<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Multigranular spatio-temporal models : implementation challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Authors(s)</strong></td>
<td>Camossi, Elena, Bertolotto, Michela, Bertino, Elisa</td>
</tr>
<tr>
<td><strong>Publication date</strong></td>
<td>2008-11-07</td>
</tr>
<tr>
<td><strong>Conference details</strong></td>
<td>Presented at the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS 2008), Irvine, CA, USA, 5-7 November 2008</td>
</tr>
<tr>
<td><strong>Publisher</strong></td>
<td>Association for Computing Machinery (ACM)</td>
</tr>
<tr>
<td><strong>Item record/more information</strong></td>
<td><a href="http://hdl.handle.net/10197/1446">http://hdl.handle.net/10197/1446</a></td>
</tr>
<tr>
<td><strong>Publisher's version (DOI)</strong></td>
<td>10.1145/1463434.1463508</td>
</tr>
</tbody>
</table>

Downloaded 2023-09-25T04:01:56Z

The UCD community has made this article openly available. Please share how this access benefits you. Your story matters! (@ucd_oa)

© Some rights reserved. For more information
Multigranular Spatio-temporal Models: Implementation Challenges

Elena Camossi‡
University College Dublin
elena.camossi@ucd.ie

Michela Bertolotto
University College Dublin
michela.bertolotto@ucd.ie

Elisa Bertino
Purdue University
bertino@cs.purdue.edu

ABSTRACT
Multiple granularities provide an essential support for extracting significant knowledge from spatio-temporal datasets at different levels of details. 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 investigate the implementation issues arising when a data model and a query language are enriched with spatio-temporal multigranularity. We introduce appropriate representations for space and time dimensions, granularities, granules, and multi-granular values. Finally, we discuss how multigranular spatio-temporal conversions affect data usability and how such important property may be guaranteed.

Categories and Subject Descriptors
H.2.1 [Database Management]: Logical Design—Data models; H.2.8 [Database Management]: Database Applications—Spatial databases and GIS

Keywords
Spatio-temporal databases, spatial and temporal granularities, multiresolution, multirepresentation

1. INTRODUCTION
The analysis and monitoring of changes in the spatial configuration of certain geographical areas occurring over a period of time require the capability of representing the datasets with respect to both their spatial layout and their temporal evolution. Moreover, to trace modifications over time 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). Multigranular modelling also offers interesting capabilities also for spatial representation: 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.

Effective approaches to present the data at different granularities facilitate the analysis when 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 usually time consuming. Zoom-in operations may help in refining the mining of specific data subsets. Multigranularity, multiresolution and multiple representation have been investigated first for temporal [3] and spatial data [2] separately, and more recently for spatio-temporal data [5, 8].

Granularities intuitively represent the units of measure of data, and may be defined on both dimensions of the spatio-temporal domain. For each dimension, a connected set of granularities may be defined, with the two sets being orthogonal with respect to each other. The choice of the correct granularity allows the system to store the minimal amount of data, according to the most appropriate level of detail. Since many different granularities exist and no granularity is inherently better than another, a database system should support a wide range of granularities and should allow the user to define his/her own application-specific granularities.

The implementation of spatio-temporal multigranularity poses several interesting challenges. Adequate representations of the spatial and the temporal domains, on which the granularity mappings are given, should be devised. It is equally important to provide efficient representations for granularities, granules and multigranular values in that they affect query execution performance. Furthermore, it is crucial to guarantee that multigranular conversions preserve data semantics and usability and do not introduce errors on converted data.

In this paper we discuss these issues. In particular, we argue that supporting two separate representations for the temporal and spatial domains is crucial. Furthermore, we illustrate the design requirements for granularities and granules, propose a data type design to implement them and suggest how user defined granularities may be specified. Moreover, optimization techniques for the storage and the retrieval of multigranular values are described. Finally, the inconsistencies arising from the application of multigranular conversions are discussed.

The paper is organized as follows. In Section 2 we discuss...
time and space domains; in Section 3 we investigate the representation of granules and granularities; in Section 5 we propose an efficient representation for spatio-temporal values; in Section 4 we investigate the comparison of multi-granular data; and in Section 6 we discuss conversions of multi-granular values at different granularities. Finally, in Section 7 we conclude the paper outlining future research directions.

2. TEMPORAL AND SPATIAL DOMAINS

It is often debated whether the representations of the temporal and the spatial domains should be distinct or not. In most applications, the time domain is linear, discrete and 1-dimensional. The spatial domain, as modeled by the vector representation, is discrete, and may be either 2-, 2.5-, or 3-dimensional.

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 among them.

Due to the inherently different nature of temporal and spatial dimensions, in a multigranular model, the temporal and the spatial domains may be conveniently represented separately, because spatial and temporal granularities reflect the intrinsic characteristics of the domain where they are defined. However, such a choice may depend on the adopted platform. If the multigranular model is built on top of a spatial DBMS, one may rely on the internal support for spatial data such systems already provide. Note that this spatial representation is inherently 2-dimensional.

Moreover, the representation of the temporal domain may rely on the system time, which is accessible by programming languages. In this case, the chronon granularity [7], that is the finest temporal granularity supported by the system, corresponds to the smallest unit of time the programming language could handle.

3. GRANULARITIES AND GRANULES

Various granularities and granules representations may be adopted in a multigranular model. Dyreson et al. [6] 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. Such explicit representation is needed for a small subset of granularities, like the chronon and quantum granularities [7], which are the granularities of the time and the space domains, respectively.

In the model proposed by Camossi et al. [5] spatial and temporal granularities are related by the finer-than relationship [3]. 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. When finer-than holds between two granularities \( G \) and \( H \), we may say that given a granularity \( g \) of the finer granularity \( G \), a granularity 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.

Spatial and temporal granularities thus form two acyclic directed graphs. In these graphs, the nodes represent the granularities, and the edges represent the relationship among the granularities. We denote each of these graphs as granularity graph. Examples of granularities graphs for spatial and temporal granularities are shown in Figure 1. To simplify the illustration, the edges representing instances of finer-than that may be derived by transitivity are not shown in the graphical representation. Note that the finest granularities in the granularity graphs give the measure of the precision applied to multi-granular values. That is, they represent the tolerance on the stored data. Often such granularities correspond to the chronon and quantum granularities.

The granularities in Figure 1 may be specified by implementing minutes and \( \mu \text{ms} \) as direct mappings with the temporal and the spatial domains, respectively. Other granularities, that may be regularly subdivided in terms of minutes and \( \mu \text{ms} \), may conveniently use these mappings. Straightforward definitions may be obtained if relationships such as partition, groups-uniformly-into and group-periodically-into [3] hold among the granularities. For instance, months groups-uniformly-into years, because one year always corresponds to twelve months.

For granularities which may not be regularly represented, other relationships, like finer-than, or equivalently coarser-than, may be used. To map a granularity \( G \), we have two possibilities: 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 Figure 1, because the instances of finer-than that can be derived 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 Figure 2, 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 the different granules representations (i.e., label, index, and physical representation). In particular, the physical representation of each granularity granule in the corresponding domain as anchored temporal interval [7] or as a set of vector features is given by the operation `getExtension(index int)`. For temporal granularities, the label format of granules (e.g., “mm/dd/yyyy”) 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 2007`, through this operation we may retrieve the finer granules at granularity `months` that represent the months of year `2007`. Similarly, `getCoarser()` returns the coarser granule of a specified granularity `H` that includes `g`.

4. MULTIGRANULAR COMPARISON

Expressing the relationships among different granularities is important not only for obtaining efficient representations, but it is essential 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 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.

In this case, it is sufficient to convert one of the two values at the other granularity with a suitable granularity conversion, because the two granularities are directly related by finer-than (i.e., `provinces` is finer-than `countries`). For example, the value at granularity `countries` may be split among the different provinces of each country. But what if the two granularities were, for example, `feet` and `kms` of Figure 1? In this case a third granularity, for example `µms`, which is finer-than both granularities, may be used.

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 hoops in the granularity graph. 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`. 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 among the GLB or the LUB is essential to guarantee the comparison of every granular value with another expressed at a different granularity (but with the same inner type). Dyreson et al. [6] 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.

5. MULTIGRANULAR VALUES

Temporal and spatio-temporal values may be stored in structures that improve retrieval efficiency. For instance, temporal values may be ordered according to the order of the time domain. A more efficient representation may be obtained using B-Tree and B-Tree\(^+\) structures.

The time efficiency in query answering may also improve considering auxiliary structures that compute spatio-temporal aggregates of multigranular values at coarser granularities. Beyond indexing, such auxiliary structures may be used directly to answer queries involving aggregates [9].

For spatial data, one should consider to include in the multigranular value the vector representation of spatial granules. This representation, even in a simplified form, directly indexed through a R-Tree structure, may improve the access to spatial data when conventional GIS queries are performed.

By contrast, to improve the space efficiency 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.

For the same purpose, we may enrich the data type specification with temporal semantic assumptions [4]. 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.

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.

6. MULTIGRANULAR CONVERSIONS

An essential requirement for data conversion is that it preserves data consistency and usability. In this respect, a multigranular model additional issues arise because of the multigranular representation.

Databases maintaining geographical representations of the same area at different resolutions should guarantee topological consistency between the represented entities: a query evaluation at a coarser level must give a result consistent with that obtained from the evaluation at the more detailed level. If the applied operators do not assure the generation of consistent maps, a post-processing check must be performed.

Also non-geometric granularity conversions should be carefully applied. Spatial conversions to coarser granularities that involve aggregates, for example, are affected by anomalies like the `modifiable area unit problem` (MAUP) and the `ecological fallacy` [1]. MAUP occurs when the geographical units considered in an aggregation are arbitrary and consequently modifiable according to the purpose of the operation. By contrast, the ecological fallacy arises when aggre-
gated data are used for making inferences about individuals. Since such anomalies are in turn due to how aggregations are performed and interpreted, the same problems may occur also for temporal data.

Moreover, depending on the relationships among granularities, conversions of data at different granularities (namely, granularity conversions [5]) may be defined either as total or as partial functions. For instance, in a multigranular system supporting the finer-than relationship, conversions to coarser granularities might be defined as total functions, whereas conversions to finer granularities would be partial functions. Given two granularities $G$ and $H$, such that $G$ is finer-than $H$, a (unique) $H$-granule that includes each $G$-granule always exists. Then, a conversion function that maps $G$-values into $H$-values is always defined (supposing it is defined on the inner domain of the multigranular values). By contrast, the inverse condition may not hold, because given a coarser $H$-granule, a finer $G$-granule included in the $H$-granule is not guaranteed to exist. Such a condition would be guaranteed by the groups-into relationship [3] (or by partitions, that holds whenever both finer-than and groups-into hold [3]).

However, even when these functions are defined, conversions to finer granularities always result in undetermined values, as discussed by Katri et al. [8]. This is particularly evident for geometric conversions. The conversion of regions to their barycenters, for instance, is usually an appropriate choice for representing them as points at a coarser granularity. By contrast, when converting points to regions at a finer level of detail, the choice of the region to represent is 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 be able to infer from this value the exact value of rainfall in a particular neighbourhood of Rome, at 12:00 AM.

Furthermore, when implementing multigranular conversions we have to take into account that conversions are not always invertible. Intuitively, when converting a multigranular value to a different granularity, and then performing the inverse conversion, we would expect that the original value results. Unfortunately, when converting from a finer to a coarser granularity, we lose some details that we cannot usually recover by applying the inverse conversion to the finer granularity. By contrast, when converting from a coarser to a finer granularity, we introduce some details that we should be able to forget, if we are no more interested in them, then we may re-obtain the original value. This is equivalent to what happens with the cast operation in the object-oriented context. When we cast up an object to a superclass, and then recast it down to its original class, we are not able to re-obtain the details we forgot with the cast up. By contrast, if we cast down an object to a subclass, we are able to re-obtain the original object, when recasting up to its original class.

Finally, the application of granularity conversions may be expensive in term of computational complexity, in particular for geometric conversions whenever they are applied to extensive geographic areas. Rewriting algebraic techniques such as those applied by database query engines when preparing query execution plans could improve the performances of data access. These techniques should give precedence to the execution of target list, to reduce the amount of data interested by conversions, as well as to the execution of temporal conversions of spatio-temporal data. Moreover, they should take advantage of auxiliary structures like those described above, that would enable fast responses to queries involving temporal and spatial aggregates.

7. CONCLUSIONS

In this paper we have discussed the implementation issues of spatio-temporal multigranularity. Among such issues, the ones related to the efficient representation of granules, granularities and spatio-temporal values are particularly relevant.

A prototype implementation for a multigranular temporal model in the object-oriented context which addresses some of the issues we discussed has already been developed. We plan porting such implementation on top of an object-relational DBMS, and extending it to the spatio-temporal domain. The resulting prototype would enable to store and query spatio-temporal datasets at different levels of details. Thus we would be able to test its performances on real world datasets such as the Hurricane Isabel dataset (http://www.tpc.ncep.noaa.gov/2003isabel.shtml).

8. REFERENCES