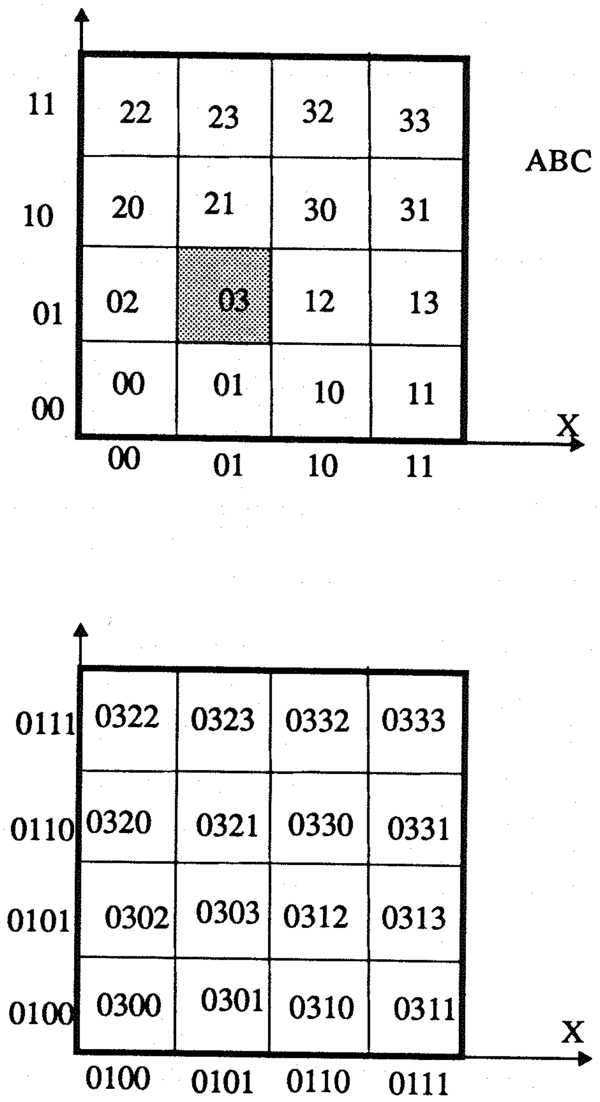


Figure 2: An Example of Tiled Universe of Coverage

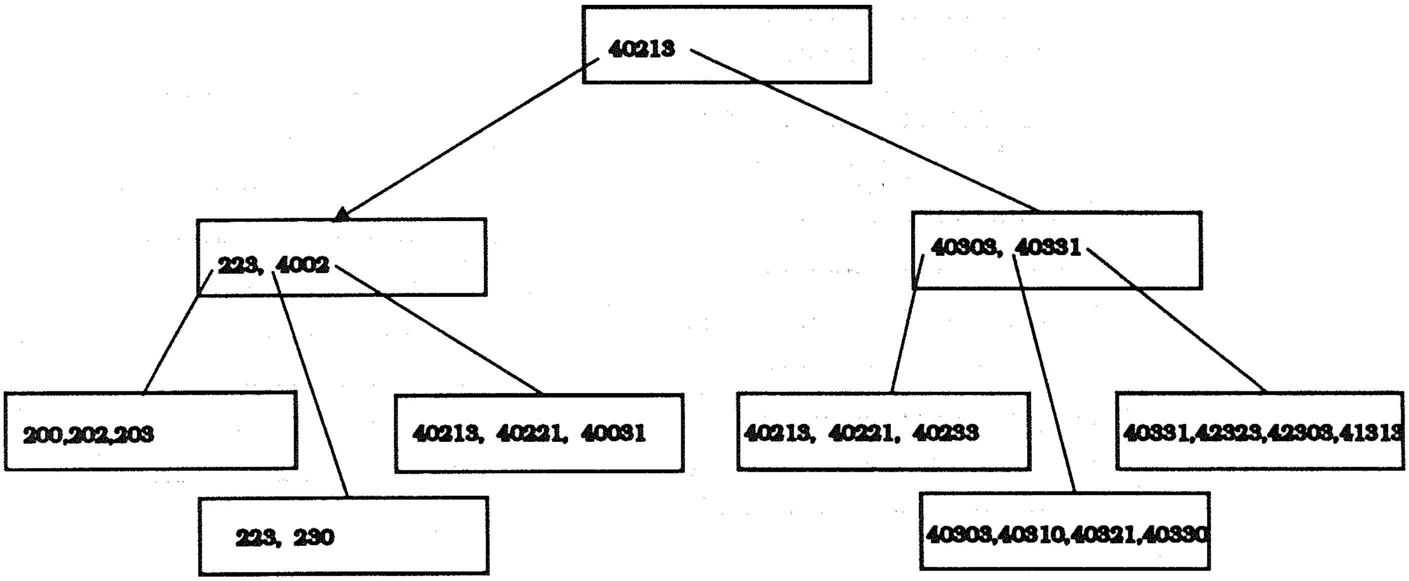


Figure 3: The B^{+}-tree index of order m = 5 for some quadcodes of Figure 2. In 32-bit representation, the first 4 bits store a level number.

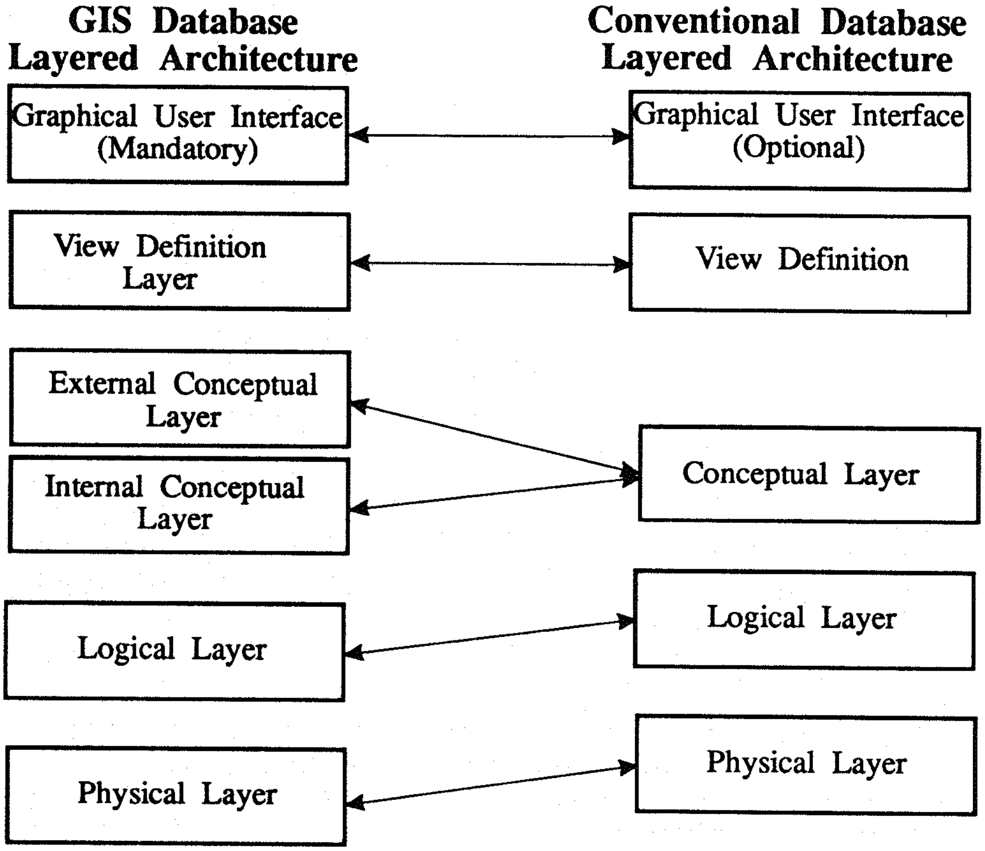


Figure 4: Correspondence between the levels of the proposed GIS and conventional database architectures

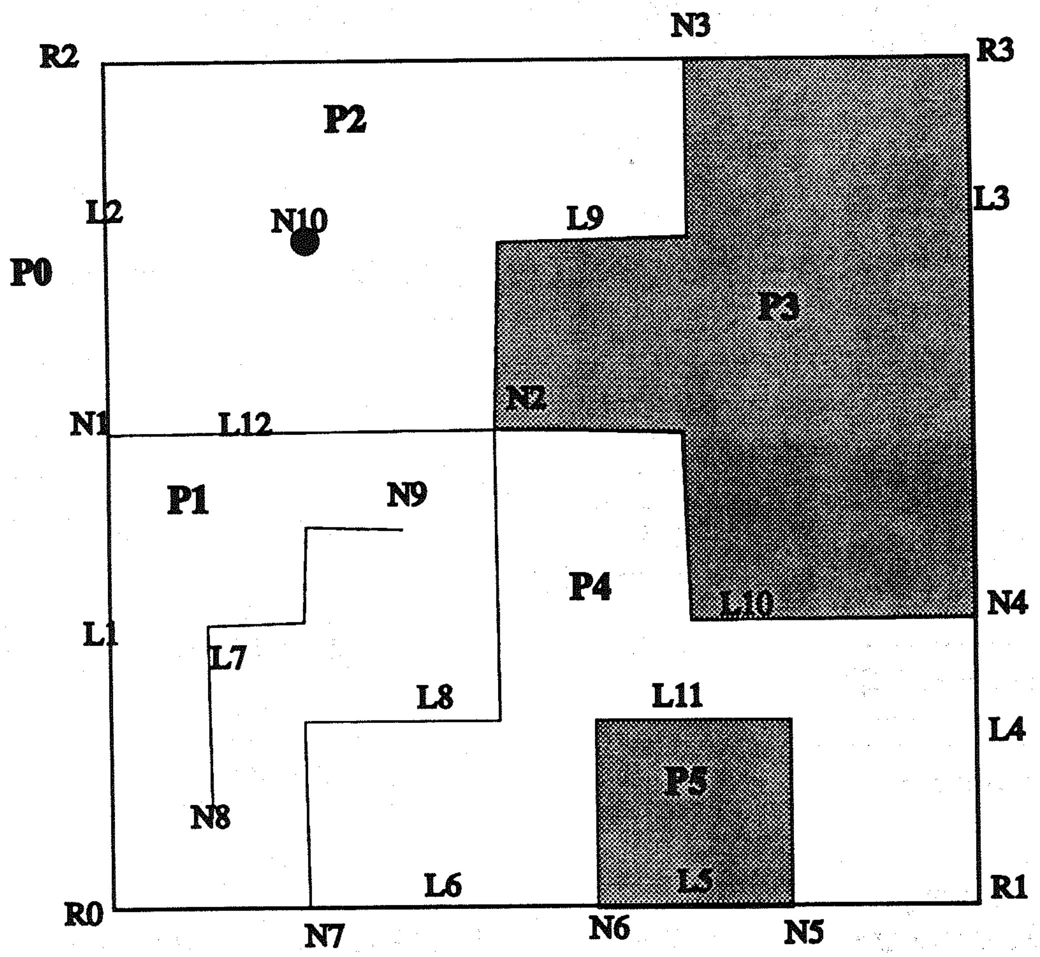


Figure 5: An illustration of the geometry of a tile layer.

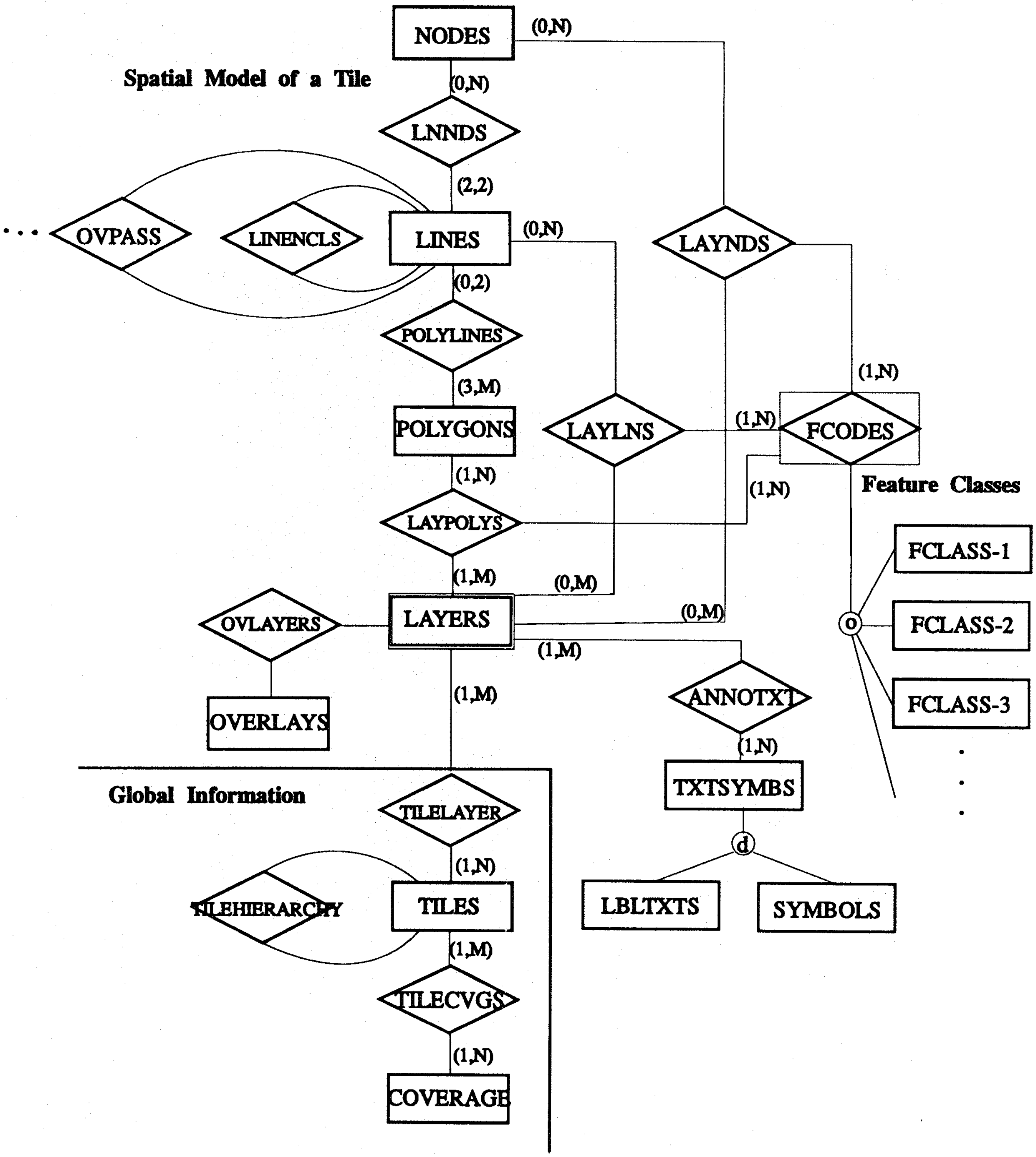


Figure 6: The EER-diagram of a 2-dimensional spatial data model

1. NODES (Nid, Xcord, Ycord, Zcord); {Nid → Xcord, Ycord, Zcord};

2. LINES (Lid, Snid, Enid, LeftPid, RightPid, NoPnts, Chain); {Lid → Snid, Enid, LeftPid, RightPid, NoPnts, Chain };

3. POLYGONS (Pid, PcntX, PcentY, PcentZ, MaxElv, MinElv, NoLns, BndLns); {Pid → PcntX, PcentY, PcentZ, MaxElv, MinElv, NoLns, BndLns}

4. LAYERS (LyCode, Tid, LyName, DispPriority); { {LyCode, Tid} → LyName, DispPriority };

5. OVRLAYERS (Ovlid, LyCode, Tid);

6. OVERLAYS (Ovlid, Tid, OverLyName); {{Ovlid, Tid} → OverLyName}

7. OVPASS (OverLid, UnderLid, Xcord, Ycord, ZcordO, ZcordU, Descript); { {OverLid, UnderLid} → Xcord, Ycord, Descript; {OverLid, Xcord, Ycord} → ZcordO; {UnderLid, Xcord, Ycord} → ZcordU }

8. LINENCLS (LyCode, ExclsLid, InclsLid, IncExc); { {LyCode, ExclsLid, InclsLid} → IncExc}

9. POLYNCLS (LyCode, ExclsPid, InclsPid, IncExc); { {LyCode, ExclsPid, InclsPid} → IncExc}

10. LAYNDS (LyCode, Nid, Fcode, RVMode, FClassName); { {LyCode, Nid, Fcode} → RVMode, FClassName };

11. LAYLNS (LyCode, Lid, Fcode, RVMode, FClassName); { {LyCode, Nid, Fcode} → RVMode, FClassName };

12. LAYPOLYS (LyCode, Pid, Fcode, FClassName); { {LyCode, Pid, Fcode} → RVMode, FClassName };

13. LBLTXT (SymbKey, String, StrLen, Style, Font)

14. SYMBOLS (SymbKey, SymbName, Fcode, Scale, SymbMethod);

15. ANNOTXT(LyCode, SymbKey, AnnotPosMRC, AbsRel, Angle, Size, Color);

Figure 7: The Final Set of Relations for the Spatial Model

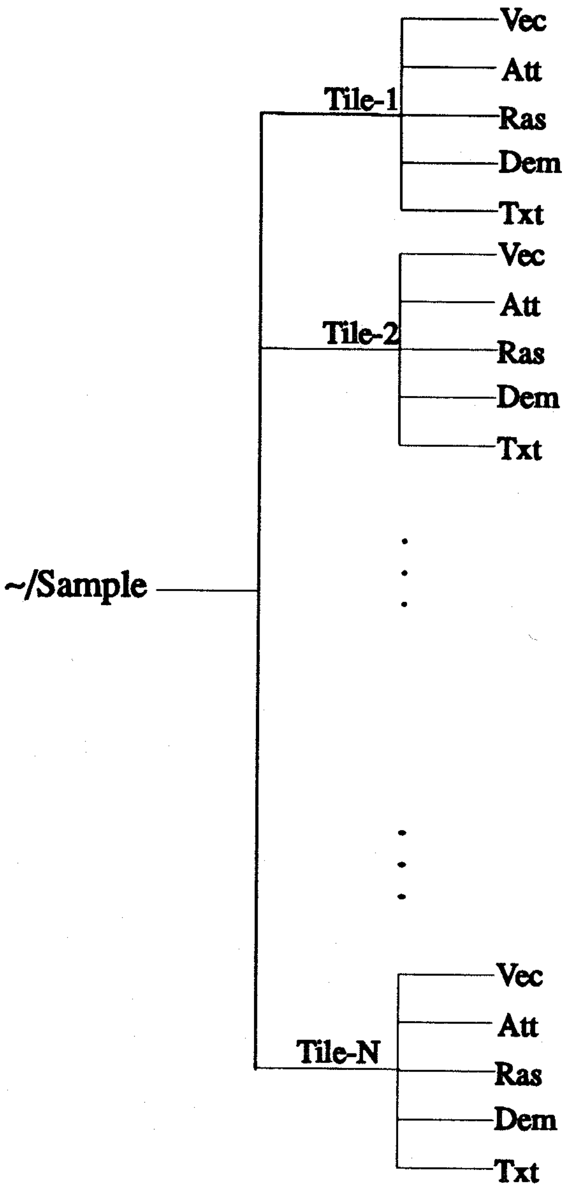


Figure 8: The hierarchical directory layout of tiles.