Spatio-temporal Multi-granularity: Modelling and Implementation Challenges

Elena Camossi\(^1\), Michela Bertolotto\(^1\), and Elisa Bertino\(^2\)

\(^1\) School of Computer Science and Informatics - University College Dublin, Belfield, Dublin 4, Ireland. Phone: +353 (0)1 7162-944/913. Fax: +353 (0)1 2697-262.
\{elena.camossi,michela.bertolotto\}@ucd.ie

\(^2\) CERIAS - Purdue University, 250 N. University Street West Lafayette, Indiana, USA 47907-2066. Phone: +1 765 496-2399. Fax: +1 765 494-0739.
bertino@cs.purdue.edu

Technical Report UCD-CSI-2009-08
October 2009

Abstract. Multiple spatial and temporal granularities are essential to extract significant knowledge from datasets at different levels of detail: they enable to zoom-in and zoom-out a dataset, enhancing the data modelling flexibility and being instrumental to boost the analysis of information. Implementing granularities poses several interesting problems. Specifically, in this paper we analyse the issues involved by enhancing a data model and a query language with spatio-temporal multi-granularity, and we figure out efficacious solutions to address all of them. In our exposition, we investigate proper representations for the spatial and the temporal domains; then we conceive an appropriate design for granules and granularities, and for multi-granular values. In particular, mutual relationships among granularities and how they affect granularities design is discussed according to their influence on data access and considering the application of multi-granular conversions. Afterward, we dedicated to the design of multi-granular spatio-temporal conversions, discussing the multiple ways in which they affect data usability and envisaging how the design of a multi-granular model and query language may guarantee such an fundamental property, reducing uncertainty on the represented values, combining concepts like topologically consistent transformation, probability distributions, invertibility and quasi-invertibility properties. In our discussion, we are influenced on our previous work on multi-granularity. Especially, we expose some of the considerations that guided the design of a multi-granular spatio-temporal model we already proposed as extension of the ODMG data model. In this paper, we describe also some relevant details of a demonstrative object-oriented prototype realized on top of ObjectStore PSE Proj, and of an object-relational prototype realized on top of ORACLE. Both prototypes, even exploiting different characteristics of the underlying data models, prove the effectiveness of the proposed design solutions.

Keywords: 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 [13, 36]). Similarly, multi-granular modelling offers interesting capabilities in the spatial domain (see [5, 35, 45]): 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 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.

For each dimension involved in a dataset, a connected set of granularities may be defined, and the different sets are independent. The choice of proper granularities allows the system to store a minimal amount of data, according to the most appropriate levels 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 applications to define their own specific granularities.

Considering in particular the problem of extending a data model with spatio-temporal multi-granularity (see for example [16, 24, 29, 44]), we have to face with several design and implementation issues, that we may broadly classify in: modelling, comparison and querying, and adapting issues.

**Modelling issues.** These concern the formal design of the basic entities that make up the multi-granular data model. Different alternatives exist for the representation of the spatial and the temporal domains, considering they are discrete or continuous, mutually dependent or independent, or application driven. It is equally important to properly design granularities and granules, whose implementation affects the efficiency of the representation of multi-granular values, and in turn on the performance of queries execution. Strategies to improve the performance of all the data storage and retrieval process have to be delineated, like, for example, auxiliary data structures, both as extension of existing spatio-temporal indexes and through the definition of novel techniques for the optimization of queries involving multi-granular aggregates. Specifically, modelling issues correspond to following questions: How do we represent the spatial and the temporal domains? How do we represent granularities and granules? How do we relate granularities and maintain a granularity lattice? How do we extend the granularity sets and, at the same time, preserve the relationships among granularities? How do we represent multi-granular spatio-temporal values?

**Comparison and Conversion issues.** To answer queries is crucial that multi-granular values be comparable, therefore, in a multi-granular model, data are converted to the same granularity to enable their comparison at the same "unit of measure". The problem of finding the best common granularity to perform a value comparison is closely related to the issues of preserving relationships among granularities. Furthermore, the functions we use to shift granularity levels, namely *multi-granular conversions*, must preserve data semantics and usability and do not introduce errors on the converted data. Specifically, addressing comparison issues we answer
to the following questions: What is the best granularity to perform a comparison? How do we support granularity conversions? How do we preserve data usability and reduce uncertainty on converted values? Can we combine concepts like topologically consistent transformation, probability distributions, invertibility and quasi-invertibility properties to reduce uncertainty?

Adapting issues. The selection of attribute granularities is based on a trade-off between application efficiency and modelling requirements. Therefore, an important feature of multi-granular models is represented by the ability to set and change the spatio-temporal granularity as well the modelling conditions change. For example, in a spatio-temporal database for environmental monitoring, the collection of meteorological parameters like the amount of rainfall, the strength and direction of the wind, the value of the atmospheric pressure, must be collected more frequently in the presence of exceptional events like hurricanes and storms to check their progress.

In this paper we address the problem of adaptability with respect to inheritance as supported by Object-oriented and Object-relational models, and consider the problem of dynamically evolving spatio-temporal objects. In particular, we address the following questions: How do we refine object attributes? Are conventional object specialization models adequate? How do we support object evolution with respect to the object state and the object granularities?

In this paper we discuss such problems, delineating design and alternative implementation solutions to solve them. In our analysis, we rely on our previous work on multiple spatio-temporal granularities [24, 26, 21]. In particular, the considerations on modelling and comparison issues refers to the ST_ODMG model [24,26], that extends the object-oriented model ODMG [27] with multi-granular spatio-temporal capabilities. The design of ST_ODMG, as we provide in [24,26] is formal: it relies on de facto standards, like the ODMG and OQL, and on agreed definitions for granularities [13]. Furthermore, it gives formal proposals for all the main components of a multi-granular data model like granularities and their relationships and a validated formalization for spatio-temporal values. Moreover, differently from other multi-granular 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 multi-granularity.

In this paper we discuss how the design ST_ODMG has been affected by modelling and comparison issues: we advise two separate representations for the temporal and spatial domains; we illustrate the design requirements for granularities and granules, describing the design of an appropriate type system to support them, and figure out a convenient specification for user defined granularities, describing the implementation details of a multi-granular temporal extension of the ODMG DDL (Data Definition Language) that enables the automatic implementation of multiple temporal granularities. Afterwards, we extensively discuss the inconsistencies arising from the conversion of multi-granular values, and describe how to prevent them. Finally, we discuss some optimization strategies to improve the performance of the storage and the retrieval of multi-granular values.

Related to the problem of adaptability, we discuss two preliminary solutions to adapt attribute granularities we are currently studying: object-oriented attributes redefinition, and evolution mod-
els, supporting run time evolutions of spatio-temporal attributes. In particular, we refer to the recently defined $ST^2$ ODMGe (Spatio-(Bi)Temporal ODMG supporting Evolutions) [21], which supports the modification of the granularities used in attribute definitions, and the deletion of attribute values at run-time.

We conclude our analysis with the description of an object-relational spatio-temporal prototype developed on top of ORACLE® 11g\textsuperscript{2}, that supports spatio-temporal multi-granularity and follows the development strategies we describe in the paper. 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 ORACLE®. Such an extension does not represent a limit to the proposed implementation because similar spatial features are provided also by other mainstream spatial database products, like Microsoft® SQL Server\textsuperscript{4}, PostgreSQL with its spatial extension PostGIS\textsuperscript{5}, and MySQL\textsuperscript{6}.

This paper extends the work presented in [25], where a preliminary discussion on multi-granularity implementation issues was presented. With respect to [25], we expand the analysis of related literature; we discuss also adaptation issues; and we propose feasible solutions to multi-granular issues. In particular, we describe how multi-granular issues may be addressed in object-oriented and object-relational multi-granular implementations.

The paper is organized as follows. In Section 2 we present the scientific literature related to this work. Then, we discuss modelling, comparison and adaptation issues, respectively in Sections 3, 4 and Section 5. In Section 6 we describe the design of a multi-granular spatio-temporal object-relational prototype that adopts the implementation solutions we propose. Finally, in Section 7 we give a final discussion, and outline future research directions.

2 Background and Related Work

Spatio-temporal multi-granularity 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., [36]). [28] the Event Calculus is extended to support temporal granularity and indeterminacy, to represent imprecise temporal events at different granularities. [36] is focused on calendric granularities, that may be applied to common calendars. Most of these approaches search for a common basis on top of which one may reason on multiple granularities in temporal databases. A consensus among the different disciplines interested in temporal granularity representation has been achieved with the formalization proposed by Bettini et al. [13], the glossary of time granularity concepts presented by Bettini et al. [13]. In the authors extend [13], who give a comprehensive discussion on temporal granularities for databases, data mining, and temporal reasoning. This formalism is used also by other works [31, 14]. The approach of [31] is logical, based on a linear temporal logic.

\textsuperscript{2} www.oracle.com Last accessed September 2009.
\textsuperscript{5} http://www.postgresql.org and http://postgis.refractions.net/ (accessed September 2009).
\textsuperscript{6} http://www.mysql.com/ (accessed September 2009).
Temporal granularity issues related to temporal databases have been investigated both for the relational and the object-oriented data models. In its first release [57] 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 [13], Dyreson et al. [33] 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, (e.g., [30]), informal, and the support of multiple temporal granularities is given as extension to the set of types of the temporal model. Moreover, in these approaches the specification and management of different granularities, e.g., how to convert from one granularity to another, is completely left to the user. None of them refers to the ODMG standard object model, which implies a strong dependence of each temporal model from the reference object model. By contrast, Bertino et al. [11] investigate the impact of temporal granularities in an object-oriented model compliant with the ODMG standard.

The representation of data at multiple levels of details, that is, at multiple granularities, is a topic of relevant interest also when modelling spatial entities. In the spatial domain, plenty of research has been undertaken on multiple representations [5] (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 [42]. Stell and Worboys [58] 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. In other research articles [20, 29, 40, 61] spatial levels of detail are organized in hierarchies. Timpf in [61] deduces the levels of detail from map series, i.e., sets of spatial data representing the same geographical area at different resolutions. The levels of detail resemble those obtained with human driven abstraction processes like aggregation, filtering and generalization.

Research on multiple resolutions addresses in particular model-oriented generalization [47], 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 [19] to preserve data usability, as, for example, the preservation of topological relationships [52]. Several model-oriented generalization operators [64] have been defined in the literature (e.g., aggregation, line simplification). For instance, Kulik et al. [45], define an algorithm performing line simplification driven by an ontology to be applied to transportation networks.
Recently, also the area of qualitative spatial reasoning has shown a growing interest in spatial representation at multiple levels of detail [18]. Most of the work in this area has focused on the imprecision, the imperfection, and on the vagueness of spatial representations [32], and on spatial rough sets [18]. 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. With this concern, Fonseca et al. [35] discuss specifically the design and the support for semantic granularities.

Few proposals in the literature address the multi-granular representation of spatio-temporal data. Claramunt and Juang [29] propose the application of nested hierarchies for modelling space and time to extract quantitative information about spatio-temporal relationships in a data set. Vangenot [62] proposes a timestamped model for multiple data representations where each representation is related to a view point and a (spatial and semantic) resolution. The relationships among the different representations of objects are semantic driven. The modelling of moving objects over multiple granularities is discussed also by Hornsby and Egenhofer In [39] a temporal coarsening operation is applied to geospatial lifelines that model the motion of spatial objects, to predict their movement. The coarsening operation is implemented as a decrease of the timestamping frequency of the object geospatial lifeline, that is then observed at a greater level of (temporal) detail, in order to simplify the motion prediction. Griffiths et al. [37] define the Tripod spatio-historical model, which integrates the definition of granular histories. No operators are provided to convert multi-granular data, but the histories are always internally represented at the chronon [41] granularity. Katri et al. [44] define an annotation-model for the specification of spatio-temporal data at multiple granularities. Their granularity model relies on the concepts of temporal indeterminacy [34] and spatial imprecision [32], and is compliant with [58]. 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.

The European project MurMur [51] addresses multiple resolutions through multiple representations, supporting perceptions, which include different point of views and spatial resolutions. More recently, the granularity conversions system defined by Camossi et al. [24] have been applied by Bertino et al. [10] to define a multi-granular spatio-temporal object-relational model. Wang and Liu [63] adopt the same definition of spatial granularity as [24], addressing uncertain spatio-temporal regions. Conversely, Belussi et al. [7] define spatio-temporal granularities as historical evolution of spatial granules, to ease the search of valid spatial granules in a given instant. Their approach relies on the mapping of spatial multiple granularities and granules onto graph structures (multidigraphs), which encompass labelling functions for granules and their mutual (topological) relationships, disregarding value conversions.

3 Modelling Issues

The representation of spatio-temporal data at multiple granularities poses several mutually interconnected problems. In the following, we debate in particular on the implementation of the spatio-
temporal domain, and the design of efficient data-types and structures for granules, granularities, and multi-granular 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 multi-granular 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 [41], 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 [13], in[24] 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 overlap\(^7\). 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 multi-granular model. Dyreson et al. [33] 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 [24] spatial and temporal granularities are related by the *finer-than* relationship [13] 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*.

\[ \begin{array}{c}
H \\
G
\end{array} \]

*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 drown in the figure. Note that the finest granularities in the granularity graphs give the measure of the precision applied for multi-granular 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 [41], 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

---

\(^7\) Temporal granules, according to the definition in [13], do not overlap, while spatial granules may touch along the boundaries.
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 [24], 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.

![Fig 2: Examples of granularity graphs. (a) Temporal granularities. (b) Spatial granularities.](image)

![Fig 3: Granularity and Granule data types](image)
A design that implements this specification of granules and granularities is given in Fig. 3, where the corresponding UML abstract data types are depicted. 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 [41] 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. 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.

User-Defined Granularities The prototype described in [11] implements T_ODMG, a multi-granular temporal object-oriented model. It includes a base set of temporal granularities implementing Gregorian calendar granularities, which may be extended with user defined granularities. The user describes a new granularity in the database schema definition with a very intuitive specification that exploits the relationships between granularities, according to the approach we described above. The T_ODMG Data Definition Language (DDL) has been extended accordingly. Afterwards,
the system processes the schema definition and automatically generates the granularity implementation, which is given as a Java class, relying on the implementation of the existing granularities.

Example 1. The following is an example of definition of two granularities that are not originally supported by the prototype: bimesters (i.e., two months) and semesters (i.e., six months). Their semantics relies on granularities months and years, therefore we exploit the relationship groups-periodically-into [13], represented by the keywords composedBy and compose, to define them.

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

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

Given such a specification, the implementation for the new granularities bimesters and semesters is generated relying on the implementation of months and years, which is already provided. 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. Such a specification, which is particularly suitable for temporal granularities, may be applied also to spatial granularities with even subdivisions (e.g., meters and kilometers). Symbolic specifications for temporal granularities are proposed in [14], relying on collection and slice formalisms.

### 3.3 Storage of Multi-granular Spatio-temporal Values

A multi-granular spatio-temporal database schema may include conventional, multi-granular spatial, temporal, and spatio-temporal attributes. In [24] multi-granular 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 multi-granular spatio-temporal. In Figures 5 and 6 we report the Unified Modeling Language (UML, http://www.uml.org/) schemas that define multi-granular spatio-temporal types.
The following is an example of spatio-temporal value, defined at temporal granularity \textit{years} and at spatial granularity \textit{countries}, with an alphanumeric inner type, which represents (the names of) some of the European Heads of government:

\[
\{\langle 2004, \{\langle \text{France, ‘Raffarin’} \rangle, \langle \text{UK, ‘Blair’} \rangle \rangle \}_\text{countries}, \\
\langle 2007, \{\langle \text{France, ‘Fillon’} \rangle, \langle \text{UK, ‘Brown’} \rangle \rangle \}_\text{countries}\}_\text{years}.
\]

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 \textit{years} and \textit{countries}, where each country is depicted through a closed polyline.

**Spatio-temporal Values Optimization** Various policies may be adopted to reduce the size of multi-granular spatio-temporal values and to improve the efficiency of values retrieval.

To improve the efficiency in space of the representation of multi-granular values, we may coalesce the values defined for contiguous granules. Coalescing is straightforward for temporal val-
ues, because intervals of granules may be used whenever the value defined for contiguous granules is the same (e.g., \{⟨2000,100⟩, ⟨2001,100⟩, ⟨2003, 100⟩⟩ \text{years} could be represented as \{⟨2000-2003,100⟩⟩ \text{years}).

With the same purpose, we may enrich the data type specification with \textit{temporal semantic assumptions} [15]. 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: \{⟨1/1/2000,1500⟩, ⟨1/2/2000,1500⟩, …, ⟨1/10/2000,1500⟩, ⟨1/11/2000,2100⟩⟩ \text{days} could be equivalently represented as: \{⟨1/1/2000,1500⟩,⟨1/11/2000,2100⟩⟩ \text{days} specifying that for this value the \textit{persistence} semantic assumption [15] holds.

Similar strategies may be adopted for multi-granular 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 and Conversion Issues

With comparison issues we indicate the problems arising when comparing multi-granular 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.

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 multi-granular values in queries. For instance, in a query we might require to compare the values of seasonal sales of two similar products,

\footnote{Note that this representation requires the values be ordered.}
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 [24], 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 vice versa. 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 multi-granular 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. [33] 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 Multi-granular Conversions

Multi-granular models should enable to convert multigranular spatio-temporal information at differ-
et 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 expres-
sive power of spatio-temporal query languages. In this respect, a key feature in a multi-granular
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 improving the execution performance. Moreover,
additional issues arise because of the multi-granular representation. For example, guaranteeing that
data consistency and usability are preserved is more challenging than in traditional models, specif-
ically 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 multi-granular values. In the rest of the section they are discussed separately, figuring out design solutions to address them.

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 [24] the conversion of multi-granular geometrical features is obtained through the composition of model-oriented and cartographic map generalisation operators that guarantee topological consistency [12, 54]. In particular, in [12] 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 [24], 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 [54] 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.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Generalization</th>
<th>Refinement</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_contr</td>
<td>1_contr: Contracts an open line to a point</td>
<td>exp_pl2: Expands a point into a line</td>
</tr>
<tr>
<td>exp_p2l</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r_contr</td>
<td>r_contr: Contracts a simple connected region and its boundary to a point</td>
<td>exp_p2r: Expands a point into a region</td>
</tr>
<tr>
<td>exp_p2r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r_thin</td>
<td>r_thin: Reduces a region and its boundary to a point</td>
<td>exp_l2r: Expands an open line into a region</td>
</tr>
<tr>
<td>exp_l2r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>l_merge</td>
<td>l_merge: Merge two lines sharing an endpoint into a single line</td>
<td>l_split: Splits a line into two lines</td>
</tr>
<tr>
<td>l_split</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r_merge</td>
<td>r_merge: Merge two regions sharing a boundary line into a single region</td>
<td>r_split: Splits a region into two regions sharing a boundary line</td>
</tr>
<tr>
<td>r_split</td>
<td></td>
<td></td>
</tr>
<tr>
<td>p_abs</td>
<td>p_abs: Eliminates an isolated point from a region</td>
<td>add_p2l: Add a point inside a region</td>
</tr>
<tr>
<td>add_p2r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>l_abs</td>
<td>l_abs: Eliminates an isolated line from a region</td>
<td>add_l2r: Add a line inside a region</td>
</tr>
<tr>
<td>add_l2r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>l_simpl</td>
<td>l_simpl: Removes a shape point from a line</td>
<td>add_p21: Add a shape point to a line</td>
</tr>
</tbody>
</table>

Table 1: Geometric Operators

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 multi-granular 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.](image)

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.](image)

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 conver-
sions is given by the relationship partitions, that holds whenever both finer-than and groups-into
hold [13]). 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).

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) [49]. 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 polical elections rely
on the intermediate results 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 individ-
uals. 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 com-
puted every time a multi-granular 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 [24] the conversions to coarser granularities described in Table 2 are supported.
They are referred to as coercion functions, and perform selection and aggregation.

Conversion to Finer Granularities Even when they are defined functions, conversions to finer
granularities intrinsically result in undetermined values [44,34]. This is particularly evident for
geometric conversions. The conversion of regions to their barycenters, for instance, is usually con-
sidered 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 [17]). Therefore, when converting from coarser to
finer granularities, we have to be aware of the error, or indeterminacy, we are introducing on data.

In [24], refinement functions enable to convert multi-granular values to finer granularities. As
described in Table 3, they perform restriction and splitting of values. Restriction functions apply
downward inheritance property [56] that assumes that if a multi-granular 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 ($\text{split}[p(n)]$ and
$\text{restr}[p(n)]$).

Operations $\text{split}[p(n)]$ and $\text{restr}[p(n)]$ defined in [24], which implement “scale mass func-
tions” as defined by Dyreson et al. [33], specifically address the indeterminacy arising from non-
geometric conversions to finer granularities. The probability distribution considered in the com-
putation 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

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

Table 2: Coercion functions [24]

<table>
<thead>
<tr>
<th>conversion family</th>
<th>conversion</th>
<th>semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction</td>
<td>restr</td>
<td>The coarser value is assigned to each finer granule</td>
</tr>
<tr>
<td></td>
<td>restr[p(n)]</td>
<td>The coarser value is assigned to the finer granules, according to the probability distribution $[p(n)]$</td>
</tr>
<tr>
<td>Split</td>
<td>split</td>
<td>The coarser value is split among the finer granules</td>
</tr>
<tr>
<td></td>
<td>split[p(n)]</td>
<td>The coarser value is split among finer granules, according to the probability distribution $[p(n)]$</td>
</tr>
</tbody>
</table>

Table 3: Refinement functions [24]
Invertibility of Conversions Intuitively, when converting a multi-granular 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 \( \text{sum} \) is the inverse of \( \text{split} \), \( \text{restr} \) is the quasi-inverse of \( \text{avg} \) (see Tables 2 and 3), because \( \text{restr}(\text{avg}(v)) = \pm \Delta \), where \( \Delta \) 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].

4.3 Conversion Optimization

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 [48]. Performance may improve also applying techniques for parallel computations [38].

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 [55], R\(^*\)-Tree [6], Hilbert R-Tree [43], 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 multi-granular case (HR\(^+\)-Tree [59], MV3R-Tree [60], 2-3TR-tree [1]).

Whenever the queries to enhance involve value aggregates, auxiliary structures that summarize spatio-temporal multi-granular values at coarser granularities may be used (e.g., aggregate Historical RB-tree [50]). For example, in [23] 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.

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, as discussed in more detail in the following section.

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.

5 Adaptation Issues

Adaptivity support is a crucial requirement for almost all applications we may think of. In a spatio-temporal setting with multiple granularities, an added dimension to the problem of adaptation is represented by the evolution of attribute granularity. Being able to dynamically adapt the spatial and temporal granularities to respond to dynamic events and situations and to reflect changes in data significance is crucial in many contexts: e.g., periodic phenomena, modifications to attribute values, operation execution, data aging or privacy restrictions. Specific operations required for supporting dynamic adaptation of granularity include: 1) granularity evolution, which aggregates existing detailed data at a coarser granularity (e.g., older data that may be stored for future reference), or even refines information at a finer granularity (e.g., in data analysis); 2) granularity acquisition, which changes at run-time the granularity used when inserting new values in the database; 3) value deletion, which removes attribute values from the database, whenever they are no longer useful at a given granularity (e.g., detailed data).

The first approach we propose here to address object adaptability enables to adapt attribute granularities exploiting the potentiality of object-oriented (OO) and object-relational (OR) DBMS along the inheritance hierarchy, pre-arranging granularity modifications in the database schema. Conversely, object evolutions [21], which are described at the end of the section, adopts a flexible solution by which run time evolutions may be specified and executed.
5.1 Multi-granular Attribute Redefinition

The idea behind multi-granular attribute refinement is that the granularity at which an attribute value is stored can be changed in a subclass, to better reflect the application evolution needs. In the subclass the attribute values may be maintained at a coarser or at a finer level of detail. For instance, if at the superclass only the monthly values are recorded, in the subclass the daily changes can be maintained, improving the level of detail for the attribute. By contrast, we may reduce the detail coarsening the attribute value in the subclass.

The most critical requirement in attribute refinement is to preserve object substitutability. Whenever an object instance of a subclass is found in a context where a superclass object was expected, its attribute values must be converted to the expected granularity, leaving the whole procedure completely transparent to the user. Therefore, multi-granular conversions including both coercion and refinement functions, such those described in the previous Section, must be provided by a multi-granular model. Supplying a variety of conversions with different semantics enables to choose, for each attribute and situation, the conversion that better reflects the attribute semantics.

Substitutability impacts both attribute accesses and updates. In our previous work [11, 24] we have exploited the potentiality of OODBMS for supporting temporal multi-granularity, and we extended the $ST_{ODMG}$ DDL and prototype with attribute redefinition features. In the prototype, we distinguish the cases of object access and update. In case of object access, granularity conversions are used to compute the value to be considered in the superclass, given the value of the attribute in the subclass. By contrast, in case of object updates, granularity conversions are applied to convert the value to assign to the granularity required in the subclass. In the following example we describe how attribute refinement works in our prototype.

Example 2. The following multi-granular spatio-temporal schema includes an example of multi-granular attribute refinement\footnote{The syntax we use in this example has been first introduced in $ST_{ODMG}$: it extends the ODMG Data Definition Language to spatio-temporal multi-granularity. The same syntax has been further extended in [9] to support attribute refinement.}. We give a partial definition for class Nation, reporting the specification of attribute population, which stores the daily updates of the amount of population recorded in each municipality of a country.

```plaintext
class Nation (...) {
    attribute Temporal$_{days}$(Spatial$_{municipalities}$(int)) population;
    ...
};
```

Then, we define a class NationStatistic, which extends Nation, to collect statistical information on the countries in the database. In particular, attribute population is refined at temporal granularity years and at spatial granularity countries.

```plaintext
class NationStatistic extends Nation (...) {
    ref attribute Temporal$_{years}$(Spatial$_{countries}$(int)) population {
        { split$_{countries}$$\rightarrow$$municipalities, sum$_{municipalities}$$\rightarrow$$countries },
```
Two pairs of granularity conversions are specified for this attribute: the first refers to the spatial refinement, whereas the second deals with the temporal refinement. In each pair of conversions \( \langle af, uf \rangle \), \( af \) is the granularity conversion used to access the attribute value from an object that at compile time has type \texttt{Nation}: the attribute value has to be converted from the sub-class granularity (e.g., \texttt{countries}) to that used in the super-class (e.g., \texttt{municipalities}). In this example both the spatial and the temporal granularities have been refined in the sub-class, therefore we have two refinement conversion functions to use in the access: both \texttt{split} and \texttt{restr} (i.e., split and restriction) granularity conversions have to be applied to the value, that is converted from granularities \texttt{years} and \texttt{countries} to finer granularities \texttt{days} and \texttt{municipalities}. \( uf \) conversions are applied when updating this attribute from an object whose run-time type is \texttt{NationStatistic}, while at compile time it has type \texttt{Nation}: in this case, the conversions \texttt{sum} and \texttt{avg} (i.e., sum and average) are applied to coarser the finer value to granularities \texttt{years} and \texttt{countries}. 

To preserve data consistency, both compile and run-time checks may be applied. At compile time, the consistency of the database schema must be verified, checking first that the granularities in the superclass and in the subclass are related by some granularity relationship. In \texttt{ST2ODMG} we consider the finer-than relationship, but other relationships may be applied as well. Then, we have to check that two inverse or quasi-inverse \cite{8} granularity conversions have been specified, one to use for the attribute access and one for the attribute update. However, at run-time we may allow one to apply a different granularity conversion for the attribute access, whenever the user needs a different conversion semantics, and also in this case the granularity conversion must be compliant with the attribute refinement.

### 5.2 Evolutions of Spatio-temporal Multi-granular Objects

The recently defined \texttt{ST2ODMG} (Spatio-(Bi)Temporal ODMG supporting Evolutions) \cite{21} addresses adaptability requirements by supporting the modification of the granularities used in attribute definitions, and the deletion of attribute values at run-time. Evolutions have the form: \texttt{ON Event [IF Condition] DO Action}. Example of events are: update, delete, etc., that is occurrences that modify the database state, including evolution actions, and may have a periodic or an extemporary behaviour. Conditions are specified against database attribute values, and include also periodic checks, evaluated on valid time. Finally, evolution actions are sequences of operations that may modify the attribute granularities and delete the attribute values.

Evolutions are defined and executed at run-time and conform to the execution model of active databases. Given an instance of an \texttt{ST2ODMG} database and a set of evolutions specified for it, the database is continuously monitored. The execution of database transactions modifies the database state and triggers the evolutions whose events refer to such transactions. Therefore, the corresponding conditions are evaluated. For those triggered evolutions whose conditions evaluate to
TRUE, the corresponding actions are executed. As a consequence, the database state (or schema, in case of granularity acquisition) may be modified.

Example 3. Given class Nation we defined in Example 2, the following are two examples of evolutions we may specify to periodically obtain summarized values of the amount population of the countries in the database.

ON update Nation.population < days, municipalities >
IF every 1 years
DO evolve < days, municipalities > to < days, countries > using
sum municipalities → countries, split countries → municipalities;

ON update Nation.population < days, countries >
DO evolve < days, countries > to < years, countries > using
avg days → years, restr years → days;

The first evolution is triggered by the updates of attribute population as originally defined in class Nation, i.e., at temporal granularity days and at spatial granularity municipalities. Once one years of values (i.e., VT denotes valid time) have been recorded for this value, the evolution is executed, and the first time a new value is created for this attribute: specifically, a new granularity level (see [22]) at granularity days and countries is defined for attribute population. For each country, it stores the daily amount of population, given as the sum of the population of every municipality in the country. This evolution is executed periodically, every 1 years. Every time this new granularity level is updated (i.e., once a year), the second evolution is triggered. It results in the creation of a new granularity level, at granularity years and countries, that stores the annual amount of population of the country. This value is obtained as the average of the daily amount stored in the previous granularity level. Consequently to the execution of evolutions, the run-time type of population is Temporal\(_{days}(\text{Spatial}_{municipalities}(\text{int})) \times \text{Temporal}_{days}(\text{Spatial}_{countries}(\text{int})) \times \text{Temporal}_{years}(\text{Spatial}_{countries}(\text{int})).\) Note that the last granularity level has the same type of attribute population we defined in class NationStatistic of Example 2. However, in this case the value is automatically computed, and belongs to the same object of type Nation it refers to. By contrast, in the case of Example 2, for each country at least two objects has to be created to maintain the same information.

After the execution of evolutions, the run-time type of attribute values are Cartesian products of multi-granular types as defined in Section 3. Therefore at run-time the state of objects in the database is no longer consistent with their class definition. We formally revisited the notion of object consistency, weakening the conditions on attribute values and on objects spatio-temporal lifespan to include the side-effects of evolutions. In particular, we require that each evolution specification includes a pair of inverse and quasi inverse granularity conversions, enabling to navigate among portions of the same attribute value expressed at different granularities.

Moreover, we take advantage of attribute run-time values at multiple granularities to enhance the access strategies to multi-granular values. We demonstrate that, under certain assumptions,
object access is invariant to the execution of evolutions. In particular, the stored information may be preserved after value deletion, because the same value may be present in the database at a different granularity, and retrieved when needed. Furthermore, object access may benefit from evolutions with respect to both effectiveness and efficiency. The values resulting from the execution of granularity conversions are already materialized in the database, thus improving the performance of queries involving aggregates and granularity refinement. The existence in the database of values at different granularities makes it possible to apply two different strategies for object access. Such strategies optimize, respectively, execution efficiency, minimizing the retrieval time, and result accuracy, minimizing the indeterminacy of granular values.

6 Multi-granularity in Object-Relational DBMS

Object-Relational DBMS (ORDBMS) combine the benefits of the relational model (i.e., simple design, declarative query language) with the advanced capabilities provided by the OO paradigm (e.g., user defined data types, object inheritance, data types encapsulation). Differently from OODBMS, ORDBMS have the advantage of relying on the widely accepted standard SQL:99 [46]. Indeed, despite the considerable effort done to achieve the same result, ODMG, the reference model for OODBMS, is not yet a standard and thus it has not been fully acknowledged by commercial products. Moreover, the imperative extensions of SQL of ORDBMS enhance the computational capability of the query language, preserving the simplicity of the SQL declarative approach, which often encompass the advantage of OODBMS that do not need such extension to query the database, but have a complex use. However, the object-relational features supported by commercial products are somehow limited.

Some commercial ORDBMS provides also a preliminary spatial type system to support the handling of spatial data, overcoming the limitations of GIS products, which adopt proprietary type systems and formats and are therefore difficult to extend. Moreover, differently from GIS products which adopt a loosely coupled storage model [53] where spatial and non-spatial attributes are stored and manipulated separately, 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, is able to manipulate and query spatial data.

However, current SDBMS and GIS packages do not provide either effective functionalities to manipulate temporal data, neither the multi-granular support. In what follows, we exploit the potentialities of ORDBMS to implement a multiple spatio-temporal granularities model, adopting the design solutions discussed in the previous sections.

6.1 A Multi-granular OR Spatio-temporal Type System

In the following, we describe a multi-granular spatio-temporal implementation realized on top of ORACLE® 11g. In particular, we define the ORACLE® data types to store multi-granular spatio-temporal attribute values, and the required methods and stored procedures to manipulate them,
whose implementation is given in Java classes which have been loaded in the DBMS and executed by the embedded Java Virtual Machine. This is a specific feature of ORACLE, which has been introduced in the product since version 8, and it enables to encompass some of the limitations of PL/SQL, the imperative extension of SQL provided by ORACLE, which may be used in alternative to implement stored procedures, but whose expressive power is more limited that that provided by JAVA. By contrast, combining ORACLE\textsuperscript{R} user defined data types and implementing database stored procedures with Java, we take advantage of the completeness and the expressive power of Java, as well as of existing Java geometric libraries, like the Java Topology Suite\textsuperscript{10} to implement granularity conversions; at the same time, the implemented operations may be still used in SQL queries like as they were implemented as PL/SQL functions. However, the data design we propose is limited with respect to a full object oriented design, because, to enable their usage in SQL queries, the operation signatures must have a correspondence with PLSQL specification, adhering to the PLSQL syntax. Moreover, since the ORACLE\textsuperscript{R} layer of the implementation is used for data persistency, a full object-oriented hierarchy would be redundant for most application aims. Therefore, the resulting data types are simplified with respect to the design we provided in Section 3. In particular, we remove the data type distinction between temporal and spatial granularities, and we do not define a data type for each granularity. Nevertheless, the resulting data design suites the requirements of a multi-granular spatio-temporal type system.

In what follows, we first describe the data types and the procedures for representing granularities and granules. Then, we describe the definition of multi-granular spatio-temporal attributes. Finally, we give the specification for multi-granular conversions.

**OR Granularities and Granules** The ORACLE\textsuperscript{R} user defined data types for defining granularities are shown in Fig. 12. With respect to the design discussed in Section 3 in Fig. 3, note that the finer and coarser granularities of a given granularity are represented through their names (the corresponding instances may be retrieved through the static function `getGranularityByName`). Static function `init` sets the relationships among granularities, as described in Section 3, initializing `finer` and `coarser`. Once these variables are initialized, the finer-than relationship may be checked through `isFinerThan` and `isCoarserThan` functions; `glb` retrieves the greatest lower bound of two granularities.

For each granularity defined, a set of static functions is defined for converting different granule representations. In particular, for temporal granularities functions `index2label`, `label2index` and `getExtension` should be defined (see functions defined for granularity `days`). Moreover, functions `geometry2label` and `getExtension` are defined for spatial granularities (cf. `countries` functions). In particular, `getExtension` gives the granule mapping with the temporal and the spatial domain (i.e., ORACLE data type `SDO.Geometry` represents a two-dimensional geometry, and corresponds to the JTS Geometry data type).

An object of type `Granule` (cf. Fig. 13) has an explicit reference to the corresponding granularity, and embeds a label and an optional index representation. `coerce` and `relax` operations return

the coarser and the set of finer granules, respectively, at a different granularity, intersecting a given granule. The implementation such operations for temporal granules may be optimized relying on the inclusion property of the temporal intervals representing the domain extension of temporal granules, a solution that suits well granularities without gaps among granules and holes within the granules [13]. By contrast, for granules with these specific features, ad-hoc implementations are required. Similarly, for spatial granules, we may optimize both relax and coerce operations relying on the inclusion property of their geometries in a given coordinate reference system.

**OR Multi-granular Spatio-temporal Values** In ORACLE®, a straightforward representation of multi-granular values may be obtained through the definition of nested tables, that is tables which are logically nested into the tables of a given database schema, but which are indeed physically stored outside the main data type. For instance, the type for multi-granular value representing the names of the European Head of Government we described above may be defined as:

```sql
CREATE TYPE HOfGov_S AS OBJECT
    ( sgran GRANULE
    , name VARCHAR2(64)
    );

CREATE TYPE HOfGovS Table AS TABLE of HOfGov_S;

CREATE TYPE HOfGov_ST AS OBJECT
    ( tgran GRANULE
    , hOfGov HOfGov_S
    );

CREATE TYPE HOfGovST Table AS TABLE of HOfGov_ST;
```

Given such a specification, type `HOfGov_ST` may be used in a type or table definition as a common column type. As an alternative to nested tables, usual foreign key references may be used to rely the outer type to the tables storing the references for multi-granular temporal and spatial values.

**Multi-granular Conversions** Multi-granular conversions are implemented as ORACLE® Java stored procedures, similarly to granularities and granule functions. For example, we have the following functions for quantitative types:
CREATE TYPE NUMBER_ARRAY AS VARRAY OF NUMBER;
CREATE TYPE VARCHAR_ARRAY AS VARRAY OF VARCHAR2;

CREATE FUNCTION projection_number(value NUMBER_ARRAY) RETURN NUMBER
AS LANGUAGE JAVA
NAME 'Conversion.projection(float[]) return float';
CREATE FUNCTION main_varchar(value VARCHAR_ARRAY) RETURN VARCHAR
AS LANGUAGE JAVA
NAME 'Conversion.main(java.lang.String[]) return java.lang.String';
CREATE FUNCTION split(value NUMBER, num NUMBER) RETURN NUMBER
AS LANGUAGE JAVA
NAME 'Conversion.split(float, float) return float';

The following functions are defined instead for geometric conversions:

CREATE TYPE GEOM_ARRAY AS VARRAY OF SDO_GEOMETRY;

CREATE FUNCTION line_contraction(line SDO_Geometry) RETURN SDO_Geometry
AS LANGUAGE JAVA
NAME 'Conversion.lineContraction(Geometry) return Geometry';
CREATE FUNCTION region_merge(region GEOM_ARRAY) RETURN SDO_Geometry
AS LANGUAGE JAVA
NAME 'Conversion.regionMerge(Geometry[]) return Geometry';
CREATE FUNCTION add_point2region(point SDO_Geometry, region SDO_Geometry) RETURN SDO_Geometry
AS LANGUAGE JAVA
NAME 'Conversion.add_point2region(Geometry,Geometry) return Geometry';

7 Conclusions

In this paper we have discussed design and implementation issues to support spatio-temporal multi-granularity in data model. For a better comprehension of the details, we referred to a multi-granular model we previously defined [24]. Nevertheless the problems we discussed apply to every data model providing a multi-granular 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, multi-granular conversions presented in the paper are specifically designed to prevent data inconsistency and to reduce indeterminacy.
We have also described an existing multi-granular temporal object-oriented prototype implementation addressing some of the issues we discussed, thus demonstrating the solutions we propose are feasible. We also developed an object-relational prototype, which is also described in the paper. The resulting prototype enables to store and query spatio-temporal datasets at different levels of details.

We observe that the discussion we provide in this paper is focussed on the feasibility of multi-granular type system implementation. In particular, we did not discussed in detail the performance of the solutions we proposed, which require further investigation on real world datasets (e.g., the Hurricane Isabel dataset, http://www.tpc.ncep.noaa.gov/2003isabel.shtml).

Furthermore, several open issues remain to face with. In particular, a main challenge is the definition of formal models and type systems for both static and dynamic multi-granular features we described in this paper. Multi-granular type systems are relevant also for programming languages manipulating spatio-temporal objects. For example, when declaring a variable one may have to specify, in addition to the variable type, the spatio-temporal granularity of the variable. Assignments of a value to a variable must then take into account not only the types of the value and the variable, but also their spatio-temporal granularities. Static type checking of programs would then need to be extended by, for example, allowing such an assignment provided that a conversion function be defined for the granularities of the value and the variable, respectively. Consistency properties, such as assuring the correct combination of spatial and temporal type constructors, would also need to be devised and techniques for their analysis be devised.

Finally, to support evolutions tools for the analysis of “evolution” triggers must be supported to detect non-terminating executions and indeterministic executions. Note that such issues have been extensively investigated in the area of active DBMS and no good solutions exist. However because we deal with a specialized domain, that is, the evolution of granularities, effective solutions to these issues could perhaps be found.

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

CREATE OR REPLACE TYPE GRANULARITY AS OBJECT
EXTERNAL NAME 'stExt.Granularity' LANGUAGE Java
USING SQLDATA (
    , granularityType VARCHAR2(32) EXTERNAL NAME 'granulartype'
    , finer granularityArray EXTERNAL NAME 'finerName'
    , coarser granularityArray EXTERNAL NAME 'coarserName'
    , granularityType VARCHAR2(32)
    , labelFormat VARCHAR2(32)
    , MEMBER FUNCTION getFiner RETURN GRANULARITY
    , MEMBER FUNCTION getCoarser RETURN GRANULARITY
    , STATIC FUNCTION getGranularityByName (g VARCHAR2) RETURN GRANULARITY
    , STATIC FUNCTION init
    , STATIC FUNCTION isFinerThan (g1 GRANULARITY, g2 GRANULARITY) RETURN boolean
    , STATIC FUNCTION isCoarserThan (g1 GRANULARITY, g2 GRANULARITY) RETURN boolean
    , STATIC FUNCTION glb (g1 GRANULARITY, g2 GRANULARITY) RETURN GRANULARITY
    , STATIC FUNCTION DAYS_INDEX2LABEL(i NUMBER) RETURN VARCHAR2
    , STATIC FUNCTION DAYS_LABEL2INDEX(l VARCHAR2) RETURN NUMBER
    , STATIC FUNCTION DAYS_GETEXTENSION(i NUMBER) RETURN CLOSEINTERVAL
    , STATIC FUNCTION COUNTRIES_GEOMETRY2LABEL(g GEOMETRY) RETURN VARCHAR
    , STATIC FUNCTION COUNTRIES_GETEXTENSION(l VARCHAR) RETURN SDO_GEOMETRY
    );

CREATE OR REPLACE TYPE GRANULARITY AS OBJECT
EXTERNAL NAME 'stExt.Granularity' LANGUAGE Java
USING SQLDATA (
    , granularityType VARCHAR2(32) EXTERNAL NAME 'granulartype'
    , finer granularityArray EXTERNAL NAME 'finerName'
    , coarser granularityArray EXTERNAL NAME 'coarserName'
    , granularityType VARCHAR2(32)
    , labelFormat VARCHAR2(32)
    , MEMBER FUNCTION getFiner RETURN GRANULARITY
    , MEMBER FUNCTION getCoarser RETURN GRANULARITY
    , STATIC FUNCTION getGranularityByName (g VARCHAR2) RETURN GRANULARITY
    , STATIC FUNCTION init
    , STATIC FUNCTION isFinerThan (g1 GRANULARITY, g2 GRANULARITY) RETURN boolean
    , STATIC FUNCTION isCoarserThan (g1 GRANULARITY, g2 GRANULARITY) RETURN boolean
    , STATIC FUNCTION glb (g1 GRANULARITY, g2 GRANULARITY) RETURN GRANULARITY
    , STATIC FUNCTION DAYS_INDEX2LABEL(i NUMBER) RETURN VARCHAR2
    , STATIC FUNCTION DAYS_LABEL2INDEX(l VARCHAR2) RETURN NUMBER
    , STATIC FUNCTION DAYS_GETEXTENSION(i NUMBER) RETURN CLOSEINTERVAL
    , STATIC FUNCTION COUNTRIES_GEOMETRY2LABEL(g GEOMETRY) RETURN VARCHAR
    , STATIC FUNCTION COUNTRIES_GETEXTENSION(l VARCHAR) RETURN SDO_GEOMETRY
    );

Fig. 12: Oracle Data Type 'Granularity'
CREATE TYPE GRANULE_LABEL_ARRAY AS VARRAY OF VARCHAR2(32);

CREATE OR REPLACE TYPE GRANULE AS OBJECT
    EXTERNAL NAME 'stExt.Granule' LANGUAGE Java
    USING SQLDATA
    ( granularity REF GRANULARITY EXTERNAL NAME 'granularity'
    , index NUMBER EXTERNAL NAME 'index'
    , label VARCHAR2(32) EXTERNAL NAME 'label'
    , MEMBER FUNCTION getGranularity RETURN Granularity
    EXTERNAL NAME 'getGranularity () return stExt.Granularity'
    , MEMBER FUNCTION getLabel RETURN VARCHAR
    EXTERNAL NAME 'getLabel () return java.lang.String'
    , MEMBER FUNCTION getIndex RETURN NUMBER
    EXTERNAL NAME 'getIndex () return long'
    , STATIC FUNCTION relax(g VARCHAR2, g1 Granularity, g2 Granularity) RETURN
        GRANULE_LABEL_ARRAY
        EXTERNAL NAME 'Granule.relax (java.lang.String,stExt.Granularity,stExt.Granularity) return
        java.lang.String[]'
    , STATIC FUNCTION coerce(g VARCHAR2, g1 Granularity, g2 Granularity) RETURN VARCHAR2
        EXTERNAL NAME 'Granule.coerce (java.lang.String,stExt.Granularity,stExt.Granularity) return
        java.lang.String'
    );

Fig. 13: Oracle Data Type 'Granule'