

CERIAS Tech Report 2009-04
Implementation Challenges in Spatio-temporal Multigranularity
by Elena Camossi, Michela Bertollo
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

Implementation Challenges in Spatio-temporal Multigranularity

Elena Camossi *

*School of Computer Science and Informatics - University College Dublin, Belfield,
Dublin 4, Ireland. Phone: +353 (0)1 7162-913. Fax: +353 (0)1 2697-262*

Michela Bertolotto

*School of Computer Science and Informatics - University College Dublin, Belfield,
Dublin 4, Ireland. Phone: +353 (0)1 7162-913. Fax: +353 (0)1 2697-262*

Elisa Bertino

*CERIAS - Purdue University, 250 N. University Street West Lafayette, Indiana,
USA 47907-2066. Phone: +1 765 496-2399 Fax: +1 765 494-0739*

Abstract

Multiple granularities are essential to extract significant knowledge from spatio-temporal datasets at different levels of detail. They enable to zoom-in and zoom-out spatio-temporal datasets, thus enhancing the data modelling flexibility and improving the analysis of information. In this paper we discuss effective solutions to implementation issues arising when a data model and a query language are enriched with spatio-temporal multigranularity. We propose appropriate representations for space and time dimensions, granularities, granules, and multi-granular values. In particular the design of granularities and their relationships is illustrated with respect to

the application of multigranular conversions for data access. Finally, we describe how multigranular spatio-temporal conversions affect data usability and how such important property may be guaranteed. In our discussion, we refer to an existing multigranular spatio-temporal model, whose design was previously proposed as extension of the ODMG data model.

Key words: Spatio-temporal databases, spatial and temporal granularities, multiresolution, multirepresentation

1 Introduction

The capability of representing spatio-temporal datasets with respect to both their spatial layout and their temporal evolution is fundamental to analyse and monitor the changes in the spatial configuration of a geographical area over a period of time. Moreover, to trace modifications according to different temporal frequencies, the history of the areas under observation has to be maintained and retrieved at multiple temporal granularities (e.g., years, months, decades). Similarly, multigranular modelling offers interesting capabilities in the spatial domain: from support to automated cartography, to efficient browsing over large datasets, to structured solutions in wayfinding, planning and design, to rendering of virtual reality environments. The approaches able to present the data at different granularities represent an effective solution to facilitate the analysis when fewer or additional details are required for specific subsets of

* Corresponding author

Email addresses: elena.camossi@ucd.ie (Elena Camossi),
michela.bertolotto@ucd.ie (Michela Bertolotto), bertino@cs.purdue.edu
(Elisa Bertino).

the data. For example, zoom-out operations may improve the efficiency of spatio-temporal data mining algorithms, which are time consuming [3]. On the other hand, zoom-in operations may help in refining the mining of specific data subsets.

Granularities intuitively represent the units of measure of a dataset, and may be defined on all data dimensions (i.e., Space and Time for spatio-temporal data). For each dimension, a connected set of granularities may be defined, and the different sets are independent. Multigranularity, multiresolution and multiple representation have been investigated first for temporal [11,28] and spatial data [6,27,35] separately, and more recently for spatio-temporal data [14,19,23,34].

The choice of the correct granularity allows the system to store the minimal amount of data, according to the most appropriate level of detail. In many applications different granularities may exist, neither of which is inherently better than the others. Therefore, a database system for such applications should support a wide range of granularities and allow the user to define his/her own application-specific granularities.

Extending a data model with spatio-temporal multigranularity entails to address several design and implementation issues. The first category of problems one has to face with *is* *modelling issues*, that are related to the formal design of the base entities that make up the multigranular data model. First of them, the modelling of the spatial and the temporal domains on which the granularity mappings are given, that, for instance, may be considered discrete or continuous, mutually dependent or independent, and may be application driven. It is equally important to find a proper design for granularities and

granules, whose implementation affects the efficiency of the representation of multigranular values, that in turn impacts on the performance of queries execution.

Another category of problems, which arise when implementing the query language, *is comparison issues*. Indeed, in multigranular queries it is crucial to guarantee that a common level of detail for two multigranular data of the same type (i.e., number, alphanumeric, geometric information) always exists in order these data be compared in queries. To obtain that, granularities must be mutually related and multigranular data have to be converted to different levels of detail. Furthermore, the functions we use to shift granularity levels, namely *multigranular conversions*, must preserve data semantics and usability and do not introduce errors on the converted data.

Finally, *optimization issues* must be considered in order to improve the performance of all the data storage and retrieval process. Strategies for a fast access to multigranular spatio-temporal data include the definition of auxiliary data structure, both as extension of existing spatio-temporal indexes and through the definition of novel techniques for the optimization of queries involving multigranular aggregates.

In this paper we discuss these issues, proposing different implementation solutions. In particular, the advisability of two separate representations for the temporal and spatial domains is discussed. Furthermore, we illustrate the design requirements for granularities and granules, proposing a data type design to implement them and suggesting how user defined granularities may be specified. Moreover, optimizations for the storage and the retrieval of multigranular values are described. Finally, the inconsistencies arising from the application

of multigranular conversions are discussed, proposing effective solutions to prevent them.

In this discussion, we refer to the multigranular model and query language we formally defined in [19,21]. This model extends ODMG [22], the reference model for object-oriented databases, with multigranular spatio-temporal capabilities. The design in [19,21] is formal: it relies on de facto standards, like the ODMG and OQL, and on agreed definitions for granularities [11]. Furthermore, it provide formal proposal for all the main components of a multigranular data model like granularities and their relationships and a validated spatio-temporal type system. Moreover, differently from other multigranular models proposed in the literature, granularity conversions and data access have been properly defined. Therefore, it is an appropriate candidate to refer in our discussion, which aim to investigates very closely how to provide an established implementation to spatio-temporal multigranularity.

To demonstrate the feasibility of the proposed solutions, we developed an object-relational spatio-temporal prototype which supports spatio-temporal multigranularity, built on top of ORACLE[®] 11g (www.oracle.com), which is based on the implementation considerations we provide,. In its design, we take advantage of the object-relational features, like extensibility of the design and data type encapsulation, and of the spatial type system already provided by the DBMS. However, this does not represent a limit to the proposed implementation because the same capabilities are also provided by other mainstream spatial database products, like Microsoft[®] SQL Server[®] ¹, PostgreSQL with

¹ <http://www.microsoft.com/sqlserver/2008/en/us/default.aspx> (accessed February 2009).

its spatial extension PostGIS², MySQL[®]³. Moreover, we describe some implementation details of two existing multigranular object-oriented prototypes implementations which solve other issues we consider in our dissertation. In particular, we describe a DLL extension for the automatic implementation of multiple temporal granularities.

This paper extends the work presented in [20], where a preliminary discussion on multigranularity issues was presented. With respect to [20], we discuss feasible solutions to multigranular issues. In particular, we describe how multigranular issues have been addressed in both object-oriented and object-relational multigranular prototypes we developed.

The paper is organized as follows. First, In Section 2 we present the scientific literature related to this work. Then, we discuss modelling, comparison and optimization issues, respectively in Sections 3, 4 and Section 5. In Section 6 we describe the design of a multigranular spatio-temporal object-relational prototype that adopts the implementation solutions we propose. Moreover, we show also a working solution to the automatic implementation of user defined temporal granularities. Finally, in Section 7 we give a final discussion, and outline future research directions.

² <http://www.postgresql.org> and <http://postgis.refrains.net/> (accessed February 2009).

³ <http://www.mysql.com/> (accessed February 2009).

2 Background and Related Work

Spatio-temporal multigranularity has been mainly investigated separately in the temporal and spatial domains. The pioneering research work on temporal granularities is by Anderson [2], but many other proposals aimed at formalising temporal granularities (e.g., [28]). A consensus among the different disciplines interested in temporal granularity representation has been achieved with the formalization proposed by Bettini et al. [11], who give a comprehensive discussion on temporal granularities for databases, data mining, and temporal reasoning.

Temporal granularity issues related to temporal databases have been investigated both for the relational and the object-oriented data models. In its first release [44] TSQL2, the temporal extension of the SQL-92 standard, supported multiple granularities, but many important issues, such as scaling from one granularity to another, were not considered. Elaborating on the theoretical framework of Bettini et. al [11], Dyreson et al. [25] solve some of the problems of TSQL2 related to the treatment of multiple granularities. In particular, they introduce the notion of “scaling mass function” to address the indeterminacy of temporal conversions, and discuss how to deal with multiple calendars.

The introduction of multiple temporal granularities in an object-oriented data model poses additional issues with respect to the relational context, due to the semantic richness of such a model. In contrast to the relational context, the introduction of temporal granularities in object-oriented data models is, in most cases, informal. By contrast, Bertino et al. [9] investigate the impact of temporal granularities in an object-oriented model compliant with the ODMG

standard.

In the spatial domain, plenty of research has been undertaken on multiple representations [6] (i.e., sequences of representations of a given object based on different decompositions, each corresponding to a given level of detail) and multiple resolutions (i.e., the data model reporting the different representations includes also information on how the representations are linked together). In the GIS context, much research addresses the development of data models for the multiresolution representation of geographic maps [32]. Stell and Worboys [45] formally define a “stratified map space”, which denotes a set of maps representing the same spatial extent at different granularities related to form a granularity lattice by conversion operators.

Research on multiple resolutions addresses in particular model-oriented generalization [37], which applies techniques used in cartography for representing spatial data at different levels of abstraction, by taking into account also the semantics of data and some notion of consistency to preserve data usability. Several model-oriented generalization operators [48] have been defined in the literature (e.g., aggregation, line simplification).

Recently, also the area of qualitative spatial reasoning has shown a growing interest in spatial representation at multiple levels of detail [16]. Most of the work in this area has focused on the imprecision, the imperfection, and on the vagueness of spatial representations [24]. All these concepts are closely related to the notion of spatial granularity. Indeed, granularities are intended as a fundamental issue for the definition of a spatial ontology that formalize all the above mentioned concepts.

Few proposals in the literature address the multigranular representation of

spatio-temporal data. Claramunt and Juang [23] propose the application of nested hierarchies for modelling space and time to extract quantitative information about spatio-temporal relationships in a data set. Griffiths et al. [29] define the Tripod spatio-historical model, which integrates the definition of granular histories. No operators are provided to convert multigranular data, but the histories are always internally represented at the *chronon* [31] granularity. Katri et al. [34] define an annotation-model for the specification of spatio-temporal data at multiple granularities. Their granularity model relies on the concepts of temporal indeterminacy [26] and spatial imprecision [24]. The resulting model and the granularity systems are effective only for data specification, because the conversion from a granularity to another is completely left to the user.

3 Modelling Issues

With modelling issues we indicate the difficulties of multigranular data modelling strictly related to data representation. *Representation issues* are modelling problems, which arise when representing information at different levels of detail. They include the effective implementation of the spatio-temporal domain, and the design of efficient data-types and structures for granules, granularities, and multigranular values.

3.1 Temporal and Spatial Domains

It is often debated whether the temporal and the spatial domains in a spatio-temporal model have to be orthogonal or should instead space depend on

time, reflecting the historical change of geo-referenced data. The choice may depend on the specific application, but the first solution enables to represent also how the spatial domain may depend on time. Moreover, it is more flexible to represent space and time separately, to reflect the intrinsic differences of the two domains. Indeed, in most applications, the time domain is linear, discrete and one-dimensional. The spatial domain, may be either continuous, as in the raster representation, or discrete, as it is provided by the vector support. Space is usually two- (e.g., in cartography), two and half (e.g., in digital elevation models), or three-dimensional (e.g., in urban models).

A crucial difference between the spatial and the temporal domains is that all the operations connected with the temporal domain rely strictly on its monotonic order. By contrast, operations involving spatial objects mainly depend on the topological relationships holding among them.

For the implementation of the spatial representation, whenever the multigranular model is built on top of a SDBMS, one may rely on the internal support for spatial data such systems already provide. By contrast, the representation of the temporal time line, whenever discrete, may rely directly on the CPU time, which is accessible by programming languages. In this case, the *chronon* granularity [31], that is the finest temporal granularity supported by the system, corresponds to the smallest unit of time the programming language can handle.

If space and time have an orthogonal representations, also the spatial and temporal granularities, which reflect the intrinsic characteristics of the domain on which they are defined, must belong to orthogonal sets.

3.2 Granularities and Granules

Different proposals exist formalizing the notions of granularities, in particular for the temporal domain (see Section 2). Relying on the approach given in [11], in [19] temporal and spatial granularities are defined as mappings from an index set \mathcal{IS} to the power set of the temporal and the spatial domains, respectively. For instance, *days*, *weeks*, *years* are temporal granularities; *meters*, *kilometers*, *feet*, *yards*, *provinces* and *countries* are spatial granularities. The temporal domain is totally ordered.

Each subset of the temporal and spatial domains corresponding to a single granularity mapping is referred to as a (temporal or spatial) *granule*, i.e., given a granularity G and an index $i \in \mathcal{IS}$, $G(i)$ is a granule of G that identifies a subset of the corresponding domain. Granules give the temporal bounds and the areas where spatio-temporal values are valid. For instance, we may say that a value reporting the measure of the daily temperature in Rome is defined for the first and the second day of January 2000. In this example, we may apply the labels “01/01/2000”, “02/01/2000”, and “*Rome*” to denote two temporal granules at granularity *days* and one spatial granule at granularity *municipalities*, respectively. The interior of granules of the same granularity cannot overlap⁴. Moreover, non-empty temporal granules must preserve the order of the temporal domain.

According to the formalization adopted, various granularities and granules representations may be adopted in a multigranular model. Dyreson et al. [25]

⁴ Temporal granules, according to the definition in [11], do not overlap, while spatial granules may touch along the boundaries.

discuss the implementation of temporal granularities, but the approach they propose may be used also for spatial granularities.

A granularity may be conveniently specified with respect to its relationships with other granularities, avoiding its exhaustive mapping onto the domain. Indeed, granularities differ according to how they partition their domain of reference. In [19] spatial and temporal granularities are related by the *finer-than* relationship [11] and its inverse *coarser-than* (see Fig. 1. When finer-than holds between two granularities G and H , we may say that given a granule g of the finer granularity G , a granule h of the coarser granularity H always exists that properly includes g . In this case we also say that H is coarser-than G . Both finer-than and coarser-than are transitive relationships. According to this relationship, for example, granularity *days* is finer-than *months*, and granularity *months* is finer-than *years*. Likewise, *municipalities* is finer-than *countries*.

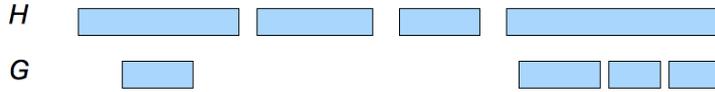


Fig. 1: *The finer-than relationship.*

Assuming that granularities are related by the finer-than relationship, spatial and temporal granularities form two Directed Acyclic Graphs (DAG). In these graphs, the nodes represent the granularities, and the edges represent the relationships among the granularities. We denote each of these graphs as *granularity graph*. Examples of granularities graphs for spatial and temporal granularities are depicted in Fig. 2. To simplify the illustration, the edges representing instances of finer-than that may be derived by transitivity are not drawn in the figure. Note that the finest granularities in the granularity graphs

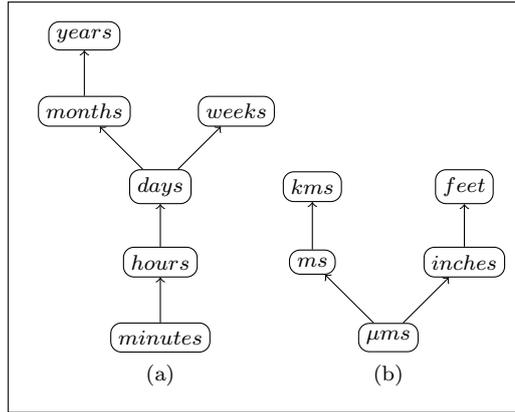


Fig. 2: *Examples of granularity graphs. (a) Temporal granularities. (b) Spatial granularities.*

give the measure of the precision applied for multigranular values, i.e., they represent the *tolerance* on the stored data. Often such granularities correspond to the *chronon* and *quantum* granularities.

Given a granularity system as that depicted in Fig. 2, an explicit representation is needed only for the *chronon* and *quantum* granularities [31], that are *minutes* and μms , which are implemented as direct mappings with the temporal and the spatial domains, respectively. Other granularities, that may be regularly subdivided in terms of minutes and μms , may conveniently use these mappings. Straightforward definitions may be obtained whenever other relationships, such as *partition* and *groups-periodically-into* [11], hold among the granularities. For instance, *months* groups-periodically-into *years*, because one year always corresponds to twelve months. These granularity properties have been exploited to extend the DDL of the multigranular temporal model described in [9] with the syntax for user-defined granularities, which are automatically generated by the model engine. This prototype is briefly described in Section 6. Such a specification, which is particularly suitable for temporal granularities, may be applied also to spatial granularities with even subdivi-

sions (e.g., *ms* and *ks*). Symbolic specifications for temporal granularities are proposed in [12], relying on *collection* and *slice* formalisms.

Granularities that may not be regularly represented may be effectively modelled adopting weaker granularity relationships. If we assume for example a model relying on finer-than and coarser-than, such as that described in [19], we have two possibilities to map a granularity G : we may either specify all granularities coarser-than (finer-than) G , or represent only those for which G is the coarser among the finer granularities. In this second case the set of relationships represented is similar to those represented in the graphs of Fig. 2, because the instances of finer-than we may derive by transitivity are discarded. Moreover, we may represent both finer-than and coarser-than, giving the specification for a bi-directional granularity graph.

A design that implements this specification of granules and granularities is given in Fig. 3, where the corresponding UML abstract data types are depicted.

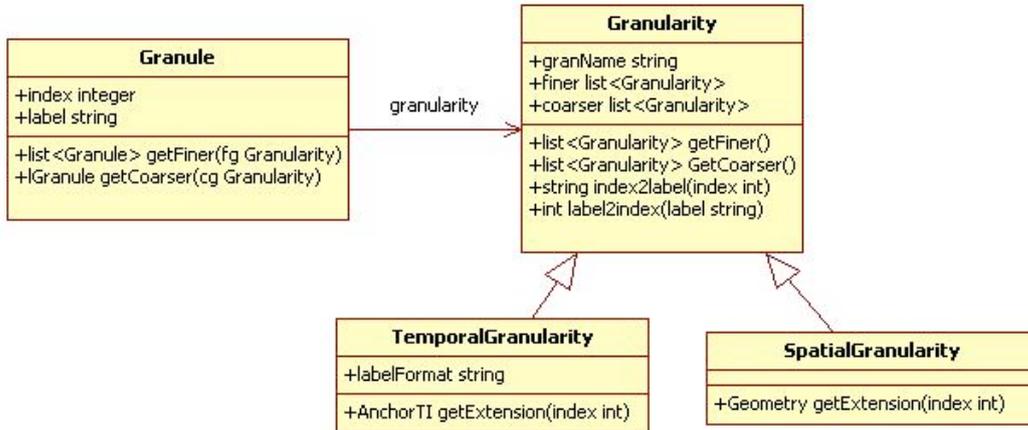


Fig. 3: *Granularity and Granule data types*

For each granularity, two operations, namely `getFiner()` and `getCoarser()`,

are given for retrieving its finer and its coarser granularities. Moreover, we specify also the conversions among different granule representations (i.e., label, index, and physical representation). In particular, the physical representation of each granule in the corresponding domain as anchored temporal interval [31] or as a set of geo-referenced features is given by the operation `getExtension(index int)`. For example, given granule g with index i , representing county Dublin in Ireland, `getExtension(i)` would return the geometry reported in Fig. 4.

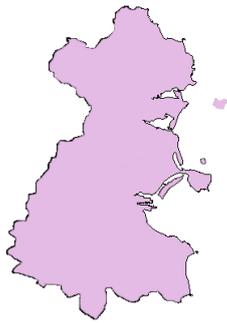


Fig. 4: *County Dublin*

For temporal granularities, the label format of granules (e.g., “mm/dd/yyyy” for days) is also stored.

Spatial and temporal granules have to conform to data type `Granule`, specifying the granularity of reference, a numeric index and an alphanumeric label. Given granule g at granularity G , the granules of a finer granularity K that are included in g are retrieved by applying `getFiner()`. For example, given the granule at granularity *years* 2010, through such operation we retrieve the finer granules at granularity *months* that represent the months of year 2010. Similarly, `getCoarser()` returns the unique coarser granule of a specified granularity H that includes g (see Fig. 1). For example, given granule g representing the month of May 2010, `g.getCoarser()` would retrieve the

granule representing year 2010.

3.3 Storage of Multigranular Spatio-temporal Values

A multigranular spatio-temporal database schema may include conventional, multigranular spatial, temporal, and spatio-temporal attributes. In [19] multigranular data are defined as instances of the types **Spatial** and **Temporal**, which are specified by a granularity (spatial and temporal, respectively) and an inner type. The inner type for **Spatial** may be a conventional type, i.e., a type without spatio-temporal characteristics, or a geometric type, i.e., a vector type, e.g., Point, Line, Polygon. The inner type for **Temporal** may be a conventional type, or a **Spatial** type. In the latter case, the resulting type is multigranular spatio-temporal. In Figures 5 and 6 we report the Unified Modeling Language (UML, <http://www.uml.org/>) schemas that define multigranular spatio-temporal types.

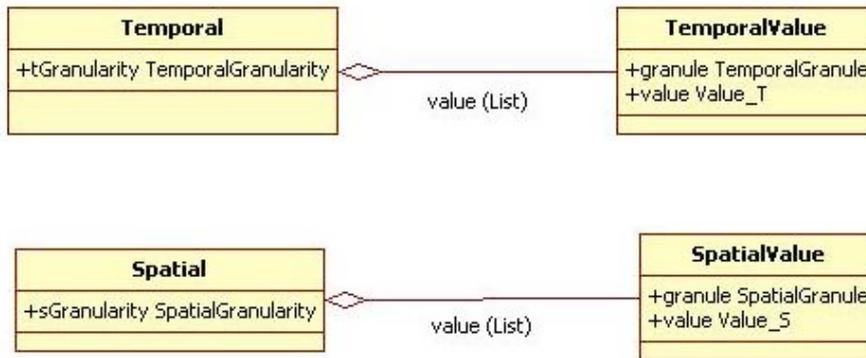


Fig. 5: *Spatio-Temporal data types*

The following is an example of spatio-temporal value, defined at temporal granularity *years* and at spatial granularity *countries*, with an alphanumeric inner type, which represents (the names of) some of the European Heads of

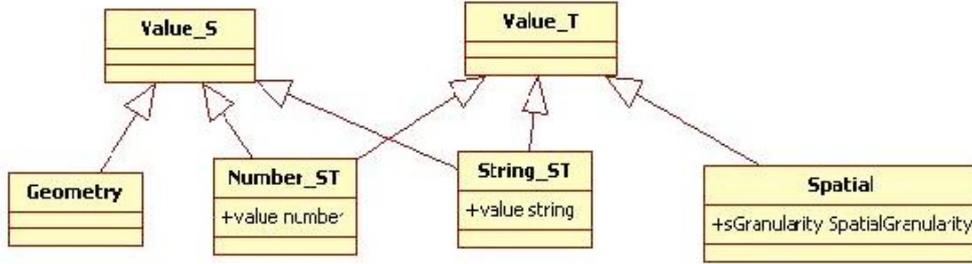


Fig. 6: *Spatio-Temporal inner types*

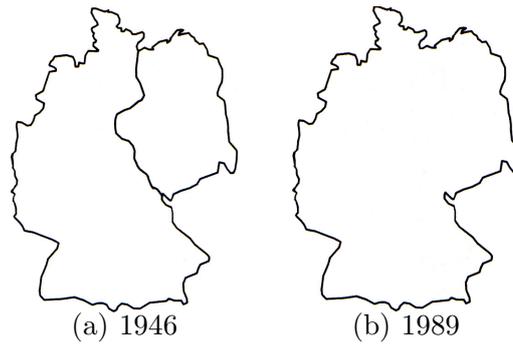


Fig. 7: *A spatio-temporal geometric value representing the German borders government:*

$$\{ \langle 2004, \{ \langle \text{France}, \text{'Raffarin'} \rangle, \langle \text{UK}, \text{'Blair'} \rangle \}^{\text{countries}} \rangle, \langle 2007, \{ \langle \text{France}, \text{'Fillon'} \rangle, \langle \text{UK}, \text{'Brown'} \rangle \}^{\text{countries}} \rangle \}^{\text{years}} .$$

By contrast, in Fig. 7, where the historical changes in the German political boundaries are shown, we represent a spatio-temporal value defined at granularities *years* and *countries*, where each country is depicted through a closed polyline.

Various policies may be adopted to reduce the size of multigranular spatio-temporal values and to improve the efficiency of values retrieval.

To improve the efficiency in space of the representation of multigranular values, we may coalesce the values defined for contiguous granules. Coalescing

is straightforward for temporal values, because intervals of granules may be used whenever the value defined for contiguous granules is the same (e.g., $\{\langle 2000,100\rangle, \langle 2001,100\rangle, \langle 2003, 100\rangle\}^{years}$ could be represented as $\{\langle 2000-2003,100\rangle\}^{years}$).

With the same purpose, we may enrich the data type specification with *temporal semantic assumptions* [13]. For instance, in a data structure storing the historical values of a bank account, one may assume that the value of the deposit does not change between two operations. In this case, the following value: $\{\langle 1/1/2000,1500\rangle, \langle 1/2/2000,1500\rangle, \dots, \langle 1/10/2000,1500\rangle, \langle 1/11/2000,2100\rangle\}^{days}$ could be equivalently represented as: $\{\langle 1/1/2000,1500\rangle, \langle 1/11/2000,2100\rangle\}^{days}$ specifying that for this value the *persistence* semantic assumption [13] holds⁵.

Similar strategies may be adopted for multigranular spatial values, whenever homogeneous areas may be identified in the represented data. In this case, the values may be conveniently converted to coarser granularities, improving the complexity in space of the representation. For instance, in a map storing the values of temperature of a given area, the spatial values may be coalesced in homogeneous (coarser) areas, and, at request, the value may be easily refined to a finer granularity.

4 Comparison Issues

With comparison issues we indicate the problems arising when comparing multigranular data, in particular in queries, including how to guarantee the existence of a common level of detail for performing a comparison, and the issues involved by converting values at different granularities.

⁵ Note that this representation requires the values be ordered.

4.1 *Guaranteeing a common representation*

Expressing the relationships among different granularities is important not only for obtaining smart representations, but also for enabling the comparison of multigranular values in queries. For instance, in a query we might require to compare the values of seasonal sales of two similar products, one stored at spatial granularity *countries* and one at temporal granularity *provinces*, to decide which one to sell in our chain of shops. To perform a meaningful comparison, we may not compare such values as they are, but they have to be expressed at the same spatial granularity.

Assuming a granularity system relying on finer-than, such as that described in [19], it is sufficient to convert one of the two values at the other granularity with a suitable granularity conversion, because *provinces* is finer-than *countries*. For example, the value at granularity *countries* may be split among the different provinces of each country or viceversa. But what if the two granularities were for example *feet* and *kms* in Fig. 2 In this case a third granularity, for example μms , which is finer-than both granularities, may be used.

In the most general case, given two multigranular values, one at granularity G and one at granularity H such that G and H are not directly related by finer-than, such values may be compared if the two values may be represented (i.e., converted) at the same granularity K , that is finer-than or coarser-than both G and H . K is chosen as the granularity that minimizes the number of conversions applied, that correspond to the number of hops in the granularity graph (see Fig. 2). If K is the coarsest among the granularities in the graph that are finer-than G and H , K is referred to as the *greatest lower bound*

(*GLB*) of G and H . For example μms is the *GLB* of *feet* and *kms*. Otherwise, if K is the finest among the granularities in the graph that are coarser-than G and H , it is referred to as the *least upper bound* (*LUB*) of G and H .

The existence of one between the *GLB* or the *LUB* is essential to guarantee the comparison of every granular value with another expressed at different granularity (but with the same inner type). Dyreson et al. [25] observe that pairs of granularities that do not have a unique *GLB* are very rare. The lack of a unique *GLB* is a situation that arises much less frequently than the lack of a unique *LUB*. Furthermore, the assumption of the existence of a unique *GLB* for each pair of granularities is enforced whenever a granularity is used to represent the domain, i.e., the *chronon* and the *quantum* granularities are included in the granularities graphs, as suggested in the previous section.

4.2 Multigranular Conversions

Multigranular models should enable to convert multigranular spatio-temporal information at different granularities, to improve or reduce the level of detail used for data representation. Conversions are essential in order to represent data at the most appropriate level of detail for each task, and they enable the consistent comparison of data at different granularities, thus improving the expressive power of spatio-temporal query languages. In this respect, a key feature in a multigranular model is the support for different conversion semantics, that is, for different transformations for converting values between different granularities.

Specific problems arise for the implementation of conversions, like the definition of the domain of a conversion, the design of optimal strategies for im-

proving the execution performance. Moreover, additional issues arise because of the multigranular representation. For example, guaranteeing that data consistency and usability are preserved is more challenging than in traditional models, specifically for spatial data. Moreover, the interpretation of aggregated values at coarser granularities require to deal with imprecise values, and conversion to finer granularities introduce indeterminacy on values. Finally, granularity conversions are usually non-invertible, and this may affect the definition of the query execution plan. All these problems must be addressed when implementing granularity conversions, and their side effects must be considered when defining the strategy for answering queries on multigranular values. In the rest of the section they are discussed separately, figuring out design solutions to address them.

4.2.1 Spatial Conversions Problems

A DBMS which enables to maintain geographical representations of the same area at different resolutions has to guarantee the topological consistency between different representation of the same entity at different resolutions: a query evaluated at a coarser level must give a result consistent with that we would obtain at a more detailed level. In Fig. 8 a topological consistent transformation of a spatial value representing a city is reported. The city is represented at progressively coarser levels of detail as (a) set of regions; (b) single region; (c) point; and the topological relationships with the roads crossing it, represented in red as polylines, are preserved along the transformations. If the spatial operators applied to solve the query do not assure to generate consistent maps, a posteriori check must be performed.

In [19] the conversion of multigranular geometrical features is obtained through

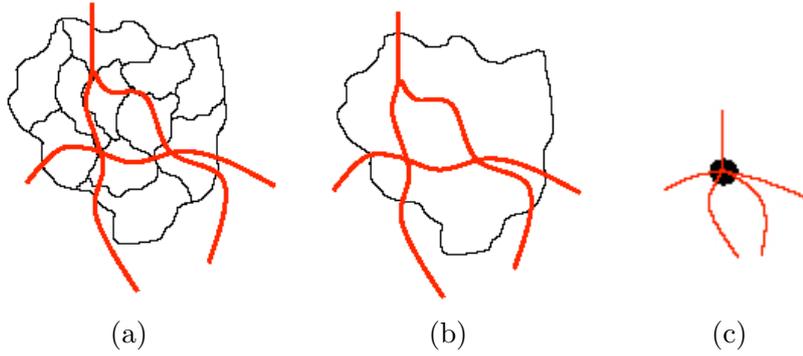


Fig. 8: *A topological consistent transformation*

the composition of model-oriented and cartographic map generalisation operators that guarantee topological consistency [10,41]. In particular, in [10] it has been proved that each consistent (i.e., preserving topological relationships) map transformation may be defined as a composition of the set of atomic operators supported in [19], which are characterized in terms of minimal morphisms in the category of a specific class of abstract cell complexes. This set has been extended with the topologically consistent algorithm proposed by Saalfeld [41] for line simplification, and with the refinement operators that perform the inverse functions. Such operators, which are represented in Table 1, may be classified with respect to the semantics of the conversion performed. *Contraction* and *thinning* operators reduce the dimension of vector features, whereas *expansion* operators increase it; *merge* operators merge adjacent features of the same dimension into a single one, while *splitting* operators subdivide single features in adjacent features of the same dimension; *abstraction* and *simplification* operators discard isolated features from polygons and remove shape points from a line, respectively, whereas *addition* operators add isolated features to polygons and shape points to lines.

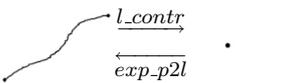
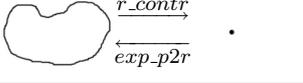
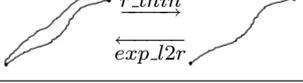
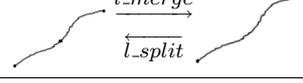
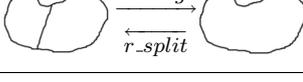
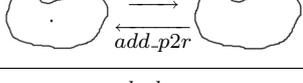
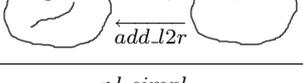
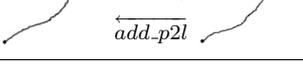
Operator	Generalization	Refinement
	l_contr : Contracts an open line to a point	exp_p12 : Expands a point into a line
	r_contr : Contracts a simple connected region and its boundary to a point	exp_p2r : Expands a point into a region
	r_thin : Reduces a region and its boundary to an open line	exp_l2r : Expands an open line into a region
	l_merge : Merge two lines sharing an endpoint into a single line	l_split : Splits a line into two lines
	r_merge : Merge two regions sharing a boundary line into a single region	r_split : Splits a region into two regions sharing a boundary line
	p_abs : Eliminates an isolated point from a region	add_p2r : Add a point inside a region
	l_abs : Eliminates an isolated line from a region	add_l2r : Add a line inside a region
	l_simpl : Removes a shape point from a line	add_p2l : Add a shape point to a line

Table 1: Geometric Operators

4.2.2 Partial Mappings of Conversions

Granularity conversions may be either total or partial functions depending not only on the values domain to which they are applied, but depending also on the relationships among granularities. Therefore, even when applied to legal spatio-temporal values, for some granular values, they may be undefined.

For instance, when the finer-than relationship holds between granularities, conversions to coarser granularities may be defined as total functions, whereas conversions to finer granularities may be partial functions. Indeed, according to the definition of finer-than, given two granularities G and H such that G is finer-than H , a (unique) H -granule that includes each G -granule always exists (see Fig. 1). Then, supposing a conversion function that maps G -values into H -

values is defined on the inner domain of the multigranular values, it is always defined. By contrast, the inverse condition may not hold. Therefore, when converting from coarser to finer granules, the conversion may be undefined.

Consider for example the situation depicted in Fig. 9, where granules of granularities *weeks*, *days*, and *schooldays* are shown. Between these granularities, finer-than holds. In particular, *schooldays* is finer-than *weeks*. When converting from *schooldays* to *weeks*, no problem arises, because when the conversion function is defined on a school day or on a set of school days, a week that includes them always exist (e.g., the days from the 17th to the 21st of December are included in the 51st week). By contrast, the inverse conversion may be undefined for vacation weeks (the 52nd week in the example). In this case, the domain of the function is defined (i.e., the 52nd week), but no school days exist for these weeks.



Fig. 9: Example of granularities related by the finer-than relationship.

The inverse situation arises when the *groups-into* relationship holds between granularities (see Fig. 10). In this case, conversions to finer granularities are (potentially) total functions, but when converting a value to a coarser granularity we have to be aware that a target granule may not exist.

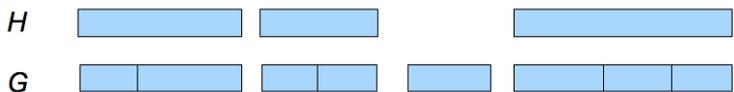


Fig. 10: The groups-into relationship.

In the above cases, the system could check in advance the existence of the target granules, thus avoiding the execution of a granularity conversion when they do not exist, even if values are defined for the corresponding portion of the spatio-temporal domain. This a-priori check would enable to save execution time, because usually granularity conversions are expensive operations. To distinguish these cases from situations where no value is actually defined at other granularities, a message error or warning may be returned, instead of a generic "undefined value".

To prevent partial mappings of the spatio-temporal domain between different granularities, other, more restrictive relationships may be adopted. For instance, a total definition for all conversions is given by the relationship *partitions*, that holds whenever both *finer-than* and *groups-into* hold [11]). However, supporting partition would not allow to support a wide range of granularities, such as *businessweeks* (*-days*, *-months*). In particular, granularities with *gaps* between granules may not be supported if granularities with completely partition the domain are considered (e.g., Gregorian calendar granularities, boundary subdivisions).

4.2.3 *Conversions to Coarser Granularities of Quantitative Values*

Conversions of quantitative values to coarser granularities involving the computation of aggregates may be affected by anomalies that have been widely studied, in particular in the spatial field, where they are known as the *Modifiable Area Unit Problem* (MAUP) and the *Ecological Fallacy* (EF) [5]. MAUP occurs when the geographical units considered in an aggregation are arbitrary and consequently modifiable according to the purpose of the operation. For example, in countries where the political elections rely on the intermediate re-

sults obtained in political subdivisions and there is not a direct proportion of the results with respect to the votes obtained, (for which, in turn, overall the rule "one man one vote" does not hold), different results may be obtained considering different intermediate aggregations (e.g., introducing new regions or provinces, or removing some of them on purpose).

By contrast, EF arises when aggregated data are misused for making inferences about individuals. For instance, knowing that the average salary of the population of a country is 1,500 Euro, does not mean that every person in that area earns exactly 1,500 Euro.

Both anomalies rely in turn due on how aggregations are performed and interpreted, thus the same problems may occur also for scaling of temporal data.

To give the user the possibility to evaluate correctly a granular value, this should give also a measure of the imprecision, or the indeterminacy, it is affected with. This error should be computed every time a multigranular conversion is applied. Moreover, a support wider as possible of conversion functions for quantitative (non-geometric) values should give the user the possibility to chose which conversion less affect the value.

For instance, in [19] the conversions to coarser granularities described in Table 2 are supported. They are referred to as *coercion functions*, and perform *selection* and *aggregation*.

4.2.4 Conversion to Finer Granularities

Even when they are defined functions, conversions to finer granularities intrinsically result in undetermined values [34,26]. This is particularly evident for geometric conversions. The conversion of regions to their barycenters, for

conversion family	conversion	semantics
<i>Aggregation</i>	max	Selects the maximum value defined
	min	Selects the minimum value defined
	avg	Computes the average of the values defined
	sum	Computes the sum of the values defined
<i>Selection</i>	main	Selects the most frequent value
	all	If all the finer values are the same, selects this value; otherwise it is undefined
	proj(n)	Selects the n th defined value
	first	Selects the first defined value
	last	Selects the last defined value

Table 2: Coercion functions [19]

instance, is usually considered an appropriate choice for representing them at a coarser granularity. By contrast, when scaling to a finer granularity and converting points to regions, for instance, the choice of the region to represent may be totally arbitrary. Similarly, the conversion of non-geometric data to a finer representation is affected by indeterminacy. For instance, given the value of rainfall stored for the 1st of June 2000 in Rome, we may not infer from this value the exact value of rainfall in a particular neighbourhood of Rome, at 12:00 AM (See also [15]). Therefore, when converting from coarser to finer granularities, we have to be aware of the error, or indeterminacy, we are introducing on data.

In [19], *refinement functions* enable to convert multigranular values to finer granularities. As described in Table 3, they perform *restriction* and *splitting* of values. *Restriction* functions apply downward inheritance property [43] that assumes that if a multigranular value is defined as v in a granule g , value v also refers to any *finer* granule g' included in g . For each attribute value for which downward inheritance is not appropriate (e.g., a temporal value storing the salary of an employee), *split* functions subdivide each coarser value among the finer granules included in it either uniformly (i.e., all the finer values will be the same), or according to a non-uniform distribution (`split[p(n)]` and `restr[p(n)]`).

Operations `split[p(n)]` and `restr[p(n)]` defined in [19], which implement “scale mass functions” as defined by Dyreson et al. [25], specifically address the indeterminacy arising from non-geometric conversions to finer granularities. The probability distribution considered in the computation may be defined, for instance, taking into account the semantics of the attribute value converted, or considering the distributions of legacy values. Different implementations for these functions have to be provided for each probability distribution supported. The parameterization of *restriction* and *split* functions with a probability distribution

conversion family	conversion	semantics
<i>Restriction</i>	<code>restr</code>	The coarser value is assigned to each finer granule
	<code>restr[p(n)]</code>	The coarser value is assigned to the finer granules, according to the probability distribution [p(n)]
<i>Split</i>	<code>split</code>	The coarser value is split among the finer granules
	<code>split[p(n)]</code>	The coarser value is split among finer granules, according to the probability distribution [p(n)]

Table 3: Refinement functions [19]

4.2.5 Invertibility of Conversions

Intuitively, when converting a multigranular value to a different granularity, and then performing the inverse conversion, we would expect to obtain the original value. Unfortunately, granularity conversions rarely are invertible: when converting from a finer to a coarser granularity, we lose some details that we cannot usually re-obtain by applying the inverse conversion to the finer granularity, unless we do not save the finer values in an auxiliary structure. By contrast, when converting from a coarser to a finer granularity, we introduce details that we should be able to forget, if we are no longer interested in them; in this case we may re-obtain the original value.

Then, we may identify pair of conversions which are invertible, and pairs that are not. For instance, we shown in [8] that `sum` is the inverse of `split`, `restr` is

the *quasi-inverse* of `avg` (see Tables 2 and 3), because $\text{restr}(\text{avg}(v)) = \pm\Delta$, where Δ is a maximum quantifiable error introduced by the application of conversions.

The property of invertibility is important for the definition of the query execution plan, whenever more values at different granularities are available for the same spatio-temporal snapshot (see Fig. 11). In this case, a query result may be affected by a quantifiable error, according to it is computed starting from available values at coarser or finer granularities [8].

5 Optimization Issues

The application of granularity conversion can be expensive in term of the use of computational resources, in particular for geometric conversions whenever they are applied to extensive geographic areas. Optimizations, in this case, are possible only at algorithm level [38]. Performance may improve also applying techniques for parallel computations [30].

Other optimization strategies may be applied to reduce the amount of data on which conversions are applied. For instance, rewriting algebraic techniques such those applied by database query engines when preparing the queries execution plans could improve the performance of data access. These techniques should give precedence to the application of target clauses, to reduce the amount of data affected by conversions, as well as to the execution of temporal conversions on spatio-temporal data on that of spatial conversions.

Traditional Index structures for ordered (e.g., temporal) and spatial vector data, like BTree+ and RTree and its variants (e.g., R⁺-Tree [42], R^{*}-Tree [7],

Hilbert R-Tree [33], PR-Tree [4]), may be applied to improve the efficiency of the retrieval of spatio-temporal values. Moreover, auxiliary data structures for spatio-temporal data have been proposed in the literature, and some of them may be extended to the multigranular case (HR⁺-Tree [46], MV3R-Tree [47], 2-3TR-tree [1]).

Whenever the queries to enhance involve value aggregates, auxiliary structures that summarize spatio-temporal multigranular values at coarser granularities may be used (e.g., aggregate Historical RB-tree [39]). For example, in [18] a data structure for the efficient storage and access to dynamic attributes has been proposed. Dynamic attributes are historical attributes whose values are tuples of temporal values, maintained at different levels of detail according to the age of data and to the execution of expiration conditions. An expiration condition is given by an expiration frequency and by a reaction policy to be applied when data expire: either evolution at a coarser granularity, or deletion of values, or both. For instance, it is possible to specify that an attribute may be evolved to a coarser granularity after a period of time, obtaining summarized information (through aggregation, selection, or user defined operations) from historical data.

In [17], such *evolution* capability has been extended to the spatio-temporal domain, enabling to obtain summarized information also from spatial and spatio-temporal data, according to given areas of interest. In Fig. 11 an example of such an auxiliary structure for a spatio-temporal value recording tax information is depicted. For this value (represented on top of the figure), two different aggregate structures are provided (at the figure bottom). The first one stores temporal aggregates (computed every five years), while the second summarizes tax information according to the macro area of reference.

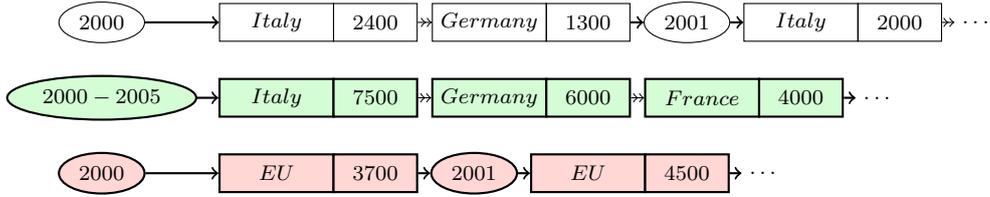


Fig. 11: *Auxiliary structure for spatio-temporal data*

6 Implementing Multigranularity

In our previous work [9,19] we have exploited the potentiality of object-oriented (OO) databases for supporting temporal multigranularity. In particular, the design of the temporal prototype described in [9] has been extended according to the model described in [19] integrating vector data and their conversions, thus enabling the application to the the spatio-temporal domain.

The use of OODBMS models for supporting multigranularity implies to face with additional issues regarding object inheritance, such as attribute redefinition and method override.

In what follows, we exploit the potentiality offered by the object-relational (OR) model, which integrate the benefits of relational models (i.e., simple design, declarative query language) with the advanced capabilities provided by the object-oriented approach (e.g., support for defining user defined data types, object inheritance, data types encapsulation), for implementing multiple spatio-temporal granularities.

Although the object-relational features supported by commercial products are somehow limited, the resulting design has the advantage of relying on the widely accepted standard SQL:99 [36]. Indeed, despite the considerable effort done to achieve the same result, ODMG, which is the reference model for object-oriented databases, is not yet a standard and thus it has not been fully

acknowledged by commercial products.

Furthermore, some commercial DBMS adopting the OR approach already provide a preliminary extension for handling spatial data. They may be preferred to GIS products for user-defined queries, because the latter are proprietary systems, thus difficult to extend. Moreover, in GIS products, spatial and non-spatial attributes are stored and manipulated separately according to a loosely coupled storage model [40]. By contrast the so-called Spatial DBMS (SDBMS) provide an integrated approach to the representation of spatial data that enables to store both spatial and non-spatial aspects of data within the same tables. Therefore SQL (extended with functionalities to directly support spatial queries) may be used to manipulate and query spatial data. However, current SDBMS and GIS packages do not provide either effective functionalities to manipulate temporal data, or multigranular support.

To enhance its computational capability, ORDBMS products provide the so called imperative extensions of SQL. OODBMS do not need similar extensions, because the query languages may be uniformly integrated with OO programming languages, which rely on the same paradigm. As a matter of fact, the complexity of use of OO query languages, if compared to the simplicity of the declarative approach of SQL, often encompasses this advantage.

Therefore, in the following we give the implementation detail of an OR multigranular implementation realized on Oracle 11g, which adopts the design solutions discussed in the previous section.

To conclude the section, we briefly describe also a syntax for the definition of user defined granularities which is included in the prototype in [9], that demonstrates the effectiveness of an efficient representation for granularities.

6.1 A Multigranular OR Spatio-temporal Type System

As we pointed out above, commercial DBMS products currently implement only a subset of object-relational features: for instance, methods override is not supported, and only very simple operations may be performed through types methods. Therefore, in what follows the data types for granularities, granules and multigranular types are somewhere redundant from a design point of view, and method declaration appears only in the leaves of the type hierarchy. The operations that have not been implemented as methods of user defined data types are realized through PL/SQL stored procedures and functions, which allow for improved performances than external procedures. In particular, functions may be used directly in SQL queries.

6.1.1 Granularities and Granules

In our prototype, we followed the design guidelines we described in Sections 3 and 4. The design of granularities and granules, whose Oracle data types are reported in Fig. 12, is compliant with the abstract data types of Fig. 3.

```
CREATE TYPE Granularity AS (  
  granID NUMBER,  
  granName VARCHAR(32),  
  labelFormat VARCHAR(32),  
  finer List<REF(Granularity)>,  
  coarser List<REF(Granularity)>  
)  
METHOD getFiner() RETURN LIST<Granularity>,  
METHOD getCoarser() RETURN LIST<Granularity>;  
  
CREATE TYPE Granule AS (  
  g REF(Granularity),  
  label VARCHAR(32),  
  index NUMBER  
);
```

Fig. 12: Oracle Data Types for granularities and granules

Each granularity is represented by a unique identifier, its name, and the format of the labels of its granules. The lists of finer and coarser granularities are also

provided, with the methods `getFiner` and `getCoarser` for returning them.

Type *Granularity* may be conveniently used to create a table storing the main information on the granularities supported by the system. Once a new granularity is created, it is sufficient to add a new tuple in this table. Each granularity is completely represented thanks to the implementation of the operations that convert the different granules representations: `index2label`, `label2index` and `getExtension`. In particular, the last operation maps a granularity granule (identified through its index) to the corresponding domain.

These operations have been implemented as stored procedures and functions for each granularity supported by the model. We have for example the following functions for granularity *days*

```
days_index2label(index NUMBER) RETURN VARCHAR
days_label2index(label VARCHAR) RETURN NUMBER
days_getExtension(IN index NUMBER) RETURN Anch.TemporalInterval
```

and the following operations for granularity *countries*:

```
countries_index2label(index NUMBER) RETURN VARCHAR
countries_label2index(IN label VARCHAR) RETURN NUMBER
countries_getExtension(IN index NUMBER) RETURN SDO.Geometry.
```

The data type *SDO_Geometry*, returned by the function `countries_getExtension`, represents the top hierarchy type for vector features as provided by the SDBMS (e.g., ORACLE *SDO_Geometry*). *Anch_TemporalInterval* is a data type for anchored temporal interval [31] (i.e., temporal intervals specified by the starting and the ending instants, then referring to the time domain).

```

CREATE TYPE SpatialValue AS (
  granule Granule,
  value Value_S
);
CREATE TYPE TemporalValue AS (
  granule Granule,
  value Value_T
);

CREATE TYPE Spatial AS (
  g Granularity,
  value List(SpatialValue)
) UNDER Value_T;
CREATE TYPE Temporal AS (
  g Granularity,
  value TemporalValue
);

```

Fig. 13: *Multigranular Data Types*

A *Granule* is represented by the reference to its granularity, by its label and granularity index (cf. Fig. 12). Other stored procedures are required to implement the operations `getFiner` and `getCoarser` defined for type *Granule* that return the (set of) finer or coarser granules of a granularity that intersect a given granule. The implementation differs according to the granularity category. For instance, for temporal granularities such operations rely on the inclusion property of the anchored temporal intervals representing temporal granules, and the algorithm applied is similar for all granularities without *gaps* among granules and *holes* within the granules [11]. For those particular granularities, specific implementations are required.

The implementation of these operations for spatial granularities is more difficult, and requires to handle the extensive mapping of each granularity to the spatial domain.

6.1.2 *Multigranular Spatio-temporal Values*

Spatio-temporal values are implemented as structured lists of pairs of granules and values ($\langle \textit{granule}, \textit{value} \rangle$) as defined in Figures 5. The corresponding ORACLE data types are shown in Fig. 13.

Data types *SpatialValue* and *TemporalValue* define a single pair ($\langle \textit{granule},$

```

CREATE TYPE Value_S AS (
);

CREATE TYPE Value_T AS (
);

CREATE TYPE Number_S AS (
  value number
) UNDER Value_S;

CREATE TYPE Number_T AS (
  value number
) UNDER Value_T;

CREATE TYPE String_S AS (
  value VARCHAR(128)
) UNDER Value_S;

CREATE TYPE String_T AS (
  value VARCHAR(128)
) UNDER Value_T;

CREATE TYPE Geometry_S AS (
  value Geometry
) UNDER Value_S;

```

Fig. 14: *Inner Data Types for Multigranular Values*

value >), while the granularity is referred to in the “wrapping” types *Spatial* and *Temporal*, that include the list of the pairs (< *granule*, *value* >) that represent the granular values. Note that type *Spatial* may be an inner type of *Temporal*, then spatio-temporal values are implemented as nested lists.

Inner types of granular values are defined according to the design specification given in Section 3, and are reported in Fig. 14. Two types hierarchies are created, *Value_S* and *Value_T*, that give inner types for spatial and temporal values, respectively. In particular, the definition of inner types of multigranular vector data relies on the data type already provided by the DBMS, that is *SDO_Geometry*.

Given such a specification, the internal representation of a spatio-temporal value storing the names of the European Heads of government would be represented as in Fig. 15.

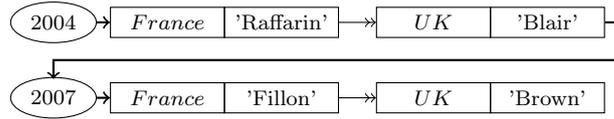


Fig. 15: *List for spatio-temporal data*

6.2 Automatic Implementation of User-Defined Granularities

The prototype in [9] includes a base set of temporal granularities implementing Gregorian calendar granularities. This set may be further extended with user defined granularities, whose implementation is automatically provided by the prototype, provided a granularity definition given by the user in the database schema. Indeed, the multigranular temporal Data Definition Language (DDL) has been extended with a set of instructions to specify new granularities relying on their relationships with existing ones, as discussed in Section 3. The user specification is processed and the new granularities are then implemented, relying on the implementation of the existing granularity.

For instance, we could define granularities *bimesters* (2 months) and *semesters* (6 months) relying on granularities *months* and *years*, exploiting the relationship groups-periodically-into [11]. In the prototype DDL such a relationship is specified by the use of the keywords `composedBy` and `compose` within the same definition, as illustrated in the following example:

```
granularity bimesters {
    1 composedBy 2 months;
    3 compose 1 semesters;
}
```

```
granularity semesters {
```

```
1 composedBy 3 bimesters;  
2 compose 1 years;  
}
```

Given the above specifications, the prototype automatically generates the implementation for the new granularities *bimesters* and *semesters*, relying on the existing implementation of *months* and *years*. In particular, the conversion among different granules representations (e.g., indexes, temporal intervals, label) relies on the regular subdivisions of these temporal granularities.

With this support we may easily specify a set of granularities for each application, allowing to tailor granularities to the domain represented and to specific requirements.

7 Conclusions

In this paper we have discussed implementation issues of spatio-temporal multigranularity. For a better comprehension of the details, we referred to a multigranular model we previously defined [19]. Nevertheless the problems we discussed apply to every data model providing a multigranular support.

Among these problems, the issues related to the efficient representation of granules, granularities and spatio-temporal values are particularly relevant. We showed a possible solution defined as extension of the object-relational model as supported by most commercial database products. Furthermore, multigranular conversions presented in the paper are specifically designed to prevent data inconsistency and to reduce indeterminacy.

We have also described an existing multigranular temporal object-oriented

prototype implementation addressing some of the issues we discussed, thus demonstrating the solutions we propose are feasible. We plan the porting of this implementation to the object-relational paradigm, and its extension to the spatio-temporal domain. The resulting prototype would enable to store and query spatio-temporal datasets at different levels of details. Thus we will be able to test its performances on real world datasets such as the Hurricane Isabel dataset (<http://www.tpc.ncep.noaa.gov/2003isabel.shtml>).

Acknowledgements

Research presented in this paper was funded by a Strategic Research Cluster grant (07/SRC/I1168) by Science Foundation Ireland under the National Development Plan. The authors gratefully acknowledge this support. The work of Elena Camossi is supported by the Irish Research Council for Science, Engineering and Technology.

References

- [1] M. Abdelguerfi, J. Givaudan, K. Shaw, and R. Ladner. The 2-3TR-tree, a trajectory-oriented index structure for fully evolving valid-time spatio-temporal datasets. In *Proc. of the 10th ACM international symposium on Advances in geographic information systems*, pages 29–34, 2002.
- [2] T.L. Anderson. Modeling time at the conceptual level. In *Proceedings of the International Conference on Databases: Improving Usability and Responsiveness*, pages 273–297, 1982.
- [3] G. Andrienko, D. Malerba, M. May, and M. Teisseire. Mining spatio-temporal data. *Journal of Intelligent Information Systems*, 27(3):187–190, 2006.
- [4] L. Arge, M. de Berg, H. J. Haverkort, and K. Yi. The Priority R-Tree: A Practically Efficient and Worst-Case Optimal R-Tree. In *Proc. of SIGMOD International Conference on Management of Data*, pages 347–358. ACM, 2004.
- [5] T. C. Bailey and A. C. Gatrell. *Interactive Spatial Data Analysis, Second Edition*. Longman, 1995.

- [6] S. Balley, C. Parent, and S. Spaccapietra. Modelling Geographic Data with Multiple Representations. *International Journal of Geographical Information Science*, 18(4):327–352, 2004.
- [7] N. Beckmann, H-P. Kriegel, R. Schneider, and B. Seeger. The R*-Tree: an efficient and robust access method for points and rectangles. In *ACM SIGMOD Record*, pages 322–331, 1990.
- [8] E. Bertino, E. Camossi, and G. Guerrini. Access to Multigranular Temporal Objects. In *Proceedings of the 6th Int. Conference On Flexible Query Answering Systems (FQAS 2004)*, number 3055 in Lecture Notes in Artificial Intelligence, pages 320–333. Springer-Verlag, 2004.
- [9] E. Bertino, E. Ferrari, G. Guerrini, and I. Merlo. T_ODMG: An ODMG Compliant Temporal Object Model Supporting Multiple Granularity Management. *Information Systems*, 28(8):885–927, 2003.
- [10] M. Bertolotto. *Geometric Modeling of Spatial Entities at Multiple Levels of Resolution*. PhD thesis, Università degli Studi di Genova, 1998.
- [11] C. Bettini, S. Jajodia, and X. Wang. *Time Granularities in Databases, Data Mining, and Temporal Reasoning*. Springer-Verlag, 2000.
- [12] C. Bettini and R. De Sibi. Symbolic Representation of User-defined Time Granularities. *Annals of Mathematics and Artificial Intelligence, Special Issues on Temporal Representation and Reasoning*, 30(1-4):53–92, 1999.
- [13] C. Bettini, X.S. Wang, and S. Jajodia. Temporal semantic assumptions and their use in databases. *IEEE Transactions on Knowledge and Data Engineering*, 10(2):277–296, 1998.
- [14] T. Bittner. Reasoning about qualitative spatio-temporal relations at multiple levels of granularity. In F. van Harmelen, editor, *Proceedings of the 15th European Conference on Artificial Intelligence*. IOS Press, 2002.
- [15] T. Bittner. Indeterminacy and Rough Approximation. In *Proceedings of the 16th International FLorida Artificial Intelligence Research Society (FLAIRS) Conference*, pages 450–454. AAAI Press, 2003.
- [16] T. Bittner and J.G. Stell. Stratified Rough Sets and Vagueness. In *Proceedings of the 2003 Conference on Spatial Information Theory*, number 2825 in Lecture Notes in Computer Science, pages 270–286, 2003.
- [17] E. Camossi. *Spatio-temporal Multigranularity in an Object Data Model*. PhD thesis, Università degli Studi di Milano, 2005.
- [18] E. Camossi, E. Bertino, G. Guerrini, and M. Mesiti. Handling Expiration of Multigranular Temporal Objects. *Special Issue of Journal of Logic and Computation*, 14(1):23–50, 2004.
- [19] E. Camossi, M. Bertolotto, and E. Bertino. A multigranular Object-oriented Framework Supporting Spatio-temporal Granularity Conversions. *International Journal of Geographical Information Science*, 20(5):511–534, 2006.

- [20] E. Camossi, M. Bertolotto, and E. Bertino. Multigranular Spatio-temporal Models: Implementation Challenges. In *Proceedings of the 16th International Symposium on Advances in Geographic Information Systems (ACM GIS08)*, 2008.
- [21] E. Camossi, M. Bertolotto, and E. Bertino. Querying multi-granular spatio-temporal objects. In *Proceedings 19th International Conference on Database and Expert Systems Application (to appear)*, Lecture Notes in Computer Science, pages 390–403. Springer Verlag, Berlin, 2008.
- [22] R. Cattell, D. Barry, M. Berler, J. Eastman, D. Jordan, C. Russel, O. Schadow, T. Stanienda, and F. Velez. *The Object Database Standard: ODMG 3.0*. Morgan-Kaufmann, 2000.
- [23] C. Claramunt and B. Jiang. Hierarchical Reasoning in Time e Space. In *Proceedings of 9th International Symposium on Spatial Data Handling*, pages 41–51, 2000.
- [24] M. Dukham, K. Mason, J.G. Stell, and M.F. Worboys. A formal approach to imperfection in geographic information. *Computer, Environment and Urban Systems*, 25:89–103, 2001.
- [25] C.E. Dyreson, W.S. Evans, H. Lin, and R.T. Snodgrass. Efficiently Supporting Temporal Granularities. *IEEE Transactions on Knowledge and Data Engineering*, 12(4):568–587, 2000.
- [26] C.E. Dyreson and R.T. Snodgrass. Supporting Valid-time Indeterminacy. *ACM Transactions on Database Systems*, 23(1):1–57, 1998.
- [27] F. Fonseca, M.J. Egenhofer, C. Davis, and G. Câmara. Semantic Granularity in Ontology Driven Geographic Information Systems. *Annals of Mathematics and Artificial Intelligence, Special Issue on Spatial and Temporal Granularity*, 36(1-2), 2002.
- [28] I.A. Gorawalla, Y. Leontiev, M.T. Öszu, D. Szafron, and C. Combi. Temporal Granularity: Completing the Puzzle. *Journal of Intelligent Information Systems*, 16(1):41–63, 2001.
- [29] T. Griffiths, A.A.A. Fernandes, N.W. Paton, and R. Barr. The Tripod spatio-historical data model. *Data Knowledge and Engineering*, 49(1):23–65, 2004.
- [30] R. Healey, S. Dowers, B. Gittings, and M. J. Mineter, editors. *Parallel Processing Algorithms For GIS*. CRC, 1997.
- [31] C.S. Jensen, C.E. Dyreson, M. Bohlen, J. Clifford, and al. A Consensus Glossary of Temporal Database Concepts. In O. Etzion, S. Jajodia, and S.Sripada, editors, *Temporal Databases: Research and Practice*, number 1399 in Lecture Notes in Computer Science, pages 366–405. Springer-Verlag, 1998.
- [32] B. Jiang and C. Claramunt. A Structural Approach to the Model Generalization of a Urban Street Network. *Geoinformatica*, 8(2):157–171, 2004.

- [33] I. Kamel and C. Faloutsos. Hilbert R-Tree: An Improved R-Tree Using Fractals. In *Proc. of the 20th Int'l Conf. on Very Large Databases*, pages 500–509, 1994.
- [34] V. Khatri, S. Ram, R.T. Snodgrass, and G. O'Brien. Supporting User Defined Granularities and Indeterminacy in a Spatiotemporal Conceptual Model. *Annals of Mathematics and Artificial Intelligence, Special Issue on Spatial and Temporal Granularity*, 36(1-2):195–232, 2002.
- [35] L. Kulik, M. Duckham, and M.J. Egenhofer. Ontology driven Map Generalization. *Journal of Visual Language and Computing*, 16(2):245–267, 2005.
- [36] J. Melton, A. Simon, and J. Gray. *SQL:1999 - Understanding Relational Language Components*. Morgan Kaufmann, 2001.
- [37] J-C. Muller, J.P. Lagrange, and R.Weibel, editors. *GIS and Generalization: methodology and practice*. London: Taylor and Francis, 1995.
- [38] M. Neun, D. Burghardt, and R. Weibel. Automated processing for map generalization using web services (to appear). *GeoInformatica*, 2009.
- [39] D. Papadias, Y. Tao, P. Kalnis, and J. Zhang. Indexing Spatio-Temporal Data Warehouses. In *Proc. of the 18th International Conference on Data Engineering*, pages 166–175. IEEE, 2002.
- [40] P. Rigaux, M. Scholl, and A. Voisard. *Spatial Databases with Application to GIS*. Morgan Kaufmann, Academic Press, 2002.
- [41] A. Saalfeld. Topologically Consistent Line Simplification with the Douglas-Peucker Algorithm. *Cartography and Geographic Information Science*, 26(1):7–18, 1999.
- [42] T Sellis, N Roussopoulos, and C Faloutsos. The R⁺-Tree: A Dynamic Index for Multi-dimensional Data. In *Proc. of the 13th Int'l Conf. on Very Large Databases*, pages 507–518, 1987.
- [43] Y. Shoham. Temporal Logics in AI: Semantical and Ontological Considerations. *Artificial Intelligence*, 33(1):89–104, 1987.
- [44] R.T. Snodgrass. *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, 1995.
- [45] J.G. Stell and M. Worboys. Stratified Map Spaces: A Fomal Basis for Multi-Resolution Spatial Databases. In T.K. Poiker and N. Chrisman, editors, *Proceedings of 8th International Symposium on Spatial Data Handling (International Geographical Union)*, pages 180–189, 1998.
- [46] Y. Tao and D. Papadias. Indexing Spatio-Temporal Data Warehouses. In *Proc. of the 13th Int'l Conf. on Scientific and Statistical Database Management*, pages 223–232. IEEE, 2001.
- [47] Y. Tao and D. Papadias. MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries. In *Proc. of the 27th International Conference on Very Large Data Bases*, pages 431–440, 2001.

- [48] R. Weibel and G. Dutton. Generalizing Spatial Data and Dealing with Multiple Representations. In P.A. Longley, D.j Maguire, M.F. Goodchild, and D.W. Rhind, editors, *Geographical Information Systems: Principles, techniques, management and applications*, chapter 10. New York: John Wiley, 1999.