preprint.sty

Managing Alternatives and Data Evolution in GIS

Claudia Bauzer Medeiros
DCC - IMECC - UNICAMP
Campinas, Brazil
cmbm@dcc.unicamp.br

Geneviève Jomier
Université Paris IX
Paris - France
jomier@etud.dauphine.fr

Abstract

This paper presents a solution for managing spatio-temporal data in a GIS database. This solution allows efficiently storing and handling temporal data and alternatives using a version mechanism. It can be used for different types of GIS-based applications, such as urban planning, environmental control and utility management.

1 Introduction

Geographic Information Systems (GIS) involve querying and analyzing massive amounts of georeferenced data, which is either organized in flat files or, more recently, managed by underlying database systems (DBMS). GIS queries may be roughly classified into four different families [Aro89]:

- presentation of the stored data;
- determination of spatial relationships between different phenomena;
- simulation and comparison of alternative scenarios based on combination of data layers; and
- prediction of the future.

Existing systems provide different facilities for handling the first two types of questions. These facilities usually consist of combining a query processor, a spatial data handler and graphical display tools on top of a data management system.

The two other types of queries, however, involve handling of spatio-temporal relations. The simulation of scenarios can be performed in specific situations, in a limited scale, using controlled parametrization of data values (see, for instance, a description of how this can be done in [HQGW93]).

The management of simulation results is however awkward, especially when the user wants to compare alternatives. The same applies to temporal evolution. Users are forced to manage time themselves, embedding appropriate code into their applications. Examples of queries demanded of GIS that fall into these categories are:

- Prevision of future based on recorded past: " What is the probability of flooding occurring in a certain area, given information collected along previous years? What are the possible damages – extent and intensity?"
- Analysis of temporal data evolution: "What has been the observed evolution of the stress in a vegetation colony? How has it been progressing and at which speed?"
- Comparative analysis of simulated scenarios: "What are the alternative for installing and expanding sanitation facilities, given observed trends in urban expansion?"
- Comparative analysis between actual data and simulated scenarios: "Given the actual state of a given area, how accurate were simulations performed to determine soil erosion in this area?

The difficulties posed to answering these queries involve factors that cannot be handled adequately by present GIS. The first issue is due to the nature of GIS data, which requires special indexing and buffering techniques. This area is the one that has merited the most attention from database researchers (e.g., [AS91, Fra91]). Present systems still have shortcomings in terms of performance and querying facilities. The addition of time and alternatives introduces other problems that are not satisfactorily supported in GIS.

This paper presents a solution to these problems. This solution is based on the DBV version model of [CVJ91]. This model, now being simulated on the O2 database system, allows efficiently keeping track of data versions in a database. It supports the creation of alternative scenarios, the identification of georeferenced features, and allows the management of scenarios and features through time with considerable savings in space. This solution is going to be tested in the DOMUS[1] environmental planning project, using real-life data layers from non-settled areas in the state of São Paulo, Brazil.

2 Characterization of GIS application demands

The rapid growth in GIS has resulted in a large number of systems, each of which with its own data storage and handling characteristics. In fact, it is only recently that

To appear in the ACM/ISCA Workshop on Advances in Geographic Information Systems, November 1993, Baltimore, USA

[1] domus(the latin for *home* – the Earth) is a joint project of researchers of the Computer Science Department and the Geosciences Institute at Unicamp.

GIS started to be implemented using DBMS. Most GIS are still based on a spatial data handler coupled to a sequential file manager, without any database facility (such as logical independence, storage management or query language support). Systems that use databases rely on combining a relational DBMS with special handlers which manipulate specific aspects of georeferenced data. The coupling of relational database systems to GIS data processing requirements has been done according to two architectures:

- proprietary systems – a special-purpose relational data base is tightly coupled with spatial data processing modules. Users cannot access the database directly and data cannot be migrated to standard relational systems;
- relational systems – a standard DBMS is used as a basis for spatial data access functions. Users can access the database directly, and data can be ported into other systems. Nevertheless, most special purpose features (e.g., geometric and image processing modules) are implemented by external packages.

The introduction of DBMS to support GIS has improved the services provided to end-users. However, these users still need more sophisticated data handling tools which are not yet supported.

Recently, there have been some prototypes developed on top of object-oriented systems, which aim to extend the services provided by relational DBMS(e.g., [KT92, ZM92, SV92]), but there is a lack of experimentation using real data.

GIS use basically two types of data: vector and raster. Data of different natures are stored in *layers* – also called *themes* or *chloropeth maps*. These layers are combined in different ways in order to process a query. One of the most common functions is the *map overlay*, employed on raster data.

The type of data used in an application depends on the domain and user requirements. Utility management (e.g., telephone or electricity planning) uses primarily vector data. Environmental control and natural resource planning use mostly raster data. When applications require merging the two kinds of data (e.g., outlining a road across a forest) this is done in two ways:

- image superposition – the vector data is drawn on top of the raster data, but there is no value processing. Users see the resulting images but cannot access the data directly.
- data conversion – vector data is converted to raster (or vice-versa) using builtin functions. The user can afterwards combine the different data sources to create new regions.

GIS demand that DBMS keep track of large amounts of georeferenced data, of different natures, collected using heterogeneous devices, and at different time periods. 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, and temporal data and alternatives can be managed.

Present GIS still lack facilities for providing the following services:

- automatic representation and management of spatio-temporal data evolution;
- handling of alternatives;
- identification of appropriate georeferenced features (data elements) to be combined in space and time.

These are the same type of problems that are faced by version mechanisms (even though the latter have not yet considered georeferenced data). Given the complexity of managing time and versions, GIS do not support these factors.

3 The DBV version mechanism

Versions are a means of storing different states of a given entity, thereby allowing the control of alternatives and of temporal data evolution. Versions are usually organized in a directed acyclic graph, which accompanies the history of data evolution across successive design trials.

The management of versions in databases has centered on different ways for keeping files. Research has appeared mostly in the context of software management (CASE systems) and CAD/CAM projects (e.g., [KSW86, Kat90, BBA91, TG92, KS92]). The subjects discussed cover the creation and manipulation of entity versions, their identification, the handling of time, status, authorization, and concurrency mechanisms. In object-oriented systems, this is aggravated by the intricate composition relationships between objects. Versions are also commonly used as a solution to concurrency control, especially for long transactions.

An important issue is the maintenance of *configurations*. A configuration is a set of versions of entities that represent some identifiable unit in the universe modelled. In many situations, the configuration becomes the versioning unit (i.e., users are not allowed to create versions of isolated entities, only of identifiable units). This is often the case of CAD environments.

Existing approaches support versions by means of chains of pointers, which keep track of connections among entity versions. There is often confusion between version (pointer and file) management and the underlying data model.

The DBV mechanism [CJ90, CVJ91] has a different approach. In this model, instead of keeping track of versions of individual entities, the problem is treated from a point of view where a unit of versioning, called *database version* – *Dv* for short – is a state of the universe modelled by the database (rather than just parts thereof).

The DBV approach does not use links between entity versions to determine to which consistent state they belong, as in the standard approach. It is based instead on the principle that the creation of an entity version $\mathcal{E}$ entails the creation of a new consistent logical database version *Dv* which will contain $\mathcal{E}$. The DBV approach sees therefore an augmented database that contains as many as necessary successive and alternative identified states of the modelled universe, and not only one state, as in conventional databases. The evolution of a DBV can be likened to the creation of new states of the universe in time. Each state is mapped onto a database version *Dv*, which is consistent and can evolve independently. Creating a new version for an entity corresponds to the appearance of another universe state. Temporal and alternative data can also be managed by this model – the database can be seen as a sequence of temporal states.

In the standard approach, when some entity version $\mathcal{E}$ is created, several chains have to be established: some chains link other versions of the same entity to its new version $\mathcal{E}$; other chains connect $\mathcal{E}$ to other entity versions to form a new configuration, maintaining configuration integrity. In the

DBV model, the connection of $\mathcal{E}$ to a configuration is instead achieved by an adhoc identification mechanism which uses "version stamps". Each *Dv* contains a logical version of each entity in the modelled universe. Identical logical versions of an entity are mapped on the same physical version of this entity using these "version stamps". Thus the creation of a new database version *Dv* does not require physical duplication of versions of entities. Keeping up consistent configurations is automatically performed by the version manager by examining tables of version stamps. Implementation details appear in [CJ90].

In the DBV model, time becomes an implicit feature by allowing timestamps to be used as part of version stamps. Thus, temporal queries do not require handling of special attributes; rather, they are processed by the versioning mechanism, which puts together data that belongs to the same *Dv* (temporal) state.

In order to properly associate an entity with its versions, the DBV mechanism relies on the notion of *identity*. In object-oriented systems, this is easily solved by taking advantage of the object id (oid) concept. In relational systems, internal surrogates can be used to the same purpose.

GIS naturally support the identity notion, which can be associated to the coordinates of features. This can be done using either raster or vector data, and thus the underlying storage model does not affect DBV version management. Thematic layers can furthermore be handled as version configurations. Thus, georeferenced data are prime subjects for DBV mechanism management.

4 Versions in GIS – other approaches

A good introduction to the problems of handling spatio-temporal data in GIS are the set of papers in [FCF92]. Research on versions has not dealt with GIS related problems. The reverse is also true: there are very few reports of use of version mechanisms in GIS(e.g., [Bat92, NTE92]). Their use in handling alternatives is never mentioned. Their support for temporal data is discussed from a file manager point of view. Existing papers usually are based on stressing the application needs, and database version management problems are glossed over.

The GFIS [Bat92] system uses a standard relational DBMS coupled to a geographic data manager. Version management is left to the database system, and is geared towards controlling concurrent access. There is no possibility of selecting versions for queries, or of handling sequences of past states.

[NTE92] discuss different data structures for implementing versions on top of tables using an object-oriented language. The paper provides a comparative analysis of these structures, but does not apply them to real data.

5 Conclusions

This paper presented a solution for the management of spatio-temporal data in GIS which consists in using the DBV version mechanism. The mechanism is orthogonal to the data model and the concurrency control issues, which are complicating factors in other version models. It solves the problem of handling of alternatives and of data evolution by associating identifiers to each data feature, and using timestamps in the creation of version identifiers.

Furthermore, this mechanism can be used to process other GIS functions, such as map overlay queries (where identifiers are used to match features in different layers), or map customization (combining alternative versions for a region).

We intend testing this solution against spatio-temporal data available in the DOMUS project, as part of an environmental planning project. Tests will use georeferenced data about the Cantareira region in the São Paulo state (roughly, 2.000 km2) [PMB93].

Acknowledgements

The research described in this paper was partially financed by grants FAPESP 91/2117-1, CNPq 453176/91, and CNPq 452357/93-4.

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.

[Bat92] P. Batty. Exploiting relational database technology in a GIS. *Computers and Geosciences: An international journal*, 18(4):453–462, 1992.

[BBA91] M. Borhani, J-P Barthès, and P. Anota. Versions in Object-Oriented Databases. Technical Report UTC/GI/DI/N 83, Universite de Technologie de Compiegne, 1991.

[CJ90] W. Cellary and G. Jomier. Consistency of Versions in Object-Oriented Databases. In *Proc. 16th VLDB*, pages 432–441, 1990.

[CVJ91] W. Cellary, G. Vossen, and G. Jomier. Multiversion Object Constellations for CAD Databases. Technical Report 9105, Justus-Liebig Universitat Giessen, 1991.

[FCF92] A. Frank, I. Campari, and U. Formentini, editors. *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space.* Lecture Notes in Computer Science 639. Springer-Verlag, 1992.

[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.

[HQGW93] N. Hachem, K. Qiu, M. Gennert, and M. Ward. Managing Derived Data in the GAEA Scientific DBMS. In *Proc 19th VLDB*, pages 1–12, 1993.

[Kat90] R. H. Katz. Toward a Unified Framework for Version Modelling in Engineering Databases. *ACM Computing Surveys*, 22(4):375–408, 1990.

[KS92] W. Kafer and H. Schoning. Mapping a Version Model to a Complex-Object Data Model . In *Proc IEEE Data Engineering Conference*, pages 348–357, 1992.

[KSW86] P. Klahold, G. Schlageter, and W. Wilkes. A General Model for Version Management in Databases. In *Proc XII VLDB*, pages 319–327, 1986.

[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.

[NTE92] R. Newell, D. Theriault, and M. Easterfieldy. Temporal GIS - modeling the evolution of spatial data in time. *Computers and Geosciences: An international journal*, 18(4):427–434, 1992.

[PMB93] F. Pires, C. B. Medeiros, and A. Barros. Modelling Geographic Information Systems using an Object Oriented Framework. In *Proc XIII International Conference of the Chilean Computer Science Society*, pages 217–232, 1993.

[SV92] M. Scholl and A. Voisard. *Building and Object-oriented System – the Story of O2*, chapter Geographic Applications – an Experience with O2. Morgan Kaufmann, California, 1992.

[TG92] V. Tsotras and B. Gopinath. Optimal Versioning of Objects . In *Proc IEEE Data Engineering Conference*, pages 358–365, 1992.

[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.