An Object-Relational Approach to the Representation of Multi-granular Spatio-Temporal Data

Elisa Bertino[1], Dolores Cuadra[2], and Paloma Martínez[2]

[1] CERIAS and CS Department, Purdue University, Recitation Building,
656 Oval Drive, West Lafayette, IN 47907-2086, USA.
bertino@cerias.purdue.edu
[2] CS Department, Carlos III University, Avd. Universidad 30, 28911 Leganés, Spain
Fax: 34-91-6249430
dcuadra@inf.uc3m.es, pmf@inf.uc3m.es

Abstract. The notion of spatio-temporal multi-granularity is fundamental when modeling objects in GIS applications in that it supports the representation of the temporal evolutions of these objects. Concepts and issues in multi-granular spatio-temporal representations have been widely investigated by the research community. However, despite the large number of theoretical investigations, no comprehensive approaches, have been proposed dealing with the representation of multi-granular spatio-temporal objects in commercially available DBMSs. The goal of the work that we report in this paper is to address this gap. To achieve it, the paper first introduces an object-relational model based on OpenGis specifications described in SQL3. Several extensions are developed in order to improve the semantics and behavior for spatio-temporal data types introducing an approach to represent the temporal dimension in this model and the multi-representation of spatio-temporal granularities.

1 Introduction

Since 1970 when Codd [5] proposed the relational model, database system technology has introduced several important changes in the way data are stored and managed. The relational data model has had a great impact on commercial products, mainly because of the development of the SQL language, which included several additional features with respect to those specified by the theoretical definition of the relational model. Since its initial definition, SQL has been widely extended [9]. A relevant set of extensions has dealt with the introduction of object modeling capabilities [21], resulting in the notion of object-relational data model. Such a model combines the simplicity and the power of SQL the ability of describing new data types with their associated operations, typical of the object-oriented approach. The object-relational data model is thus a powerful model combining the best aspects of two different approaches.

The SQL3 standard [10] is the reference language for the object-relational model. It has been defined by extending the previous SQL92 standard with the ability to modify, retrieve and define the data types needed to represent a large variety of application domains. Examples of those extensions include: XML, MULTISET (like the ARRAY data type, without implicit order), BIGINT and others.

O. Pastor and J. Falcão e Cunha (Eds.): CAiSE 2005, LNCS 3520, pp. 119 – 134, 2005.

An important class of applications is represented by spatial and GIS applications. These applications are relevant in a number of different domains, such as transportation, urban planning, homeland security, and environmental protection. Spatial data management is often combined with temporal data management, because in many cases one needs to record the temporal evolution of spatially-related entities. The resulting data models are thus termed spatio-temporal data models. One of the most crucial issues when dealing with spatio-temporal databases is the management of the information concerning moving objects in a spatial context. Such an issue represents an important requirement in several application domains, like air traffic control and habitat control of endangered species and so on. So far, the GIS systems have handled the spatial and the non-spatial data separately, which increases the complexity of maintaining data integrity. The use of an object-relational database system to manage this kind of data represents a good alternative, in particular because such a system is able to homogeneously and efficiently manage user-defined information, and to improve integrity for data of different nature.

Most of the spatial-temporal data approaches proposed so far do not exploit the powerful modeling and management features that are provided by recent versions of commercially available DBMSs. Another main limitation of current approaches is that they do not support multiple granularities in the representation of spatio-temporal data. Multiple granularities, defined as a set of measure units for space and time, are crucial in facilitating the management of information for applications such as air traffic control, meteorological forecast and so forth [3], [17]. The goal of the work reported in this paper is to address such limitations by developing an object-relational approach to the management of spatio-temporal data supporting multiple granularities for both space and time.

In particular, in the paper we propose a temporal extension expressed in the SQL3 standard, specifically tailored to the OpenGis specifications [29]. Our start point is the object-relational model and meta-model that verifies the OpenGis specifications from which we develop an extension of the spatial and non-spatial data types defining a new data type composed the two parts; the first part is the object value and the second the valid time for this value. This extension provides the support required to model multiple spatio-temporal granularities and tools to operate on objects with different granularities in order to address their integration and inter-operability.

The remainder of this paper is organized as follows. Next section gives an overview of related work dealing with the spatial and temporal representation in the object-relational model. Section 3 introduces the notions of spatial and temporal granularity and their basic properties. Section 4 describes the OpenGis specifications in SQL3 [29] and presents our extension to supporting multiple spatial granularities. Section 5 then extends the model and meta-model developed in Section 4 by introducing time as an extension of the SQL3 data types. The last section concludes the paper and outlines future work.

2 Related Works

Initial approaches, aimed at supporting advanced data management applications, have recognized that spatial information [15], [24] and temporal information [25], [22],

[23], [19] are both crucial. However, those early approaches have dealt with space and time representations separately. Most recent proposals have proposed integrated approaches able to model both spatial and temporal aspects of data objects.

A first relevant approach has been developed by Guting et al. [16]. The approach supports the change of the position or extension for geometry through the use of abstract data type definition capabilities. It develops a set of constructors and query operators in an abstract model thus giving a compact and uniform vision for every data type. In the proposed approach [16], some data types, namely Integer, Boolean, Spatial, can be transformed in a temporal data type. Such an abstract model has been then transformed in a discrete model [14], closer to the implementation but more restricted with respect to the abstract model. The discrete model represents the object values that have a temporal dimension through the use of snapshots. This method is thus referred to as *sliced representation*. As part of their work, Guting et al. [14] have shown how the sliced representation can be mapped onto relational data structures such as records and arrays. However, they do not have specifically addressed how to map their abstract model onto the SQL3 standard.

Chen and Zaniolo [6], based on the spatial-temporal representation model proposed by Worboys [26], propose a number of SQL3 extensions aiming at supporting advanced queries. Unlike the previous proposals, they adopt a point-based approach to model the time dimension and queries are expressed through user-defined aggregate functions. Such an approach to handle time for spatio-temporal databases represented in a relational framework, it is very simple and minimizes SQL3 extensions in comparison with the complex functions one has to apply when using based-intervals approaches [13], [19]. Such an approach has then been architected by solutions supporting efficient storage [7]. Finally, a more recent paper [8] by the same authors describes how to implement these functions in ATLAS [28] in order to model specific application domains. This approach keeps the SQL philosophy; therefore it is easy to use for developers and users. A major drawback of this approach is that it is not able to support multiple temporal attributes with different granularities in the same relation.

Another approach very close to implementation has been proposed by Lee [18]. The spatial object history is modeled through the creation of relations. The developed framework is based on the creation of some special purpose relations, called *dimensions*, storing the current values of the object geometry. Besides those dimension relations, the approach requires the introduction of two additional relations keeping historical information. The approach by Lee also includes a set of macro operators that are used to execute queries. The macro operators are applied to spatial data, temporal data, and spatio-temporal data. Also, an aggregate operator is defined to manage the history of a spatio-temporal data. The main drawback of this approach is that it requires executing join operations among the dimension tables in order to produce complete spatial information. The use and maintenance of data under such an approach is thus quite complicated.

An extensive analysis of the most important approaches to represent spatio-temporal in the object-relational model has been carried out by Erwig et al. [12]. This paper presents an extensive comparison between two approaches to represent the temporal dimension in a spatio-object-relational model. The first approach is based on extending each relation storing spatial data objects, with an additional column

representing temporal information. The column essentially records the temporal validity of row values. The second approach exploits the expressive power of the object-relational model. It is based on the definition of an abstract data type (ADT) joining space and time on the same column. The second approach better represents the semantics of the original data and allows one to create relations containing time information at column level.

Compared to previous work, the main contributions of our work are summarized as follows. First, the spatial representation we adopt is based in the proposal of OpenGis specifications described through SQL3 [29]. The above approaches, by contrast, adopt much simpler spatial data models not supporting OpenGis. Second, we provide support for multiple spatial granularities. Third, we use an abstract data type approach to represent and manage the temporal dimension; thus our approach is more flexible than other approaches and supports a finer degree of control over temporal behavior. Our approach facilitates the formulation of queries in SQL and provides a representation which is more intuitive and easy to use. Finally, we propose solutions to maintain multiple granularities, in both space and time, and provide conversion functions to map data representations among different granularities. None of the above described approaches provides such functions.

3 Preliminaries: Multi-granular Spatio-Temporal Representation

In this section, we introduce the relevant notions underlying our approach to multiple granularities in both time and space aspects. From an informal view point, we can think of a temporal or spatial granularity as a discrete partition of time or space. Different partitions may be defined, thus resulting in multiple granularities.

The temporal granularity has been defined as the partitioning of a temporal domain in groups of elements ordered through an index set, where each group is perceived as an indivisible unit (a *granule*) [2]. The granularities set is denoted by G_T and its elements are related by the relationship *finer_than.* A granularity *S* is *finer_than R* if for each index i a index $j \in R$ exists such that $S(i) \subseteq R(j)$, where S(i) denotes the granule belonging to the i index position. This relation is denoted by $S \preceq_T R$. *S*emantically, when we define a granularity for an object we specify the time instants at which the object's values are relevant to the specific application domain. Days, months and years are several kinds of temporal granularity; days are finer than months and months are finer than years. Operations and comparisons between objects at different temporal granularity require the use of *conversion functions*. These functions change the temporal properties of an object from a finer granularity to a coarser granularity and we can compose a macro function with the function composition. Conversion functions can be like the ones shown in the Table 1, or aggregation functions available in SQL [4] or created according to the application's semantics.

The spatial granularity is defined as the unit of measure in a spatial reference system. It thus represents the unit according to which spatial properties, like the area, are measured. The set of the spatial granularity is denoted by G_S. The elements of G_S would be meters, kilometers, grades and others. Each one of those elements must be defined with respect to a spatial reference system. In G_S, the relation *finer_than* is similar to the temporal granularity given before and it is denoted by $M \preceq_S N$.

Table 1. Temporal and spatial conversion functions

Proj (index)	It returns, for each granule in the coarser granularity, the value corresponding to the granule of position index at the finer granularity
First, Last	First and last index in the Proj (index) function
Main	It returns, for each granule in the coarser granularity, the value which appears most frequently in the included granules at the finer granularity
All	It returns, for each granule in the coarser granularity, the value which always appears in the included granules at the finer granularity if this value exists, the null value otherwise

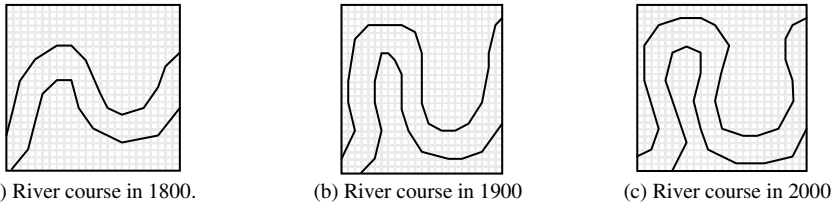
Contract functions	
l_contr	It contracts an open line, endpoints included, to a point
r_contr	It contracts a simple connect region and its bounding to a point
r_thinning	It reduces a region and its bounding lines to a line
Merge functions	
l_merge	It merges two lines sharing an endpoint into to single line
r_merge	It merges two regions sharing a boundary line into a single region
Absorption operations	
P_abs	It eliminates (abstracts) an isolated point inside a region
l_abs	It eliminates a line inside a region

Possible conversion functions are shown in Table 1; we refer the reader to [1] for additional details. The application of these conversion functions guarantees the topology consistency.

Suppose we were interested in studying the evolution of a river's course. According to this specification, we can define a time unit as century thus obtaining the representation shown in the Fig. 1. Semantically speaking, the temporal and spatial granularity integration provides, on the one hand, the timestamp at which a spatial object is observed and, on the other hand, establishes the spatial measure within a reference system.

Our approach represents spatial-temporal multiple granularities considering the spatial representation of an object with respect to one temporal granule. Therefore, we must specify the observation unit for both space and time. These objects are called *moving objects* [16]. The Fig. 1 shows the time variation in the river course with centuries as temporal granularity.

The temporal and spatial multi-representation is built when executing user queries by using the conversion functions defined for each application depending on the domain. In the next section, we show how to specify the conversion functions



(a) River course in 1800. (b) River course in 1900 (c) River course in 2000

Fig. 1. River course evolution

composition and their composition order what we denote as *composition sequence*. The composition of functions is strongly domain-dependent and it is very important to store it with the scheme definition to provide more semantics.

4 Spatial Framework: Modeling OpenGis Specifications with SQL3

In this section, we describe the OpenGis specifications in SQL3 considering the abstract data types to define the Geometry type besides the meta-scheme definition necessary to manage the spatial data in an object-relational database.

Our approach begins with the framework developed in [29] and provides an extension through the new meta-information and SQL3 functions to support the spatial granularity. The proposal is focused on this spatial framework for two reasons; first, the representation agrees with the OpenGis standard and, second, it is specified by another standard, the SQL3 language, which provides an easy implementation in any object-relational DBMS.

4.1 Spatial Data Description

We describe OpenGis from two different viewpoints. The first describes the geometry data types, their properties and functions related to the spatial relationships among geometric objects. The second describes the meta-scheme which gives support to the geometry data type. The meta-scheme is similar to a data dictionary. It is composed of relations describing the spatial domain. The meta-schema in [29] describes the spatial domain as geometric objects in $\Re^2$ space.

The relations belonging to the meta-schema cannot be modified by the user. The relation definition is fixed in the meta-schema and modifications to it are achieved through the use of views. We consider a view as a subset of meta-information that can be modified by the user.

In the spatial meta-scheme presented in [29], framework of our proposal, when we define a geometric object in an object-relational scheme, the meta-scheme stores the information concerning its dimension, its geometry type (if it is a point, a line, a polygon and so on) and the spatial reference system over it is defined. This information allows, for example, one to check whether the relationships among geometric objects can be determined. The spatial relationship functions can just apply to objects with the same granularity and spatial reference.

We introduce a notation to reference this meta-information. We denote by SOR the spatio-object-relational model and by SOR-M the meta-scheme over SOR defined in [29].

Definition 1: The SOR-M = $\{RM_1,\ldots, RM_n\}$ is a set of relations that describe the spatial domain. We define V_s as a set of views, $V_s = \{V_{s1}, \ldots, V_{si}\}$ such that $V_{sj} = \texttt{Ops}\ (RM_k,..,RM_j)$ where `Ops` is a macro-operator composed of relational algebra operators. □

The meta-information held in V_s can be modified by user. We reference to SOR-M as the set of views (V_s).

Definition 2: The Geometry data type (G) is defined as an abstract data type whose features[1] are described in two important views, *Geometry_Columns* and *Spatial_Ref_Sys*, such that *Geometry_Columns*, *Spatial_Ref_Sys* $\in V_s$. □

The geometry type in *SOR* model is represented using a discrete model because *SOR* is a model close to the implementation. The spatial representation is stored as a set of vertexes that are interpreted through the linear interpolation between them.

According to Fig. 2, the *Geometry* type has a hierarchical structure that is specialized into Point, Curve, Surface and Geometry Collection subtypes. Fig. 2 is based on Geometry Model specified in the OpenGis Abstract Specification restricted to 0, 1, and 2 dimensions for the collection types named Multipoint, MultiLineString and MultiPolygon. Therefore, Geometry, Curve, Surface, Multicurve and Multisurface are introduced as abstract data types. The subtypes of Geometry are restricted to 0, 1, and 2 dimensions in a coordinate space ($\Re^2$).

The geometry type includes basic functions that retrieve information about spatial properties and methods for testing the spatial relationships and analyzing geometric objects; besides for each type specific functions are defined. In this paper, we do not show these functions and methods because they are specified in [29]. We focus our paper on how we can define a geometry object in a relation and how to represent its spatial properties in SOR-M.

Definition 3: Let R (A_1:D_1,…,A_n:D_n) be a relation where A_i is a column defined over a domain D_i. R is a *feature relation* if $\exists$ A_i ($i \in \{1,..,n\}$) such that $A_i \in G$. A_i is called *geometry or spatial column*. □

When we define a *feature relation*, the spatial attribute definition shows just the geometry type. In the SOR model, every geometry column stores its features in the *Geometry_Columns* and it is referred to *Spatial Reference System (SRS)*. A *Spatial Reference System* (also called *Coordinate System*) is a way to assign coordinates to a location and to establish relationships between sets of such coordinates. It enables the interpretation of a set of coordinates as a representation of a position in a real world space. Any spatial data in SOR has a coordinate system associated with it through the *Spatial_Ref_Sys* view.

To define the geometry column features we introduce procedures to insert these features in SOR-M in order to make the update operations easy for the users. The *Spatial_Ref_Sys* view has defined SRS by default but the user can define other SRSs. For this reason, we create two SQL 3 procedures.

Definition 4: Let R (A_1:D_1,…,A_n:D_n) be a feature relation. We define geometry features $\forall$ $A_i \in G$ through the following procedures:

- *SP (R, A_i, Dimension_Number, DRS_ID)* is a SQL3 procedure which inserts the dimension number and the identifier DRS (DRS_ID) in the *Geometry_Columns* view when A_i is defined in a SRS by default (DRS)
- *SR (SRID, SRText)* is a SQL3 procedure to create a new spatial reference system. It inserts the identifier reference system and the specification textual in *Spatial_Ref_Sys.* The *SRText* column describes a standard specification defined in [29]. □

[1] We use the term feature to indicate an attribute of an abstract data type.

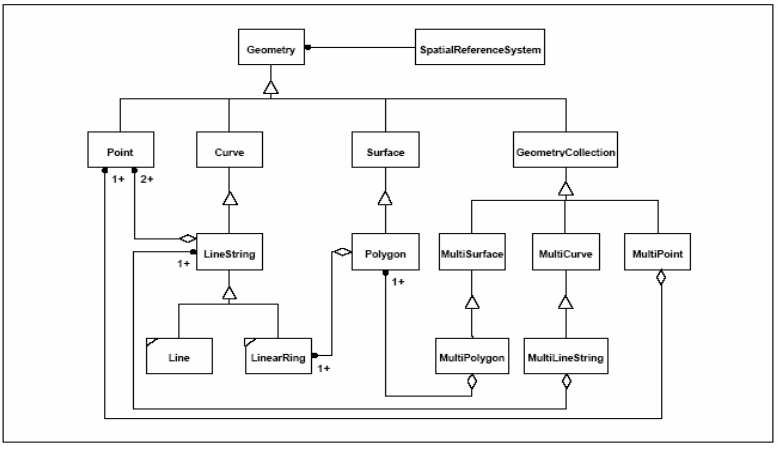


Fig. 2. Geometry description in ODMG [21]

Example 1: Let *City* be a relation defined with two attributes, *Name* of Varchar type and *Extension* of Polygon type. We define the *City feature relation* as:

```
City(Name:varchar(60),Extension:Polygon);
```

The Extension column is defined as a Polygon subtype. If we want to define the *Extension* column in City as a polygon with two dimensions at meters in a spatial reference system by default (DRS_ID = 14), we must specify them with the *SP* procedure to insert these features in the *Geometry_Columns* view:

```
SP('City', 'Extension', 2, 14);
```

The *SP* signature indicates with the number 2 that *Extension* is a two-dimensional geometry object having 14 as identifier for the spatial reference system (DRS_ID). This identifier references a row of *Spatial_Ref_Sys* view. □

4.2 Spatial Granularity Description and Proposed Extensions by *SOR* Model

The spatial granularity is a measure unit in relation to a *Spatial Reference System (SRS)*. In the previous spatial framework, the spatial granularity is referred to the *SRS* chose. We can distinguish between a spatial granularity referred to a default reference system (*DRS)* and a new spatial reference. The DRS supported in the OpenGis specifications appears in [29] and the granularity is always defined according to the default unit. The definition of a new reference system involves the insertion of a new row in *Spatial_Ref_Sys* view. In order to provide an easy interaction with the *SOR-M* to define a spatial granularity different with respect to the Unit parameter defined over DRS we create a new granularity procedure.

Definition 5: Let *SOR* be a model, we define *SG* procedure such that *SG(Table_Name, Geometry_Name, DRS_ID, Granularity, Factor)* inserts the spatial granularity of the *Geometry_Name* column in *Table_Name* relation concerning a DRS

which is identified by DRS_ID. The *Factor* parameter indicates the conversion rate relative to the unit by default supported by DRS. □

This procedure defines the spatial granularity in a geometry column when the DRS has a unit different from the default one. The SG inserts a new row in the *Spatial_Ref_Sys* view.

Example 2: Consider example 1. Suppose that the system, defined as DRS_ID = 14 with 'Meters' as default unit, is a DRS. To define the Extension granularity in hectometers, we should use the SG() procedure with the following parameters.

```
SG ('City', 'Extension', 14, 'Hms', 100.0)
```

The last parameter indicates the mapping between meters and hectometers. □

Another important aspect is to provide functions to compare geometries with different spatial granularities. Two geometries with different spatial granularities can be compared if they have associated conversion functions that modify their granularity while maintaining their topological properties. Thus, we must extend the set of functions defined in [29] to support the conversion functions and to create a view in the *SOR-M* recording how to apply them.

We extend the *SOR* model by adding the conversion functions shown in Table 1 as SQL3 functions (Table 3). *SOR-M* is extended to also include a view called *S_Coarser* that specifies what conversion functions can be applied to a geometry column.

Definition 6: Let $R(A_1:D_1, .., A_n:D_n)$ be a *feature relation*, $\forall A_i\ (i \in \{1,..,n\})\ A_i \in G$, we can define the conversion functions for A_i by modifying *S_Coarser* view with the *SC* procedure, defined as follows:

SC (Table_Name, Geometry_Column, Order, Sg, Eg, Operator_Name, Condition)

The *Order* attribute indicates the application sequence of the operator the name of which is stored in *Name_Operator*. This attribute is needed when a spatial data has conversion functions concatenation to specify the change to several coarser granularities. The *Sg* and *Eg* record the initial and final granularities necessary to apply the conversion function and *Condition* is an expression that indicates what objects are affected for this operator. □

Definition 7: Let R(A1:D1, .., An:Dn) be a *feature relation* in the SOR model. We define *SGranularity (Name_Attribute, Spatial_Ganularity): Geometry* such that SGranularity() is a SQL3 function transforming the *Name_Attribute* spatial column values to values in the indicated granularity by *Spatial_Ganularity* according to the specification in *S_Coarser* view**.** □

Example 3: Following with the example 2, we can specify that the *Extension* attribute can change to a coarser granularity through application of the r_thinning operator. The conversion function for this attribute is specified in the meta-scheme through the SC procedure the signature of which is the following:

```
SC ('City','Extension',1,'Hms','Kms','r_thinning',ALL);
```

As such signature specification shows, the Extension column can change the hectometers to kilometers granularity by applying the *r_thinning* function over all

Extension geometry. In this case, we are indicating that *Extension* values can be represented at kilometers as well. The *Order* attribute records the *1* value although in this case is not significant because we just apply one conversion operator.

The city names and theirs extensions in kilometers can be queried as:

```
Select SGranularity (Extension, Kms)
From City;
```

Table 2. Functions of spatial conversions

Contract functions	l_contr (G:Line): Point
	r_contr (G: Polygon): Point
	r_thinning (G: Polygon): Line
Merge functions	l_merge (L1: Line, L2: Line): Line
	r_merge (G1:Polygon, G2:Polygon):Polygon
Absorption operations	p_abs (G:Polygon, P:Point):Polygon
	l_abs (G:Polygon, L:Line):Polygon

The extended *SOR* model provides two important semantics restrictions. First, the spatial granularity definition through the SR procedure defines a new spatial reference system, and the SG procedure changes the granularity in a default spatial reference system. The second is the facility supporting the representation of a geometry type in several spatial granularities using the conversion functions.

5 Temporal Representation in the *SOR* Model

In the object relational model, the temporal representation can use a point-based approach or an interval-point approach and it could affect the relation's tuples, called tuple level [6], [22], [23], [8] or to the relation attribute, attribute level [13], [14]. As we discussed in the Section 2, the point-based approach avoids the coalescing problems and provides a simpler vision of spatial-temporal systems. For this reason, our proposal represents the time with a point-based approach. Furthermore, it focuses on attribute level because we believe that is closer to the real world and supports the definition of several temporal attributes in the same relation.

We consider a temporal granularity as *foreseeable* or *unforeseeable.* A foreseeable granularity describes periodical phenomena. Granules are calculated through a time function and a temporal seed. The foreseeable temporal granularity can be calculated by applying the time function to the seed, e.g. if the seed is *'12-1-2003',* and the time function is to add one day, the *'13-2-2003'* records the next timestamp. The temporal seed can be the first value observed or a value chosen by the user. The unforeseeable granularity is described by random phenomena. For example, if we want to represent the price changes of shares, we could define the unforeseeable granularity in that we do not know the time units when the share could change. This temporal granularity classification allows us to differentiate among several application domains and will be

represented in our approach by the definition of a temporal reference system within the Gregorian calendar.

The proposal will carry out a temporal representation with the definition of abstract data types provided by the standard SQL3 inside the spatial framework explained before. The new data types will take into account that the temporal dimension can be expressed as a function over time for a spatial or non-spatial attribute. This function will be provided by the DBMS or/and customized by the user to collect the information for a certain domain.

In the next section, we explain the approach we adopt to represent temporal attributes in the *SOR* model; specifically; we focus on the presentation of the *foreseeable object moving* representation.

5.1 Temporal Attributes

Until now, we can classify the relation columns as *atomic* or *spatial* depending on the data type over which they are defined. The atomic columns are defined over a predefined SQL3 data type and the spatial columns over *Geometry* type. Both types could be grouped and denoted as non-temporal columns (*NTD*). The *temporal column* will be considered as an extension of the atomic and spatial column where the changes produced along the time in those columns will be reflected. At present, the spatial o non-spatial representation shows the current values of the application domain. In this section, we describe how the time evolution is presented.

Let *SOR* be the spatial relational model. The *spatio-temporal object relational* model (*TSOR)* is defined as an extension *of SOR* where the data types are specialized by adding a time attribute. This model is supported by the meta-scheme *TSOR-M.*

Definition 8: The *TSOR-M = {TM$_1$,…, TM$_n$}* is a set of relations that describe the temporal domain. We define V_T as a set of views, $V_T = \{V_{T1}, \ldots, V_{Ti}\}$ such that $V_{Tj} = \texttt{Opt} = (TM_k,..,TM_j)$ where `Opt` is a macro-operator composed of relational algebra operators. □

By using such model extension, we can define the temporal data type as follows.

Definition 9: Let α be a data type belonging to NTD, the temporal data type based on α is denoted by *Tα* and it is defined as a pair *(Tvalue, Tgranule)* where the *Tvalue* is a valid value over α data type in the *Tgranule* time instant, such that Tgranule ∈ S where S ∈ G_T and S is the temporal granularity for Tα. □

In order to support attribute evolution we use the collection type defined in SQL 3.

Definition 10: Let $R(A_1:D_1, .., A_n:D_n)$ be a relation R. R is a *temporal relation* if ∃ A_i ($i \in \{1,..,n\}$) such that $A_i \in$ Collection_Tα and Collection_Tα records a collection of ordered pairs of Tα type. Each $A_i \in$ Collection_Tα is denoted as *temporal column.*

A particular case is when α is the G spatial data type. R is a *temporal -feature* and ∀ A_i ($i \in \{1,..,n\}$) $A_i \in$ Collection_TG, A_i is denoted as *moving column*. □

The temporal column definition in a relation requires that the meta-scheme, the *Temporal_Columns* view to be exact, be updated in order to maintain the data consistent.

Therefore, a temporal granularity is defined for each temporal column. The granularity is defined by using a SQL3 procedure, like the spatial granularity procedure.

Definition 11: Let *TSOR* model and $R(A_1:D_1, .., A_n:D_n)$ a temporal relation. The temporal granularity $\forall A_i (i \in \{1,..,n\})$ such that $A_i \in$ Collection_Tα is defined by *TG*. *TG* is a SQL3 procedure which modifies the *Temporal_Columns* view in *TSOR_M* inserting a new row with the following specifications*:*

TG (Table_Name, Temporal_Column, Unit, Granularity, Seed, Rate, Function)

The *TG* signature specifies various information. *Unit* is the temporal domain that can get the values Y (year), M (month), D (day), H (hour), Mi (minutes) and S (second), all of them belonging to SQL3 DATETIME. *Granularity* describes the chosen temporal unit and *Rate* denotes the conversion factor to obtain the next granule from *Seed*. When the time function is different from an arithmetic succession of the time, the *Function* attribute stores the user-defined function name. □

According to this definition, we have covered the *foreseeable temporal* granularities to represent applications where certain phenomenon is observed at periodic timestamp.

Moreover, we create the conversion functions described in Table 1 as SQL3 functions to facilitate the integration and comparison among attributes with different temporal granularities. The *SOR-M* is extended by including a view denoting *T_Coarser* that specify what conversion functions are applied to each temporal column.

Definition 12: Let $R(A_1:D_1, .., A_n:D_n)$ be a temporal relation, $\forall A_i (i \in \{1,..,n\})$ such what $A_i \in$ Collection_Tα, we can define the conversion specification for A_i modifying *T_Coarser* view with the *TG* procedure. The *TG* signature is*:*

TG (Table_Name, Temporal_Column, Order, St, Et, Operator_Name, Condition)

where, the *Order* attribute indicates the application sequence of the operator whose name is stored in *Name_Operator*. The *St* and *Et* record the initial and final granularity parameters necessary to apply the conversion function and *Condition* is an expression that indicates which objects are affected by this operator. □

Definition 13: Let $R (A_1:D_1, .., A_n:D_n)$ be a temporal relation in the *TSOR* model. We define *TGranularity (Name_Attribute, Temporal_Ganularity):Collection_ T*α as a SQL3 functions that transform the temporal column values, *Name_Attribute* $\in$ Tα *,* to values in the indicated temporal granularity, *Temporal_Ganularity,* according to *T_Coarser* view. □

Therefore, the *TG()* procedure and *TGranularity()* contribute to extend the *TSOR* model.

Example 4: In this example, City is a temporal feature relation (see Fig. 3) because the Extension is defined as a moving column.

```
City(Name:varchar(60),Extension:Collection_Tpolygon)
TG('City', 'Extension', 'Y', 'Century', 100);
SG('City', 'Extension', 14, 'Hms', 100.0);
```

We use the SG and TG procedures to define the spatio-temporal granularity in Extension and the SC operator to specify the spatial conversion function that can be applied.

```
SC ('City','Extension',1,'Hms', 'Kms', 'r_thinning',
ALL);
```

To determine the New York boundary at year 1900, we specify the following query using *SGranularity()* operator.

```
Select SGranularity (c.Extension[].Tvalues, 'Kms')From
City c Where c.Name='New York' and
c.Extension[].Tgranule = '1900';
```

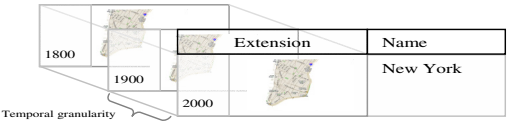


Fig. 3. Example to illustrate the City temporal feature relation

The SGranularity() realizes a query to *S_Coarser* view description to calculate the correct topology by the geometry. The *Extension* column changes from hectometers to kilometers granularity applying *r_thinning* function. □

Each temporal attribute has defined a temporal granularity and the relation is the context that joins them. For this reason, the database design aspects are more significant; we must choose the temporal attributes, in a relation, with granularities that define the domain semantics. With insertion, delete or update operations we must check the temporal validity according to the definition in the *TSOR-M*. Therefore, the temporal validity maintenance could be achieved through the temporal information that is included in the meta-scheme. The implementation of active mechanisms will allow to validate the time for each operation in the database. These mechanisms are strongly domain-dependent because describe the time relationships among the attributes of a relation.

The temporal granularity for a temporal column is an inherent restriction of the *TSOR* model. That is, for each temporal attribute, the model forces one to define a temporal granularity. A semantic restriction is defined in the *TSOR* model giving the possibility to describe the conversion functions. Therefore, this model is semantically richer and allows one to support a larger variety of application domains.

7 Conclusion and on Going Research

The proposed framework is based on [29] with the definition of the Geometry data type. We have discussed how the spatial granularity is implicitly defined in the spatial reference system described in [29]. At the spatial level, our approach proposed the

following improvements with respect to a SOR model: an extension of the meta-schema described in [29], referred to as *SOR-M*, that includes new views, functions and operators. Among the new views, the *S_Coarser* view specifies the conversion functions required to change the spatial granularity preserving the consistency of the spatial attribute. The conversion operators are added to the functions presented in [29].

With respect to the temporal level, our approach is the first proposal to represent temporal attributes in the framework described before. We explain and define how the SQL3 data types can be converted into SQL3 temporal data types. This proposal is based on [13] but our temporal treatment is addressed to point-based approach. The temporal representation based on points makes it easy the use of aggregate functions as proposed in [28]. We define, as well, new views, functions and procedures extending the SQL3 functionalities. These functionalities are necessary to keep the consistency between temporal attributes in a relation and to deal with different granularities.

The main problem when we apply the relational model is the bi-dimensional structure of the relations. This problem reverts in our approximation and generates some difficulties that we are currently investigating. In a relation with several temporal attributes, the attribute with the finest granularity will be the marker of the variability in the relation. To improve the storage problems we are investigating the use of other SQL3 structures, as the REF type in order to avoid redundant information and the use of point-based view at the query level and the interval-based view at the storage level following the approach reported in [25]. The consistency of the temporal attributes within the same relation is another problem that in this approach has been commented. As future work, we plan to develop a set of triggers to check the time validity between attributes of the same relation as well as the execution model and its implications on the inheritance relationships among object types. Finally, other future work will dealing with supporting granularities per value instead of per column like the TOOBIS project approach [31] and the development of an effective and efficient query processing technique. This is an important issue to improve the retrieval of data from temporal attributes using new indexing techniques.

References

1. Bertolotto, M. Geometric Modeling of Spatial Entities at Multiple Levels of Resolution. PhD Thesis, Università degli Studi di Genova (1998).
2. Bettini, C., Jajodia, S. & Wang, X.Time Granularities in Databases, Data Mining and Temporal Reasoning. Springer-Verlag (2000).
3. T. Bittner and B. Smith. A Unified Theory of Granularity, Vagueness and Approximation. In Proc. of COSIT Workshop on Spatial Vagueness, Uncertain and Granularity (2001).
4. Camossi, E., Bertolotto, M., Bertino, E., Guerrini, G. A multigranular Spatio-temporal Data Model. Proc. 11th ACM International Symposium on Advances in Geographic Information Systems**,** New Orleans, Louisiana, USA (1998) 94 – 101.
5. Codd, E. F. A Relational Model of Data for Large Shared Data Banks. CACM 13 (6), June (1970).
6. Chen, C.X. & Zaniolo, C. SQLst: A Spatio-Temporal Data Model and Query Language. In Proc. of Int. Conference on Conceptual Modeling/ The Entity Relational Approach (2000).

7. Cindy X. Chen, Jiejun Kong and Carlo Zaniolo. Design and Implementation of a Temporal Extension of SQL. In Proceedings of the 19th International Conference on Data Engineering (ICDE'03), pages 689-691, Bangalore, India, March (2003).
8. Cindy Chen, Haixun Wang, and Carlo Zaniolo. Toward Extensible Spatio-Temporal Databases: an approach based on User-Defined Aggregates. In Flexible querying and reasoning in spatio-temporal databases: theory and applications, Springer Geosciences/Geoinformation series (2004).
9. Date, C. J. An Introduction to Database Systems. 8th Edition. Addison-Wesley, Reading, Mass (2003).
10. Eisenberg, A., Melton, J., Kulkarni, K., Michels, J.E., Zemke, F. SQL: 2003 Has Been Published. SIGMOD Record, vol. 33, no. 1, March (2004).
11. Elmasri, R. and Navathe, S: Fundamentals of Database Systems. Addison-Wesley, (2004).
12. Erwig, M., Schneider, M. & Güting R. H. Temporal Objects for Spatio-Temporal Data Models and a Comparison of Their Representations. ER Workshop (1998).
13. Erwig, M., Güting R. H., Schneider, M. & Vazirgiannis, M. Abstract and Discrete Modeling of Spatio-Temporal Data Types. In Proceedings of the 6th ACM Symposium on Geographic Information Systems, pg. 131-136, Washington, D.C., Novembre (1998).
14. Forlizzi, L., Güting, R.H., Nardelli, E., Schneider, M. A data model and data structures for moving objects database. Proceedings of the 2000 ACM SIGMOD international conference on Management of data (2000) 319-330**.**
15. Güting, R.H. An Introduction to Spatial Database Systems. VLDB Journal, vol. 3, (1994) 357-399.
16. R.H. Gütting, M.H. Böhlen, M. Erwig, C.S. Jensen, N.A. Lorentzos, M. Schneider and M. Vazirgiannis. A Foundation for Representing and Querying Moving Objects. ACM Ttransactions on Database Systems, Vol.25, No.1 (2000).
17. V. Katri, S. Ram, R.T. Snodgrass and G. O'Brien. Supporting User Defined Granularities and Indeterminacy in a Spatiotemporal Conceptual Model. Special Issues of Annals of Mathematics and Artificial Intelligence on Spatial and Temporal Granularity, 36(1-2) (2002) 195-232.
18. Lee, Y.L. Integrating Spatial and Temporal Relationship Operators into SQL 3 for Historical Data Management. ETRI Journal, vol. 24, n. 3, June (2002).
19. Lorentzos N.A. and Mitsopoulos Y.G. SQL Extension for Interval Data. IEEE Trans. on Knowledge and Data Engineering, vol. 9, no. 3, May/June (1997) 480-499.
20. Muller, J.C, Lagrange, J.P. & Weidel, R. (eds). GIS and Generalization: methodology and practice. Taylor and Francis (1991).
21. OMG (2000). Unified Modelling Language Specification, Version 1-3. Object Management Group.
22. Snodgrass, R.T. The TSQL2 Temporal Query Language. The TSQL2 Language Design Committee, Kluwer Academic Publishers (1995).
23. Snodgrass, R. T. Developing Time-Oriented Database Applications in SQL, Morgan Kaufmann Publishers, Inc., San Francisco, July (1999).
24. Schneider, M. Spatial Data Types for Database Systems-Finite Resolution Geometry for Geographic Information Systems. LNCS 1288, Springer-Verlag (1997).
25. Toman, D. A Point-Based Temporal Extension of SQL. In Proceedings of the 6th International Conference on Deductive and Object-Oriented Databases (1997)103-121.
26. Worboys, M. A Unified Model for Spacial and Temporal Information. The Computer Journal, 37(1) (1994) 26-34.
27. Wang, H. & Zaniolo, C. User Defined Aggregates in Object-Relational Systems. In 16th International Conference on Data Engineering, Feb (2000).

28. Wang, H. & Zaniolo, C. Using SQL to Build New Aggregates and Extenders for Object Relational Model. Proc. of the 26th International Conference a Very Large Databases (2000) 166-175.
29. www.opengis.org
30. www.oracle.com
31. www.mm.di.uoa.gr/~toobis/