# A Multidimensional Model Representing Continuous Fields in Spatial Data Warehouses

Alejandro Vaisman
Universidad de Buenos Aires, Argentina and
Hasselt University and
Transnational University of Limburg, Belgium
avaisman@dc.uba.ar

Esteban Zimányi
Université Libre de Bruxelles, Belgium
ezimanyi@ulb.ac.be

## ABSTRACT

Data warehouses and On-Line Analytical Processing (OLAP) provide an analysis framework supporting the decision making process. In many application domains, complex analysis tasks often require to take geographical information into account. Several proposals exist for integrating OLAP and Geographic Information Systems (GIS). However, there are very few attempts to support continuous fields, i.e., phenomena that are perceived as having a value at each point in space and/or time. Examples of such phenomena include temperature, altitude, or land use. In this paper, we extend a conceptual multidimensional model with continuous fields, showing that this can be achieved by defining an appropriate data type that encapsulates the different operations needed for manipulating such fields. We also define a query language based on relational calculus that allows expressing spatial OLAP queries involving continuous fields, and use this language to formally characterize this class of queries.

## Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Spatial Databases and GIS; H.4.2 [**Information Systems Applications**]: Decision Support

## Keywords

GIS, OLAP, Query Languages

## 1. INTRODUCTION

In the last few years, efforts have been carried out to integrate On-Line Analytical Processing (OLAP) [13] and Geographic Information Systems (GIS). This integration is usually called SOLAP (standing for Spatial OLAP), a paradigm aimed at being able to explore spatial data by drilling on maps, in the same way as OLAP operates over tables and charts. This concept was introduced by Rivest *et al.* [23], who also describe the desirable features and operators a SOLAP system should have. SOLAP concepts and operators have been implemented in a commercial tool called JMAP[1]. A survey on the topic can be found in [4]. Moreover, the need for sophisticated GIS-based Decision Support Systems (DSS), for the analysis of organizational data with respect to geographic information, is encouraging OLAP and GIS vendors to increasingly integrate their products.

These advances in data analysis technologies attracted the attention of many GIS practitioners (for instance, in the environmental domain), who envisioned the possibility of performing complex analysis tasks. This raises new challenges, like the need to handle *continuous fields*, which describe the distribution of physical phenomena that change continuously in time and/or space. Examples of such phenomena are temperature, pressure, or land elevation. Besides physical geography, continuous fields (from now on, fields), like land use and population density, are used in human geography as an aid in spatial decision making process.

Although some work has been done to support querying fields in GIS (see Section 2), the area of spatial multidimensional analysis of continuous data is still almost unexplored. Integrating spatiotemporal continuity within multidimensional structures poses numerous challenges [2]. Further, existing multidimensional structures and models dealing with discrete data, are not adequate for the analysis of continuous phenomena. Multidimensional models and associated query languages are thus needed, to support continuous data.

The main contribution of this paper is a conceptual multidimensional model that supports fields. This model extends the one introduced in [16] in a natural way. We discuss the rationale underlying the choice of this model, study the problems that a conceptual multidimensional model for fields must address, and show how our proposal accounts for these problems. We also show why the few existing proposals fail in this attempt. We then characterize multidimensional queries over fields, denoting this class of queries SOLAP-CF (standing for SOLAP with Continuous Fields). Along the lines of previous work [27], for this characterization we make use of the relational calculus supporting aggregate functions, and extend it with a *field* data type. We build on the type system defined by Güting *et al.* [10], and extension their *abstract model* with a *field* data type.We provide an in-depth study of the operators that this data type must include to support (and extend to the multidimensional setting) the classic Map Algebra introduced by Tomlin [26], and its extensions proposed by Câmara *et al.* [5], and Cordeiro *et al.* [6]. Tomlin's work was later extended to support spatiotemporal data by Mennis *et al.*[17], yielding the so-called

[1]`http://www.kheops-tech.com/en/jmap/solap.jsp`

*cubic* map algebra. We show how our type system supports multidimensional analysis over time-varying fields, and formally define the class of STOLAP-CF queries. We also provide a comprehensive set of SOLAP-CF and STOLAP-CF queries. Finally, we discuss, also following [10], different choices for the implementation of the abstract model, using regular gridded digital elevation models (DEM), and triangulated irregular networks (TIN) as data structures [15].

This paper is organized as follows. Section 2 provides an overview of related work dealing with fields. Section 3 presents the conceptual model, and introduces the *field* data type as well as the relational calculus we use to express multidimensional queries. Section 4 studies the extension of SOLAP with spatial fields, while spatio-temporal fields are covered in Section 5. In Section 6 we discuss different possibilities for a discrete model that can implement our proposal. We conclude in Section 7.

## 2. RELATED WORK

*Fields* are well-known in physics, where, for example, magnetic or gravitational fields are such that every spatial location has a value for a magnetic or gravitational force, respectively. They are modeled mathematically by a function that maps a spatial location to a force vector. In GIS, *fields* model phenomena that can be represented by a function of space and time [12]. A field is formally defined as composed of [20]: (a) a domain $\mathcal{D}$ which is a continuous set; (b) a range of values $\mathcal{R}$; and (c) a mapping function $f$ from $\mathcal{D}$ to $\mathcal{R}$.

In a pioneering work on defining *algebra for fields*, Tomlin [26] proposed a so-called map algebra, based on the notion that a map is used to represent a continuous variable (e.g., temperature). There are three types of functions in Map algebra: *local*, *focal*, and *zonal*. Local functions compute a value at a certain location as a function of the value(s) at this location in other map layer(s), allowing queries like "Compute the total desert land in a country, where a region is classified as desert if the annual rain is less than 500 mm per year." Focal functions compute each location's value as a function of existing values in the neighboring locations of existing layers (i.e., they are characterized by the topological predicate *touches*), allowing aggregate queries like "Local altitude in clay soil regions, in a map containing soils distribution in some portion of land". Zonal functions (characterized by the topological predicate *inside*), compute a location's new value from one layer (containing the values for a variable), associated to the zone (in another map) containing the location. An example of a query associated to these functions is "Total area in a province, with elevation greater than 1000 m above sea level". Câmara *et al.* [5] and Cordeiro *et al.* [6] formalized and extended these functions, supporting more topological predicates. We base our proposal on this work, and on the proposal of Mennis *et al.* [17], where map algebra operators are extended to query time-varying fields. Therefore, the model and query language we present in this paper cover those proposals, and extend them to the multidimensional setting.

Further, Paolino *et al.* [20] introduced *Phenomena*, a visual language for querying continuous fields, based on a conceptual model where users view the world as consisting of both continuous fields and discrete objects, and are able to manipulate them in a uniform manner.

Regarding *fields and multidimensional models*, the joint contribution of the GIS and OLAP communities to this problem has been limited. Shanmugasundaram *et al.* [24] propose a data cube representation that deals with continuous dimensions not needing a predefined discrete hierarchy. They focus on using the known data density to calculate aggregate queries without accessing the data. The representation reduces the storage requirements, but continuity is addressed in a limited way. Ahmed *et al.* use interpolation methods to estimate (continuous) values for dimension levels and measures, based on existing sample data values [2]. Continuous cube cells are computed on-the-fly, producing a continuous representation of the discrete cube. Sequels of this proposal introduce SOLAP concepts, and a SOLAP application supporting some form of continuous data [1]. These proposals are based on a data model devised for OLAP, not for *spatial* OLAP, which we believe does not favor comprehensive representation of spatial dimensions. Opposite to this, our approach is based on a conceptual multidimensional model designed with spatial data in mind. Thus, continuous fields are introduced as a natural evolution of this model.

With respect to *standards*, the ISO standard 19123:2005 [11] defines a conceptual schema for fields, referred to as coverages. A coverage is defined as a function from a spatial, temporal, or spatiotemporal domain to an attribute range. It associates a position within its domain to a record of values of defined data types. Examples of coverages include rasters, triangulated irregular networks, point coverages, and polygon coverages. This standard has an associated Implementation Specification for Grid Coverages defined by the Open Geospatial Consortium [18].

Several *tools* support fields. For example, GeoRaster[2] is a feature of Oracle Spatial that allows storing, indexing, querying, analyzing, and delivering raster data, and its associated metadata. GeoRaster provides specialized data types and associated operators, as well as an object relational schema, which can be used to store and manipulate multidimensional raster layers.

In spite of the many existing proposals, only recently a precise definition of spatial and spatiotemporal OLAP queries was proposed by the authors of the present paper [27]. That work defines a taxonomy of models that integrates OLAP, spatial data, and moving data types, defining, for each of the classes in this taxonomy, the queries that they must support. We extend this classification to support fields, defining two new query classes, for spatial and spatiotemporal OLAP over fields: SOLAP-CF and STOLAP-CF.

## 3. PRELIMINARIES

### 3.1 The MultiDim model

In this paper we extend the *MultiDim* model [16] to support fields. We give next a brief review of the main features of this model. A *multidimensional schema* is as a finite set of dimensions and fact relationships. A *dimension* comprises at least one *hierarchy*, which contains at least one level. A hierarchy with only one level is called a *basic hierarchy*. Several levels can be related to each other through a binary relationship that defines a partial order $\preceq$ between levels. Given two consecutive related levels $l_i, l_j$, if $l_i \preceq l_j$ then $l_i$ is called *child* and $l_j$ is called *parent*. A level representing the less detailed data for a hierarchy is called a *leaf level*, related to

[2]`http://download.oracle.com/docs/html/B10827_01/geor_intro.htm`

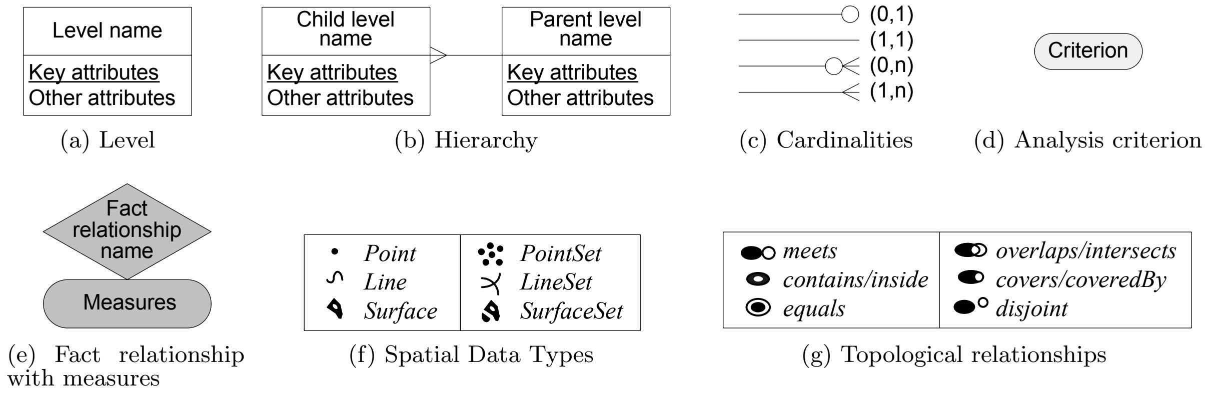

Figure 1: Notation of the MultiDim model.

at least one *fact relationship*. The latter represents an $n$-ary relationship between two or more leaf levels. If these levels are spatial, the relationship may also be topological and requires a spatial predicate, e.g., *intersection*. These operators are indicated in the model, as we explain below. A fact relationship may contain measures that can be spatial or thematic. The latter are numeric and express analysis needs in a quantified form, the former can be represented by a geometry or field, or calculated using spatial operators, such as distance or area.

The dimension levels contain category attributes and property attributes. A *category attribute* of a parent level shows how child members are grouped for applying aggregation functions to measures. A category attribute in a leaf level indicates the aggregation level of a measure in the associated fact relationship, e.g., monthly sales if a leaf level of a time dimension is represented by a month. A *property attribute* contains additional features of a level; it can be spatial (represented by a geometry or field) or thematic (alphanumeric data types). The spatiality of a level depends on whether it has at least one spatial property attribute. Similarly, the spatiality of a hierarchy (resp. dimension) depends on whether it has at least one spatial level (resp. hierarchy). Additionally, a hierarchy name is derived from the name of its leaf level and a criterion name; similarly, a dimension name is obtained from its leaf level name.

Figure 1 presents the graphical notation for representing different elements in the multidimensional model we described above. Further, for representing the geometry of spatial levels, measures, and different kinds of spatial predicates in the *MultiDim* model, we use MADS (standing for Modeling of Application Data with Spatio-temporal features)[21], a spatio-temporal conceptual model that allows to define intuitive, easy-to-understand schemas supporting a wide variety of spatio-temporal features associated to any kind of object in the data model.

Throughout the paper we use the following (simplified) real-world example. The Agriculture Agency of a country collects information about the crops produced at land plots. The application has maps describing the location of land plots in counties, as well as the political division of the country into states and counties.

Figure 2 shows the conceptual schema depicting the above scenario using the MultiDim model explained in the previous section. There is one fact relationship, Yield, which has two measures: production and cropArea. The fact relationship Yield is related to three dimensions: Crop, LandPlot, and Time. Dimensions are composed of levels and hierarchies. For example, the LandPlot dimension is composed of three levels, LandPlot, County, and State, related by one-to-many parent-child relationships. For example, the three levels in the LandPlot dimension are spatial; they have a geometry representing a region. Similarly, the attribute capital in State, and the measure commonArea in the fact relationship, are spatial as well. Finally, topological relationships may be represented in fact relationships and in parent-child relationships. For example, the topological relationships in the LandPlot hierarchy indicate that a land *overlaps* a county (it may be located in more than one county) while a county is *covered* by its parent state.

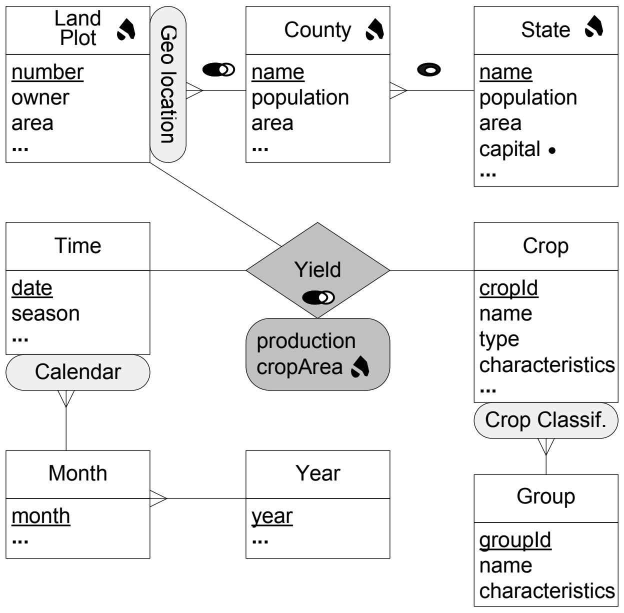

Figure 2: An example of a spatial data warehouse.

## 3.2 Extending the conceptual model

To address continuous field scenarios, we extend the data types defined by Güting *et al.* [10]. We work at the *abstract model level*, which simplifies the model definition, making it independent of the implementation choice, thus making the presentation clearer. We define a new *field* data type, its corresponding operations, and use them in a relational calculus supporting aggregate functions. We start with a quick overview of the data types, and refer to this work for the complete definition of the type system and the corresponding operations. There is a set of *base types*: int, real, bool, string, and an *identifier type* id, used to identify dimension level members. There are also *time types* namely instant and periods, the latter being a set of time intervals. Finally, there are four *spatial data types*, point, points, line, and region. A value of type point represents a point in the Euclidean plane. A points value is a finite set of points. A line value is a finite set of continuous curves in the plane. A region is a finite set of disjoint parts called *faces*, each of which may have holes. All the above types are called *discrete* types.

*Field types* capture the variation in space of base types. They are obtained by applying a constructor field$(\cdot)$. Hence, a value of type field(real) (e.g., representing altitude) is a continuous function $f$ : point $\rightarrow$ real. Field types have associated operations that generalize those of the discrete types. This is called *lifting*. For example, the comparison predicates (e.g., $>$) with signature $\alpha \times \alpha \rightarrow$ bool are generalized by allowing any argument to be a field, as in field$(\alpha) \times$ field$(\alpha) \rightarrow$ field(bool). Intuitively, the semantics of such lifted operations is that the result is computed at each point in space using the non-lifted operation. Formally, we define the semantics of the field type by means of its *carrier set,* i.e., a set containing the possible values for the type.

*Definition 1. (Carrier set of field)* Let $\alpha$, with carrier set $\overline{A}_\alpha$, be a type to which the moving(.) operator is applicable. The carrier set for moving$(\alpha)$ is given by:

$$A_{field}(\alpha) = \{f | f : \overline{A}_{point} \rightarrow \overline{A}_\alpha\}$$

There are also *moving types*, that capture the evolution over time of base types, spatial types, and field types. They are obtained by applying a constructor moving$(\cdot)$. For example, a value of type moving(point) (e.g., representing a vehicle that changes its position in time) is a continuous function $f$ : instant $\rightarrow$ point. Similarly, a value of type moving(field(real)) defines a continuous function $f$ : instant $\rightarrow$ (point $\rightarrow$ real). It can be used to represent temperature, which varies on time and space. Operations on non-temporal types are generalized (or *lifted*) to moving types. For example, a distance function with signature moving(point) $\times$ moving(point) $\rightarrow$ moving(real) computes the distance between two moving points and returns a moving real, i.e., a real-valued function of time. As in the case of field types, the semantics of lifted operations for moving types is that the result is computed at each time instant using the non-lifted operation.

*Definition 2. (Carrier set of moving(field))* Let $\alpha$, with carrier set $\overline{A}_\alpha$, be a type to which the field(.) operator is applicable. The carrier set for field$(\alpha)$ is given by:

$$A_{moving}(field(\alpha)) = \{f | f : \overline{A}_{instant} \rightarrow \overline{A}_{field}(\alpha)\}$$

Definition 3 summarizes the discussion above, spelling out the data types we support in this paper.

*Definition 3. (Data types)* Let us denote $\Gamma$ a set of *discrete types*, composed of a set of *base types* $\beta$, a set of *time types* $\tau$, and a set of *spatial types* $\xi$. The set of *field types* $\phi$ is obtained by applying the field constructor to elements of $\beta$. Further, the set of *moving types* $\Theta$ is obtained by applying the moving constructor to elements of $\beta$, $\xi$, and $\phi$.

## 3.3 A relational calculus supporting aggregates

We address the issue of querying data warehouses, using a relational representation of the MultiDim conceptual model. A dimension level is represented by a relation of the same name, with an implicit identifier attribute denoted id, an implicit geometry attribute (if the level is spatial), and other explicitly indicated attributes. The id attribute (e.g., County.id) identifies a particular member in a dimension instance. Dimension levels in hierarchies (e.g., LandPlot) have also an additional attribute containing the identifier of the parent level (e.g., LandPlot.county), and there is a referential integrity constraint between such attributes and the corresponding parent (e.g., County.id). A fact relationship is represented by a relation of the same name having an implicit id attribute, one attribute for each dimension, and one attribute per measure. There is a referential integrity constraint between the dimension attributes in the fact relationship (e.g., Yield.district) and the identifier of the corresponding dimension (e.g., LandPlot.id).

In the remainder of the paper we use a query language based on the tuple relational calculus (e.g., [8]) extended with aggregate functions and variable definitions. We first show that this language expresses standard SOLAP and spatial OLAP queries. Then, we show that extending this calculus with *field* types is enough to express multidimensional queries over fields. We now introduce the language through an example. Consider the following relations from the data warehouse shown in Fig. 2.

County(<u>id</u>, geometry, name, population, area, . . . , state)
State(<u>id</u>, geometry, name, population, area, capital, . . .).

The following query asks the name and population of counties in California.

$\{c.\mathsf{name}, c.\mathsf{population} \mid \mathsf{County}(c) \wedge \exists s\, (\mathsf{State}(s) \wedge$
$\quad c.\mathsf{state} = s.\mathsf{id} \ \wedge s.\mathsf{name} = \text{‘California’})\}$

Suppose that we want to compute the total population of counties of California. A first attempt to write this query would be:

$\mathsf{sum}(\{c.\mathsf{population} \mid \mathsf{County}(c) \wedge \exists s\, (\mathsf{State}(s) \wedge$
$\quad c.\mathsf{state} = s.\mathsf{id} \ \wedge s.\mathsf{name} = \text{‘California’})\})$

Notice that however, since the relational calculus is based on sets (i.e., collections with no duplicates), if two counties in California happen to have the same population, they would appear only once in the set to which the sum operator is applied. We adopt Klug's approach [14], where this problem is solved by using aggregate operators that take as argument a set of tuples (instead of a set of values) and specifying on which column the aggregate operator must be applied. Therefore, the above query is written as follows.

$\mathsf{sum}_2(\{c.\mathsf{id}, c.\mathsf{population} \mid \mathsf{County}(c) \wedge \exists s\, (\mathsf{State}(s) \wedge$
$\quad c.\mathsf{state} = s.\mathsf{id} \ \wedge s.\mathsf{name} = \text{‘California’})\})$

In this case, the sum operator is applied to a set of pairs $\langle \mathsf{id}, \mathsf{population} \rangle$ and computes the sum of the second attribute.

Finally, suppose that we want to compute the total population by state provided that it is greater than 100,000. In this case we need a recursive definition of queries and variables that bind the results of inner queries to outer queries:

$$\{s.\mathsf{name}, \mathsf{totalPop} \mid \mathsf{State}(s) \wedge \mathsf{totalPop} = \mathsf{sum}_2( \{c.\mathsf{id}, c.\mathsf{population} \mid \mathsf{County}(c) \wedge c.\mathsf{state} = s.\mathsf{id}\}) \wedge \mathsf{totalPop} > 100,000 \}$$

Here, the outer query fixes a particular state $s$ and the inner query collects the population of counties for that state. The sum of these populations is then bound to the variable totalPop. Notice that this query corresponds to an SQL query with the GROUP BY and HAVING clauses.

We denote $\mathcal{R}_{agg}$ the relational calculus with aggregate functions defined above. It is easy to prove that $\mathcal{R}_{agg}$ has the same expressive power than Klug's calculus [14].

## 3.4 Expressing OLAP and SOLAP queries

In this section we characterize the classes of OLAP and spatial OLAP (SOLAP) queries, based on the language $\mathcal{R}_{agg}$ defined in Section 3.3. In Section 4 we characterize multidimensional queries over fields. We start by showing examples of OLAP queries.

1. For land plots located in counties of California and crops in the cereals group give the maximum production by month.

$$\{l.\mathsf{number}, c.\mathsf{name}, m.\mathsf{month}, \mathsf{maxProd} \mid \mathsf{LandPlot}(l) \wedge \mathsf{Crop}(c) \wedge \mathsf{Month}(m) \wedge \exists g\, (\mathsf{Group}(g) \wedge c.\mathsf{group} = g.\mathsf{id} \wedge g.\mathsf{name} = \text{'Cereal'} \wedge \mathsf{maxProd} = \mathsf{max}_1( \{y.\mathsf{production} \mid \mathsf{Yield}(y) \wedge y.\mathsf{landPlot} = l.\mathsf{id} \wedge y.\mathsf{crop} = c.\mathsf{id} \wedge \exists u, \exists s, \exists t\, ( \mathsf{County}(u) \wedge \mathsf{State}(s) \wedge \mathsf{Time}(t) \wedge l.\mathsf{county} = u.\mathsf{id} \wedge u.\mathsf{state} = s.\mathsf{id} \wedge s.\mathsf{name} = \text{'California'} \wedge y.\mathsf{time} = t.\mathsf{id} \wedge t.\mathsf{month} = m.\mathsf{id})\})\}$$

2. For each county, give the number of landplots where, for at least one crop, the production in March 2008 was greater than 100,000 tons.

$$\{c.\mathsf{name}, \mathsf{nbLandPlots} \mid \mathsf{County}(c) \wedge \mathsf{nbLandPlots} = \mathsf{count}(\{l.\mathsf{id} \mid \mathsf{LandPlot}(l) \wedge \exists p\, (\mathsf{Crop}(p) \wedge \mathsf{sum}_2(\{y.\mathsf{id}, y.\mathsf{production} \mid \mathsf{Yield}(y) \wedge y.\mathsf{crop} = p.\mathsf{id} \wedge y.\mathsf{landPlot} = l.\mathsf{id} \wedge l.\mathsf{county} = c.\mathsf{id} \wedge \exists t\, (\mathsf{Time}(t) \wedge y.\mathsf{time} = t.\mathsf{id} \wedge t.\mathsf{date} \geq 1/3/2008 \wedge t.\mathsf{date} \leq 31/3/2008)\}) > 100,000)\})\}$$

*Definition 4. (OLAP queries)* Let us call $\mathcal{R}_{agg}$ the relational calculus with aggregate functions defined in Section 3.3. The class of OLAP queries includes all the queries expressible in $\mathcal{R}_{agg}$.

Next, we give some examples of SOLAP queries, where spatial features come into play. Therefore, we need the spatial data types defined in Section 3.1.

3. Total area of landplots located within 10 km from Orange county that intersect San Diego county.

$$\mathsf{sum}_2(\{l.\mathsf{id}, l.\mathsf{area} \mid \mathsf{LandPlot}(l) \wedge \exists c_1, \exists c_2\, (\mathsf{County}(c_1) \wedge \mathsf{County}(c_2) \wedge c_1.\mathsf{name} = \text{'Orange'} \wedge \mathsf{distance}(l.\mathsf{geometry}, c_1.\mathsf{geometry}) < 10 \wedge c_2.\mathsf{name} = \text{'San Diego'} \wedge \mathsf{intersects}(c_2.\mathsf{geometry}, l.\mathsf{geometry}))\})$$

| Class | Operations |
|---|---|
| Projection to Domain/Range | defspace, rangevalues, point, val |
| Interaction with Domain/Range | atpoint, atpoints, atline, atregion, at, atmin, atmax, defined, takes,concave, convex, flex |
| Rate of change | partialder_x, partialder_y |
| Aggregation operators | integral, area, surface, favg, fvariance, fstdev |
| Lifting | (all new operations inferred) |

Table 1: Classes of operations on field types.

Note that this query does not use a fact relationship. The function distance verifies that the geometries of the land plot and the Orange county are less than 10 km away from each other and the predicate intersects verifies that the land plot intersects the San Diego county.

4. For land plots that are located in the border of the San Diego county, compute the yield by acre for the production of cereals in 2008.

$$\{l.\mathsf{number}, \mathsf{yield} \mid \mathsf{LandPlot}(l) \wedge \exists c\, (\mathsf{County}(c) \wedge c.\mathsf{name} = \text{'San Diego'} \wedge \mathsf{touches}(l.\mathsf{geometry}, c.\mathsf{geometry}) \wedge \mathsf{yield} = \mathsf{sum}_2(\{y.\mathsf{id}, y.\mathsf{production} \mid \mathsf{Yield}(y) \wedge y.\mathsf{landplot} = l.\mathsf{id} \wedge \exists p, \exists g\, (\mathsf{Crop}(p) \wedge \mathsf{Group}(g) \wedge y.\mathsf{crop} = p.\mathsf{id} \wedge p.\mathsf{group} = g.\mathsf{id} \wedge g.\mathsf{name} = \text{'Cerals'} \wedge \exists t\, (\mathsf{Time}(t) \wedge y.\mathsf{time} = t.\mathsf{id} \wedge t.\mathsf{date} \geq 1/1/2008 \wedge t.\mathsf{date} \leq 31/12/2008)\})/\mathsf{area}(l.\mathsf{geometry}))\}$$

Here, the outer query fixes a landplot that satisfies the topological predicate touches and the inner query computes the total production of cereals during 2008 for that landplot, which is then divided by its area.

*Definition 5. (SOLAP queries)* Let us call $\mathcal{R}^{\xi}_{agg}$ the language $\mathcal{R}_{agg}$ augmented with spatial types in $\xi$. The class of SOLAP queries is the class composed of all the queries that can be expressed by $\mathcal{R}^{\xi}_{agg}$.

## 4. EXTENDING SOLAP WITH CONTINUOUS FIELDS

### 4.1 The field data type and its operations

We now introduce field types in order to extend SOLAP to support fields. A *field type* is a function from the spatial domain to a base type. Field types are obtained by applying the field type constructor to a type $\alpha$. In this section we discuss *nontemporal* field types, *temporal* field types are covered in next section. Notice that field types are partial functions, i.e., they may be undefined for certain regions of space. Further, a type constructor inspace yields for, a type $\alpha$, a corresponding type whose values are pairs consisting of a point in space and a value for $\alpha$.

Field types come equipped with a set of operations, which may be grouped in several classes, shown in Table 1. We discuss next some of these operations.

A set of operations realize the *projection into the domain and range.* Operations defspace and rangevalues return, respectively, the projection of a field type into its domain and range. For values of inspace types, operations point and val return the point and the value of the type.

Another set of operators allow the *interaction with domain and range.* Operations atpoint, atpoints, atline, and atregion restrict the function to a given subset of the space defined by a spatial value. Operation at restricts the function to a point or to a point set (a range) in the range of the function. Predicates atmin and atmax reduce the function to the points in space when its value is minimal or maximal, respectively. The defined predicate allows checking whether the spatial function is defined at a subset of the space defined by a spatial value. Analogously, predicate takes checks whether the function somewhere assumed (one of) the value(s) from the range given as the second argument. Operations concave and convex restrict the function to the points where it is concave or convex, respectively. Finally, operation flex restricts the function to the points where convexity changes.

*Rate of change* operators compute how a field changes across space. Functions partialder_x and partialder_y give, respectively, the partial derivative of the function defining the field with respect to the one of the axis $x$ and $y$. For example, partialder_x is defined by

$$\frac{\partial f}{\partial x}(x,y) = \lim_{h\to 0} \frac{f(x+h,y)-f(x,y)}{h}.$$

There are three basic *field aggregation operators* that take as argument a field over numeric values (int or real) defined over a spatial extent $S$ and return a real value. They are defined as follows [20]:

- integral: $\iint_S f(x,y)\,dxdy$
- area: $\iint_S dxdy$
- surface: $\iint_S \sqrt{1+\frac{df^2}{dx}+\frac{df^2}{dy}}\,dxdy$

From these operators, other derived operators can be defined (prefixed with an 'f' (field) in order to distinguish them from the usual aggregation operators generalized to fields, which we discuss below).

- favg: integral/area
- fvariance: $\iint_S \frac{(f(x,y)-\mathsf{favg})^2}{\mathsf{area}}\,dxdy$
- fstdev: $\sqrt{\mathsf{fvariance}}$.

All operations on discrete types are generalized for field types. This is called *lifting* (following [10]). An operation op for discrete types is lifted to allow any of the argument types to be replaced by the respective field type and also return a corresponding field type. As an example, the + operator with signature $\alpha \times \alpha \to \alpha$ has lifted versions where one or both of its arguments can be field types and the result is a field type (e.g., $\mathsf{field}(\alpha) \times \alpha \to \mathsf{field}(\alpha)$). Intuitively, the semantics of such lifted operations is that the result is computed at each point using the non-lifted operation. Several definitions of an operator may be applied when combining two fields that are defined on different spatial extents. A first solution could be that the result is defined in the *intersection* of both extents, and be undefined elsewhere. Another solution could be that the result is defined on the *union* of the two extents, and a default value (e.g., 0 for the addition) is used for combining over the extents that belong to only one field. *Aggregation* operators are also uplifted in the same way. For instance, an uplifted avg operator combines several fields, yielding a new field where the average is computed at each point in space. These uplifted aggregation operations correspond to Tomlin's *local functions* [26]. Notice that these operators are used in particular for granularity transformations. For example, when transforming a temperature field of granularity day to granularity month an uplifted average could be used.

## 4.2 SOLAP-CF queries

We now define the class of multidimensional queries over fields, which we denote SOLAP-CF queries. We modify our running example in Fig. 2 by adding the field dimensions SoilType, SoilPHLevel, and Temperature, a field measure suitability (with values between 0 and 10)), and a measure avgTemp, as shown in Fig. 3. *Non-temporal* field levels and measures are identified by the f(🌍) pictogram, while *temporal* ones are identified by the f(🌍,🕓) pictogram. For example, the level SoilType is a field, which means that each point in the space of interest has a value for its soil type (like clay or limestone). Similarly, Temperature is a *temporal field*, which means that each point in the space of interest has a value of temperature, and this value changes at each time instant. Field levels have a geometry attribute, which keeps track of the value of the field at each point in space (and time, for temporal fields). The type of such attribute is $\mathsf{field}(\alpha)$ or $\mathsf{moving}(\mathsf{field}(\alpha))$, depending on whether the field is temporal or not. Here we discuss non-temporal fields, while Section 5 is devoted to temporal fields.

Notice that in our model the field dimensions are not connected to a fact relationship. This contrasts with traditional multidimensional models, as well as models introducing fields in spatial data warehouses (e.g., [2]). In these approaches, the dimension instances are the possible values of the underlying domain (probably obtained through interpolation). Since we consider continuous domains, there may be an infinite number of instances, each one corresponding to one possible value of the domain. Therefore, field dimensions contain only one instance, and the attributes of the field dimension correspond to metadata describing it. Also, the model allows the combination of spatial data represented by geometries and by fields. Thus, users can decide to represent a spatial phenomenon using either of the two representations depending on their specific needs. This enables complex analysis scenarios, and supports, in particular Tomlin's zonal operators [26], as we show below.

In our example there is a *field hierarchy* composed of levels SoilType and SoilGroup. This models a two-level soil classification, such as the one in World Reference Base for Soil Resources[3]. Alternatively, other hierarchies for soil classification could be used, such as the USDA Soil Taxonomy that contains 6 levels. For field data, it is usual practice to create hierarchies to reduce the resolution of data in order to speed-up display and reduce the level of detail, while keeping the same spatial extent. Different techniques can be applied for these task, like spatial aggregation of fields [3], clustering data based on approximation methods [22], and map layer generation operation [9], among other ones.

*Field measures*, represented by a field data type, are also supported. An example is the measure suitability in the fact relationship Yield, which could be precomputed in the preprocessing stage as a function of many factors (for example, soil type, soil pH level, and temperature). In addition,

[3]http://www.fao.org/ag/agl/agll/wrb/default.stm

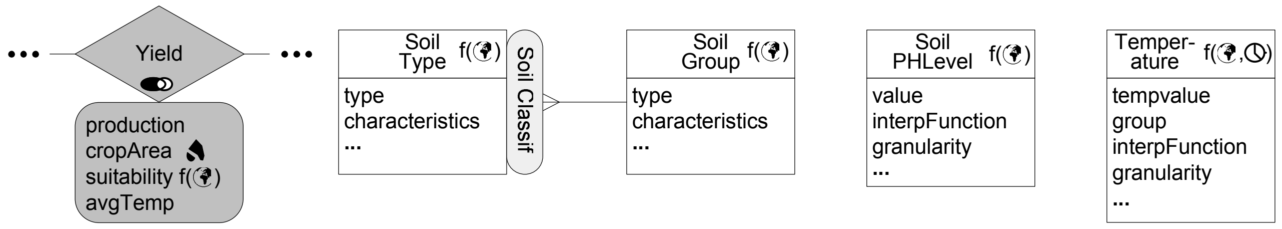

Figure 3: Field dimensions and field measures added to the example in Fig. 2.

traditional numerical measures can be calculated from field data. An example is measure avgTemp, which keeps the average temperature (a real value) of each instance of the fact relationship, and which is computed from the dimensions Temperature, LandPlot, and Time. An issue that arises with field measures is their *aggregation along hierarchies*. It is well-known that aggregation functions for numerical measures can be distributive (e.g., *sum*), algebraic (e.g., *average*), or holistic (e.g., *median*). This classification also applies for spatial aggregation functions manipulating geometries [25]. For example, spatial distributive aggregates include *convex hull* and *geometric intersection*, spatial algebraic functions include *center of n geometric points* and *center of gravity*, and spatial holistic functions include *equipartition* and *nearest-neighbor index*. Since field measures are defined over some spatial extent, similar aggregation functions can be used.

We characterize next the class of SOLAP-CF queries. We start by giving some examples of these queries.

5. Total area of clay soil in the state of Utah.

$$\{\mathsf{area}(\mathsf{intersection}(s.\mathsf{geometry}, \mathsf{defspace}(\mathsf{at}(t.\mathsf{geometry}, \text{'Clay'})))) \mid \mathsf{State}(s) \wedge \mathsf{SoilType}(t) \wedge s.\mathsf{name} = \text{'Utah'}\}$$

In this query, the function at restricts the SoilType field to the points in space that have the value 'Clay', and function defspace yields the region containing such points. This region is then intersected with the region of the Utah state and the area operator is applied to the intersection.

6. For land plots having at least 30% of their surface of clay soil, give the average suitability value for wheat on February 1st, 2009.

$$\{l.\mathsf{number}, \mathsf{favg}(y.\mathsf{suitability}) \mid \mathsf{LandPlot}(l) \wedge \exists s, \exists y, \exists t, \exists c\, (\mathsf{SoilType}(s) \wedge \mathsf{Yield}(y) \wedge \mathsf{Time}(t) \wedge \mathsf{Crop}(c) \wedge y.\mathsf{landplot} = l.\mathsf{id} \wedge y.\mathsf{time} = t.\mathsf{id} \wedge t.\mathsf{date} = 1/2/2009 \wedge c.\mathsf{name} = \text{'Wheat'} \wedge (\mathsf{area}(\mathsf{defspace}(\mathsf{atregion}(\mathsf{at}(s.\mathsf{geometry}, \text{'Clay'}), l.\mathsf{geometry})))/\mathsf{area}(l.\mathsf{geometry})) \geq 0.3)\}$$

Here, the soil type field is restricted to the value 'Clay' by means of the function at, and then is restricted to the geometry of the land plot with function atregion. The operator defspace obtains the geometry of the restricted field, the area of this geometry is computed, and this is finally divided by the total area of the land plot. The favg function is applied to compute the overall average suitability as a real value.

7. Give parcels that, having a minimum pH value of less than 4, had average suitability greater than 8 for any kind of crop during March 2009.

| Class | Operations |
|---|---|
| Projection to Domain/Range | deftime, rangevalues, inst, val, locations, trajectory, routes, traversed |
| Interaction with Domain/Range | atinstant, atperiods, initial, final, present, at, atmin, atmax, passes |
| Rate of change | derivative, speed, turn, velocity |
| Lifting | (all new operations inferred) |

Table 2: Classes of operations on moving types.

$$\{l.\mathsf{number} \mid \mathsf{LandPlot}(l) \wedge \exists s\, (\mathsf{SoilPHLevel}(s) \wedge \mathsf{element}(\mathsf{rangevalues}(\mathsf{atmin}(\mathsf{atregion}(s.\mathsf{geometry}, l.\mathsf{geometry})))) < 4 \wedge \mathsf{favg}(\{\mathsf{avg}(y.\mathsf{suitability}) \mid \mathsf{Yield}(y) \wedge \exists t, \exists m\, (\mathsf{Time}(t) \wedge \mathsf{Month}(m) \wedge y.\mathsf{landPlot} = l.\mathsf{id} \wedge y.\mathsf{time} = t.\mathsf{id} \wedge t.\mathsf{month} = m.\mathsf{id} \wedge m.\mathsf{month} \leq 3/2009\}) > 8)\}$$

Here, function atregion restricts the pH field to the geometry of the landplot; function atmin further restricts it to the points of minimum value. Function rangevalues obtains the set of values of the field (a singleton), and element is used to get the value from the singleton. In the inner query, the set of suitability fields of the land plot during March 2009 are first aggregated into a single field with function avg and then favg is used to obtain the overall average value.

Definition 6 formalizes the class of SOLAP-CF queries.

*Definition 6. (SOLAP-CF queries)* Let us call $\mathcal{R}_{agg}^{\xi\phi}$ the language $\mathcal{R}_{agg}$ augmented with spatial types $\xi$ and field types $\phi$. The class of SOLAP-CF queries, is the class composed of all the queries that can be expressed by $\mathcal{R}_{agg}^{\xi\phi}$.

## 5. SPATIO-TEMPORAL OLAP WITH CONTINUOUS FIELDS

Temporal fields model phenomena whose value change along time and space. Typical examples are temperature or precipitation. We model temporal fields using the moving types defined by Güting *et al.* [10], introduced in Section 3. Moving types have a set of associated operations, shown in Table 2. In addition, a type constructor intime yields, which, for a type $\alpha$, a corresponding type whose values are pairs consisting of a time instant and a value for $\alpha$.

A set of operations realize the *projection into the domain and range*. Operations deftime and rangevalues return, respectively, the projection of a moving type into its domain and range. For values of intime types, the two operations inst and val return the instant and the value of the type. Several operations project moving objects into the plane. The projection of a moving point into the plane may consist of points and lines, returned by the operations locations and trajectory, respectively. The projection of a moving line into the plane may consist of lines and regions, returned by the operations routes and traversed. Finally, the projection of a moving region into the plane consists in a region, which is also returned by the operation traversed.

Another set of operators allow the *interaction with domain and range*. Operations atinstant and atperiods restrict the function to a given time or set of time intervals. Operations initial and final return, respectively, the (instant,value) pairs for the first and last instant of the definition time. Operation at restricts the function to a point or to a point set (a range) in the range of the function. Predicates atmin and atmax reduce the function to the times when it was minimal or maximal, respectively. The present predicate checks whether the temporal function is defined at an instant of time, or is ever defined during a given set of intervals. Analogously, predicate passes checks whether the function ever assumed one of the values from the range given as second argument.

The derivative operator takes as argument a moving real and yields as result a moving real. Four operations compute the *rate of change* for points: speed yields the usual concept of speed of a moving point at all times as a moving real; mdirection returns the direction of the movement, i.e., the angle between the $x$-axis and a tangent to the trajectory of the moving point; turn yields the change of direction at all times; and velocity returns the derivative of the movement as a vector-based function.

Finally, as was the case for field types, all operations on nontemporal types are generalized (or lifted) for moving types. As an example, the '=' operator has lifted versions where one or both of its arguments can be moving types and the result is a moving boolean. Intuitively, the semantics of such lifted operations is that the result is computed at each time instant using the non-lifted operation.

When fields change across time, they are denoted *temporal*. Further, spatio-temporal OLAP (STOLAP) accounts for spatial objects evolving over time. Moving types, defined by moving($\alpha$) (where $\alpha$ is a spatial data type or a field type), allow to define SOLAP over continuous fields (STOLAP-CF). In order to give examples of STOLAP-CF queries, we consider the Temperature dimension of our running example in Fig. 3, which varies over time and space, as indicated by the f(🌍,🕓) pictogram.

Q8. For each landplot and each month, give a field computing the average temperature of the month at each point in the land plot

$$\begin{aligned}&\{l.\mathsf{number}, m.\mathsf{month}, \mathsf{temp} \mid \mathsf{LandPlot}(l) \wedge \mathsf{Month}(m) \wedge \\ &\quad \mathsf{first} = \min(\{t.\mathsf{date} \mid \mathsf{Time}(t) \wedge t.\mathsf{month} = m.\mathsf{id}\} \wedge \\ &\quad \mathsf{last} = \max(\{t.\mathsf{date} \mid \mathsf{Time}(t) \wedge t.\mathsf{month} = m.\mathsf{id}\} \wedge \\ &\quad \mathsf{temp} = \mathsf{avg}(\{\mathsf{atperiods}(\mathsf{atregion}(t.\mathsf{geometry}, l.\mathsf{geometry}), \\ &\qquad \mathsf{range}(\mathsf{first}, \mathsf{last})) \mid \mathsf{Temperature}(t)\})\}\end{aligned}$$

Variables first and last contain the first and last days of a month, respectively. The temperature field is restricted to an specific month with function atperiods, and to the geometry of the land plot with function at. A field tempMonth is computed for each land plot and month by applying the avg operator. Function atregion restricts the temperature fields to the points belonging to the geometry of the land plot.

Q9. Land plots of clay soil in the state of Utah with an average temperature of 20 °C in March 2009 and with suitability (at every point of the landplot) for a crop of wheat at June 1st, 2009 greater than 1.4.

$$\begin{aligned}&\{l.\mathsf{number} \mid \mathsf{LandPlot}(l) \;\wedge\; \exists s, \exists t, \exists y, \exists m, \exists c\, (\mathsf{State}(s) \;\wedge \\ &\quad \mathsf{SoilType}(t) \;\wedge \mathsf{Yield}(y) \;\wedge\; \mathsf{Time}(m) \;\wedge\; \mathsf{Crop}(c) \;\wedge \\ &\quad s.\mathsf{name} = \text{`Utah'} \;\wedge \mathsf{intersects}(l.\mathsf{geometry}, \mathsf{intersection}( \\ &\qquad s.\mathsf{geometry}, \mathsf{defspace}(\mathsf{at}(t.\mathsf{geometry}, \text{`Clay'})))) \;\wedge \\ &\quad \mathsf{favg}(\mathsf{avg}(\{\mathsf{atperiods}(\mathsf{at}(u.\mathsf{geometry}, l.\mathsf{geometry}), \mathsf{range}( \\ &\qquad 1/3/2009, 31/3/2009)) \mid \mathsf{Temperature}(u)\})) = 20 \;\wedge \\ &\quad y.\mathsf{landPlot} = l.\mathsf{id} \wedge y.\mathsf{time} = m.\mathsf{id} \wedge m.\mathsf{date} = 1/6/2009 \wedge \\ &\quad y.\mathsf{crop} = c.\mathsf{id} \;\wedge\; c.\mathsf{type} = l.\text{`Wheat'} \;\wedge \\ &\quad \mathsf{defspace}(\mathsf{at}(y.\mathsf{suitability}, \mathsf{range}(1.4, 10))) = l.\mathsf{geometry})\}\end{aligned}$$

Function intersection computes the region of clay soil in Utah, and intersects verifies that the land plot overlaps this region. In the inner query, the temperature field, restricted to the geometry of the landplot and to March 2009, is aggregated with the avg operator, resulting in a field to which the field aggregation operator favg is applied to obtain the average as a real value, which is then compared to 20. After obtaining the instance of the fact relationship relating the landplot, the date(June 1st 2009), and the wheat crop, the suitability field for this instance is restricted to the points that have a value in the range [1.4,10], the region containing those points is obtained with function defspace, and it is verified that this region equals the geometry of the landplot, ensuring that every point satisfies the condition.

We now formally define the class of STOLAP-CF queries.

*Definition 7. (STOLAP-CF queries)* Let us call $\mathcal{R}_{agg}^{\xi\phi\Theta}$ the language $\mathcal{R}_{agg}$ augmented with spatial types $\xi$, field types $\phi$ and moving spatial types $\Theta$. The class of STOLAP-CF queries contains all the queries expressed by $\mathcal{R}_{agg}^{\xi\phi\Theta}$.

## 6. DISCRETE MODELS

In this section we study discrete models for implementing the abstract model of Section 3.

### 6.1 Spatial Fields

We consider two discrete models for spatial fields. The first one is based on a set of points defining the vertices of a regular square grid of size $\delta$. Each point (or vertex) has an associated value for the field. Bilinear interpolation is used for obtaining the value of the field at a point located between the vertices of a grid cell. A second discrete model is based on triangulated irregular networks (or TINs) defined by a set of irregularly distributed nodes and lines that are arranged in a network of non-overlapping triangles. The value of the function at a point is obtained from a linear interpolation from the values of the vertices of its containing triangle. The choice of these two models is based on current methods and tools that manipulate field (or raster) data. Other discrete models may be designed to suit particular applications. Note that the models and interpolation methods discussed here are included in the OpenGIS specification for coverage geometry and functions of the OGC [19].

We begin by defining a set $SPoint_\beta$ whose elements describe sample 2D points with an associated value from a base type $\beta$ (i.e., an integer, real, boolean, or string value). Let us denote $D_\beta$ the carrier set of type $\beta$.

*Definition 8. ($SPoint_\beta$)*

$$SPoint_\beta = \{(x, y, z) \mid x, y \in D_{Point} \wedge z \in D_\beta\}$$

Here, $D_{Point}$ is the carrier set that implements the type Point in the discrete model. The type $SPoint_\beta$ allows us to define the two alternative discrete models for spatial fields, described next.

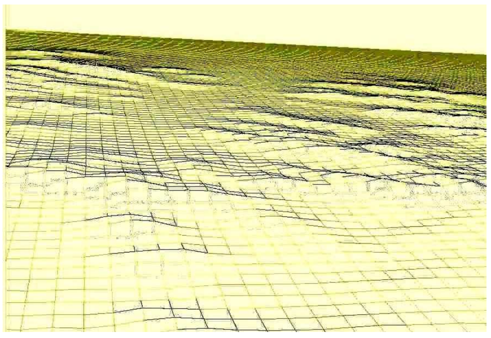

Figure 4: A grid representing altitude.

***Regular Grids.*** We first define the carrier set for type $Grid_\beta^\delta$.

*Definition 9. (The carrier set of $Grid_\beta^\delta$)*

$$D_{Grid_\beta^\delta} = \{P \mid P \subset SPoint_\beta \wedge P \text{ defines the vertices of a regular square grid of size } \delta \text{ in } \mathbb{R}^2\}$$

A set of points $P$ defines a regular square grid of size $\delta$ in $\mathbb{R}^2$ if for each pair of adjacent points $p_i, p_j, p_i \neq p_j$, either $p_i = (x_i, y_i)$, $p_j = (x_i \pm \delta, y_i)$, or $p_i = (x_i, y_i)$, $p_j = (x_i, y_i \pm \delta)$. Figure 4 illustrates a grid obtained using this model to represent altitude.

The semantics of this type is defined by means of a bilinear interpolation, i.e., an interpolation performed over the $x$ and $y$ axes. Formally, given four points $P_1, .., P_4 \subset SPoint_\beta$, such that $P_1 = (x_1, y_1, z_1)$, $P_2 = (x_1 + \delta, y_1, z_2)$, $P_3 = (x_1, y_1 + \delta, z_3)$, $P_4 = (x_1 + \delta, y_1 + \delta, z_4)$, to obtain the value $z$ of a point $P_{i,j} = (x + \Delta_x, y + \Delta_y, z)$ we first linearly interpolate in the $x$-direction, yielding two points $P_i = (x_1 + \Delta_x, y_1, z_i)$, and $P_k = (x_1 + \Delta_x, y_1 + \delta, z_k)$, such that $z_i = z_1 * \frac{\delta - \Delta_x}{\delta} + z_2 * \frac{\Delta_x}{\delta}$. Analogously, $z_k = z_3 * \frac{\delta - \Delta_x}{\delta} + z_4 * \frac{\Delta_x}{\delta}$. Finally, the value $z$ corresponding to $P_{i,j}$ is $z = z_k * \frac{\delta - \Delta_y}{\delta} + z_4 * \frac{\Delta_y}{\delta}$.

***Triangulated Irregular Networks.*** We now define the discrete model based on TINs. For this, we first define a set $STriangle_\beta$. Elements in this set describe sample triangles whose vertices are in $SPoint_\beta$. These triangles are typically obtained through a Delaunay triangulation [7]. In this method, triangles are generated such that no point lies inside the circumcircle of any triangle (the unique circle that passes through each of the triangle's vertices).

*Definition 10. (The carrier set of $TIN_\beta$)*

$$STriangle_\beta = \{(p_1, p_2, p_3) \mid p_1, p_2, p_3 \in SPoint_\beta\}$$

$$D_{TIN_\beta} = \{T \mid T \subset STriangle_\beta \wedge T \text{ defines a tesselation of } \mathbb{R}^2\}$$

A set $T$ of triangles defines a tesselation of $\mathbb{R}^2$ if

1. $\forall t_1, t_2 \in T, t_1^\circ \cap t_2^\circ = \emptyset$, where $t_1^\circ$ denotes the interior of $t_1$
2. $\bigcup_{i=1}^{|T|} t_i = \mathbb{R}^2$

The semantics of this type is defined by linear interpolation of the plane surfaces described by the triangles. Formally, a plane surface defined by three points $P_1 = (x_1, y_1, z_1)$, $P_2 = (x_2, y_2, z_2)$, and $P_3 = (x_3, y_3, z_3)$ has a formula that may be expressed in matrix determinant form by:

$$\begin{vmatrix} x & y & z & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix} = 0$$

Points in the planar triangle defined by the three $(x_i, y_i)$ coordinates (including the vertices) may be interpolated by evaluating the determinant of the above matrix.

## 6.2 Spatio-Temporal Fields

We now generalize the discrete models previously presented to spatio-temporal fields. This is realized by applying the discrete model of Güting *et al.* [10] to the temporal dimension. In short, the approach consists in representing spatio-temporal types by a so-called *sliced representation.* The idea is to decompose the temporal development of a type along the time dimension into fragment intervals called *slices*, such that within each slice this development can be represented by some kind of "simple" function. This is realized by defining for each type $\alpha$ a *unit type*, a pair consisting of a time interval and a value describing the "simple" function. Unit types for the base types and spatial types are defined as in [10]. For example, the unit type ureal represents moving real values. The carrier set for this type is

$$D_{\mathsf{ureal}} = Interval(Instant) \times \{(a, b, c, r) \mid a, b, c \in \mathsf{real}, r \in \mathsf{bool}\}.$$

The semantics of a unit type is given by a function of time $\iota$ defined during the unit interval. In the case of ureal, this semantics is given by

$$\iota((a, b, c, r), t) = \begin{cases} at^2 + bt + c & \text{if } \neg r \\ \sqrt{at^2 + bt + c} & \text{if } r. \end{cases}$$

Therefore, a ureal value can represent a quadratic polynomial in $t$ or a square root of such a polynomial.

We begin by generalizing the regular grid concrete model for the spatio-temporal case. For this we define a set $UPoint_\beta$ whose elements describe sample 2D points with an associated couple of values from a base type $\beta$, and then define the carrier set for type $UGrid_\beta^\delta$.

*Definition 11. ($UPoint_\beta$)*

$$UPoint_\beta = \{(x, y, z_1, z_2) \mid x, y \in D_{Point} \wedge z_1, z_2 \in D_\beta\}$$

*Definition 12. (The carrier set of $UGrid_\beta^\delta$)*

$$D_{UGrid_\beta^\delta} = Interval(Instant) \times \{P \mid P \subset UPoint_\beta \wedge P \text{ defines the vertices of a regular square grid of size } \delta \text{ in } \mathbb{R}^2\}$$

A value $((t_1, t_2), \{P_1, \ldots, P_n\})$ where $P_i = (x, y, z_1, z_2)$ represents that at point $(x, y)$ the value of the field changed from $z_1$ to $z_2$ during the time interval $[t_1, t_2]$. Formally, each unit point $P_i$ defines a linear interpolation given by

$$\iota((t_1, t_2), (x_1, x_2, z_1, z_2), t) = mt + z_1 - mt_1$$

where $m = \frac{y_2 - y_1}{x_2 - x1}$ is the slope of the straight line connecting the points $(t_1, z_1)$ and $(t_2, z_2)$.

There are at least two possible generalizations of the TIN concrete model for the spatio-temporal case. In the first one, the vertices of the triangles do not evolve on time, only the value of the field evolves at each vertex does. In an alternative generalization, computationally more complex, the vertices of the triangles evolve on time as well as the value of the field on each vertex. For the sake of space we only consider next the first generalization.

*Definition 13. (The carrier set of $UTIN_\beta$)*

$$UTriangle_\beta = \{(P_1, P_2, P_3) \mid P_1, P_2, P_3 \in UPoint_\beta\}$$

$$D_{UTIN_\beta} = Interval(Instant) \times \{T \mid T \subset UTriangle_\beta \wedge \\ T \text{ defines a tesselation of } \mathbb{R}^2\}$$

A value $((t_1, t_2), \{T_1, \ldots, T_n\})$ where $T_i = (P_1, P_2, P_3)$ and $P_j = (x_j, y_j, z_1^j, z_2^j)$ represents the fact that for each triangle $T_i$ the planar surface defined by its three vertex points $(x_j, y_j)$ changed linearly its slope during the time interval. More formally, the planar surface defined by $T_i$ at an instant $t$ is obtained from the linear interpolation of the unit points composing its vertices, as given by Definition 12.

## 7. CONCLUSION AND FUTURE WORK

We have presented a conceptual multidimensional model for spatial data analysis, where continuous fields support is achieved extending the MultiDim model for spatial OLAP [16], with a *field* data type, following the approach of Güting *et al.* [10]. We also provided a thorough analysis of the operators associated to this data type. In addition, we defined the classes of SOLAP-CF and STOLAP-CF queries, i.e., non-temporal and temporal multidimensional queries including continuous fields, respectively. We discussed possible discrete implementations of the abstract model, for spatial and spatio-temporal fields. Our future research direction is headed to the implementation of a prototype based on the models studied in this paper.

**Acknowledgments.** This research has been partially funded by the Research Foundation Flanders, project G.0344.05, the European Union under the FP6-IST-FET programme, Project n. FP6-14915, GeoPKDD: Geographic Privacy-Aware Knowledge Discovery and Delivery, and the Argentina Scientific Agency, project PICT 2004 11-21.350.

## 8. REFERENCES

[1] T. O. Ahmed. Continuous spatial data warehousing. In *9th International Arab Conference on Information Technology*, 2008.

[2] T. O. Ahmed and M. Miquel. Multidimensional structures dedicated to continuous spatiotemporal phenomena. In *BNCOD*, pages 29–40, 2005.

[3] C. Bailey-Kellogg, F. Zhao, and K. Yip. Spatial aggregation: Language and applications. In *AAAI/IAAI, Vol. 1*, pages 517–522, 1996.

[4] Y. Bédard, S. Rivest, and M. Proulx. Spatial online analytical processing (SOLAP): Concepts, architectures, and solutions from a geomatics engineering perspective. In *Data Warehouses and OLAP: Concepts, Architectures and Solutions*, chapter 13, pages 298–319. IRM Press, 2007.

[5] G. Câmara, D. Palomo, R. C. M. de Souza, and D. de Oliveira. Towards a generalized map algebra: Principles and data types. In *GeoInfo*, pages 66–81, 2005.

[6] J. P. Cordeiro, G. Câmara, U. F. Moura, C. C. Barbosa, and F. Almeida. Algebraic formalism over maps. In *GeoInfo*, pages 49–65, 2005.

[7] B. Delaunay. Sur la sphère vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, 7(6):793–800, 1934.

[8] R. Elmasri and S. Navathe. *Fundamentals of Database Systems*. Addison-Wesley, fifth edition, 2007.

[9] E. Erwig and M. Schneider. Formalization of advanced map operations. In *International Symposium on Spatial Data Handling*, pages 3–17, 2000.

[10] R. H. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.

[11] ISO TC 211. Geographic information – Schema for coverage geometry and functions. ISO 19123:2005, 2005.

[12] K. K. Kemp. Fields as a framework for integrating gis and environmental process models. *Transactions in GIS 1(3)*, pages 219–246, 1997.

[13] R. Kimball. *The Data Warehouse Toolkit*. J. Wiley and Sons, Inc., 1996.

[14] A. Klug. Equivalence of relational algebra and relational calculus query languages having aggregate functions. *Journal of the ACM*, 29(3):699–717, 1982.

[15] M. Kumler. An intensive comparison of triangulated irregular networks (tins) and digital elevation models (dems). *Cartographica*, 31(2):1–99, 1994.

[16] E. Malinowski and E. Zimányi. *Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications*. Springer, 2008.

[17] J. Mennis, R. Viger, and C. Tomlin. Cubic map algebra functions for spatio-temporal analysis. *Cartography and Geographic Information Science, 32(1)*, pages 17–32, 2005.

[18] Open Geospatial Consortium Inc. OpenGIS Implementation Specification: Grid Coverage. OGC 01-004, Version 1.00, 2001.

[19] Open Geospatial Consortium Inc. OpenGIS Abstract Specification: Topic 6: The Coverage Type and its Subtypes. OGC 07-011, Version 4, 2007.

[20] L. Paolino, G. Tortora, M. Sebillo, G. Vitiello, and R. Laurini. Phenomena: a visual query language for continuous fields. In *GIS*, pages 147–153, 2003.

[21] C. Parent, S. Spaccapietra, and E. Zimányi. *Conceptual Modeling for Traditional and Spatio-Temporal Applications: The MADS Approach*. Springer-Verlag New York, Inc., NJ, USA, 2006.

[22] S. Prasher and X. Zhou. Multiresolution anlgamation dynamic spatial data cube generation. In *Proceedings of the 15th Australasian Database Conference, ADC 2005*, pages 103–111, Dunedin, New Zealand, 2004.

[23] S. Rivest, Y. Bédard, and P. Marchand. Toward better suppport for spatial decision making: Defining the characteristics of spatial on-line analytical processing (SOLAP). *Geomatica*, 55(4):539–555, 2001.

[24] J. Shanmugasundaram, U. M. Fayyad, and P. S. Bradley. Compressed data cubes for olap aggregate query approximation on continuous dimensions. In *KDD*, pages 223–232, 1999.

[25] S. Shekhar and S. Chawla. *Spatial Databases: a Tour*. Prentice Hall, 2003.

[26] D. Tomlin. *Geographic Information Systems and Cartographic Modelling*. Prentice-Hall, 1990.

[27] A. Vaisman and E. Zimányi. What is spatio-temporal data warehousing? In *11th International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, 2009.