

A DBMS Repository for the Application Domain of Geographic Information Systems*

Emmanuel Stefanakis and Timos Sellis

Department of Electrical and Computer Engineering
National Technical University of Athens
Zographou, Athens, Greece 15773
e-mail: {stefanak,timos}@cs.ntua.gr
tel. +30-1-7721402, fax +30-1-7722459

Abstract. Much attention has been devoted in the past to support classes of applications which are not well served by conventional database systems. Focusing on the application domain of geographic information systems (GISs), several architectural approaches have been proposed to implement commercial or prototype systems and satisfy the urgent needs for operational spatial data handling. All these approaches adopt an application layer, which lies on the top of the database management system (DBMS) repositories and aims at supplementing the set of capabilities offered by the underlying repositories. The execution of operations within the application layer is frequently accompanied by up-/down-loading of voluminous data from/to the database and consequently it should be avoided. What is usually suggested as an alternative solution is pushing the operations of the application layer into the DBMS repositories. After a classification of operations available in GIS packages, the paper examines the features of a DBMS repository for the application domain of GISs.

Keywords: GIS architectures, DBMS repositories, GIS operations.

1 Introduction

Traditional database management systems (DBMSs) have only dealt with alphanumeric domains. These systems have proved not to be suitable for non-standard applications, such as computer-aided design (CAD), geographic information systems (GISs) and image databases, which are characterized by the existence of more complex domains.

Much attention has been devoted in the past to support classes of applications that are not well served by conventional relational systems. Focusing on the application domain of GISs, four architectural approaches have been adopted generally, to implement commercial or prototype systems and satisfy the urgent needs for operational spatial data handling. These approaches are [3, 12, 16]:

- *Single conventional DBMS:* In this approach, both spatial and non-spatial data are represented in tabular form in the pure relational model. Operators needed to define and manipulate spatial entities are contained in an application layer which is built on the top of the conventional DBMS.

* This article appears in the Proceedings of the 7th International Symposium on Spatial Data Handling (SDH'96), Delft - The Netherlands, August 1996.

- *Partial conventional DBMS*: In this approach, a conventional DBMS is used to represent thematic information associated to spatial entities, while a separate subsystem is used to handle spatial data. Operators needed to define and manipulate spatial entities are contained in an application layer which provides a uniform interface to both subsystems.
- *Extended conventional DBMS*: In this approach, a conventional DBMS is modified to also support GIS application domains. This is accomplished by adding new constructs to a conventional DBMS so as to enhance its modeling power and provide better support for spatial applications. These new constructs include support for abstract data types (ADTs), procedural fields, complex objects, composite attributes, and so on. Additional operators needed to define and manipulate spatial entities are contained in an application layer which lies on top of the extended DBMS.
- *Object-Oriented DBMS*: In this approach, the object-oriented model is used as a basis for the application domain of GIS. The concepts of the underlying model (e.g., classes, inheritance, encapsulation, types, methods) are very convenient. Any additional operators needed to define and manipulate spatial entities are contained in an application layer which lies on top of the object-oriented DBMS.

The role of the *application layer* in all approaches above is to supplement the set of capabilities offered by the underlying system architectures, so that the operational needs for spatial data handling are satisfied. In other words, the application layer provides the set of GIS operations which are not available in the underlying DBMSs. Obviously, this set varies from one software package to another and heavily depends upon the system architecture. The execution of operations within the application layer is frequently accompanied by up-/down-loading of voluminous data¹ from/to the database. This is an expensive task which should be avoided in a production environment. What is usually suggested as an alternative solution is pushing the operations of the application layer into the DBMS repository.

This study examines the shrinkage of the application layer and the generation of a DBMS repository for the application domain of GISs. The discussion is organized as follows. Section 2 presents a classification of operations available in GISs, while Section 3 shows a simplified example of a sequence of GIS operations to support a real-world problem. Section 4 presents the design aspects for the DBMS repository. These involve the design of both the structure of the database and operations available in the DBMS. Sections 5 and 6 demonstrate the features of an extensible relational DBMS repository and an object-oriented DBMS repository for the application domain of GISs. Finally, Section 7 concludes the discussion.

2 Classification of GIS Operations

The operations available in geographic information systems (GIS) vary from one system to another. However, their fundamental capabilities can be expressed in terms of four types of operations [15]:

- *Programming operations*: They consist of a number of routines in the operating system level, such as supervise and direct the system operations and control the communication with peripheral devices connected to the computer.

¹ spatial data are usually voluminous

- *Data preparation operations*: They encompass a variety of methods for capturing data from different sources (e.g., digital or paper maps, land measurements), processing and storing them appropriately in the database.
- *Data presentation operations*: They encompass a variety of methods for presentation of data, such as drawing maps, drafting charts, generating reports, and so on.
- *Data interpretation operations*: These operations transform data into information and as such they comprise the heart of any geographic information system. Consequently, the discussion that follows focuses on them.

Operations for data interpretation can be viewed as dealing with a hierarchy of data [12, 15]. At the highest level, there is a library of *maps* (more commonly referred to as *layers*), all of which are in registration (i.e., they have a common coordinate system). Each layer is partitioned into *zones* (regions), where the zones are sets of locations with a common *attribute value*. Examples of layers are the land-use layer, which is divided into land-use zones (e.g., wetland, river, desert, city, park and agricultural zones) and the road network layer, which contains the roads that pass through the portion of space that is covered by the layer.

Data interpretation operations available in GISs characterize [3, 4, 12, 15]:

- *individual locations*,
- *locations within neighborhoods*, and
- *locations within zones*,

and constitute respectively the three classes of operations, i.e., *local*, *focal* and *zonal* operations. Notice that all data interpretation is done in a layer-by-layer basis. That is, each operation accepts one or more existing layers as input (the operands) and generates a new layer as output (the product).

The first class of data-interpreting operations (local operations) includes those that compute a new value for each location on a layer as a function of existing data explicitly associated with that location. The data to be processed by these operations may include the zonal values associated with each location on one or more layers. *Local operations* include:

- *Search operations*, i.e., retrieval of information associated with individual locations on a layer.
- *Classification and recoding operations*, i.e., assignment of new attribute values to individual locations on a layer.
- *Generalization operations*, i.e., reduction of detail associated with individual locations on a layer.
- *Overlay operations*, i.e., assignment of new attribute values to individual locations resulting from the combination of two or more layers.

Focal operations compute new values for every location as a function of its *neighborhood*. A neighborhood is defined as any set of one or more locations that bear a specified distance and/or directional relationship to a particular location, the *neighborhood focus*. *Focal operations* include:

- *Search operations*, i.e., retrieval of information characterizing the immediate or extended vicinity (the region of interest) of individual locations on a layer.

- *Proximity operations*, i.e., assignment of new attribute values to individual locations on a layer, which depict their distance or direction in a neighborhood with respect to the neighborhood focus.
- *Interpolation operations*, i.e., assignment of new attribute values to individual locations on a layer derived by averaging sets of two or more target values associated to selected locations in their immediate or extended vicinity.
- *Surfacial operations*, i.e., assignment of new attribute values to individual locations on a layer indicating their surfacial characteristics (slope, aspect, volume, etc.).
- *Connectivity operations*, i.e., assignment of new attribute values to individual locations on a layer derived from a running total of the results being retained in a quantitative or qualitative step-by-step fashion and considering the values associated to locations in the immediate or extended vicinity (optimum path finding, etc.).

The third and final class of data-interpreting operations (zonal operations) includes those that compute a new value for each location as a function of existing values associated with a zone containing that location. *Zonal operations* include:

- *Search operations*, i.e., retrieval of information characterizing individual locations on a layer that coincide with the zones of another layer.
- *Measurement operations*, i.e., assignment of new attribute values to individual locations on a layer that correspond to a measurement (e.g., area, length) characterizing their zones.

3 Site Selection Based on a Sequence of GIS Operations

The purpose of this section is to present a sequence of data-interpretation operations which may compose one or more procedures² to accomplish the task of *site selection* for a *residential housing development*. The basic approach to this is to create a set of *constraints*, which restrict the planned activity, and a set of *opportunities*, which are conducive to the activity. The combination of the two is considered in order to find the best locations.

In the simplified situation that follows the set of constraints and opportunities consists of³:

- vacant area (i.e., no development),
- dry land,
- level and smooth site (e.g., slope < 10%),
- nearness to the existing road network, and
- south-facing slope.

In addition all candidate sites should have an adequate size to satisfy the needs of the planning activity (e.g., between 1 and 1.5 sq km).

The whole task requires as input three layers of the region under examination:

² A *procedure* is any finite sequence of one or more operations that are applied to meaningful data with deliberate intent [15].

³ a wider set could be taken into account, but this subset is enough to illustrate some basic data-interpreting operations available in GISs.

- *hypsography* layer: the three-dimensional surface of the region (altitude values),
- *development* layer: it depicts the existing infrastructure of the region (e.g., roads, buildings, etc.), and
- *moisture* layer: it depicts the soil moisture of the region (e.g., lakes, wet-lands, dry-lands, etc.).

The procedure of site selection, based on the sets of constraints and opportunities determined above, may consist of the following sequence of operations⁴:

1. *Vacant areas*: A new layer of vacant areas is produced from the layer of development by classifying, generalizing and finally performing a selective search on the result.
 - *development-classes* = Local(classification) of *development*
 - *vacant-developed* = Local(generalization) of *development-classes*
 - *vacant* = Local(search) of *vacant-developed*
2. *Dry lands*: A new layer of dry lands is produced from the layer of moisture by classifying, reducing detail and performing a selective search on the result.
 - *moisture-classes* = Local(classification) of *moisture*
 - *dry-wet* = Local(generalization) of *moisture-classes*
 - *dry* = Local(search) of *dry-wet*
3. *Level sites*: A new layer of level and smooth sites is produced from the layer of hypsography by computing, classifying, generalizing and finally performing a selective search on the result.
 - *slope* = Focal(surfacial) of *hypsography*
 - *slope-classes* = Local(classification) of *slope*
 - *level-steep* = Local(generalization) of *slope-classes*
 - *level* = Local(search) of *level-steep*
4. *Accessible areas*: A new layer of accessible sites by the existing road network is produced implicitly from the layer of development by highlighting the road network, computing, classifying, generalizing and finally performing a selective search on the proximities.
 - *roads* = Local(search) of *development*
 - *road-proximity* = Focal(Proximity) of *roads*
 - *road-proximity-classes* = Local(classification) of *road-proximity*
 - *accessible-inaccessible* = Local(generalization) of *road-proximity-classes*
 - *accessible* = Local(search) of *accessible-inaccessible*
5. *South-facing areas*: A new layer of south-facing areas is produced from the layer of hypsography by computing, classifying, generalizing and finally performing a selective search on the aspects.
 - *aspect* = Focal(surfacial) of *hypsography*
 - *aspect-classes* = Local(classification) of *aspect*
 - *south-north* = Local(generalization) of *aspect-classes*
 - *south* = Local(search) of *south-north*

⁴ the syntax adopted for the operations is:
new-layer = Operation-class(operation-subclass) of *existing-layer* and ...

6. *Good-sites*: A new layer of sites that satisfy the set of constraints and opportunities is produced by the successive overlay of layers produced in the previous steps. Finally, good sites are highlighted by performing a selective search on the result.
 - *vacant-dry* = Local(overlay) of *vacant* and *dry*
 - *vacant-dry-level* = Local(overlay) of *level* and *vacant-dry*
 - *vacant-dry-level-accessible* = Local(overlay) of *accessible* and *vacant-dry-level*
 - *vacant-dry-level-accessible-south* = Local(overlay) of *south* and *vacant-dry-level-accessible*
 - *good-sites* = Local(search) of *vacant-dry-level-accessible-south*
7. *Candidate sites*: A new layer of sites that satisfy the set of constraints and opportunities and have adequate size is produced from the layer of good sites by measuring the sizes of zones and highlighting those that are within the predefined size interval.
 - *good-sites-size* = Zonal(measurement) of *good-sites*
 - *candidate-sites* = Local(search) of *good-sites-size*

4 Design of a DBMS Repository for GISs

Depending on the underlying architecture (Section 1) commercial and prototype GISs support the execution of data-interpreting operations either entirely in the application layer or partially in the DBMS repositories and partially in the application layer. In general, the execution of operations within the application layer is accompanied by up-/down-loading of massive data from/to the database and should be avoided in order to ameliorate the response time of the system. The shrinkage of the application layer can be accomplished by enriching the DBMS repository with GIS operations.

In fact, this is not applicable in the approaches of *single conventional DBMS* and *partial conventional DBMS*. As for the former, the relational model is not rich or powerful enough to model the structural complexities of most types of spatial data, and relational algebra has not the expressive power to support data-interpreting operations. As for the latter, provided that spatial and non-spatial data reside in separate databases, processing is mostly performed in the application layer. However, the shrinkage of the application layer can be accomplished when the approaches of *extended conventional DBMS* (Section 5) and *object-oriented DBMS* (Section 6) are adopted.

This Section aims at presenting the design aspects of a DBMS repository for the application domain of GISs. This task involves the design of both the database and DBMS components.

4.1 The Database Component

A database is a collection of *entities*. Each entity is associated with a number of *attributes* describing its characteristics. Entities with common attributes compose a *class* of entities. The links among entities are described through *relationships*.

For the application domain of GISs, the *zones* (e.g., a city, a woodland area, a road network, etc.) constitute the entities in the database. A zone is a set of

individual locations with common attribute values. Zone attributes have a *spatial* and a *thematic* component. The spatial component specifies the zone location with respect to a given coordinate system (e.g., $[\phi, \lambda]$ of Athens), while the thematic component describes the qualitative or quantitative properties associated with the zone (e.g., archaeological interest and population of Athens).

From the literature two main concepts for representing the spatial component of a zone are known: the *raster* and the *vector* models. In the raster model each zone is represented by the set of individual locations which constitute vertices of a regular grid in space of a fixed resolution and fall within the zone. In the vector model the zone is represented by the set of individual locations which describe its outline⁵. The distance between successive locations defines the resolution in representation. The thematic component of a zone is represented by a set of alphanumeric data types.

Each collection of zones with common attributes composes a class of entities, such as a layer (e.g., land-use layer). The links among zones describe their relationships. The relationships may be spatial (e.g., distance between two cities or intersection of two highways) or non-spatial (e.g., the capital city of a province).

4.2 The DBMS Component

A Database Management System (DBMS) is a computerized system for managing a database. Conventional DBMSs provide some analytical operations such as simple statistics.

A DBMS repository for the application domain of GISs should provide additional operations, such as those presented in Section 2. Specifically, the following categories of operations for data-interpretation should be available:

- *Selection operations*: They perform a selective search on zones based on their spatial and non-spatial attribute values (i.e., local search).
- *Combination operations*: They compute the relationships (intersection, union, relative position, etc.) between a pair of zones or layers (i.e., overlay, focal and zonal search).
- *Analytical operations*: They process further selections and combinations of existing sets of zones (i.e. classification, generalization, proximity, interpolation, surfacial, connectivity, measurement).

The presence of *indices* over both the spatial and non-spatial attributes is required in order to facilitate and speed up the execution of operations.

5 An Extensible Relational DBMS Repository for GISs

This Section briefly presents the features of an extensible relational DBMS repository for the application domain of GISs. The relational model [6] represents the data in a database as a collection of relations (tables), which consist of rows and

⁵ In 2-D space a point feature (e.g., a village) is represented by a single individual location; a linear feature (e.g., a highway) is represented by a sequence of individual locations which trace its axis; and a polygonal feature (e.g., a forest) is represented by a sequence of individual locations which trace its boundary.

columns. Each row (tuple) in a relation is a collection of related data values. These values can be interpreted as a fact describing an entity or a relationship instance. A class of entities can be viewed as a relation in a relational DBMS. The records of a relation represent the entities of a class, while the columns accommodate the attribute values characterizing those entities. All attribute values are defined over a set of domains, each of which has one or another form of an alphanumeric data type. The relational algebra provides a collection of operations to manipulate entire relations. These operations are used to select tuples from individual relations and to combine related tuples from several relations. The result of each operation is a new relation, which can be further manipulated by the relational algebra operations. The relational algebra operations are divided into two groups. One group includes set of operations from the mathematical set theory, (e.g., union, intersection, difference and cartesian product), while the other group consists of operations developed specifically for relational databases (e.g., select, project, join). Several index structures for files (e.g., hashing technique, B-tree variations) are adopted in order to speed up the execution of these operations. In an extensible relational DBMS, like Postgres [14] and Starburst [8], new data types (Abstract Data Types), operations and specialized indices can be incorporated into the system.

In an extensible relational DBMS a zone can be viewed as a tuple in one or more relations, which compose classes of zones (e.g., layers). Attribute values are of alphanumeric data types for thematic data (e.g., city population or vegetation type) or of user defined ADTs (e.g., point, line, polygon boundary) for geometric data (e.g., road or forest geometry). The minimum set of operations provided by relational algebra is extended to support GIS operations. Spatial index structures, such as grid-files, quadtrees and R-trees [11], are employed, along with standard alphanumeric indices, to speed up the execution of operations.

Several prototype systems have been built on top of extensible relational DBMS to support spatial applications. Haas and Cody [7] describe how Starburst can be extended with user defined ADTs and operations to support the needs of two real-world applications and show how some sample queries may be processed efficiently. The Geo++ system [10] is built on top of Postgres. The system adopts the geometric capabilities of Postgres (R-tree index structure; primitive data types: point, line segment, path and box) and makes use of additional data types, as well as operators to incorporate the functionalities of commercial GISs. Aref and Samet [2] describe the architecture of SAND system. In SAND spatial and non-spatial data are linked bidirectionally. Spatial attributes are stored in spatial data structures. A spatial data structure is chosen based on the attribute's spatial data type (e.g., point, line, polygon, etc.), and serves as an index for spatial objects and an environment for the execution of spatially related operations. Non-spatial attributes are maintained in database relations. Two logical links are maintained between spatial and non-spatial attributes of an entity: forward and backward links. The linked instances and the links form what is termed a spatial relation.

6 An Object-Oriented DBMS Repository for GISs

This Section briefly presents the features of an object-oriented DBMS repository for the application domain of GISs. The object-oriented model [6] represents the data in a database as a collection of objects. An object represents a physical entity,

a concept, an event or some aspect of interest to the database application and is characterized by its identity, state and behavior. The object identity is maintained via an object identifier. The state is stored in data attributes, which can also be objects. The behavior is encoded in subroutines and functions, called methods, which are encapsulated into objects. Objects with common state and behavior form classes of objects, which are organized into type hierarchies. A subclass inherits both data attributes and methods from its superclass (inheritance). The execution of operations is supported by appropriate index structures for objects.

In an object-oriented DBMS repository for GISs, a zone can be viewed as an object, with a unique identifier, a set of spatial and non-spatial attribute values, and a collection of operations describing its behavior. Zones with common data attributes and behavior (e.g., a layer) can be viewed as a class of objects.

A few attempts to build prototype systems on top of object-oriented DBMS to support spatial applications have been made in the past. Milne et al. [9] examine the construction of a system on top of the object-oriented DBMS, ONTOS. ONTOS provides a persistent class library, written in C++, and allows the construction of new persistent classes using C++. The authors extend ONTOS with spatial object classes, a graphical user interface, graphic display module and geographical database module. They also report an impressive performance gain over a proprietary relational DBMS (Oracle) and an extended relational DBMS with special GIS capabilities (SIRO-DBMS [1]). A couple of attempts have been made to build an object-oriented GIS on top of the object-oriented DBMS O₂. David et al. [5] have implemented GeO₂ system. The conceptual data model of GeO₂ is made of a semantic data model, which is an extension of the Entity-Relationship data model [6], and a localization data model, which is defined by an ADT. Three levels of data structures (spaghetti, network and map structure) are provided to support the efficient and appropriate storage of geographic coordinates and relationships between entities. Scholl and Voisard [13] implemented a system in which the database consists of a set of maps, where a map is a relation that has at least one spatial attribute. Spatial attributes are represented by ADTs (e.g., points, lines, polygons, etc.). As for map operations, they are implemented as methods on map objects.

7 Conclusion

The design of a DBMS repository for the application domain of GISs is a difficult task due to peculiarities of spatial data and operations involved. The contribution of the paper can be summarized as follows:

- After a classification of operations available in current GIS packages, it is shown how a sequence of them may compose one or more procedures to support the spatial decision-making process.
- The fundamental design aspects of a DBMS repository for the application domain of GISs are presented.
- The features of both an extensible relational and an object-oriented DBMS repositories for spatial applications are given briefly and some representative prototype systems developed in the past are summarized.

Future research in the area includes:

- The design and implementation of an object-oriented DBMS repository for the application domain of GISs.
- The incorporation of operations available in commercial GIS packages into the DBMS repository and the shrinkage of the application layer.
- The development of efficient optimization strategies for query processing and the evaluation of the system performance versus other commercial and prototype packages.

References

1. D.J. Abel. A database toolkit for Geographical Information Systems. *International Journal of Geographical Information Systems*, 3(2):103–115, 1989.
2. W.G. Aref and H. Samet. Extending a DBMS with spatial operations. In O. Guenther and H.J. Schek, editors, *Advances in Spatial Databases (Proceedings of the 2nd Symposium on Large Spatial Databases - SSD'91)*, pages 299–318. Springer-Verlag, LNCS-525, Zurich, Switzerland, 1991.
3. S. Aronoff. *Geographic Information Systems: A Management Perspective*. WDL Publications, 1989.
4. J.K. Berry. Fundamental operations in computer-assisted map analysis. *International Journal of Geographical Information Systems*, 1(2):119–136, 1987.
5. R. David, I. Raynal, G. Schorter, and V. Mansart. Geo₂: Why objects in a geographical DBMS. In D. Abel and B.C. Ooi, editors, *Advances in Spatial Databases (Proceedings of the 3rd Symposium on Large Spatial Databases - SSD'93)*, pages 264–276. Springer-Verlag, LNCS-692, Singapore, 1993.
6. R. Elmasri and S.B. Navathe. *Fundamentals of Database Systems*. Benjamin-Cummings, 1989.
7. L.M. Haas and W.F. Cody. Exploiting extensible DBMS in integrated Geographic Information Systems. In O. Guenther and H.J. Schek, editors, *Advances in Spatial Databases (Proceedings of the 2nd Symposium on Large Spatial Databases - SSD'91)*, pages 423–449. Springer-Verlag, LNCS-525, Zurich, Switzerland, 1991.
8. L.M. Haas et al. Starburst mid-flight: As the dust clears. *IEEE Transactions on Knowledge and Data Engineering*, 2(1):143–160, 1990.
9. P. Milne, S. Milton, and J.L. Smith. Geographical object-oriented databases: A case study. *International Journal of Geographical Information Systems*, 7(1):39–55, 1993.
10. P. van Oosterom and T. Vijlbrief. Building a GIS on top of the open DBMS Postgres. In *Proceedings of the 2nd European Conference on Geographical Information Systems (EGIS'91)*, 1991.
11. H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1989.
12. H. Samet and W.G. Aref. Spatial data models and query processing. In W. Kim, editor, *Modern Database Systems*, pages 339–360. ACM Press, 1995.
13. M. Scholl and A. Voisard. Object-oriented database systems for geographic applications: An example with O₂. In G. Gambosi, M. Scholl, and H.W. Six, editors, *Geographic Database Management Systems*, pages 103–137. Springer-Verlag, 1992.
14. M. Stonebraker and L. Rowe. The design of Postgres. In *Proceedings of the ACM-SIGMOD Conference*, pages 340–355, Washington, D.C., 1986.
15. C.D. Tomlin. *Geographic Information Systems and Cartographic Modeling*. Prentice Hall, 1990.
16. T. Vijlbrief and P. van Oosterom. The GEO++ System: An extensible GIS. In *Proceedings of the 5th International Symposium on Spatial Data Handling*, pages 40–50, Charleston, South Carolina, 1992.

This article was processed using the L^AT_EX macro package with LLNCS style