Using Versions in GIS

Claudia Bauzer Medeiros
Geneviève Jomier

Relatório Técnico DCC–94-05

Junho de 1994

Using Versions in GIS

Claudia Bauzer Medeiros[*] Geneviève Jomier[†]

Abstract

Geographic information systems GIS have become important tools in public planning activities (e.g, in environmental or utility management). This type of activity requires the creation and management of alternative scenarios, as well as analysis of temporal data evolution. Existing systems provide limited support for these operations, and appropriate tools are yet to be developed.

This paper presents a solution to this problem. This solution is based on managing temporal data and alternatives using the DBV version mechanism. It provides efficient handling and storage of versions, and supports the creation of alternatives for decision-making activities.

A reduced version of this report appeared in the Proceedings of the DEXA'94 Conference – 5th International Conference on Database and Expert Systems Applications, Athens, Greece.

Keywords: versions, GIS, georeferenced data evolution.

1 Introduction

Geographic Information Systems (GIS) are automated systems that manipulate *georeferenced data* – data about geographic phenomena associated with their spatial relationships and location on the terrestrial surface. The term *georeferenced entity*, in this paper, refers to any type of

---

[*] DCC-UNICAMP - Campinas, Brazil. cmbm@dcc.unicamp.br

[†] Université Paris IX Dauphine - Paris, France. jomier@lamsade.dauphine.fr

entity whose components are georeferenced (be it a house, a residential plot or a geographic region).

The number and type of applications and analyses that can be performed by a GIS are as large and diverse as the available geographic data sets [Aro89]. Examples are urban planning, natural resources administration, facility management, demography, cartography, and archaeology [MGR91].

GIS data contains *spatial* and *thematic* components [KBS91]. Thematic (non-spatial) components are alphanumeric values related to georeferenced entities, e.g., the name of a mountain, or the type of vegetation cover. Texts or images are considered to be unconventional thematic data. Spatial data has two different properties: *geometric properties* such as spatial location, size and shape of spatial objects; and *topological properties* such as connectivity, adjacency, inclusion and containment, modelling relationships between geometric data.

Data of different natures are stored in *thematic layers* – also called *themes* or *chloropeth maps*. These layers are combined in different ways in order to process a query. Layers contain alphanumeric data and spatial information. A fauna layer of an area, for instance, contains textual data (e.g., animal species) and spatial information (area occupied by species).

GIS queries involve one or more of the following issues [Aro89, Flo91, Peu93]:

1. *What* kind of phenomenon is this? (describe non-spatial characteristics of an entity).
2. *Where* is this phenomenon located? (describe the spatial characteristics of a given georeferenced entity, which comprises its location, topological and geometrical characteristics, and often its relationships with other entities).
3. *When* was this data collected? (determine data validity period).
4. What did this entity look like at some *past* period? What will happen to it in some *future* period? (examine previous states of an entity and predict its future evolution, given its recorded behavior).

5. *What* would happen to an entity *if* certain events were to take place? (simulation and comparison of alternative scenarios based on changing existing data).

Database research for GIS support has been centered on *spatial databases*, especially in what concerns algorithms for storing and accessing spatial data (e.g., [Sam89, Fra91]). Spatial structures usually support *geometric operations*, which treat geographic phenomena as if they were points, line segments and polygons (the so-called *vector* format). Related research includes geographic query languages (e.g., [RFS88, Ooi90]) and the development of GIS prototypes using new generation databases (e.g., [vOV91, SV92, MMS93, LL93, SA93]).

Thus, database researchers are concerned with providing users with fast access to georeferenced data by means of spatial indices. They are, furthermore, developing different types of query languages and mechanisms in order to allow processing the first two kinds of queries (i.e., *what, where*). Existing systems also provide different facilities for handling these two 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 remaining kinds of queries, however, involve other types of knowledge and distinct storage and indexing facilities. *When* processing (queries 3 and 4) involves temporal database management, itself a matter of intensive research (see, for instance, the bibliography in [Soo91] or the discussion of open issues in [JCG+92]).

The simulation of scenarios (query 5) is supported by some systems for 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 combination of simulation results is, however, not allowed, especially when the user wants to compare alternatives. Users have to store the different scenarios in separate files, and have to handle themselves the management of these files, by embedding appropriate code into their applications.

This paper extends the work of [MJ93], presenting and detailing a framework which allows processing queries 1 through 5. This solution is based on the notion of database versions, and adopts the DBV version model of [CVJ94]. This model, now being simulated on the O2 database system, allows efficiently keeping track of data versions in a database.

The DBV model enables the simultaneous management of distinct alternative scenarios, which can be compared for planning purposes. Georeferenced feature evolution through time can be managed by this model with considerable savings in space. As well, the DBV model can be embedded in any database system and does not require that the end-user control the different data configurations.

The main contributions are:

- analysis of the problem of georeferenced data evolution from a database point of view, rephrasing this problem in a versioning framework;
- description of how the DBV version model can support the management of this evolution, using concrete examples. So far, GIS have not considered version mechanisms, given the complexity of the factors involved.

The research discussed in this paper is part of the DOMUS[1] environmental planning project, which uses real-life data layers from non-settled areas in the state of São Paulo, Brazil.

The rest of this paper is organized as follows. Section 2 characterizes GIS applications demands from a database point of view, and points out problems in database support to these applications. Section 3 gives an overview of the DBV mechanism. Section 4 presents a detailed example of how this mechanism can support management of scenarios and data evolution. Section 5 shows how other version mechanisms are unable to perform this task satisfactorily. Finally, section 6 presents conclusions and directions for future research.

---

[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.

2 GIS application requirements

GIS demand that DBMS keep track of massive 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 [GB90, MJ93].

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 the early systems, data was organized in flat files. New systems are based on relational database management systems (DBMS). However, several of the requirements of GIS applications are not provided by standard relational DBMS. Thus, special data handlers have been developed to interact with the stored relations and allow the management and display of georeferenced data. Nevertheless, these commercial systems still lack the extensibility and flexibility desired by end-users, especially in what concerns support for planning activities.

The coupling of database systems to GIS data processing requirements has been done according to the following architectures [MP94]:

- 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 – e.g., [Mor92].

- extensible systems – these use the facilities provided by extensible relational or object-oriented DBMS embedding the spatial dimension in the system. The formulation of spatial queries is directly supported in the extensible query language.

Relational databases do not provide an adequate underlying model to support most types of geographic data [Ege92]: the use of tables with fixed number of attributes does not allow flexibility in the management of georeferenced data, nor the incremental development of applications. Thus, researchers have directed their attention to new architectures.

New architectures rely on extensible relational (e.g., [HC91, vOV91]), object-oriented (e.g, [KT92, ZM92, SV92]) or rule-based (e.g., [SA93, SRD$^+$91]) systems. A comparison in performance and flexibility of relational and extended relational systems is found in [SFGM93]. Object-oriented systems are presented as a solution to the need for representing the dynamics of the real world (using methods). The possibility of progressively building up complex entities by the repeated application of orthogonal constructors helps constructing complex scenarios. Finally, rule-based systems are designed with specific applications in mind, and aim to help users in their queries, by limiting the size of the query universe.

In all of the above, there is no consideration for temporal queries or comparison of scenarios (such as queries 3 through 5 of section 1). 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 is aggravated by the introduction of the time element, not supported in commercial databases. Thus, users are forced to manage time values themselves, if they want to analyze data evolution.

Another issue concerns data integration over time. Georeferenced data may be collected at different time periods. This creates another type of value inconsistency, which is due to the temporal evolution of georeferenced entities. Thus, if the mapping of a region takes several months (or sometimes years), differences will occur which must be taken into consideration. As pointed out by [Flo91], another related problem is the difference in time scales. Some phenomena (e.g., vegetation) fluctuate according to a seasonal cycle, whereas others (e.g, temperature) may vary on a daily basis. Thus, when posing queries that consider the evolution of georeferenced phenomena for a given region, these factors must be taken into consideration.

GIS use basically two data formats: *vector* and *tesselation*. The tesselation format is usually called *raster*, which is a specific form of tesselation. In this format, spatial objects are described as polygonal units of space – cells – in a matrix. Each cell contains one thematic value (i.e., there cannot be two types of soil for a given cell). The vector format treats spatial entities as points, lines and polygons, using lists of coordinate pairs. Boundaries of regions are stored precisely, and several attributes can be associated to a single element.

The type of data used in an application depends on its domain and on user requirements, as well as on the scale of the problem analyzed. Utility management (e.g., telephone or electricity planning) or cartography use primarily vector data. Environmental control and natural resource planning use mostly raster data. Nevertheless, there is no clearcut definition of which type of format is more adequate for a given application, and recent results show that both should be provided in a database system [Cou92, Goo91].

In this paper we assume that the vector-raster conversion poses no problem, and that data is kept in vector format. This type of format allows the identification of entities (by means of keys in the relational model or oids in an object model). It is, therefore, more adequate for index keeping and managing of individual georeferenced phenomena by a DBMS.

3 The DBV Version Mechanism

As seen in the previous section, present GIS still lack facilities for providing the following services:

- automatic representation and management of data evolution in time;
- management of alternative scenarios for planning purposes.

These are the same type of problems that are faced by version mechanisms (even though the latter have not yet considered georeferenced data). Thus, it is only natural to examine the feasibility of adopting versions to allow such services.

3.1 Versions in databases

Versions are a means of storing different states of a given entity, thereby allowing the control of alternatives and of temporal data evolution. 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, LST93]). 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 proposed for dealing with concurrency control, especially for long transactions. In this last context, different users are granted access to copies (i.e., versions) of the same set of data, thus allowing them to work in parallel.

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. This is often the case of CAD environments.

3.2 An overview of the DBV mechanism

Version mechanisms must put together entity versions to reflect a given state of the modelled universe. Existing approaches support this by means of chains of pointers, which keep track of connections among versions of a given entity, as well as among entities that belong to a given version state. Thus, the database is perceived as a set of entities connected by several linked chains. There is furthermore often confusion between version management and the underlying data model.

The DBV mechanism [CJ90, CVJ94] has a different approach. The main principles of this model are the following [GJ94]:

- The user should always be able to manage entity versions within a specific, identifiable, context. This context is called *database version* – *Dv* for short. *Dv* represents in fact a "version", or identifiable state, of the modelled universe.

- Each *Dv* contains a logical version of each (identified) entity of the database. The value of this version may be *nil*, meaning that it does not exist in a given *Dv*.

- If several logical versions of an entity have the same value, there is no physical replication. Rather, these logical versions share the same physical version. The mechanism of mapping logical/physical versions of entities is transparent to the user.

Thus, instead of considering versions of an entity in isolation, the model allows managing them within their appropriate framework.

Rather than keeping track of versions of individual entities, the problem is treated from a point of view where the unit of versioning is the *Dv* context, which corresponds to a state of the universe modelled by the database, regardless of the underlying data model. The notion of version as seen by the user (the *logical database versions*) is independent of what is actually stored (*physical versioning*), and there is no replication of data.

From a logical point of view, the database is perceived as a set of consistent database versions *Dv*, which can evolve independently of each other. The user works in this augmented database by selecting the *Dv* context(s) of interest. Once the desired *Dv* is selected, the user can treat it as a database on its own, querying and updating it. Logical operations on *Dv* are translated into actual physical versioning operations by the version management system. Temporal and alternative data are naturally managed by this model.

The versioning of an entity is logically performed by versioning the entire logical database to which it belongs. The logical independence of *Dv* allows defining two types of update transactions: those that manipulate some *Dv*; and transactions that derive a new *Dv*. The latter can be described in two steps. Let $\mathcal{E}$ be an entity of a given *Dv*, for which the user wants to create a new version $\mathcal{E}_\infty$:

- first, a (logical) copy of *Dv* is created, corresponding to a new Dv_1 database version, identical to *Dv*;
- second, Dv_1 is updated (i.e., $\mathcal{E}_\infty$ is created from $\mathcal{E}$, and additional updates are performed in order to maintain the consistency inside Dv_1).

Thus, a multiversion database is a set of logical *consistent* states, each of which corresponds to a different version of the world, created by the user. Each *Dv* is, therefore, a unit of consistency and can evolve independently.

Physically, in order to properly associate an entity with its versions, the DBV mechanism relies on the notion of *identity*: each *Dv* has an associated identifier which is used for managing purposes. The identifier mechanism associates every logical entity version $\mathcal{E}_\rangle$ to its proper context Dv_i.

The retrieval of each *Dv* is automatically ensured by the version manager by examining tables of identifiers. Implementation details appear in [CJ90].

For instance, time may be supported by mapping timestamps into the DBV identifiers. 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 identifiable temporal state.

4 Applying the DBV model to GIS

This section shows an extended example of how the DBV approach allows handling GIS queries for evolution of data and comparison of alternatives.

Consider the following sequence of queries:

- Analysis of temporal data evolution: "What has been the observed modifications of the forest cover in a given area for a specific time period?"

- Prevision of future based on recorded past: “ What is the probability of this forest cover decreasing in area, given information collected along this period? What are the possible damages – extent and intensity?”
- Comparative analysis between actual data and simulated scenarios: “Given the actual state of the forest cover in the area, how accurate were simulations performed along this period to determine its evolution during the same period?

All these queries concern the same type of theme – vegetation – for the same geographical area. They require searching through different (historical) scenarios, first identifying the area and then its forest cover.

The first query requires doing a statistical analysis of a historical sequence of vegetation data. The second query requires using the previous analysis to perform prediction of phenomena. The third query assumes that the two other queries were periodically posed in the past, and that their results were stored. Thus, it demands comparison of predicted behavior and actual observed behavior for several periods in the database's history.

As explained in the previous section, in the DBV approach, the user is provided with an integrated view of the world: it is perceived as a set of database states (*Dv*), each corresponding to an independent consistent version of the user's universe. The evolution of georeferenced entities is accompanied by the corresponding evolution of *Dv* states. For the user, there is no predefined link between different *Dv*, which allows working either over one single context or navigating across contexts. This corresponds to what the user manipulates in reality, since no georeferenced phenomenon can be treated in isolation.

There is no difference in treatment for actual (measured) data values or (alternative) predicted values. Thus, several *Dv* may exist for the same time period, each describing a state of the world – either an actual recorded state or some alternative artificial state generated for planning purposes. Therefore, for any time period, the database may contain a set of logical databases: the modelled real world and different simulated scenarios. For vegetation cover, for instance, one can keep track of several parallel scenarios by modifying distinct parameters, e.g,, rainfall or evolution of human settlements.

Thus, for any of the three queries, the processing is performed as:

- (i) Select all *Dv* within the specified time period;

  (The version manager performs this operation by accessing the identifier of each *Dv*, which in this case will contain a timestamp identification)

- (ii) For each such *Dv*, select the area and its cover, by performing standard GIS (nontemporal) database queries;

  (Each *Dv* selected in the previous step is seen by the user as an independent consistent database. Thus, it can be queried independently of the rest of the DBV database, regardless of other existing versions.)

- (iii) Perform the simulation operations on the set of areas and covers obtained from the execution of the two previous steps.

Physically, the creation of a logical database does not require physical duplication of entities, just creation of identifiers and recording of data changes (differential information). These changes and the corresponding identifiers are used to build the complete (logical) database states *Dv*.

The *Dv* are built by gathering together all entities present at a given database state, by means of special index structures. This means retrieving all entities that have compatible identifier values.

In addition, query processing may be speeded up by using the notion of configuration. In fact, a configuration characterizes a unit of work inside a context (and thus of consistency). Thus, users may decide to specifically identify a configuration containing the area and its cover (requested in the three queries). This will speed up version processing for this type of query (in a way similar to precomputed views).

Finally, for specific situations, the user may also require the creation of a *constellation*. This is a set of multiversion entities together with all their components. Constellations can vary in size and be created dynamically by the user. Thus, in the example of the vegetation cover queries, the user may specify that a constellation be created for the specific area and its vegetation cover. Then, further queries of the same nature will be simplified to retrieving one constellation, which will have all the recorded versions of the area and its vegetation cover across all *Dv* (both real and alternative scenarios). For details on constellation management the reader is referred to [CVJ91].

5 Other version mechanisms and GIS

A good introduction to the problems of handling spatio-temporal data in GIS are the set of papers in [FCF92], which cover different issues. They range from problems in database support of time [Sno92] to discussing the concept of a region in creating study scenarios [Gut92]. The need for flexible mechanisms to allow managing of these scenarios is stressed in several papers.

Database research on versions has not dealt with problems related to GIS, and there are very few reports of GIS using version mechanisms (e.g., [Bat92, NTE92]). The main reasons for this are:

- Most version management mechanisms available in database systems become cumbersome when it comes to managing the evolution of instances. They require the maintenance of complex data structures to allow following data evolution.
- GIS data is complex and occupies considerable space. Thus, its management already presents so many challenges to a DBMS that there is no question of coupling it to the usual version mechanisms.
- In many cases, it is impossible to follow the evolution of phenomena across time periods, since entities may disappear or suffer unexpected modifications.

Thus, even though versioning solves users' problems, it has not, so far, been seriously considered in the GIS context, and is used at most to support parallel access to data. Their use as a means to manage temporal data evolution is not considered. Rather, researchers consider (historical) versions of entire files – e.g., the sequence of file versions for a given thematic layer. Finally, when versions are associated with georeferenced entities, there is no concern with how to manage them from a database point of view.

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 parallel 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.

We now briefly review how traditional version schemes would cope with the queries discussed in section 4. In such schemes, two types of approach are possible:

- **Snapshot view** The complete data files (layers, with spatial and thematic information) are stored, together with time stamp indication. (I.e., the database is in fact a set of database snapshots, where each snapshot contains several thematic layers.)

  Thus, in order to answer the first two queries, the system has to:

  - (i) retrieve the entire layer files for the time period;
  - (ii) for each layer, select the desired area and determine its forest cover;
  - (iii) produce the time series analysis desired.

  In order to answer the third query, the database must contain not only the entire layers for every period of interest, but also layer files describing simulation results.

  This first solution, though relatively simple to process, entails massive occupation of storage, and is therefore not feasible for practical purposes. The snapshot approach requires the actual recording of the entire database, and thus the variety and periodicity of stored phenomena is limited, due to size constraints.

- **Historical chain view** The history of entities is maintained through a linked list of data values and timestamps: only differential values are kept. (The database is seen as a conglomerate of linked chains in all directions.)

  In this case, queries can only be answered for entities whose history has been maintained through chains. This requires that the database designer has previous knowledge of all possible queries that will involve version manipulation. Alternatively, these chains can be maintained for every entity and value in the database. Whereas the first alternative limits user exploratory activity, the second alternative requires a heavy overhead of pointers.

  Supposing the historical chain of the designated area is available, then the queries are processed by the following procedure:

  - (i) find the area and its vegetation cover in the present;
  - (ii) follow back pointers of this area and cover, retrieving past information;
  - (iii) produce the time series analysis desired.

  Finally, the third query requires that not only actual historical chains be maintained, but also prevision chains for the same entity. This, again, complicates the housekeeping algorithms.

  The pointer version mechanisms soon become too cumbersome to manage when each time period contains many entities that vary in different ways, as is typically the case in geographical applications.

It is interesting to compare these approaches to the DBV solution. The snapshot approach favors users who need to access entire contexts. However, it does not automatically support navigation through these contexts, since there is no sharing of entities across the snapshots. For instance, any update to an entity in a given snapshot must be manually performed by the user in the other snapshots. The historical chain approach is geared towards management of versions of individual entities, through manipulation of their chains. However, it does not automatically support the building of contexts, which must be performed by the user.

The DBV model, on the other hand, automatically supports both working within a context and comparing entity versions across contexts. This is achieved thanks to the fact that this model allows separating the logical versioning from the physical versioning level. It thus combines the advantages of the other approaches, without the inconvenients: it does not imply the waste of space of the snapshot approach, and neither does it demand the complex computation procedures of the historical chain approach.

6 Conclusions

This paper presented a solution for the management of evolution of georeferenced data in GIS which consists in using the DBV version mechanism. This solution allows the development of automated tools to keep track of different versions of the same georeferenced entity through time. This facility enables users to create and manage alternative scenarios, as well as to keep track of temporal data evolution. This type of support has so far been unavailable in commercial GIS, though required by different kinds of planning applications.

The use of a version mechanism, as discussed in this paper, seems to be an obvious choice to cope with GIS users' demands. However, GIS databases do not consider this type of facility, since the handling of georeferenced data presents in itself many problems. Furthermore, available version management systems are complex and cannot readily satisfy GIS requirements.

The DBV mechanism, on the other hand, allows efficiently keeping track of data and schema versions in a database, with considerable savings in space and computation time, as compared to other database version mechanisms. It allows dissociating the issues of context and configuration consistency from version maintenance, which is not possible in other version models.

Its main advantage, from a GIS point of view, is that it allows users to access entire consistent database states for any given entity version. Thus users can create different scenarios by just modifying individual entities, and need not worry about keeping them within their appropriate context. This is achieved without additional overhead, by the appropriate management of version identifiers (as opposed to traditional mechanisms that require handling pointer chains). Finally, the DBV model is orthogonal to the underlying data model and to concurrency control, which are complicating factors in other version models.

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.

[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.

[Cou92] H. Couclelis. People Manipulate Objects (but Cultivate Fields): Beyond the Raster-Vector Debate in GIS. In *Proc International Conference on GIS - From Space to Territory: Theories and Methods of Spatial Reasoning*, Springer Verlag Lecture Notes in Computer Science 639, pages 65–77, 1992.

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

[CVJ94] W. Cellary, G. Vossen, and G. Jomier. Multiversion Object Constellations: a new Approach to Support a Designer's Database Work. *To appear, Engineering with Computers*, 1994.

[Ege92] M. Egenhofer. Why not SQL! *International Journal of Geographical Information Systems*, 6(2):71–86, 1992.

[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.

[Flo91] R. Flowerdew. *Geographical Information Systems - volume I*, chapter Spatial Data Integration, pages 375–387. John Wiley and Sons, 1991.

[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.

[GB90] O. Guenther and A. Buchman. Research Issues in Spatial Databases. *ACM Sigmod Record*, 19(4):61–68, 1990.

[GJ94] S. Gancarski and G. Jomier. Le Modele de Versions de Bases de Donnees: Formalisation. In *Submitted for publication*, 1994.

[Goo91] M. Goodchild. Spatial Analysis with GIS: Problems and Prospects. In *Proc GIS/LIS'91*, volume 1, pages 40–48, 1991.

[Gut92] J. Guttenberg. Towards a Behavioral Theory of Regionalization. In *Proc International Conference on GIS - From Space to Territory: Theories and Methods of Spatial Reasoning*, Springer Verlag Lecture Notes in Computer Science 639, pages 110–121, 1992.

[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.

[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.

[JCG$^+$92] C. Jensen, J. Clifford, S. Gadia, A. Segev, and R. Snodgrass. A Glossary of Temporal Database Concepts. *ACM Sigmod Record*, 21(3):35–43, 1992.

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

[KBS91] H. Kriegel, T. Brinkhoff, and R. Schneider. The Combination of Spatial Access Methods and Computational Geometry in Geographic Database Systems. In *Proc 2nd Symposium Spatial Database Systems*, pages 5–22. Springer Verlag Lecture Notes in Computer Science 525, 1991.

[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.

[LL93] Y. Leung and K. S. Leung. An Intelligent Expert System Shell for Knowledge-based Geographical Information Systems: 1 - the tools. *International Journal of Geographical Information Systems*, 7(1):189–200, 1993.

[LST93] G. Landau, J. Schmidt, and V. Tsotras. Efficient Support of Historical Queries for Multiple Lines of Evolution. In *Proc. IEEE Data Engineering Conference*, pages 319–326, 1993.

[MGR91] D. J. Maguire, M. F. Goodchild, and D. W. Rhind, editors. *Geographical Information Systems - volume II - Applications.* John Wiley and Sons, 1991.

[MJ93] C. B. Medeiros and G. Jomier. Managing Alternatives and Data Evolution in GIS. In *Proc. ACM/ISCA Workshop on Advances in Geographic Information Systems*, pages 34–37, 1993.

[MMS93] P. Milne, S. Milton, and J. Smith. Geographical Object-oriented Databases: a Case Study. *International Journal of Geographical Information Systems*, 7:39–56, 1993.

[Mor92] S. Morehouse. The ARC/INFO geographic information system. *Computers and Geosciences: An international journal*, 18(4):435–443, 1992.

[MP94] C. B. Medeiros and F. Pires. Databases for GIS. *ACM Sigmod Record*, 23(1):107–115, 1994.

[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.

[Ooi90] B. C. Ooi. *Efficient Query Processing in Geographic Information Systems.* Springer Verlag Lecture Notes in Computer Science, 1990.

[Peu93] D. Peuquet. What, Where and When - a Conceptual Basis for Design of Spatiotemporal GIS Databases. In *Proc. ACM/ISCA Workshop on Advances in Geographic Information Systems*, pages 117–122, 1993.

[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.

[RFS88] N. Roussoupolos, C. Faloutsos, and T. Sellis. An Efficient Pictorial Database System for PSQL. *IEEE Transactions on Software Engineering*, SE-14(5):639–650, 1988.

[SA93] R. Subramanian and N. Adam. Ill-defined Spatial Operators in Geographic Databases: their Nature and Query Processing Strategies. In *Proc. ACM/ISCA Workshop on Advances in Geographic Information Systems*, pages 88–93, 1993.

[Sam89] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1989.

[SFGM93] M. Stonebraker, J. Frew, K. Gardels, and J. Meredith. The Sequoia 2000 Benchmark. In *Proc ACM SIGMOD Conference*, pages 2–11, 1993.

[Sno92] R. Snodgrass. Temporal Databases. In *Proc International Conference on GIS - From Space to Territory: Theories and Methods of Spatial Reasoning*, Springer Verlag Lecture Notes in Computer Science 639, pages 22–63, 1992.

[Soo91] M. Soo. Bibliography on Temporal Databases. *ACM SIGMOD RECORD*, 20(1):14–23, 1991.

[SRD$^+$91] A. Skidmore, P. Ryan, W. Dawes, D. Short, and E. O'Loughlin. Use of an Expert System to Map Forest Soils from a Geographical Information System. *International Journal of Geographical Information Systems*, 5(4):431–446, 1991.

[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.

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

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

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

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

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

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

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

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

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

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

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

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

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

Relatórios Técnicos – 1993

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

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

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

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

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

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

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

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

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

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

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

93-12 **Boole's conditions of possible experience and reasoning under uncertainty,** *Pierre Hansen, Brigitte Jaumard, Marcus Poggi de Aragão*

93-13 **Modelling Geographic Information Systems using an Object Oriented Framework,** *Fatima Pires, Claudia Bauzer Medeiros, Ardemiris Barros Silva*

93-14 **Managing Time in Object-Oriented Databases,** *Lincoln M. Oliveira, Claudia Bauzer Medeiros*

93-15 **Using Extended Hierarchical Quorum Consensus to Control Replicated Data: from Traditional Voting to Logical Structures,** *Nabor das Chagas Mendonça, Ricardo de Oliveira Anido*

93-16 **$\mathcal{LL}$ – An Object Oriented Library Language Reference Manual,** *Tomasz Kowaltowski, Evandro Bacarin*

93-17 **Metodologias para Conversão de Esquemas em Sistemas de Bancos de Dados Heterogêneos,** *Ronaldo Lopes de Oliveira, Geovane Cayres Magalhães*

93-18 **Rule Application in GIS – a Case Study,** *Claudia Bauzer Medeiros, Geovane Cayres Magalhães*

93-19 **Modelamento, Simulação e Síntese com VHDL,** *Carlos Geraldo Krüger e Mário Lúcio Côrtes*

93-20 **Reflections on Using Statecharts to Capture Human-Computer Interface Behaviour,** *Fábio Nogueira de Lucena e Hans Liesenberg*

93-21 **Applications of Finite Automata in Debugging Natural Language Vocabularies,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

93-22 **Minimization of Binary Automata,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

93-23 **Rethinking the DNA Fragment Assembly Problem,** *João Meidanis*

93-24 EGOLib — **Uma Biblioteca Orientada a Objetos Gráficos,** *Eduardo Aguiar Patrocínio, Pedro Jussieu de Rezende*

93-25 **Compreensão de Algoritmos através de Ambientes Dedicados a Animação,** *Rackel Valadares Amorim, Pedro Jussieu de Rezende*

93-26 GeoLab: **An Environment for Development of Algorithms in Computational Geometry,** *Pedro Jussieu de Rezende, Welson R. Jacometti*

93-27 **A Unified Characterization of Chordal, Interval, Indifference and Other Classes of Graphs,** *João Meidanis*

93-28 **Programming Dialogue Control of User Interfaces Using Statecharts,** *Fábio Nogueira de Lucena e Hans Liesenberg*

93-29 EGOLib – **Manual de Referência,** *Eduardo Aguiar Patrocínio e Pedro Jussieu de Rezende*

Relatórios Técnicos – 1994

94-01 **A Statechart Engine to Support Implementations of Complex Behaviour,** *Fábio Nogueira de Lucena, Hans K. E. Liesenberg*

94-02 **Incorporação do Tempo em um SGBD Orientado a Objetos,** *Ângelo Roncalli Alencar Brayner, Claudia Bauzer Medeiros*

94-03 **O Algoritmo KMP através de Autômatos,** *Marcus Vinícius A. Andrade e Cláudio L. Lucchesi*

94-04 **On Edge-Colouring Indifference Graphs,** *Celina M. H. de Figueiredo, João Meidanis, Célia Picinin de Mello*

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

`reltec@dcc.unicamp.br`