Modelling Geographic Information Systems using an Object Oriented Framework

Fatima Pires *Claudia Bauzer Medeiros*
Ardemiris Barros Silva

Relatório Técnico DCC–13/93

Junho de 1993

Modelling Geographic Information Systems using an Object Oriented Framework

Fatima Pires Claudia Bauzer Medeiros*
Ardemiris Barros Silva†

Abstract

Geographic information systems demand the processing of complex data using specialized operations, not available in traditional database systems. Even though there exist commercial systems that provide some of these facilities, there is a lack of proper support, which should cover not only the implementation but also the design stage. This paper answers this latter need, providing a framework for modelling geographic information systems using the paradigm of object orientation. The framework is a result of joint work of database researchers and GIS users, in developing environmental control applications.

1 Introduction

Data intensive geographic applications such as cartography, urban or environmental planning are built using Geographic Information Systems (GIS). A GIS is an automated system that performs data management and retrieval operations for a special class of data – *georeferenced data*. This term usually refers to data about geographic phenomena associated with its (geographical) location (e.g., vegetation or soil for a given

*Research partially financed by grants FAPESP 91/2117-1 and CNPq 500869/91-0
†FEC - UNICAMP - CP 6152

region). The basis for storing physical location information of georeferenced data is *spatial data* [Fra91], which is the general name for special data structures and storage techniques to represent multidimensional entities and their distribution in space.

Recent literature on GIS is mostly of two kinds: research performed by computer scientists, and research conducted by end-users (e.g., cartographers, biologists). The first type of research centers around spatial data structures, data modelling and development of database functions, without specific applications in mind. The second type of work is concerned with problems in specifying families of applications, without taking database support into consideration. This paper combines both approaches. It presents a framework for modelling GIS applications for object-oriented databases which is a result of joint work of database researchers and end users, for environmental planning and control.

1.1 Characterization and problems of GIS applications

One trait common to most GIS applications is their dependence on *thematic layers*, which consist of data collections about specific geographic features (the themes) for a given region – e.g., hidrography layer, geology layer. A GIS must provide users with facilities for choosing and combining different layers (the so-called *overlay* operations) in order to obtain the desired views of a region being analyzed. The functions required by a GIS must handle maintenance and analysis of spatial and non-spatial data alike. The most complicated cases arrive when both types of data must be combined in order to allow proper manipulation (e.g., overlay or connectivity operations involving different themes for a given region).

GIS queries may be classified into three different families [Aro89]:

- presentation of the stored data – e.g, show a map;
- determination of patterns followed by data – e.g., show population according to socio-economic standards
- predictions of future based on present data – e.g., determine regions prone to earthquakes

The use of databases to support GIS applications is so recent that many researchers still talk about "computer-based GIS" (as opposed to "paper-based" systems) [Aro89]. Most automated systems are based on a spatial data handler module coupled to a sequential file manager. These systems do not, therefore, take advantage of the facilities provided by a DBMS, such as logical independence, query language support, storage management, concurrency control. The fundamental question is how to embed the spatial aspects in a data model and support this by a DBMS such that acceptable interfaces (query languages and pictorial interfaces) can be developed.

Among open problems in the area one can mention:

- the absence of a database system that provides efficient support to data used in GIS (e.g., handling simultaneously standard and spatial data). An approach to solve this is extending a DBMS with spatial data handling procedures [AS91]. Data comes from heterogeneous sources and is traditionally stored and handled in separate thematic layers, which must be combined to provide information about a region (using the so-called layer-based functions). As remarked in [Kim90], research in this area is oriented towards adaptation of main memory data structures, and disk based structures have performance problems.

- the need for adequate querying and interface generation facilities, to allow users to see multidimensional spatial objects (as in CAD systems), with the associated geographic features – e.g., vegetation, climate and soil. Query facilities must allow users to navigate through and combine thematic layers.

- the lack of proper support for geometric functions (e.g., contiguity, connectivity, contour). Each application domain needs particular data interpolation and interpretation procedures, which at present are coded into the applications, thus increasing their dependence on data.

- the inexistence of an appropriate data model for GIS applications.

Researchers on databases and geosciences seem to agree that object-oriented models provide a good basis for developing GIS applications [KT92]. Results in this area are recent, and there is a lack of experience with real data.

This paper presents an approach to deal with the modelling problem. It describes a framework for designing a database to support GIS applications, using the object-oriented paradigm. This database is now being implemented using the O2 [Da90] object-oriented system. Layer-based and spatial data handling functions are being implemented as methods, some of which are based on computational geometry algorithms transplanted to disk operations.

The paper's main objective is to discuss modelling problems and to present the general modelling strategy, which can be used by other researchers interested in developing GIS applications. Implementation details will therefore not be discussed.

This paper is organized as follows. Section 2 presents a discussion of database modelling for GIS applications. Section 3 presents our data modelling steps. Section 4 shows how to use these steps in modelling an environmental control application. Finally, section 5 presents conclusions and discusses the present state of the project.

2 Data models for GIS

Geographic data, as described in [Aro89], has four main characteristics:

- its geographic position (coordinates)
- its attributes (non-spatial and spatial data)
- its spatial relationships (which allow relating geographic objects in space and time)
- its time components (which allow determining when some feature existed and for which period it was valid) [BFAT91].

GIS spatial data may be either in raster or in vector format. The former usually derives from images collected by satellite scans. These images are divided into regularly shaped pieces, handled in matrices of *cells*, where each cell is associated with a set of values for the points inside the cell, or a single value, an average for the points. Vector format is based on storing data coordinates, and considering them as sets of points, lines and polygons to which geometric functions are applied.

One comparison of GIS data models in 2-dimensional space [FG90] divides them according to the view of reality provided: *field view, object view*. Both are based on a tuple representation, where each attribute corresponds to a geographical feature and its (x,y) coordinates. The *field* view sees geographic entities on the plane as variables whose values are defined everywhere. Regions and segments are the model's basic entities. The *object* view sees these entities as sets of independent objects spread on the plane: points, lines and areas.

Another general classification of GIS data models is proposed in [WD91]. It divides such models in *layer-based spatial models* and *object-based spatial models*. The authors propose a third type of model, which is a result of combining characteristics from both kinds.

A *layer-based model* is based on considering data as stored on a set of layers containing spatial data and thematic information. Layers can be combined to build new layers, using *local, focal* and *zonal* operations, depending on the space unit considered.

Object-based spatial models combine the properties of object orientation and special geometric attributes – point, line and polygon. In this framework, a forest may be modelled by a polygon, plus components describing its features such as vegetation, soil and fauna. Operations on this model may be *set-based, topological, metric* or *euclidean*.

[FG90]'s classification considers the end-users' point of view. [WD91]'s classification uses database concepts. However, it is based on the assumption that spatial models that use geometric operations are all object-oriented. This, however, is seldom the case. Commercial systems that implement geometric operations are all relational-based. We propose a different classification, based on the underlying data model and its

support by a DBMS:

- *flat file model* – these are the models that do not depend on any underlying database. The design is based on partitioning data according to the procedures that will be applied, and queries consist of sequences of procedures executed on these partitions. The operations performed are usually raster-based, on layers, corresponding to the layer-based model of [WD91];

- *relational model* – these models are based on relational systems. Data is organized in tables. Storage management is often handled by two systems: the relational DBMS supports alphanumeric data, and another system processes spatial data. One example is the ARC-INFO [Env92] commercial system, which is based on combining thematic layers with the vector model using a relational system. Layers can contain either raster generated data or vector-based data. However, each type of data is handled by a different facility (e.g., textual data by a relational DBMS, image data by special procedures), so that the model must be supported by a complex combination of software modules that interact with the DBMS.

- *extensible relational model* – these rely on facilities provided by extensible relational models. This is an intermediate step between the previous model and object-oriented models, in the sense that data nesting is allowed and procedures can be attached to relation fields. Data nesting helps handling vector data – e.g., the value of one nested attribute may be a set of coordinates, representing a polygon. Examples of the use of this approach are [HC91], which describes the modelling and implementation of GIS using Starburst, and [vOV91], which add R-tree and graphical display modules on top of POSTGRES.

- *object-oriented model* – based on variations of the object-oriented paradigm. There follows some examples of this type of GIS.

[Lor91] discusses GIS querying facilities in an object-oriented system based on non-first normal form objects. [KT92] analyzes the representation of spatial relationships between objects in the Zenith object manager. Geographic objects are modelled as aggregations of geometric objects (lines, points and polygons). These geometric objects contain spatial and non-spatial (textual) properties. The Zenith system is an object manager that allows defining attributes (properties) for objects and relationships. Furthermore, it supports versions, which enables users to study the evolution of specific georeferenced objects. The paper's emphasis is on how this manager supports geographic relationships, which are implemented as links among objects.

[ZM92] propose coupling an object model with an expert system in order to allow processing GIS applications. The paper concentrates on defining core classes (point, line, polygon) and combining them into composition and inheritance relationships. The implementation is based in the CLIPS/COOL programming environment, a C-based expert system with object oriented features. This is not a persistent language, and therefore is not database supported.

[BDK92] model GIS cartographic applications using two levels of abstraction: *Map* and *Geometry*. The *Map* level is a set of tuples, containing spatial and non-spatial components. Spatial components belong to the *Geometry* level. It contains elements for cartographic processing –point, arc and polygon – and objects which are sets of these elements, for set-based operations – adjacency, intersection, containment. This model was implemented in the V1 prototype of the O2 system and the modelling was directed at this implementation. Therefore, some compromises had to be made to allow implementation. The same model was used, with modifications, in [Voi92].

3 The modelling framework

This section describes our modelling framework. This framework is based on a combination of results from the GIS modelling literature and our personal experience in working with real-life problems. Thus, it mixes

modelling from a database point of view and GIS modelling concerns as expressed by end-users.

3.1 The database view

From the database modelling point of view, most traditional GIS applications present the advantage that there is very little update activity after data is loaded. Thus, most transactions are read-only. Many concerns present in conventional database applications (e.g., constraint maintenance, schema and data evolution, version control) can be ignored[1]. Integrity verification and data adjustment can be processed during load time. Modelling issues can thus be restricted to

- choice of data model
- choice of base and complex data types
- choice of data manipulation functions for queries

Choice of data model

We settled on an object-oriented model. Since there is no standard definition for such models, we chose [Bee89]'s class-based framework. An object is an instance of a class. It is characterized by its state (contents) and behavior (methods), and is subject to inheritance and composition properties. Objects can be composed into more complex objects using constructors. The database schema is defined by its composition and inheritance graphs, as well as the applicable methods.

Object-orientation supports reuse and evolution, which we believe will be needed for future GIS applications. It provides the following facilities which are needed by GIS:

- It supports the creation of new data types based on aggregating existing objects. An example, for instance, is the specification of

[1] We believe this has been the case because applications are not supported by databases, which renders update operations very complex. Once users obtain proper database support, applications will start demanding treatment of data evolution.

a geographic region as an aggregate of area, mountain range and hidrography components. The area, the rivers and the range of mountains are objects defined elsewhere.

- It provides means for establishing complex relationships among objects, by means of composition and inheritance links. One example is the description of contiguity relationships between different roads in a given area.

- It allows classifying phenomena in taxonomies according to their characteristics, and accessing them via this taxonomy using the inheritance relationships – e.g., minerals or fauna.

- It permits modelling of the dynamic behavior of natural phenomena (using methods), and describing their physical properties. Thus, the system can easily provide answers to pattern and prediction queries.

Choice of Data types

Unlike previous work on GIS modelling, we do not explicitly differentiate between spatial and non-spatial data. We consider georeferenced objects to be aggregates of other objects, some of which may have spatial characteristics, and are subject to geometric processing methods. Furthermore, like other GIS data models, we allow both raster and vector-based data. To combine both types of data, we transform raster data into vector data, by treating it as a grid of cells. Each cell is represented by a point whose coordinates are determined by the cell's center. The cell's non-spatial attributes are represented by their average value. The choice of vector-based format is due to the vast experience in the field of computational geometry in implementing geometric functions. It is necessary, however, to adapt the existing in-memory algorithms to handle large volumes of data stored on disk. Raster images are also kept as components, thus allowing them to be displayed using graphical methods.

We propose the following base types

- **Basic geographic feature types.** These are non-spatial features describing geographic phenomena independent of their location – e.g., Soil, Minerals.
- **Image.** Bitmaps containing pictorial data
- **Geometric feature types.** They are the basis for all spatial operations:
  Point, Line, Polygon – similar to other models
  Spatial_object – describes the spatial characteristics of geographic objects. It is an aggregate of other Spatial_objects and Images.

Any **Geographic object** is a combination of **Basic Features, Images**, **Spatial_objects** and other **Geographic objects**.

Most object-oriented data models for GIS create geographic objects by establishing *composition* links between non-spatial and spatial objects. We decided, instead, to model geographic objects using *inheritance*. Thus, instead of modelling the usual

"a geographic object *contains* spatial objects".

we model the fact:

" a geographic object *is-a* spatial object"

This is closer to the semantics of geographic applications. In the other type of modelling, geometric operations on sets of geographic objects are performed on their spatial component, which is perceived as a "different" type of component. In our model, geometric operations are methods sent to the *Spatial_object* class, which is inherited by all geographic object classes. Thus, they are perceived as being applied to the whole geographic object. For instance, to determine if there is an intersection between two regions, users of the first type of model must pose the query

Region1.spatial_component **intersects** Region2.spatial_component

whereas users of our model pose the query

Region1 **intersects** Region2

Choice of data manipulation functions

There are basically three types of functions: spatial functions, non-spatial functions and functions that combine features from both. The first are geometric functions, which are application independent. The other two types depend on the application and user needs.

Since we choose to treat spatial data in vector format, the **Point, Line, Polygon** objects are the basis of spatial operations. The use of points as a basic type allows a collection of spatial operators, such as union, intersection and difference, as well as predicates as overlap, containment and proximity. Types **Line, Polygon** are in principle superfluous, since they are sets of **Points**. However, while point sets are good for expressing certain relationships, they lack the power for expressing others such as topological neighborhood. For this reason, it is more convenient to keep the three basic types apart, each with specific methods. The type **Point** can be further specialized by creating 2D and 3D subtypes, but we will not enter into this level of detail.

This paper is not concerned with implementation issues and therefore we do not discuss the details of methods executing geometric operations. We adopt the operations of [Aro89]:

- correction and transformation of coordinates – geometric transformations, edge matching, coordinate thinning;
- measurement – length, area, distance;
- neighborhood – topographic functions, intersection, contour, interpolation;
- connectivity – contiguity, proximity, network, intervisibility

3.2 The GIS-user view

The modelling of GIS applications involves basically three kinds of variables from the end-user point of view:

- primary thematic layers
- derived thematic layers
- granularity spread

Primary thematic layers

GIS manipulate a variety of thematic features: vegetation, climate, road network, human settlements, industrial development zones and others. We call *primary layer* any layer that can be built from a single thematic feature. Each family of applications concentrates on a subset of these layers. An important issue is to define what are the primary layers needed for an application.

Our work concerns the development of applications for environmental control in non-settled areas. Thus, according to user requirements, we need data on the following primary layers: geological structure, relief, climate, hidrography, soil, vegetation and fauna.

Derived data layers

Primary layers are not enough to model a geographic region. Sometimes, it is necessary to combine data from different layers in order to create another layer. We call this type of layer *derived layer*, to indicate that it contains features which cannot be inferred from any single feature (as opposed to primary layers). Derived layers can be dynamically generated at query time, or created and stored as any other layer. They can undergo any layer-based operation. Their creation and materialization mechanisms can be treated as if they were database views.

For instance, a *Climate* (derived) layer needs to combine data on *temperature, atmospheric pressure, winds, humidity, rainfall, solar exposure* which come from different primary layers. Climate information is used so frequently in GIS applications that it is advisable to keep it materialized.

Thematic data suffers from the fact that it is collected by different technologies and methodologies, according to the feature represented. Building a derived layer requires taking this into consideration. For instance, socio-economic data is collected by field interviews and income

tax information. Vegetation data is collected by means of satellite surveys and ground data checking. An environmental planning application may want to check vegetation zones against human settlement characteristics. This requires finding some common unit, at query time, to combine both layers. Since derived layers depend on user needs, the functions to build them are application dependent.

Granularity spread

We name *granularity spread* the scale, amount and level of data detail to be supported by the database. Different types of users may want to analyze the same feature from a finer or rougher level of detail. Normally, only primary layers undergo this type of analysis.

For instance, biologists may want to examine the vegetation of a region from a low-level point of view – individual plant species and their characteristics. Geographers, however, work from a more abstract point of view and are only interested in dividing an area in vegetation zones – brush, prairie, forest – without concerning themselves with isolated species.

Environmental control applications must cater to all levels of users and must therefore support a wide spread of level of detail. This is usually done bottom-up (i.e., starting from a basic level to more complex levels, similar to building complex objects from atomic values). In our example in section 4, for instance, we start from *Mineral* to *Rock* (both basic geographic features) and use *Rock* to build the *Geology* hierarchy.

3.3 Modelling steps

Given that we have chosen the object-oriented model, modelling consists in determining class hierarchies and their composition links, and defining appropriate methods.

The basis for all geometric operations is the *Point* class, which supports all geometric manipulation methods. This is similar to other modelling proposals in object oriented GIS (e.g., [KT92, Voi92]). The difference lies in that this special class is transparent to end-users, in the sense that spatial properties are treated as any other properties.

The first modelling step is to define the **Spatial_object** class. Besides standard geometric functions, the user may need application- specific spatial functions (e.g., the cartographic operations of [Voi92]). The schema type specification can be described, at high level, as follows. The *tuple* constructor is used to combine heterogenous components, as in all object-oriented models.

class Point : point specification
class Line: tuple(line_name, set(Point))
class Polygon: tuple (poly_name, set(Line))
class Spatial_object: tuple (obj_name, geometry (Points, Lines, Polygons), image)

A parallel step concerns defining **Basic geographic features**, which contain atomic values (e.g., integers,strings), images and other basic_features:

class B_feat_name: tuple (atomic values, images, **B_feat**)

Next, geographic_object classes are built combining atomic values, basic_features and geographic_object features. These classes can represent objects for different primary and derived layers, at varying granularity spreads. As seen in section 3, geographic objects inherit from spatial_objects.

class Geo_obj_name **inherit from Spatial_object** :
tuple (atomic,**B_feat, Geo_obj**).

The resulting classes are all combined into a high-level **Region** class, which contains all components of interest for a combined analysis of geographic features.

From a joint database designer and end-user point of view, the modelling steps can therefore be summarized as follows:

1. define the basic types **Point, Line, Polygon, Spatial_object**;
2. define the **Basic feature** classes;
3. define primary data layer geographic_objects using the lowest granularity desired;
4. repeat until all layers to be stored are defined

(a) define primary data layer objects at higher granularity

(b) define derived data layer objects

5. define **Region** class

4 Modelling example - environmental control

This section describes the modelling of an environmental control application using the steps of section 3. The schema is written using an O2-like syntax, where all types are public. Specific language-dependent details are ommitted since they do not contribute to the understanding of the modelling process. The point coordinate system chosen is UTM (Universal Mercator).

1) Point, Line, Polygon, Spatial_object

class **Point** type tuple (latitude, longitude:Real)
class **Line** type tuple (name: String, coord: set(Point))
class **Polygon** type tuple (name: String, segment: set(Line))
class **Spatial_object** type tuple (name: string, geo_image:Image
    class **Spatial_ob_polygon** inherits Spatial_object
                type tuple(geometry: Polygon)
    (spatial_object classes with points and lines are defined similarly)

2)a) Basic features – primary, low-level

class **Plant** type tuple (name, species, genus, family: String, photo: Image)
    (other basic_feature classes are **Mineral, Temperature, Pressure, Fauna**)

2)b) Basic features – primary, higher-level

class **Rock** type tuple (name, formation: String; composition: set(Mineral))

3) and 4) Geographic objects – primary, varying levels

class **Soil** inherit Spatial_ob_polygon
type tuple (humus, granulometry: Integer; texture, origin, structure: String)
class **Geology** inherit Spatial_ob_polygon type tuple (gtype:String)
class **Fold** inherit Geology type tuple (rock: set(Rock))
(other Geology subclasses are **Fault, Contact**)
class **Topology** inherit Spatial_ob_polygon
type tuple (altitude: Integer)
(Topology subclasses are **Depression, Valley, Mountain**)
class **Vegetation** inherit Spatial_ob_polygon
type tuple (vcover: set (Plant))

Other primary classes are **Hidrology, Fauna_zone, Temp_zone, Pressure_zone, Rain_zone**. The last three divide areas in zones according to average temperature, atmospheric pressure and rainfall indices.

4) Geographic_objects – derived, varying levels

class **Climate_zone** inherit Spatial_ob_polygon
type tuple (ctype:String, temp: Temp_zone, press: Pressure_zone, rain: Rain_zone
class **Mountain_range** inherit Spatial_ob_polygon
type tuple (name: String, topo: Topography, veg: set(Vegetation))

Climate_zone is an example of a derived class that must be computed with a user-defined function on temperature, pressure and rainfall data.

5) Region class

The **Region** class is a tuple containing **Soil, Crop, Forest, Hidrog-**

raphy, Depression, Fold, Fault, Climate_zone.

The figure shows the inheritance graph for geographic objects. Phenomena marked with asterisks correspond to stored derived layers. Given the data types, users can already define queries, even if methods are not present:

- Create a soil_texture view grouping regions by texture of soil

  group r in Region by (texture: r.soil.texture)

  (the result is a set of Region tuples organized according to the regions' soil texture fields)

- find hot_regions – annual average temperature > 80F and rainfall > 200mm

  define hot_temp as
  select tuple (lat:zt.latitude, long: zt.longitude)
  from zt in Temp_zone where zt.value > 70F

  define hot_rain as
  select tuple (lat:zr.latitude, long: zr.longitude)
  from zr in Rain_zone where zr.value > 200

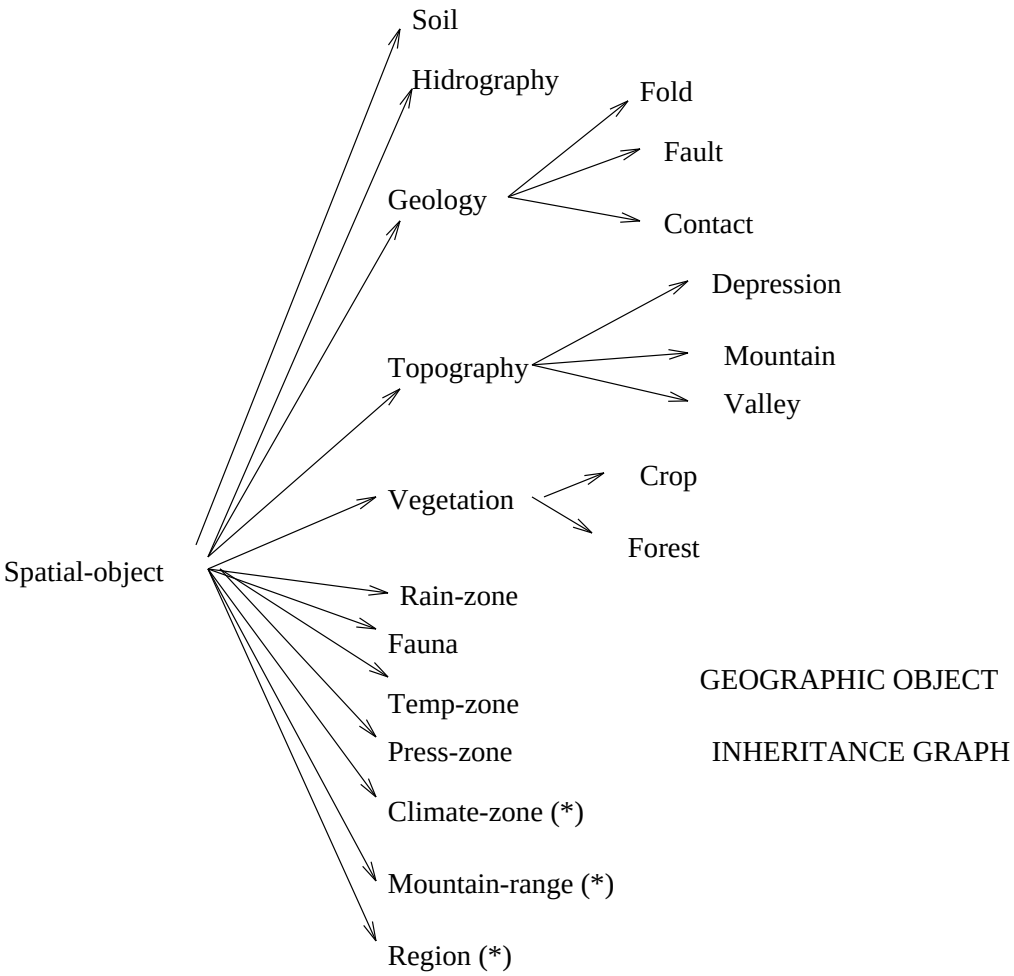
  the above definitions specify two views that return sets of space points where temperature and rainfall are as desired.

  Query: hot_temp + hot_rain (intersects both sets of tuples)

5 Conclusions

This paper presented a framework for modelling GIS applications using object-oriented databases. The modelling combines features from layer (raster-based) and vector data and the object-oriented framework. Instead of using small data samples or randomly generated data, the problem is discussed based on the experience of modelling a real-life application in the area of environmental control.

In order to specify the application, it was necessary to interact intensively with the end-users – in this case, biologists, geologists and



GEOGRAPHIC OBJECT

INHERITANCE GRAPH

engineers – who had little experience with database systems. The users were, however, experts in handling GIS queries using a flat-file system (the first type of model in our taxonomy). Thus, one nontrivial task in the modelling activity was to understand users' standard sequences of file operations for answering a query. From those, it was necessary to determine the appropriate base objects and classes, and a set of views.

Present and future work involves both modelling and implementation issues. We have started to implement one application, using data from the Cantareira Region, São Paulo State – an area of 2.000 km^2. The area's primary, low level layer data is stored in sequential character string files which are presently manipulated by the users' flat file GIS manager. The first step under way is to transform this data into geographic objects to be handled by the O2 database system. We expect to have a first prototype ready by the beginning of August. The results obtained will be used to improve future modelling procedures.

Future work also includes transforming algorithms in computational geometry (e.g., connectivity functions) to be applied to spatial data stored on disk, in order to answer more complex queries on georeferenced data. As well, there is an interest in creating appropriate interface functions to let users interact directly with geographic objects.

References

[Aro89] S. Aronoff. *Geographic Information Systems.* WDL Publications, Canada, 1989.

[AS91] W. Aref and H. Samet. Extending a DBMS with Spatial Operations. In *Proc. 2nd Symposium Spatial Database Systems*, pages 299–317. Springer Verlag Lecture Notes in Computer Science 525, 1991.

[BDK92] F. Bancilhon, C. Delobel, and P. Kanellakis, editors. *Building and Object-oriented System – the Story of O2*, chapter Geographic Applications – an Experience with O2. Morgan Kaufmann, California, 1992.

[Bee89] C. Beeri. Formal Models for Object-oriented Databases. In *Proc. 1st International Conference on Deductive and Object-oriented Databases*, pages 370–395, 1989.

[BFAT91] R. Barrera, A. Frank, and K. Al-Taha. Temporal Relations in Geographic Information Systems. *ACM Sigmod Record*, 20(3):85–93, 1991.

[Da90] O. Deux and al. The Story of O_2. *IEEE Transactions on Knowledge Bases and Data Engineering*, 2(1), 1990.

[Env92] Environmental Systems Research Institute, Inc. *ARC-INFO: GIS Today and Tomorrow*, 1992.

[FG90] A. Frank and M. Goodchild. Two Perspectives on Geographical Data Modelling. Technical Report 90-11, National Center for Geographic Information and Analysis, 1990.

[Fra91] A. Frank. Properties of Geographic Data: Requirements for Spatial Access Methods . In *Proc. 2nd Symposium Spatial Database Systems*, pages 225–234. Springer Verlag Lecture Notes in Computer Science 525, 1991.

[HC91] L. Haas and W. Cody. Exploiting Extensible DBMS in Integrated Geographic Information Systems . In *Proc. 2nd Symposium Spatial Database Systems*, pages 423–449. Springer Verlag Lecture Notes in Computer Science 525, 1991.

[Kim90] W. Kim. Special Issue – Directions for Future Database Research and Development. In *ACM SIGMOD RECORD*, december 1990.

[KT92] Z. Kemp and R. Thearle. Modelling Relationships in Spatial Databases . In *Proc 5th International Symposium on Spatial Data Handling*, pages 313–322, 1992. Volume 1.

[Lor91] R. Lorie. The Use of a Complex Object Language in Geographic Data Management. In *Proc. 2nd Symposium Spatial*

Database Systems, pages 319–337. Springer Verlag Lecture Notes in Computer Science 525, 1991.

[Voi92] A. Voisard. *Bases de Donnees Geographiques: du Modele de Donnees a l'INterface Utilisateur*. PhD thesis, Université de Paris-Sud Orsay, 1992.

[vOV91] P. von Oosterom and T. Vijlbrief. Building a GIS on Top of the Open DBMS POSTGRES. In *Proc European GIS Conference*, 1991.

[WD91] M. Worboys and S. Deen. Semantic Heterogeneity in Distributed Geographic Databases. *ACM Sigmod Record*, 20(4):30–34, 1991.

[ZM92] F. Zhan and D. Mark. Object-Oriented Spatial Knowledge Representation and Processing: Formalization of Core Classes and their Relationships. In *Proc 5th International Symposium on Spatial Data Handling*, pages 662–671, 1992. Volume 2.

Relatórios Técnicos – 1992

01/92 **Applications of Finite Automata Representing Large Vocabularies,** *C. L. Lucchesi, T. Kowaltowski*

02/92 **Point Set Pattern Matching in d-Dimensions,** *P. J. de Rezende, D. T. Lee*

03/92 **On the Irrelevance of Edge Orientations on the Acyclic Directed Two Disjoint Paths Problem,** *C. L. Lucchesi, M. C. M. T. Giglio*

04/92 **A Note on Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams,** *W. Jacometti*

05/92 **An (l, u)-Transversal Theorem for Bipartite Graphs,** *C. L. Lucchesi, D. H. Younger*

06/92 **Implementing Integrity Control in Active Databases,** *C. B. Medeiros, M. J. Andrade*

07/92 **New Experimental Results For Bipartite Matching,** *J. C. Setubal*

08/92 **Maintaining Integrity Constraints across Versions in a Database,** *C. B. Medeiros, G. Jomier, W. Cellary*

09/92 **On Clique-Complete Graphs,** *C. L. Lucchesi, C. P. Mello, J. L. Szwarcfiter*

10/92 **Examples of Informal but Rigorous Correctness Proofs for Tree Traversing Algorithms,** *T. Kowaltowski*

11/92 **Debugging Aids for Statechart-Based Systems,** *V. G. S. Elias, H. Liesenberg*

12/92 **Browsing and Querying in Object-Oriented Databases,** *J. L. de Oliveira, R. de O. Anido*

Relatórios Técnicos – 1993

01/93 **Transforming Statecharts into Reactive Systems,** *Antonio G. Figueiredo Filho, Hans K. E. Liesenberg*

02/93 **The Hierarchical Ring Protocol: An Efficient Scheme for Reading Replicated Data,** *Nabor das C. Mendonça, Ricardo de O. Anido*

03/93 **Matching Algorithms for Bipartite Graphs,** *Herbert A. Baier Saip, Cláudio L. Lucchesi*

04/93 **A lexBFS Algorithm for Proper Interval Graph Recognition,** *Celina M. H. de Figueiredo, João Meidanis, Célia P. de Mello*

05/93 **Sistema Gerenciador de Processamento Cooperativo,** *Ivonne. M. Carrazana, Nelson. C. Machado, Célio. C. Guimarães*

06/93 **Implementação de um Banco de Dados Relacional Dotado de uma Interface Cooperativa,** *Nascif A. Abousalh Neto, Ariadne M. B. R. Carvalho*

07/93 **Estadogramas no Desenvolvimento de Interfaces,** *Fábio N. de Lucena, Hans K. E. Liesenberg*

08/93 **Introspection and Projection in Reasoning about Other Agents,** *Jacques Wainer*

09/93 **Codificação de Seqüências de Imagens com Quantização Vetorial,** *Carlos Antonio Reinaldo Costa, Paulo Lício de Geus*

10/93 **Minimização do Consumo de Energia em um Sistema para Aquisição de Dados Controlado por Microcomputador,** *Paulo Cesar Centoducatte, Nelson Castro Machado*

11/93 **An Implementation Structure for RM-OSI/ISO Transaction Processing Application Contexts,** *Flávio Morais de Assis Silva, Edmundo Roberto Mauro Madeira*

Departamento de Ciência da Computação — IMECC
Caixa Postal 6065
Universidade Estadual de Campinas
13081-970 – Campinas – SP
BRASIL

reltec@dcc.unicamp.br