GIS Databases: From Multiscale to MultiRepresentation[1]

Stefano Spaccapietra*, Christine Parent**, Christelle Vangenot*

* Swiss Federal Institute of Technology Lausanne (EPFL)
EPFL-DI-LBD, 1015 Lausanne, Switzerland
{spaccapietra, vangenot}@epfl.ch
** University of Lausanne - HEC Inforge
1015 Lausanne, Switzerland
Christine.Parent@hec.unil.ch

Abstract. *Cartography is one of the major application areas using geographical databases. Whether it is for the business of producing paper maps for sale, or whether it is for displaying maps on a screen to visualize the result of a query, we need computer systems that know how to represent the same geographical area at different scales. The concept of multiscale database has become popular in the GIS domain as a way to enforce consistency between representations and reduce the global update load. Scaling, however, is just one of the facets that may lead to keeping several representations for the same real-world object. Viewpoint and classification are two major abstractions in the design process that also generate multiple representations. This paper investigates the generic issues and solutions to achieve flexible support of multiple representation in a GIS database.*

1 Introduction

Geographic data has become quite popular. It plays a major role in information services to citizens, as one of the most common concerns in everyday life is locating something we are looking for, or finding a way to reach it. It is the essence of an increasing variety of societal management applications that range from land management and ecological monitoring to housing or traffic control. Finally, its economic importance is recognized by businesses that discover the benefits of geomarketing strategies.

Maps are the most natural way to convey geographical information, and they are excellent support to visualize analytical data about phenomena that have a geographical extent. This includes geography-compliant maps, that show items of interest as faithfully as possible with respect to their real-word location and shape, as well as schematic maps (e.g., city transport systems, airline connections diagrams, train networks, facility management networks), where the focus is on correct

[1] This work is supported by EEC and OFES as part of the MurMur project within the context of the 5th Framework IST Programme (project number 10723). C.Vangenot is supported by FNRS (Swiss National Research Fund) under contract 2100-046664.

(topological) connections and readability rather than on precisely locating lines and nodes.

A map is drawn according to a given scale. At different scales, the same information is usually drawn differently (not just magnified or reduced) because of:

- drawing conventions that may change from one scale to another one,
- items that may appear/disappear or be aggregated/de-aggregated because their size make them visible/invisible depending on the chosen scale,
- shape of visible items may be modified (made simpler or more precise),
- or simply the information is not available at the requested scale.

To maintain consistency and avoid redundancy, the ideal solution would be a database where geometry information is kept at the most precise scale, and all visualizations at less precise scales are automatically derived, mainly through cartographic generalization processes [21, 35]. Unfortunately, this derivation cannot be fully automated. Hence, map production systems have to explicitly store several representations of the geometry of objects (usually one per scale range). This can be done by either keeping a separate database per scale range, or by using a multi-scale database, i.e., a database where spatial objects may be associated to a variety of geometric representations that are scale dependent. More accurately, the latter should be referred to as a multi-resolution database (scale is a concept that refers to map drawing). Resolution is usually defined as the minimum geometric dimension that an object must have in order to be of interest, and consequently represented in the database. The resolution of information in the database is the resolution that either was used at data acquisition, or the one that results from a cartographic generalization process. If different resolutions are associated to the same objects, we can talk about multi-resolution objects.

Beyond cartography, multiple representations of geographic data are needed to serve multi-disciplinary user communities, as the same piece of land may support analysis, planning, and forecast activities by city administrations, environmentalists, sociologists, botanists, zoologists, etc. Altogether, the variety of representations that may be recorded in a database for a given object extends over different facets, such as:

- multiple geometry (within the same or different geometric abstract data types) may characterize the same object in different contexts,
- coexistence of many abstraction levels in object classification, which may result in the simultaneous, independent representation of a composed object (i.e., an object that is built by aggregating a number of component objects, whether the aggregation is based on geometric, temporal or semantic criteria) and its component objects,
- coexistence of many abstraction levels in object description, which may result in attributes having a hierarchical value domain (i.e., a domain composed of a hierarchically structured set of values, such that values at a node are more precise than the value in the parent node), and
- multiple representations in terms of thematic information, which corresponds to maintaining several viewpoints in traditional databases.

Commercial systems poorly support the need for multiple representation. Few GISs can explicitly represent objects with multiple geometry. Current DBMSs provide limited support for multiple thematic representation. However, the situation may soon evolve as the database and GIS research communities have been active in developing proposals for new object identification and description schemes. Database researchers proposed concepts such as roles, prototypical objects, deputy objects, or aspects. GIS researchers focused on issues such as inter-level connectivity in multiple level data sets, scale transition relationships, or stratified map space. Interoperable environments have also been addressed to allow interconnecting related representations from different information sources. This paper surveys the issues that have been addressed.

2 A Framework for Multi-Representation

We assume that the real world of interest that is to be represented in the database is composed of objects, their links in between and their static and dynamic properties (attributes and methods). As representations may vary according to different criteria, the representation space may be seen as a multi-dimensional space, where each dimension (or axis) relates to one of the criteria in use. Dimensions we are particularly interested in here[2] are:

- the spatial resolution dimension: coordinates on this axis represent the spatial resolution ranges for which representations hold;
- the observer's, or viewpoint dimension: coordinates on this axis represent the different viewpoints for which representations are elaborated;
- the classification dimension: coordinates on this axis represent object instances as members of a given object type.

A point in this 3-dimensional space is the representation of an object instance as a member of the population of a given object type, and according to a given viewpoint and to a given resolution range. Notice that two points may hold identical values, e.g., two viewpoints sharing the same representation for a given object instance at a given resolution.

The 3D metaphor can easily characterize alternatives in schema definition (how the data is presented to users) and database definition (how instances are grouped into databases). For example, current single-resolution spatial databases correspond to forming a database with representations that lie on a same plane orthogonal to the resolution axis. A standard map is built from representations that lie on a single straight line parallel to the classification axis; the position of the line is determined by the map scale and the target viewpoint. Systems that support objects with multiple geometry get rid of the resolution axis and work in 2D representation space. Solutions that decompose the representation space into fragments (sub-cubes, planes or lines)

[2] For sake of simplicity, we limit ourselves to three dimensions. However, more could be considered, e.g., a time dimension that would support representations at different points in time.

are likely to require interschema/interdatabase links to be able to associate/retrieve different representations of the same real world object.

Looking at the state of art and on practical applications, it is easy to see that researchers usually focus on one dimension only. Multi-resolution databases, views and multi-instantiation are separate research areas, each one pursuing its own dimension. For sake of simplicity, our survey hereinafter discusses the dimensions separately.

In the resolution dimension, the following choices may be found:

- each object has a single representation (i.e., one database instance) which includes multiple geometries, and all object instances are stored in a single multi-resolution database,
- each object has multiple, interconnected representations (one per resolution range) and
  - there is a single-schema database that stores all representations,
  - there is a multiple-schema database (one schema per resolution range)
  - there are several single-schema databases (one per resolution range), each one storing representations that are homogeneous in resolution,
  - there are several multi-resolution databases.

In the viewpoint dimension, similar choices may be identified:

- each object has a single representation (i.e., one database instance) which includes multiple roles, and
  - all object instances are stored in a single-schema database,
  - all object instances are stored in a multiple-schema database (one schema per viewpoint),
- each object has multiple, interconnected representations (one per role) and
  - there is a single-schema database that stores all representations,
  - there is a multiple-schema database (one schema per viewpoint)
  - there are several single-schema databases (one per viewpoint), each one storing representations that belong to the same viewpoint,
  - there are several multi-viewpoint databases.

Complementary aspects that will also be discussed are inheritance issues, related to the third dimension (object classification) and rules for object creation.

3 Multiple resolution

Data about the same geographical space may be collected at various resolution levels, to serve different applications within an organization. For instance, the French National Mapping Agency (IGN) maintains several databases about France, each one used to produce maps in a specific scale range. Multi-resolution data may also be needed for one single application, as is the case, for instance, in embedded navigation, where only parts of the navigation process need detailed information (e.g., the departure and arrival areas), while for the rest of the navigation only coarse level

information is needed (e.g., for traveling on a highway section). Finally, multi-resolution data may just be a consequence of integrating data from various digital sources that have been independently set up. This situation becomes more and more common: with the focus on data reuse, justified by high data acquisition costs, data integration has become one of the major challenges in GIS applications.

3.1. One object, one multi-resolution instance

To move from single-resolution to multi-resolution databases, one solution (assuming a discrete, vector approach) is to allow an object instance to bear multiple geometries. Each geometry is qualified with the relevant resolution range. The different geometries, other than points, are mainly acquired either through separate data collection processes, or via interactive, cartographic generalization processes, and have to be explicitly input into the database. This approach follows the representation principle: one object in the real world translates into one instance in the database. Proposals by Frank & Timpf [11, 32], Kidner, Jones & al. [15, 17], Bedard [3], and Vangenot [34] represent variations within this trend.

Multiple resolution, however, does not reduce to multiple geometries. The focus on objects changes from one resolution level to another: more details bring in more objects, less details result in objects being aggregated to form new objects of a different type. Relationships between objects may change, including topological relationships [14]. Thematic attributes of objects, and even thematic attribute values may change [28, 29]. A multi-resolution database has to keep track of all links that are needed to retrieve a consistent subset of database representations for each user interested in data at a specific resolution. Aggregation links, for instance, are necessary to support intelligent zooming [11].

A specific case in the category in this section is raised by federated databases. Here, users access the federated database via a single integrated schema, which describes virtual multi-resolution instances, but real instances are distributed over a set of underlying, mono-resolution databases that participate into the federation [7, 25].

If this integrating approach is also used for the viewpoint and classification dimensions, the result is one instance holding all possible representations. Because of the complexity of changes that representation of the real world undergoes when moving from one resolution to another one, keeping all facets in a single-instance framework may become cumbersome. For instance, displaying a map at a given scale requires examining all object instances to find out if they have a geometry defined that corresponds to the requested scale and that is located in the space to be covered by the map. This leads to building spatial indexes that depend on resolution. Similar impact makes other traditional functionality (e.g., query processing, access rights enforcement) more complex to implement.

3.2. One object, many single-resolution instances

One way to reduce complexity is to split the representation of a real-world object into multiple, interconnected representations, each one materialized as an object instance

in the database. The question on how to split may be addressed independently from the user perspective and from the system perspective. On the one hand, database designers have to decide how information will be presented to users (hopefully, the way users would like to see it). On the other hand, the way information is actually stored may be quite different, as the criterion here is system performance or site autonomy, not user-friendliness. What follows has to be understood as pertaining to the user perspective.

Splitting may be along one dimension only: resolution, viewpoint, or classification. Splitting, as we have stated, means having multiple object instances for the same real-world object. If the split is by resolution (the case we are discussing in this section), the different instances will bear different geometries, such that each geometry is appropriate within a given resolution range. The existence of multiple instances rises three questions:

- how the instances are classified: into one class in one database schema, into different classes in the same schema, or into different classes belonging to different schemas;
- how the instances are related: implicitly, through their identification mechanism, or explicitly through links (e.g., association or generalization links); and
- which properties are associated to each instance: all properties explicitly or only properties specific to the resolution of the instance, with other properties inherited from other instances.

If all instances are classified into a single class, say Building, users will have to resort to a more complex identification scheme (typically, the "normal" identifier plus a code corresponding to the resolution level) to denote the instance they are interested in: e.g., values for building-id may be <building#**.**resolution-code>, such as 372**.**r1, 372**.**r2, etc. If each instance is in a different class, identification will go through the class name plus the normal identifier. In other words, it is the class name that will include the resolution code (e.g., Building-r1, Building-r2, …). Current proposals for multiple instances all go for the second solution. More specifically, they recommend to group into one schema object types that pertain to the same resolution level. Simply stated, multiple resolution objects are handled through a set of single-resolution schemas. The schemas may eventually map to a single physical database, as in Timpf's Map Cube model [33]. They may actually be implemented as views over a global, multi-resolution schema. Or they may map to different databases, one per resolution range [18].

Regarding inter-instance links, implicit linking through identifiers is possible but not recommended. It leaves the entire burden to users, provides little support for consistency and is likely to lead to poor performance. Explicit definition of links is hence supported by all proposals for multiple instances. Depending on whether the object types belong to the same schema or not, links will be just a specific kind of association, or a new type of interschema link. Within the same schema, the semantics of such a link is that the linked instances "represent the same object at different resolution levels". This is very similar to the semantics of the traditional is-a link, where linked instances represent the same object at different semantic resolution levels, but it does not obey the inclusion semantics that characterizes the is-a link in

current database systems. Indeed, a change in resolution may result in a different set of objects representing the reality of interest. For example, assuming a database on roads, moving to a coarser resolution may cause small roads to disappear (they fall below the threshold) and roads that run in parallel (e.g., highway lanes) to be merged into a new road object. As a consequence, two types for the same objects at different resolution will generally have intersecting populations, rather than one included into the other. This needs a different link than the is-a link. It may even require several links between the two types, to express links that may be one-to-one, zero-to-one, one-to-many, or many-to-many depending on which instances are considered.

As for properties, associating to each instance the whole set of properties that are relevant for that instance guarantees completeness of the representation, flexibility and self-contained manipulability. However, this will also need a number of integrity constraints to ensure that properties that are resolution-independent hold the same value in all instances. As checking integrity constraints is time-consuming (hence, lowers performance), modern database systems provide an inheritance mechanism associated to the is-a link. Unfortunately, as we have just seen, is-a links are not always appropriate for multi-resolution classifications. More research is needed to extend the inheritance approach to object types with intersecting populations.

4 Multiple viewpoint

A viewpoint is what determines a given representation for some reality of interest, among all possible representations. A viewpoint usually expresses information requirements from a given set of users that show homogeneity in their requirements. A viewpoint definition holds a specification of both the data structure (object and relationship classes, attributes) and the rules for data usage (e.g., methods and integrity constraints). As change in the classification of objects is the topic of the next section, we will limit our discussion here to changes in the descriptive part, i.e., the attributes (which extends to methods if database design uses an object-oriented model).

The fact that different users may have different viewpoints is known from the very beginning in the database field. Support for this diversity is achieved by allowing definition of personalized views over an underlying global database schema. However, the extent of flexibility in the view definition mechanism has significantly changed with the evolution of database technology. Systems developed in the 70s offered very little flexibility. They supported sub-schemas over the database schema, where differences between the two mainly stemmed from allowing sub-setting (selection) and renaming operations in the definition of a subschema.

Relational systems focused on the definition of a derived, virtual table, called a view, from existing tables. Relational systems achieve maximum restructuring flexibility, as arbitrary algebraic expressions may be used to build a view (although the use of binary operators, e.g., join, may result in a view that does not support update operations). This power in flexibility directly results from the poor semantics that is embedded in flat relational tables. As the only structure that is supported is the tuple structure, users can easily build a new tuple structure by relating attributes from whichever table they want. However, view definition by restructuring operations means that support is limited to representations that are derivable from existing ones.

For representations that are not 100% derivable the entire burden is on the users. Users are responsible for adding the necessary artificial keys and foreign keys to link related tables, and for providing the procedures to enforce the desired consistency rules.

Object-oriented, or object-relational, database systems fail in supporting similar flexibility. Object identity and complex object structures both make view definition a problem that is not easy to solve. Using binary operations results in generation of new objects, which rises the problem of providing a new object identity and keeping the link between the new object and the objects it stems from. Combining unary operations (e.g., projection and selection) in the definition of a view raises the issue of how to insert the view as a new object type in the type hierarchy. This issue has no solution that obeys the rules of classical object-oriented data models. Complex object structures induce hierarchical arrangement of data that is not simple to restructure (and generates new objects). For these reasons most systems based on the object-oriented approach limit view definition to views that can be constructed using only selection and renaming operations (i.e., object preserving operations). We are back in the 70s, but with a more powerful paradigm. On the other hand, compared to relational systems, object-oriented systems provide additional support for multiple representations through generalization/specialization hierarchies that materialize links between instances that represent the same real world object by sharing system-generated object identifiers. However, this is known to be insufficient (in terms of expressive power, user-friendliness, and practicality) to provide full flexibility in multiple representation support.

View definition implements the two facets, presentation and implementation, that we introduced at the beginning of Section 3.2. Users are presented with object types and instances that are formatted according to their specific viewpoint. The system collapses all descriptions into a single multi-viewpoint object. Because users navigate only within their own viewpoint, there is no need to provide them with facilities to view data according to another viewpoint. Because of the collapsing into a single object type, the object type by definition materializes the link between alternative viewpoints on the same objects. As for the facilities introduced by generalization/specialization hierarchies and is-a links, they are discussed hereinafter.

A notable exception is the TROPES data model [20], where the focus is on a single instance solution visible to users. Each object type then bears multiple descriptions that are qualified by the name of the viewpoint they implement.

Views in GIS have been addressed in [5]. Rather than talking about schemas and viewpoints in a database terminology, some authors use more GIS-oriented concepts. For example, Stell and Worboys [31] see the database organized as a stratified map space, where each map gathers objects that share the same semantic and spatial granularity. Maps are grouped by map spaces, i.e., sets of maps showing the same schema at different granularities. The stratified map space is the set of all maps organized according to a hierarchy based on different granularity levels. Transformation functions allow navigating in a stratified map space. Finally, a sheaf is a set of stratified map spaces where each space covers a different spatial or semantic area.

5 Multiple Classification

Because modeling is expressing general rules about the world of interest, classification is the most fundamental abstraction in the data modeling process. It allows to get rid of the details, and talk in terms of object classes, their relationships and the properties we want to attach to them. It is also a very subjective abstraction. Classification of the same set of objects is very likely to change when a different viewpoint on data is taken. Classification may also change in time, whenever objects acquire new properties or loose properties in their evolution. Even from a single viewpoint it may be desirable to classify a given object into multiple classes, as classification is not necessarily partitioning. Semantic and object-oriented data models support this by providing the is-a link to define generalization/specialization hierarchies. However, is-a links only support classification refinement and taxonomic reasoning. They are not appropriate for arbitrary classifications, where two sets of objects are related but neither one is included in the other (intersection semantics). To support intersecting classes, some approaches allow multiple inheritance: the intersection class may then be modeled as a subtype of the two initial classes. Beyond the fact that this modeling trick results in the creation of artificial classes (where artificial means not of interest for the application), its scope is restricted to classes that belong to the same generalization/specialization hierarchy (because of consistency rules on object identity).

Another limitation of current generalization/specialization hierarchies is their static aspect. Objects are not allowed to move from one class to another. Moreover, because of dynamic binding implementations, objects are not allowed to belong to two leaf classes. This set of constraints is not acceptable when the focus is on data modeling. While an ultimate, consensus solution is not yet available to escape from this too rigid framework, significant research efforts have already produced a number of proposals which, in different ways and using different terminology, aim at supporting the role concept [2]. A role is an alternative classification of an object, such that an object may become a member of several role classes, remain a member for some time and then release its membership. Objects can move from one role class to another [4, 24]. Role classes may be static, which means their type is defined in the schema, or they can be created and deleted dynamically during application execution [24]. In most approaches role classes are seen as a transient repository for objects from a given object type, called the base object type. For example, objects of the base type Person may temporarily belong to role classes such as Student, Worker, Retired. This is similar to generalization/ specialization hierarchies, except that objects can move around and belong to many leaf classes at the same time. This transient aspect leads naturally to propose keeping the lifecycle of objects in roles [27]. In [13, 27] an object can be instantiated several times as different instances of the same role. This allows representing, for example, a person who registers as a student in two different institutions.

An additional requirement for role classes is to accept instances from different object types that do not belong to the same generalization hierarchy. For example, a Car-owner role may be populated with instances from the Person type and instances from the Company type (both companies and persons may own cars). The category concept [10] was proposed to cope with this situation in the context of the Entity-

Relationship model. In the context of object models, this requirement is easier to achieve in proposals that do not require the existence of a base object type [12, 16, 19]. In the latter models, the role type concept replaces the object type concept. Objects can enter the database through creation in any of the roles that accept creation operations, and then move around according to inter-role links (which can be bi-directional or not depending on application constraints).

Roles provide a solution to support many representations of a single object, such that each representation is materialized into one database instance. This scheme is also referred to as multi-instantiation, although this term is sometimes used to specifically denote models where every type is considered as a role type [12]. It allows to easily support properties and relationships that are role-specific. Thus, the role concept conveys both a change in classification and a change in viewpoint. It has been investigated by many authors, resulting in many variations in the rules that define the allowed data structures (namely, relationships between roles and the corresponding object type) and the allowed lifecycles (how objects can move around in roles) [see, e.g., 1, 6, 16, 19, 22, 23].

6 Inheritance

Moving from objects to roles, i.e., from mono- to multi-instantiation, rises the issue of which inheritance mechanism, if any, should be associated to the inter-role links. It is indeed not possible to just reuse the object-oriented combination of automatic inheritance, late binding, refinement, redefinition and overloading. These concepts and mechanisms are strongly related to the inclusion semantics and mono-instantiation rules of the generalization/specialization hierarchies that are embedded in object-oriented data models.

Two basic alternatives have been proposed to replace or complement the automatic inheritance and late binding approach: either static, explicitly defined inheritance, or inheritance on demand in query formulation. An example of the former is known as delegation: the definition of an object/role type includes attributes whose value is not stored within the instance of that object/role type, but derived from the corresponding homonym attribute in the corresponding instance belonging to another object/role type. Reference in a query to one of these derived attributes automatically results in accessing the other instance to get the requested value. The net effect is similar to inheritance, but this inheritance is limited to the subset of attributes that the designer freely chooses. Actually, most proposals go for some mix of automatic inheritance and delegation. For example, object types and role types are organized into a mixed hierarchy, where they may be linked by is-a links or by role links. Automatic inheritance with late binding is the rule for types linked by is-a links, whereas role links obey the delegation principle [13].

The second solution, specifying the desired inheritance as part of query formulation, is a sort of adjustable dynamic binding, driven by users' specifications rather than by static schema definitions. When accessing an object, the user has to specify the multi-instantiation context to be considered for the query. That is to say, which other object/role types can be accessed to find the desired property (attribute or method) if not found in the type directly denoted in the query. We refer to this as the scope of the query. Moreover, the user can specify in which population the object

instance to start with is to be taken. We refer to this as the selected viewpoint for the execution of the query. The combination of these two specifications, viewpoint and scope, gives the user complete control on which object properties have to be accessed [12].

This is particularly relevant in spatio-temporal databases. Spatio-temporal databases use system-defined attributes to hold spatial and temporal information. These attributes have standard names, such as "geometry", "lifecycle", or "timestamp". If both a superclass and its subclass have specific spatial or temporal information, an attribute with the same name will exist in both classes. For instance, one may want to keep the lifecycle of somebody both as a Person and as an Employee, where obviously the two lifecycles hold different values for the same person. A traditional dynamic binding mechanism would automatically return the value in the subclass. Actually, dynamic binding proceeds from the idea of genericity versus specificity, and that genericity is seen as a way to abstract from specificity in denoting a method, while keeping specificity as the goal in executing the denoted method. But in the lifecycle example there is no such idea. The two values have different semantics, and there is no reason to substitute one by the other. An application interested in lifecycles of Person objects would not be willing to get instead lifecycles of objects in Employee, Student, etc. The same applies to spatial information. Assume the superclass has spatiality at 1/10'000 resolution and the subclass has spatiality at 1/250'000 resolution. An application drawing a map at 1/10'000 would definitely not care of spatiality existing at 1/250'000. Once more, a solution is needed that provides more flexibility and user control on accessing rules. One proposal based on the viewpoint and scope idea may be found in [8].

7 Object Creation

When an object deserves multiple representations in distinct instances, the question rises whether there are rules governing creation of instances and their migration from the population of a type to the population of some other type. For example, in proposals that assume the co-existence of a base object type (holding properties that are inherent to the object) and multiple role types (holding properties specific to the role), objects must be created at first in the base object type. Once created, they can generate additional instances in the role types, but cannot migrate to role types (where by migration we mean disappearing from the source population and appearing in the target population). Consistently, objects cannot be deleted in the base type as long as they are still represented in a role type.

The workflow that governs the membership behavior of an object can be defined and constrained in different ways. One approach is the definition of membership predicates for each object/role type. This allows automatic acquisition of new roles: when an object instance is modified, its new value is confronted with the membership predicates and whenever the predicate is satisfied the instance is classified as member of that population [23, 24]. Predicates may also be checked on demand, rather than automatically on modification. Inference rules may be associated to each object/role type, specifying which other types may or may not be populated by an instance migrating or being generated from this type [19, 24, 27]. Kambayashi & Peng [16] propose to associate transformation functions to migration/generation paths, to

compute values and structure for the target instance from one or more source instances. Transformations between representations have also been addressed in [7, 18].

8 Conclusion

Support for multiple representations has been an active research domain, in particular over the last decade. However, it is our feeling that only recently it has come out as a the next major step forward in data modeling technology. Clearly the focus on reaching operational solutions for object-oriented technology in database management has driven most of the attention from the research and development world. But the perspectives that object-based approaches made visible to users have made users more demanding in terms of satisfaction of their requirements. This gives a substantial new impetus to more flexible representation schemes that can support full customization despite information sharing.

This paper proposed a generic framework to address the multiple representation problem, making clear that different phenomena contribute to a diversification of representations. We have investigated the related issues and solutions, showing that, despite similarities, the approach may differ from one dimension to the other. It may also differ in between the users' view and the implementation view. We focus on multiple representation of objects, but the concern extends to relationships, including topological relationships [9, 14].

The issues we addressed are of great relevance in the GIS world, and directly apply to multi-resolution geographical databases. The MurMur European project, in which we are involved, aims at specifying and developing a spatio-temporal data model that provides concepts and facilities to fully support multi-resolution and multi-representation. The MADS data model [26] serves as initial framework. The project started January 1st, 2000 and will last for 30 months. More about the project may be found in [30].

References

[1] A. Albano, G. Ghelli, R. Orsini. Fibonacci: A Programming Language for Object Databases, Very Large Data Bases Journal, 4(3), p. 403-444, 1995.

[2] C. W. Bachman. The role concept in data models, Proceedings of the Third International Conference on Very Large Data Bases, VLDB'77, Tokyo, Japan, p. 464-476, October 6-8, 1977.

[3] Y. Bédard. Visual modeling of spatial databases: Towards Spatial extensions and UML, Geomatica, 53(2), p.169-186, 1999.

[4] W. W. Chu, G. Zhang. Associations and roles in object-oriented modeling, Proceedings of the 16th International Conference on Conceptual Modeling, ER'97, Los Angeles, California, USA, p. 257-270, November 3-5, 1997.

[5] C. Claramunt. Un modèle de vue spatiale pour une représentation flexible de données géographiques. Ph.D. Thesis, Université de Bourgogne, Dijon, France, 1998.

[6] S. Coulondre, T. Libourel. Des critères dans les classes : Homogénéisation de la gestion des rôles, Proceedings of 15èmes Journées Bases de données Avancées, BDA'99, Bordeaux, France, p. 263-281, October 25-27, 1999.

[7] T. Devogele. Processus d'intégration et d'appariement de bases de données géographiques: Application à une base de données routière multi-échelle. PhD Thesis, Université de Versailles, Institut Géographique National, 1998.

[8] P. Donini, S. Monties. Qualified Inheritance in Spatio-Temporal Databases, IAPRS, Vol. XXXIII, Proceedings of the XIX Congress of the International Society for Photogrammetry and Remote Sensing, Amsterdam, July 16-23, 2000.

[9] M.J. Egenhofer, E. Clementini, P. Di Felice. Evaluating inconsistencies among multiple representations. Proceedings of the Sixth International Symposium on Spatial Data Handling, SDH'94, p. 901-920, Edinburgh, Scotland, 1994.

[10] R. Elmasri, J. Weeldreyer, A. Hevner. The Category Concept: An Extension to the Entity-Relationship Model, International Journal on Data and Knowledge Engineering, 1(1), 1985.

[11] A. Franck, S. Timpf. Multiple representations for cartographic objects in a multi-scale tree: An intelligent graphical zoom, Computers & Graphics, 18(6), 1994.

[12] M. Gentile. An object-oriented approach to manage the multiple representations of real entities, EPFL PhD Thesis no 1490, 1996.

[13] G. Gottlob, M. Schrefl, B. Röck. Extending object-oriented systems with roles, ACM Transactions on Information Systems, 14 (3), p.268-296, 1996.

[14] T. Jen. Formalisation des relations spatiales topologiques et application à l'exploitation des bases de données géographiques, PhD Thesis, Université Paris XI Orsay, 1999.

[15] C.B. Jones, D.B. Kidner, L.Q. Luo, G.L. Bundy, J.M. Ware. Database design for a multi-scale spatial information system, International Journal of Geographical Information Systems, 10(8): 901-920, 1996.

[16] Y. Kambayashi, Z. Peng. Object deputy model and its applications, Proceedings of the Fourth International Conference on Database Systems for Advanced Applications, DASFAA'95, p. 1-15, Singapore, April 11-13, 1995.

[17] D. Kidner, C. Jones. A Deductive Object-Oriented GIS for Handling Multiple Representations, Proceedings of the Sixth International Symposium on Spatial Data Handling, SDH'94, p. 882-900, Edinburgh, Scotland, 1994.

[18] T. Kilpelaïnen. Maintenance of topographic data by multiple representations, Proceedings for the Annual Conference and Exposition of GIS/LIS '98, Forth Worth, Texas, p. 342-351, November 10-12, 1998.

[19] Q. Li , F. H. Lochovsky. ADOME: An advanced object modeling environment, IEEE Transactions on Knowledge and Data Engineering, 10(2), p. 255-275, 1998.

[20] O. Marino Drews. Raisonnement classificatoire dans une représentation à objets multi-points de vue. PhD Thesis, Université Joseph Fourier Grenoble I, 1993.

[21] J.C. Müller, J.P. Lagrange, R. Weibel, F. Salgé. Generalization: State of the art and issues, in J.C. Müller, J.P. Lagrange and R. Weibel, editors, GIS and Generalization: Methodology and Practice, p. 3-17. Taylor & Francis, 1995.

[22] H. Naja. La représentation multiple pour l'ingénierie, L'objet, 4(2), p.173-191, 1998.

[23] E. Odberg. Category classes: flexible classification and evolution in object-oriented databases, Proceedings of Advanced Information Systems Engineering, CAiSE'94, Utrecht, The Netherlands, p. 406-420, June 6-10, 1994.

[24] M.P. Papazoglou, B.J. Kramer, A. Bouguettaya. On the representation of objects with polymorphic shape and behavior, Proceedings of the 13th International Conference on Entity-Relationship Approach, ER'94, Manchester, UK, p. 223-240, December 13-16, 1994.

[25] C. Parent, S. Spaccapietra, T. Devogele. Conflicts in Spatial Database Integration, Proceedings of the 9th International Conference on Parallel and Distributed Computing Systems, PDCS '96, Dijon, France, p. 772-778, September 25-27, 1996.

[26] C. Parent, S. Spaccapietra, E. Zimanyi. Spatio-Temporal Conceptual Models: Data Structures + Space + Time, Proceedings ACM-GIS'99, Kansas City, November 6-7, 1999.

[27] B. Pernici. Objects with Roles, Proceedings of ACM Conference on Office Information Systems, Cambridge, Massachusetts, p. 205-215, 1990.

[28] P. Rigaux, M. Scholl. Multi-scale partitions: Applications to spatial and statistical databases, Proceedings of the 4th International Symposium on Advances in Spatial Databases, SSD'95, Portland, Maine, Springer-Verlag LNCS 951, p. 170-183, August 6-9, 1995.

[29] M. Scholl, A. Voisard, J.-P. Peloux, L. Raynal, P. Rigaux. Systèmes de Gestion de Bases de Données Géographiques, Spécificités, International Thomson Publishing, 1996.

[30] S. Spaccapietra, C. Parent, E. Zimanyi, C. Vangenot. MurMur: A Research Agenda on Multiple Representations, 1999 International Symposium on Database Applications in Non-Traditional Environments (DANTE'99), Kyoto, Japan, November 28-30, 1999.

[31] J. Stell, M. Worboys. Stratified Map Spaces: A formal basis for multi-resolution spatial databases, Proceedings of the 8th International Symposium on Spatial Data Handling, SDH'98, Vancouver, Canada, p. 180-189, July 11-15, 1998.

[32] S. Timpf, A. Franck. A multi-scale DAG for cartographic objects, Proceedings of Auto Carto 12, Charlotte, North Caroline, USA, p. 157-163, Feb.27-March 1, 1995.

[33] S. Timpf. Hierarchical structures in map series, Ph.D. thesis, Technical University Vienna, 1998.

[34] C. Vangenot. Multiresolution Representation. Concepts for the description of multiple representation databases, (in French), International Journal of GIS and Spatial Analysis, Hermes, Paris, 8(1-2), p.121-148, 1998.

[35] R. Weibel, G. Dutton. Generalizing spatial data and dealing with multiple representations, In P. Longley, M.F. Goodchild, D.J. Maguire, D.W. Rhind, editors, Geographical Information Systems: Principles, Techniques, Management and Applications, vol. 1, 2nd edition, Geoinformation International, 1999.