A Generic Data Model and Query Language for Spatiotemporal OLAP Cube Analysis

Leticia I. Gómez
Instituto Tecnológico de Buenos Aires
Buenos Aires, Argentina
lgomez@itba.edu.ar

Silvia A. Gómez
Instituto Tecnológico de Buenos Aires
Buenos Aires, Argentina
sgomez@itba.edu.ar

Alejandro A. Vaisman
Université Libre de Bruxelles
Brussels, Belgium
avaisman@ulb.ac.be

ABSTRACT

Nowadays, organizations need to use OLAP (On Line Analytical Processing) tools together with geographical information. To support this, the notion of SOLAP (Spatial OLAP) arouse, aimed at exploring spatial data in the same way as OLAP operates over tables. SOLAP however, only accounts for discrete spatial data. More sophisticated GIS-based decision support systems are increasingly being needed, to handle more complex types of data, like continuous fields. Fields describe physical phenomena that change continuously in time and/or space (e.g.,temperature). Although many models have been proposed for adding spatial information to OLAP tools, no one allows the user to perceive data as a cube, and analyze any type of spatial data, continuous or discrete, together with typical alphanumerical discrete OLAP data, using only the classic OLAP operators (e.g., Roll-up, Drill-down). In this paper we propose an algebra that operates over data cubes, independently of the underlying data types and physical data representation. That means, in our approach, the final user only sees the typical OLAP operators at the query level. At lower abstraction levels we provide discrete and continuous spatial data support as well as different ways of partitioning the space. We also describe a proof-of-concept implementation to illustrate the ideas presented in the paper. As far as we are aware of, this is the first proposal that allows analyzing discrete and continuous spatiotemporal data and OLAP cubes together, using just the traditional OLAP operations, thus providing a very general framework for spatiotemporal data analysis.

Categories and Subject Descriptors

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

General Terms

Theory

EDBT 2012, March 26–30, 2012, Berlin, Germany.

Keywords

Fields, OLAP, SOLAP, Map Algebra

1. INTRODUCTION

OLAP (On Line Analytical Processing) [10] comprises a set of tools and algorithms that allow efficiently querying multidimensional databases containing large amounts of data, usually called Data Warehouses. In this multidimensional model, data can be perceived as a *cube*, where each cell contains a measure or set of measures of interest. At the logical level, OLAP data are typically organized as a set of dimensions and fact tables. Modern organizations need to use OLAP analytical capabilities together with geographical information. In this direction, SOLAP (standing for Spatial OLAP), a concept introduced by Rivest et al. [17], aims at exploring spatial data by drilling on maps, in the same way as OLAP operates over tables and charts. However, SOLAP only accounts for discrete spatial data, where spatial objects are represented as geometries. More sophisticated GIS-based decision support systems are increasingly being needed, able to handle more complex types of data, like continuous fields (from now on, fields). Fields describe physical phenomena that change continuously in time and/or space, like temperature, land elevation, land use and population density, frequently used in human geography [23]. They are perceived as having a value at each point in a continuous N-dimensional spatial and/or spatiotemporal domain. Moreover, a field can be used to describe the names of countries and for those points that do no belong to any country a distinguished value can be assigned. In real-world practice, scientists and practitioners register the values of a field by taking samples at (generally) fixed locations, and inferring the values at other points in space using some interpolation method. Thus, fields can be described by a function that indicates the distribution of the phenomena or feature of interest. The most popular discrete representation for fields is the raster model, where the 2D space is divided into regular squares. The raster model is frequently used for representing soil type, temperature, among other physical phenomena. Other representations have also been proposed, like the Voronoi diagrams [13, 21] or TIN, the latter usually employed to represent an 'Altitude' field [12].

For adding spatial information to OLAP tools, many models have been proposed, at the conceptual and logical levels. We find this unnecessary from the user's point of view. We believe that a user of a spatio-temporal enabled OLAP system would like to find the data cube at the conceptual level, independently of the kind of underlying data. Such a model

would allow to analyze any type of spatial data, continuous (independently of the underlying representation) or discrete, together with typical alphanumerical discrete OLAP data, using only the classic OLAP operators, like Roll-up, Drill-down, and/or Drill-across. To achieve this, at the logical and physical levels we need different mechanisms to manage these different kinds of data and data representations.

In light of the above, in this paper we present an algebra that operates over data cubes, independently of the underlying data types and physical data representation. That means, the final user only sees the typical OLAP operators at the user query level. At the logical level, we describe operations that allow to manage different kinds of spatial data. For this, we present a discrete model for continuous fields that allows us to considered them as standard data cubes with one-level dimensions. Further, based the former, and on the work of Abelló *et al.* [1], we define mapping mechanisms that allow to analyze together cubes containing alphanumerical and spatial (discrete and continuous) data, providing support to the conceptual model. Finally, at the physical level, a model that allows representing continuous data in a discrete way, and a collection of operators over this model, gives support to our proposal at the physical level (that means, at the logical level we do not care about how continuous data are represented). Finally, we provide a proof-of-concept implementation to illustrate the ideas presented in the paper. As far as we are aware of, our proposal is the first one that allows analyzing discrete and continuous spatiotemporal data and OLAP cubes together, using the traditional OLAP operations, thus providing a very general framework for spatiotemporal data analysis.

1.1 Motivation

We motivate our work with an example from the wine industry that we use throughout the paper. John, who owns a vineyard, stores information about grape production in SOLAP cubes (i.e., cubes that include spatial dimensions). To make well-informed decisions, he needs to incorporate external information about precipitation, temperature, altitude and soil type (i.e., physical phenomena stored as *fields*). He knows that climate changes impact directly on wine production. More specifically, metrics such as average growing season temperatures can be used for establishing optimum regions for wine production, for each wine variety [9]. For example, Cabernet Sauvignon is produced in regions that span from intermediate to hot climates, with temperatures ranging from 16.5^0C to 19.5^0C in the growing season.

As a concrete example, let us consider the following scenario. The quality of the wine for each type of grape depends on the specific climatic conditions for *periods* such as bud break, flowering, ripeness and harvest. Thus, John performs a cross-analysis of the production of each kind of grape with the temperature amplitude and average precipitation during those periods, for each country region. Technically, the latter information is stored as fields. Since different kinds of data (alphanumerical, discrete spatial, and spatio-temporal fields) are involved, standard approaches require using different tools and languages, and integrating the information in an ad-hoc manner. Instead, with our approach he will only deal with typical OLAP operators, without caring about the type of data stored in a cube, and how these data are stored at the physical level. A cross-analysis like the above would be expressed as (we explain this in detail later in the paper):

```
DrillAcross(
  Dice(
    RollUp(
      DrillAcross(
        DrillAcross(
          RollUp(tempField,geoDim,region,MIN),
          RollUp(tempField,geoDim,region,MAX)),
        RollUp(precipField,geoDim,region,AVG)),
      dateDim,month,AVG),
    month>=Feb-2010 and month<=Sep-2010),
  RollUp(RollUp(grapeSOLAP,dateDim,month,AVG),
         geoDim,region,AVG),
  AVG)
```

That means, John only sees cubes (i.e., tempField, precipField, and grapeSOLAP), and does not care about the data these cubes contain. The distinction of data types and data representations is performed at the logical and physical levels (e.g., temperature can be a field *type* at the logical level, and stored as a rasterized grid at the physical level). We also explain later the key role of the Drill-Across operator for relating cubes of different kinds. As far as we are aware of, this is a completely novel approach in the OLAP field.

1.2 Related Work

The joint contribution of the GIS and database communities to the problem of analyzing fields using OLAP models has been limited. Shanmugasundaram *et al.* [18] 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. This representation reduces the storage requirements, but continuity is addressed in a limited way. Few proposals address the integration of field and OLAP data. Ahmed and Miquel [3] discuss the importance of modeling multidimensional structures for field-based data and analyze how either cell values or interpolation methods can be used for inferring values at non-sampled points. Nevertheless, neither formal definitions are provided nor methods for calculating aggregating functions in the continuous cube are described. Vaisman and Zimanyi [20] present a conceptual model for SOLAP that supports dimensions and measures representing continuous fields, and characterize multidimensional queries over fields. They define a field data type, a set of associated operations, and a multidimensional calculus supporting this data type. Gomez et *al.* [6] proposed physical data structures for implementing this set of operators. Therefore, these proposals operate at the logical and physical levels, respectively. Bimonte *et al.* [4] recently introduced a multidimensional model that supports measures and dimension as continuous field data, although they do not consider the fields as OLAP cubes, neither integrate the continuous and discrete models with an unique set of OLAP operators. In a recent work, the authors of the present paper [5] sketched a closed generic map algebra over spatio-temporal continuous fields, independent of the underlying field representation, and how this idea can be used for an integrated analysis of spatial and OLAP data. Here we generalize and formalize these ideas, and present a complete framework for analysis, along with a proof-of-concept implementation.

Regarding algebras for field analysis, Tomlin [19] proposed *Map Algebra*, an informal language for manipulating two-dimensional fields, represented as raster data. This language

basically consists in three operators denoted *Local*, *Focal* and *Zonal*. For instance, given a collection of grids, and an aggregate function, the *Local* operator returns a grid such that the value in each cell is the result of applying the aggregate function to values of the cell at the same location in the input grids. Map Algebra operators have been implemented in different GIS (Geographic Information Systems), like ArcGIS[1]. Mennis and Viger [14] generalized the concept of Map Algebra, adding the temporal dimension. More recently, Mennis [16] proposed the Multidimensional Map Algebra, an extension of Map Algebra that allows working with raster data of different dimensionality. The proposal also discusses the local, focal, and zonal operators in this mixing of raster data and how to define neighborhoods and lags. For example, a Grid and a 4D HyperCube can participate of operations.

Regarding continuous data representation, the main drawback of the raster model is that the value assigned to a cell may represent *many* sampled points (i.e., the actual sampled values are lost). That means, the space is arbitrary partitioned (technically, tessellated) without considering the sampled points. Therefore, alternatives to raster data are discussed in the literature. For example, Ledoux and Gold [13] argue that using *Voronoi diagrams* for representing fields has several advantages, and they redefine the local, focal and zonal operations when space is tessellated using Voronoi diagrams. Ledoux [21] proposes a solution for creating a 3D discrete Voronoi diagram and discuss its implementation. Other discrete representation exist (like TIN, mentioned in Section 1).

1.3 Contributions and Paper Organization

Existing proposals for adding support to continuous fields require the definition of a new data model. This actually also happens with many of the proposals for Spatial-OLAP (SOLAP). The direct consequence of this is that each time a new kind of data needs to be added, new models need to be developed. Our approach, on the contrary, focuses in keeping unchanged the traditional OLAP cube model and operators at the conceptual model, defining polymorphic operators at the logical level, and caring about the data representation at the physical level. In this sense, our approach can support also other kinds of data representations, beyond the ones addressed in this paper. We organize the presentation as follows. We start introducing a multidimensional data model over which we will build our proposal (Section 2). We then present a discrete data model for representing continuous fields and an algebra that makes use of these operators (Section 3). This data model is general enough to support any kind of underlying data representation (e.g., Voronoi, Raster, TIN). In Section 4 we show that actually fields can be considered as another kind of data cube. We detail how each typical OLAP operator can be applied over field cubes, and how a mapping mechanism allows combining cubes of different kinds thanks to the Drill-Across operator. To show the plausibility of our approach, in Section 5 we present a proof-of-concept implementation over a PostgreSQL database. We conclude in Section 6.

2. MULTIDIMENSIONAL MODEL

Many formal models for OLAP had been proposed ([7, 8, 22]). We now introduce the one that we will use in the remainder of the paper, although most of the existing models can be used as a basis for our work.

DEFINITION 1 (DIMENSION SCHEMA). *A* dimension schema *is a tuple* $\langle nameDS, \mathcal{L}, \rightarrow \rangle$ *where: (a) nameDS is the name of the dimension; (b)* $\mathcal{L}$ *is a non-empty finite set of levels, which contains a distinguished level denoted All; (c) Each level* $l \in \mathcal{L}$ *has associated a non-empty finite set of level descriptors* $LD(l)$*, each one of them having domain* Dom_{ld_i}*; (d)* $\rightarrow$ *is a partial order called a rollup relation on the levels in* $\mathcal{L}$*. This partial order defines a hierarchy in the dimension; (e) The reflexive and transitive closure of the rollup relation has a unique bottom level and a unique top level. The top level is the distinguished level All. A* spatial dimension schema *is a dimension schema where at least one level* $l \in \mathcal{L}$ *has exactly one level description with domain of type geometry (e.g. point, polygon).* □

DEFINITION 2 (CUBE SCHEMA). *A cube schema is a tuple* $\langle nameCS, D, M \rangle$ *where nameCS is the name of the cube, D is a finite set of dimension schemas, and M is also a finite set of levels called measures.* □

DEFINITION 3 (DIMENSION INSTANCE). *A dimension instance I of a dimension schema* $< nameDS, \mathcal{L}, \rightarrow >$ *consists of: (a) A finite set of members for each level* $\in \mathcal{L}$ *where each member is identified by an ID. The level All has a unique member, 'all'; (b) A mapping function* Val_{ld} *mapping level descriptor names to values in their domain; (c) A set of rollup relations of the form* $Rollup_{l_i \rightarrow l_j}$ *from the members of level* l_i *to the members of level* l_j*, for each pair of levels* l_i *and* l_j *in* $\mathcal{L}$*.* □

DEFINITION 4 (CUBE INSTANCE). *Given a cube schema* $\langle nameCS, Dims, Measures \rangle$ *such that* $|Dims| = D$*, and* $|Measures| = M$*, a dimension instance* I_i *for each* $d_i \in Dims$*,* $i = 1..D$*, and a set Points=*$\{(c_1, \ldots, c_D) |$ c_i *is a member of* $I_i, i = 1..D\}$*. A cube instance C is a partial function (let us call it data cube) with signature: dom(Points)* $\rightarrow$ *dom(*m_1*)* $\times$ *...* $\times$ *dom(*m_M*) where* $m_i \in Measures$*, i=1..M, mapping members in Points to members in Measures.* □

EXAMPLE 1 (CUBE SCHEMA AND INSTANCE). *Our producer John needs to analyze the harvest production (a measure), along different dimensions, namely the crop block and the picking date. Let us call these dimensions blockDim and dateDim, respectively. The former is a spatial dimension since it contains level descriptors with domain geometry. In order to achieve this goal, John builds a cube, denoted grapeCS, as follows.*

Grape cube schema: $\langle grapeCS$, $\langle dateDim$, $\{date, month, year, All\}$, $\{date \rightarrow month, month \rightarrow year, year \rightarrow All\}\rangle$, $\langle blockDim, \{block, district, province, g, gT, All\}$, $\{block \rightarrow district, district \rightarrow province, province \rightarrow All, block \rightarrow g, g \rightarrow gT, gT \rightarrow All\}\rangle$, $\{harvest\}\rangle$.

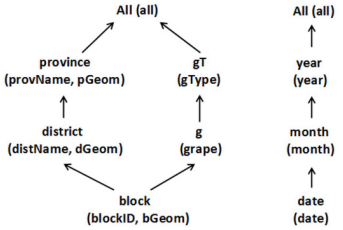
The level descriptors are: $LD(block)$=$\{blockID, bGeom\}$, $LD(district)$=$\{distName, dGeom\}$, ..., $LD(g) = \{grape\}$, $LD(gT) = \{gType\}$, ..., $LD(date)$=$\{date\}$*. Fig. 1 shows both dimension schemas.*

Table 1 depicts a relational instance of grapeCS, for blocks located in Belgium. □

[1] http://www.esri.com/software/arcgis/index.html

dateDim				blockDim									measures
date	month	year	All	blockID	bGeom	distName	dGeom	provName	pGeom	grape	gType	All	harvest
2009-08-27	Aug-09	2009	all	58381	geo01	Turnhout	geo15	Antwerpen	geo70	Chardonnay	White	all	7693
2009-10-15	Oct-09	2009	all	59315	geo47	Mechelen	geo11	Antwerpen	geo70	Kerner	White	all	7736
2009-08-19	Aug-09	2009	all	58394	geo31	Mechelen	geo11	Antwerpen	geo70	Pinot Noir	Red	all	7009
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2009-10-19	Oct-09	2009	all	58996	geo57	Leuven	geo43	Brabant	geo24	Kerner	White	all	6086
2009-08-12	Aug-09	2009	all	59025	geo88	Nijvel	geo30	Brabant	geo24	Pinot Blanc	White	all	7428

Table 1: grapeCS instance



(a) blockDim (b) dateDim

Figure 1: (a) crop dimension (b) date dimension

2.1 OLAP Cube Operators

Our proposal for an OLAP model that seamlessly supports different kinds of spatio-temporal analysis, places the data cube at the conceptual level. Thus, we briefly review the OLAP operators, with emphasis on the Drill-Across operator, which plays a crucial role in our approach.

Dice. Selects values of domain in dimensions or measures that satisfy a boolean condition σ, since both of them can be treated interchangeably [2, 7].

Slice. Reduces the dimensionality of a cube by removing one of its dimensions, i.e., a cube of N-1 dimensions is obtained from a cube of N dimensions. In order to discard the selected dimension, it must contain just one value in its domain. Thus, if the dimension has more than one value two approaches can be used: we can apply the Roll-Up (see below) operator for summarizing into a singleton [2], or the Dice operator for choosing only one value.

Roll-Up and Drill-Down. A Roll-up operator aggregates measures according to a dimension hierarchy, to obtain measures at a coarser granularity for a given dimension, based on the use of a preexisting hierarchy over this dimension. Drill-Down de-aggregates previously summarized measures and can be considered the inverse of Roll-Up. Following Agrawal et al. [2], we consider Drill-Down a high-level operation that can be implemented by tracking the (stored) paths followed during user navigation.

Drill-Across. Relates information contained in two data cubes having the same dimensions. Thus, measures from different cubes can be compared. According to Kimball [11], Drill-Across can only be applied when both cubes have not only the same schema dimensions but also the same instances. However, other authors relax this restriction. This is the approach of Abelló et al. [1], which we follow here, using two concepts: (a) *Dimension-Dimension Derivation*: Used when two dimensions come from a common concept although their structures differ. For example, they do not have the same levels because their granularities are not the same. This would be the case of a spatial dimension with 'point' granularity, and another one with 'polygon' granularity. A solution consists in rolling up the former to a common level. Even, if necessary, a new level could be introduced. (b) *Dimension-Dimension Association*: Corresponds to the case in which two cubes have different dimensions, but one of them could be defined as the association of several ones. For example in one cube we define latitude and longitude as separated dimensions; in another one we store only one dimension containing the 'point' geometry. A mapping function can solve this problem. To follow the latter approach, we need to formalize a semantic mapping between dimensions that will the allow to build a new data cube from two other ones, such that measures in both cubes appear together in the new one.

Definition 5 (Levels for Mapping). *Given a a cube schema $\langle nameCS, Dims, Measures\rangle$, a Level for Mapping, denoted LM(nameCS), is the set $\{(d,l)|d \in Dims$ and $l \in Levels(d)\}$, and this set satisfies the following condition: every dimension appears exactly once in the semantic mapping LM(nameCS).* □

Definition 6 (Cube Semantic Mapping). *Given two cube instances c1 and c2 with schema $\langle nameCS_1, Dims_1, Measures_1\rangle$ and $\langle nameCS_2, Dims_2, Measures_2\rangle$, and two sets $LM(nameCS_1)$ and $LM(nameCS_2)$, a cube semantic mapping SM(c1, c2), consists of: (a) A cube schema mapping for identifying $p1 \in LM(nameCS_1)$ that can be semantically matched to $p2 \in LM(nameCS_2)$ without using those pairs more than once in this mapping. (b) A boolean function that expresses the semantic equivalence among the members of the matched levels (in the cube schema mapping). This is the cube instance mapping.* □

Property 1 (one to one). *A cube schema mapping must satisfy: (a) $SM(p1,p2) = SM(p'1,p2) = true$ iff $p1 = p'1$; and (b) $SM(p1,p2) = SM(p1,p'2) = true$ iff $p2 = p'2$. This assures that there not exists one coordinate in one instance cube that can be mapped to more than one coordinate in the other instance cube.* □

Example 2 (Drill-Across). *Let us consider two cubes c1 and c2 with cube schemas cs1 and cs2, respectively. Cube cs1 contains the measure sales and two spatial dimensions D1 (with level description $dist_x$) and D2 (with level description $dist_y$). Both dimensions represent districts as points. Cube cs2 contains the measure population and a spatial dimension SD with two levels, dist and prov, (both with level description* geom *of type Polygon), such that dist → prov. To apply the Drill-Across operation between both cubes, we*

need to define $LM(cs1) = \{(D1, dist_x), (D2, dist_y)\}$ *and* $LM(cs2) = \{(SD, dist)\}$. *Therefore,* $SM(c1, c2)$ *specifies: (a) a cube semantic mapping between the first two pairs of LM(cs1) and the pair in LM(cs2); (b) a cube instance mapping given by the boolean function "contains(geom, Point* $(dist_x, dist_y)$*)", where geom is the descriptor of the level in LM(cs2). The Drill-Across defined in this way satisfies the one-to-one property, since although members in c1 are defined by Points (coordinates) and members in c2 are defined by Polygons, both of them represent the same concept.*

Suppose now that dimension SD contains only the level province. It would be incorrect to perform a Drill-Across between c1 and c2 since the one-to-one property is not satisfied. We cannot build a new cube with dimensions D1 and D2 (district concept) where members give context to measures recorded at different granularities (sales at the district level and population at the province level). Table 2 shows both cubes, where L_x, L_y, N_x, N_y, T_x *and* T_y *refer to the coordinates of Leuven, Nijvel and Turnhout, respectively. Also,* g_{Ant} *and* $g_{Brabant}$ *refer to the geometries of Antwerpen and Brabant, respectively. Thus, to perform a Drill-Across in this example, we must first combine dimensions D1 and D2 in c1 into only one, second to introduce a new level that represent province, and finally roll-up to the new level (to avoid the problems shown in Table 2).* □

3. DISCRETE DATA MODEL FOR FIELDS

A key point in our proposal requires showing that spatial continuous data could be seen as a data cube. Since cubes are discrete structures, we need a formal data model for representing continuous fields (CF) in a discrete way. Moreover, this model must be independent of the underlying representation of continuous data, in order to provide the required level of abstraction. Formally, a Continuous Field F consists in: (a) A continuous domain D; (b) A range of values R; (c) A mapping function $f : D \to R$. Since operators are performed over a bounded domain, the notion of Bounded Continuous Field is first introduced.

Definition 7 (Bounded CF). *A Bounded Continuous Field BF consists in: (a) a domain* $Dom(BF) = Dom_1(BF) \times \ldots\ldots \times Dom_N(BF)$, *where* Dom_i *is a closed interval in* $\mathbb{R}$, $\forall i, i = 1..N$; *(b) a list of N labels* $Labels(BF) = \langle l_1, \ldots, l_n \rangle$, *to describe semantically each dimension in the domain, where* $l_i \neq l_j$, $\forall i, i = 1..n$, $\forall j, j = 1..n$. *(c) a set of values* $R \bigcup \{\perp\}$, *denoted* $Range(BF)$, *where* $\perp$ *is a distinguished value that does not belong to R; (d) a mapping function* $Fn : Dom(BF) \to Range(BF)$. □

In practice, the values of the function are restricted to some specific area of interest, and only a finite set of samples points are known. For estimating the values at non-sampled points, different interpolation functions can be used, usually based on a partition of the space, denoted a *tessellation.* For example, a *Rasterized Tessellation* is a regular partition of a 2D or 3D domain, where samples are used to assign a value to each cell and, if more than one value exist, all values corresponding to the same cell are summarized into a single one. A *Voronoi Tessellation* over a finite set of samples is an irregular partition where the segments in the Voronoi diagram are composed of all the points equidistant to the two nearest samples, and the nodes are the points equidistant to three or more samples [13]. Given a tessellation, many interpolation functions could be used, the simplest one being the *constant* function, which assigns to every point inside a cell the value of the sampled value contained in it.

Definition 8 (Discretized Field). *An N-dimensional Discretized Field (DField) F is a Bounded CF with: (a) a non-empty finite set of K tuples of dimension N+1,* $Samples(F) = \{(x_{11}, \ldots, \ldots, x_{1n}, v_1), ..., (x_{k1}, \ldots, x_{kn}, v_k)\}$, *where* $\forall i, \forall j, i = 1..K, j = 1..N, x_{ij} \in Dom_j(F)$, *and* $\forall i, i = 1..K$, $v_i \in Range(F)$; *(b) an interpolation function* fs *over* $Samples(F)$, *used for defining Fn (Definition 7).* □

A DField contains *only sampled values.* Given an N-dimensional DField DF, we denote (a) **sampled tuple** each element s of Samples(DF); (b) **sampled point** ($sp(s)$) the first N components of a sampled tuple; (c) **sampled value** ($sv(s)$) the last component of a sampled tuple. DFields can be classified into *spatial discretized field* (SDField) and *spatio-temporal discretized field* (STDField).

Definition 9 (Spatial Discretized Field). *A Spatial Discretized Field (DField) F is a DField (see Definition 8) such that: (a) Its domain* $Dom(F)$ *is spatial; (b)* $Dom(F)$ *is tessellated such that at least one sample belongs to each geometry, i.e.,* $T(Dom(F)) = \{g_1, \ldots, g_p\}$, *and* $\forall i, i = 1..p \; \exists \; s \in Samples(F)$ *such that* $sp(s) \in g_i$. □

Borrowing ideas from Cubic Map Algebra [15], we model a spatio-temporal DField as a list of snapshots of SDFields across time, such that each snapshot is valid during a certain time interval. These time intervals induce a temporal partition of the time dimension. In each of the snapshots we can only apply the f_s function defined above. Snapshots can be of different kinds, meaning that the first one could be a raster tessellation, and the next one could be a Voronoi partition. This is formalized in Definition 10 below.

Definition 10 (Spatio-Temporal DFields). *A ST-DField F is a time-ordered sequence of N-dimensional SD-Fields* $\{F_1, F_2, ...F_K\}$, *Seq(F), such that: (a)* $Dom(F_i)$ *is the same* $\forall i, i = 1..K$, *denoted* $Dom_s(F)$; *(b) Each* F_i *is a snapshot of the field taken at time* t_{F_i}, *and has an associated time interval* $I_{F_i} \subset \mathbb{R}$ *representing its interval of validity, such that: (1)* $Dom_t(F) = \bigcup_{i=1..K} I_{F_i}$; *(2)* $I_{F_i} = [s_{F_i}, e_{F_i})$ *where* $s_{F_i} = t_{F_i} - (t_{F_{i+1}} - t_{F_i})/2$ *and* $e_{F_i} = t_{F_i} + (t_{F_{i+1}} - t_{F_i})/2$ *(thus,* $\forall i, i = 1..K \quad \forall j, j = 1..K \; I_{F_i} \bigcap I_{F_j} = \emptyset$*); (c)* $Dom(F) = Dom_s(F) \times Dom_t(F)$; *(d)* $Labels(F_i)$ *is the same* $\forall i, i = 1..K$, *and* $Labels(F) = \langle l_1, \ldots l_n, Time \rangle$ *where* $l_i \in Labels(F_1)$, $\forall i, i = 1..N$; *(e)* $Range(F_i)$ *is the same* $\forall i, i = 1..K$, *denoted* $Range(F)$; *(f)* $Samples(F) = \{(sp(s), s_{F_i} + (e_{F_i} - s_{F_i})/2, sv(s)) | \; \exists F_i \in Seq(F) \; \exists s \in Samples(F_i)\}$; *(g)* f_s *is the same* $\forall i, i = 1..K$; *(h)* $Fn(x_1, \ldots x_n, t) = Fn_i(x_1, \ldots x_n)$ *where* Fn_i *is the Fn corresponding to the snapshot* F_i *if* $t \in I_{F_i}$. □

Example 3 (DFields and STDFields). *The soil pH measures the acidity or basicity in soils. The Normalized Difference Vegetation Index (NDVI) is used to analyze the quality and development of vegetation based on remote sensing measurements of the radiation intensity emitted by plants. Fig. 2 shows a Voronoi SDField Soil pH, a raster SDField NDVI and an STDField temperature, where the first two snapshots correspond to a Voronoi representation, and the other ones, to rasterized partitions.* □

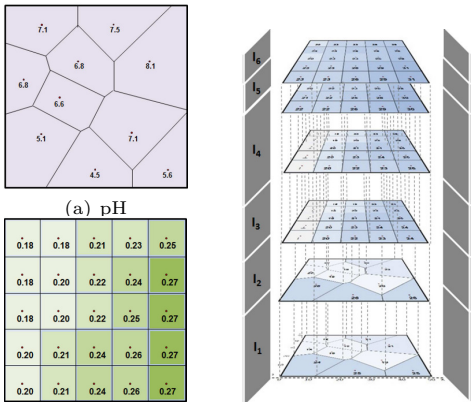
c1		
$dist_X$	$dist_Y$	sales
L_X	L_Y	300
N_X	N_Y	400
T_X	T_Y	200

c2	
geom	population
$g_{Brabant}$	15
g_{Ant}	18

DrillAcross(c1, c2)			
$dist_X$	$dist_Y$	sales	population
L_X	L_Y	300	15
N_X	N_Y	400	15
T_X	T_Y	200	18

DrillAcross(c2, c1)		
geom	population	sales
$g_{Brabant}$	15	300
$g_{Brabant}$	15	400
g_{Ant}	18	200

Table 2: On the left, c1 and c2 cubes. On the right, two Drill-Across problems.



(a) pH

(b) NDVI

(c) temperature

Figure 2: Examples of DFields

3.1 A Closed Generic Map Algebra for DFields

We mentioned in Section 1.2 that existing proposals of algebras for fields cannot handle fields with different kinds of tessellations. In this section we propose a Map Algebra that overcomes this limitation. Moreover, our meta-operators conform a closed algebra and can be nested in field expressions. The operators we propose below are at logical level, that is, at the conceptual level we still have only the typical OLAP operators. However, to produce this abstraction layer, some mappings mechanisms are needed.

Fields may come from different sources, and their dimensions can be labeled differently from each other, even if they refer to the same semantic concept. In order to operate with them, it is necessary that they are *domain compatible*, i.e., they have the same dimensionality (2D, 3D) and there exists a semantic mapping among the labels belonging to same domains. This mapping aims to identify which labels refer to the same concepts although they have different names defined in different order. For example, F1 and F2 with $Labels(F1) = \langle X, Y \rangle$, Dom(X)=[10, 30], Dom(Y)=[50, 120] and $Labels(F2) = \langle coordY, coordX \rangle$, Dom(coordY)=[50, 120], Dom(coorX)=[10, 30] are domain compatible if we define a semantic mapping by matching X with coordX and Y with coordY. We omit the complete formalization of this mapping for the sake of space. Now, we can redefine the traditional Map Algebra operators, in order to support any kind of underlying representation, making use of the notion of "Discretized Field" introduced above.

The **Generic Local Operator** receives a collection of domain-compatible DFields, and produces a new one whose values at each location p are computed applying a function f to the values at the same location p in all the input fields.

Definition 11 (Local Constructor). *Given a set of k domain-compatible DFields F_1, F_2, ..., F_k, and a function $fl : Range(F_1) \times .. \times Range(F_k) \rightarrow Rn \cup \{\perp\}$, $Local(fl, F_1, \ldots, F_k)$ builds a new DField denoted Output such that: (a) $Dom(Output) = Dom(F_1)$ (could be any other one); (b) $Labels(Output) = Labels(F_1)$; (c) $T(Dom(Output)) = T(Dom(F_1))$; (d) Samples $(Output) = \{s' | s' = \langle sp(s), v' \rangle$ and $s \in Samples(F_1)$ and $v' = fl(Fn_1(sp(s)), Fn_2(sp(s)^{*_1}), ..., Fn_k(sp(s)^{*_1}))$ where Fn_i is the function Fn of F_i and $sp(s)^{*_1}$ is the transposition according the semantic mapping (with respect to F_1) $\}$; (e) $Range(Output) = Range(fl)$; (f) $Fn(Output)$ is based on fs of F_1.* □

The **Focal** and **Zonal** operators aggregate values over a region. When an aggregate function (like Sum, Avg) is applied over a field, we must define the region of the domain where the values will be aggregated. Given an N-dimensional DField F, an aggregation region, denoted AggRegion, is an M-dimensional region with M≤N such that AggRegion ⊆ Dom(F). To approximate aggregates in this region, we need the following definition.

Definition 12 (Representative Points). *Given a DField F where $T(Dom(F)) = \{g_1, g_2, ...g_n\}$, we associate a unique point to each geometry, and denote it a* representative point, *$rep(g_i)$, $\forall i, i = 1..n$. The value of the function at $rep(g_i)$ is called a* representative value *$vrep(g_i)$.* □

In a Voronoi Tessellation there is exactly one sample for each geometry, defined as the representative point of such geometry. Analogously in the raster model (although the representative point is not necessarily the original sample).

The **Generic Focal Operator** receives a DField F and produces a new one such as at each location p its value is calculated aggregating the values of F in the neighborhood of p using a function $\mathcal{A}$. The new DField keeps the representation and labels of the input parameter. The neighborhood of a point can be defined using different criteria (e.g., for raster data as a collection of cells around a given one). Since we do not use this operator in the rest of the paper, for the sake of space we omit its formal definition.

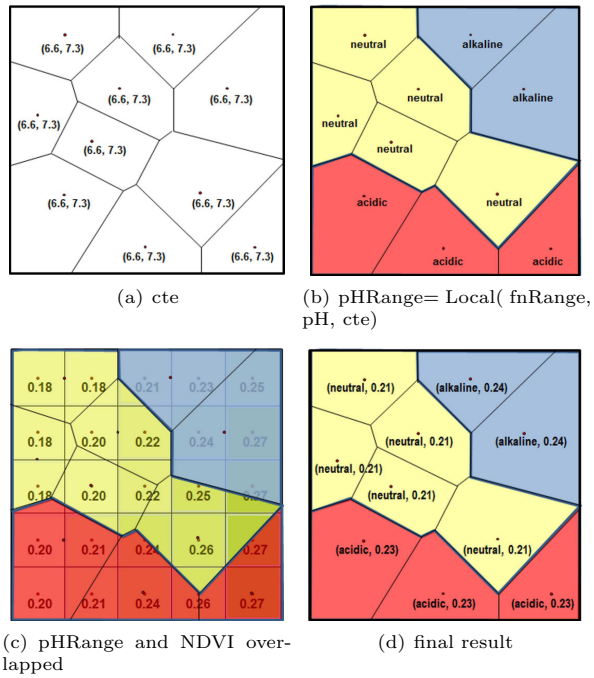
The **Generic Zonal Operator** receives two DFields F and Ref (domain-compatible), and an aggregate function $\mathcal{A}$, and produces a new DField. Ref is divided in zones where the values are the same. For each of such zones, $\mathcal{A}$ is applied over the corresponding values in F (i.e., the result contains as many groups as different representative values exist). Note that, opposite to the case of Tomlin's Map Algebra, here *the zonal operators return a field*, which makes the algebra closed. Before giving a formal definition, we need the notion of *Isopartition.*

Definition 13 (Isopartition of a DField). *Given a DField F, and $V = \{vrep(g_i) | g_i \in T(Dom(F))\}$, an isopartition of F with respect to V, denoted as $IP(F, V)$ is the set $\{t | t = \langle v, \{g_1, g_2, \ldots g_k\} \rangle, v \in V$, and $\forall i, i = 1..k, vrep(g_i) = v\}$. Note that in an isopartition, two zones with the same value can be disjoint.* □

DEFINITION 14 (THE ZONAL CONSTRUCTOR). *Let Ref and F be two N-dimensional domain compatible DFields and an aggregate function $\mathcal{A} : AggRegions \times Fields \rightarrow Rn \cup \{\perp\}$. $Zonal(\mathcal{A}, Ref, F)$ builds a new DField such that: (a) $Dom(Output) = Dom(Ref)$; (b) $Labels(Output) = Labels(Ref)$; (c) $T(Dom(Output)) = T(Dom(Ref))$; (d) $Samples(Output) = \{s' | s' = \langle sp(s), v' \rangle, s \in Samples(Ref), v' = \mathcal{A}(Reg, F)$, and $\exists \langle sv(s), \{g_1, \ldots g_k\} \rangle \in IP(Ref, V)$ where $V = \{vrep(g_i) | g_i \in T(Dom(Ref))\}$ and $Reg = \cup_{i=1..k} g_i\}$; (e) $Range(Output) = Range(\mathcal{A})$; (f) $Fn(Output)$ is based on the fs of Ref.* □

EXAMPLE 4 (GENERIC MAP ALGEBRA). *The Natural Resources Conservation Service (NRCS) classifies a pH level of a soil as 'acidic' for pH < 6.6, 'neutral' for pH between 6.6 and 7.3, and 'alkaline' for pH > 7.3. We want to generate a new DField with the average NDVI index for each zone defined by this pH classification, using the 2D Voronoi SDField pH in Fig. 2(a), and the 2D raster SDField NDVI of Fig. 2(b).*

For comparing a DField against a constant value we need to generate a constant DField (i.e., one such that all of its samples have the same value), and then apply a Local operator. In our example, we first build the SDField denoted cte *(Fig. 3(a)), with the lower and upper values of the pH, and then we apply Local(fnRange, pH, cte) where the function is defined as $fnRange(v1, v2) = \{if\ v1 < v2[0]$* then return *'acidic'* else *$if\ v1 < v2[1]$* then return *'neutral'* else return *'alkaline'}. Fig. 3(b) depicts this intermediate result. Finally, for obtaining the average by zone we apply the Zonal operator between the pHRange and NDVI SDFields. Figs. 3(c) and 3(d) show the overlapped intermediate pHRange and NDVI SDField previous to the final step and the final resulting SDField, respectively. Since the Generic Map Algebra is closed, the query can be expressed as: $Zonal(Avg, Local(fnRange, pH, cte), NDVI)$.* □



(a) cte (b) pHRange= Local(fnRange, pH, cte)

(c) pHRange and NDVI overlapped (d) final result

Figure 3: Zonal(Avg,Local(fnRange,pH,cte),NDVI)

4. DFIELDS AND OLAP CUBES

We argue that the structure of a DField fits perfectly into the concept of a data cube. A cube is composed of facts, measures and dimensions. In a DField, the fact of interest is precisely what the field represents. The (one-level) dimensions are the domains of the field. Thus, an N-dimensional DField can be modeled with N schema dimensions of the form $\langle l_i, \{l_i, All\}, \{l_i \rightarrow All\} \rangle, \forall i, i = 1..N$, and the measure given by the field value. Thus, the cube schema is given by $\langle fieldName, D, M \rangle$ where D is the set of schema dimensions of the field, and M (i.e., the measure) is the set whose unique element represents the range of the field (e.g., temperature values). Notice that we model a DField with a hierarchy of only one level. For example, the cube schema for the STDField temperature of Example 3 is given by $\langle temperature, \{\langle x, \{x, All\}, \{x \rightarrow All\} \rangle, \langle y, \{y, All\}, \{y \rightarrow All\} \rangle, \langle t, \{t, All\}, \{t \rightarrow All\} \rangle\}, \{value\} \rangle$. Thus, it contains two spatial and one temporal dimensions.

Therefore, a DField can be perceived as a typical OLAP cube over which we can apply the traditional OLAP operators. Discrete spatial data are essentially analogous to non-spatial cubes at a high abstraction level (this is implicit in Definition 1). Thus, at the conceptual model the user can perceive information as a data cube, and leave the implementation details to lower abstraction layers.

Following this line of reasoning, at the *logical* level, each OLAP cube operator might be re-defined for each added kind of data type. In our case, for the *field* data type. Moreover, these operators are polymorphic and, at the physical level, some of them need to be implemented differently depending on the specific kind of DField, i.e. SField or ST-DField, without affecting the queries that users pose. We now propose the semantics for each OLAP operator that applies to DFields. In the remainder we consider the following set of aggregate functions $\mathcal{A} = \{$MAX, MIN, AVG, SUM$\}$. These are polymorphic functions, i.e., they are defined for single or vectorial values. In the case of DFields having vectorial values, if the user chooses one of these aggregation functions as a parameter, the same function will be applied to all the components of the vectorial value. If the user wants to apply a different aggregation for each component, a user-defined function must be used.

4.1 Dice Operator on DFields

Applying a Dice operation (i.e., a selection of values in dimensions or measures that satisfy a boolean condition σ) to a DField, might introduce a discontinuity (e.g. σ=x < 20 or x>50) over its domain. In order to guarantee spatial continuity the Dice operator over a DField sets to $\perp$ the value of samples when the condition σ is not satisfied. Formally:

DEFINITION 15 (DICE OPERATOR ON DFIELDS). *The input of this operator is an N-dimensional DField F and a boolean predicate σ over dimensions and measures. The output is a new N-dimensional DField NF such that: (a) $Dom(NF) = Dom(F)$; (b) $Labels(NF) = Labels(F)$; (c) $T(Dom(NF)) = T(Dom(F))$; (d) $Samples\ (NF) = \{\langle sp(s), v' \rangle | \ s \in Samples(F), v' = sv(s)$ if σ is satisfied or $\perp$ otherwise $\}$; (e) $Range(NF) = Range(F) \cup \{\perp\}$; (f) $Fn(NF) = Fn(F)$.* □

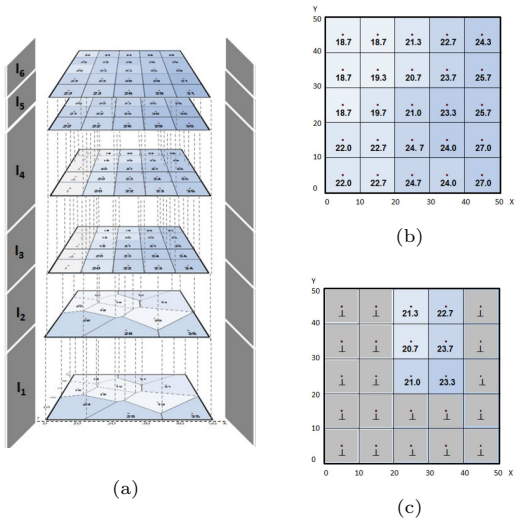


Figure 4: (a) SDField temperature. (b) auxi← Slice(temperature, t, Avg). (c) Dice(auxi, 'x ≥ 20 and v < 24')

4.2 Slice Operator on DFields

Here we cannot apply a Dice operator to produce a singleton in the dimension to be removed (as in OLAP cubes), because this would produce ⊥ values. For assuring a singleton, an aggregation to the *All* must be applied.

Definition 16 (Slice Operator on DFields). *This operator receives an N-dimensional DField F, a dimension in the domain with label J and an aggregate function Agg ∈ A}. It returns a new (N-1)-dimensional DField NF as follows: If the sliced dimension is temporal, then NF = Local (Agg, F_k, ...F_1) such that F_k is the last DField of Seq(F). Else NF is built as follows: (a) $Dom(NF) = Dom_1(F) \times ... \times Dom_{j-1}(F) \times Dom_{j+1}(F) \times ... \times Dom_n(F)$; (b) $Labels(NF) = \langle L_1(F) \times ... \times L_{j-1}(F) \times L_{j+1}(F) \times ... \times L_n(F)$; (c) $T(Dom(NF)) = T(Dom(F))$, that means, the type of tessellation is preserved after eliminating the dimension with label J. (Analogously, for a STDField the operator preserves the tessellation of each snapshot); (d) Each sample in NF satisfies the following condition: $\langle (x_1, ..., x_{j-1}, x_{j+1}, ..., x_n), v' \rangle \in Samples(NF) \Rightarrow$ ($\exists m, \exists i, 1 \leq i \leq m$, $\forall \langle (x_1, ..., x_{j-1}, x_j, x_{j+1}, ..., x_n), v_i \rangle \in Samples(F) \Rightarrow v' = Agg(v_1 ... v_m)$), (e) $Range(NF) = Range(Agg)$; (f) $Fn(NF) = Fn(F)$.*

Example 5. *Given the 3D STDField temperature of Example 3, we want to compute its values independently of the timestamp in which they were registered, only keeping those values with temperature less than 24 located in places such that coordinate x is greater or equal than 20. To remove the temporal dimension we apply the Slice operator (see Fig. 4(b)), and for selecting the samples that satisfy the required condition, we apply the Dice operator. The query reads: Dice(Slice(temperature, t, Avg), 'x ≥ 20 and v < 24'). Fig. 4(c) depicts the final result.* □

4.3 Roll-Up and Drill-Down over DFields

In general, a Roll-Up aggregates measures according to a dimension hierarchy. In our case, this aggregation is performed over an external hierarchy. For example, to obtain the average precipitation by district (i.e., polygons), an external hierarchy where a 2D Point level rolls-up to a 2D Polygon level could be introduced. In this work we only analyze the case of the Roll-Up operator over spatial hierarchies. We do not discuss Drill-Down, since it can be implemented by tracking navigation paths, as in traditional OLAP.

Spatial Roll-Up Over SDFields. A *Spatial Roll-Up over a DField* aggregates values of the DField according to geometric regions of interest that conform a spatial hierarchy, and can be solved by using the zonal operator. Technically, since the geometries in these hierarchies are not DFields, an operator, denoted *GeomToField* must be applied to convert a set of geometries in one level into a DField. Then, a Roll-Up of a DField F over a spatial dimension consists in applying the Generic Zonal operator to F, where the reference field is the DField produced by the GeomToField operator applied over the geometries in the corresponding level of the hierarchy.

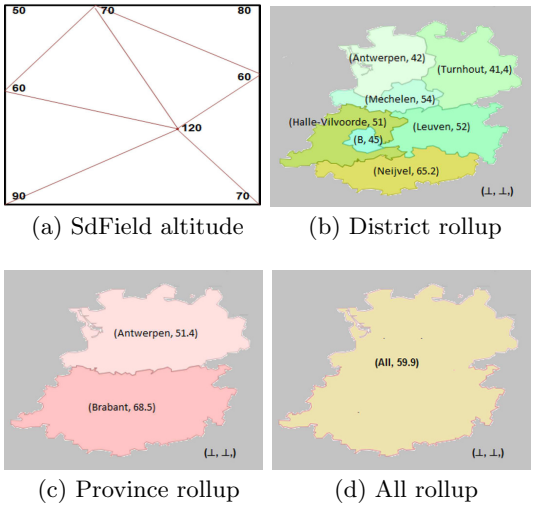
Definition 17 (GeomToField Operator). *Input: an N-dimensional SDField F, a set G of disjoint labeled geometries over an N-dimensional space. Output: a new N-dimensional DField NF as follows: (a) Dom(NF) = Dom(F) (b) Labels(NF) = Labels(F) (c) T(Dom(NF)) = ((G ∩ GDom) ∪ (GDom - UG)) where GDom is the resulting geometry of the union of all the geometries of T(Dom(F)) and UG is the union of all the geometries of G (d) Samples(NF)= {(p, v)|(∃g ∈ (G∩GDom) and p is any point of g and v = label(g)) or (∃g ∈ (GDom - UG) and p is any point of (GDom - UG) and v = ⊥)} (e) Range(NF)= labels of G ∪ {⊥} (f) Fn= Cte ∈ (F).* □

Definition 18 (Spatial Roll-Up over a DField). *Input: an N-dimensional SDField F, a geometric spatial dimension instance I with schema ⟨dimName, L, →⟩, a level l ∈ L containing a level descriptor whose domain type is an N-dimensional geometry, an aggregate function Agg ∈ A}. Output: Zonal(GeomToField(F, G), F, Agg) where Zonal is the generic operator defined in Section 3, and G is the set of geometries of l, each one labeled with the ID of the member that corresponds to it.* □

Example 6 (Spatial Roll-Up). *We have the SDField altitude (Fig. 5 (a)), and the hierarchy instance of the blockDim schema in Example 1. There is also the dimension instance containing only the Antwerpen and Brabant provinces, and the districts in each one of such provinces. A first Roll-Up aggregates the altitudes over districts obtaining a new SDField F2 (Fig. 5 (b)). A second Roll-Up to the province level calculates the average altitude by province, obtaining F3. (Fig. 5 (c)).* □

Spatial Roll-Up over a STDField. Here, we apply the spatial Roll-Up to each snapshot of the temporal sequence seq(F) in the STDField. Formally:

Definition 19 (Spatial Roll-up over STDFields). *Input: an N-dimensional STDField F, a spatial dimension instance I with schema ⟨dimName, L, →⟩, a label l ∈ L with description in the geometry domain, an aggregate function Agg ∈ A}. Output: a new N-dimensional STDField NF such that: (a) Dom(NF) = Dom(F); (b) Labels(NF) =*



(a) SdField altitude (b) District rollup

(c) Province rollup (d) All rollup

Figure 5: Spatial Roll-Up over SDField altitude

.

Labels(F); (c) Seq(NF) = {Zonal(GeomToField(F_i, G), F_i, Agg) such that $F_i \in Seq(F)$ and G is the set of geometries of l, each one labeled with the ID of the member that corresponds to it }; (d) Range(NF) = Range(Agg) ∪ {⊥}; (e) Fn(NF) = Fn(F). □

Temporal Roll-Up Over STDFields. A temporal Roll-Up over a STDField F (a temporal sequence of SDFields, Seq(F)) aggregates values of the STDField according to the members (temporal intervals) of an external temporal hierarchy. For each member m of a hierarchy level, a *Generic Local* operator is applied over all the snapshots in Seq(F) whose intervals have non-empty intersection with it. The youngest snapshot is chosen as the first parameter of this operation. Thus, the output is a new STDField that contains as many summarized SDFields as members in the corresponding level of the temporal hierarchy.

Definition 20 (Temporal Roll-Up). *Input: an N-dimensional STDField F, a temporal dimension instance I with schema $\langle H, \mathcal{L}, \rightarrow \rangle$, a label $l \in \mathcal{L}$, an aggregate function $Agg \in \mathcal{A}$. Output: a new N-dimensional STDField NF as follows: (a) Dom(NF) = Dom(F) (b) Labels(NF) = Labels(F) (c) Seq(NF) = $\{F_m | \exists m \in l$ and $F_m = Local(Agg, F_{k+q}, \ldots, F_k)$ and $F_{k+i} \in R(m)\ \forall i, i = 1 \ldots q$ where $R(m) = \{F_j | F_j \in Seq(F)$ and $I_j \cap m \neq \emptyset\}$ (d) Range(NF)= Range(Agg) ∪ {⊥} (e) Fn(NF) = Fn(F)* □

Example 7 (Temporal Roll-Up). *Consider an STDField temperature with intervals $I1 = [03/01/09; 31/03/09)$, $I2 = [01/04/09; 31/05/09)$, $I3 = [01/06/09; 31/07/09)$, $I4 = [01/08/09; 31/10/09)$, $I5 = [01/11/09; 30/11/09)$, $I6 = [01/ 12/09; 31/ 12/09]$, a temporal dimension with schema $\langle TDS, \{Date, Quarter, All\}, \{Date \rightarrow Quarter\}, \{Quarter \rightarrow All\}\rangle$, and an instance H of TDS with $Members(Quarter) = \{Q1{-}09, Q2{-}09, Q3{-}09, Q4{-}09\}$. RollUp(temperature, H, Quarter, Avg) produces a new STDField with only four snapshots, corresponding to the aggregation of the snapshots in Seq(temperature) into the quarters of 2009 (See Fig. 6).* □

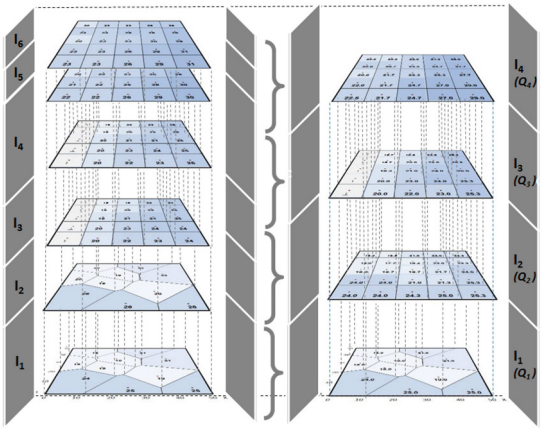


Figure 6: RollUp(temperature, H, Quarter, Avg)

4.4 The Drill-Across Operation

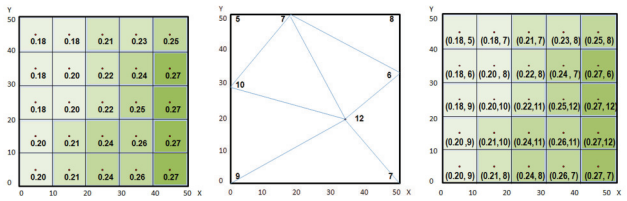
Drill-Across is the only operator that mixes two cubes. We consider two cases: (a) Both cubes are DFields; (b) One cube is a DField and the other one is a traditional OLAP cube. Putting together DField cubes and traditional OLAP cubes, requires polymorphic operators at the logical level, since Drill-Across produces different results depending of its operands (i.e., the kinds of cubes). In the case of two DFields, this represents no problem: both cubes contain hierarchies with exactly one level in a continuous domain. However, when we have a field and a non-field cube, the problem becomes more involved, since we need to chose the cube to be returned. In this paper we propose the resulting cube to be the one coming from the traditional OLAP side.

Drill-Across between DFields. When we perform a Drill-Across between two DFields, both cubes contain hierarchies with exactly one level in a continuous domain. In this case, both cubes must be domain-compatible, and the Drill-Across reduces to apply a *Generic Local* operator.

Definition 21 (Drill-Across between DFields). *The operator receives two domain compatible DFields F1 and F2, and returns a new SDField $NF = Local(fl, F1, F2)$, where $fl : Range(F1) \times Range(F2) \rightarrow Range(F1) \times Range(F2)$ such that $fl(a, b) = (a, b)$ (i.e., a vector composed of both values).* □

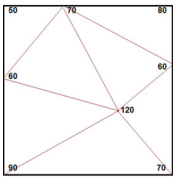
Example 8. *For a (raster) SDField NDVI and a (TIN) SDField Altitude, we can apply DrillAcross(NDVI, Altitude) and build a cube where the two measures can be analyzed together. Fig. 7 shows the resulting field.* □

Drill-Across between DField and OLAP Cubes. Conceptually, a semantic mapping that assures the one-to-one property between a continuous cube (i.e., a cube with infinite members) and a discrete one is not possible unless we summarize values in the continuous domain. When performing a Drill-Across operation between a DField and an OLAP cube, the latter has associated an external hierarchy, over which the former is summarized. In the following, for



(a) (b) (c)

Figure 7: (a) SDField NDVI. (b) SDField Altitude. (c) Drill-Across (NDVI, Altitude).



(a) SDField altitude

distName	dGeom	harvest
Turnhout	geo15	7120
Mechelen	geo11	7542
...	...	...
Leuven	geo43	6830
Nijvel	geo30	7499

(b) SOLAP grapes

distName	dGeom	harvest	altitude
Turnhout	geo15	7120	41.4
Mechelen	geo11	7542	54.0
...	...	...	....
Leuven	geo43	6830	52.0
Nijvel	geo30	7499	65.2

(c) Output SOLAP

Figure 8: Drill-Across(altitude, grapes, Avg)

the sake of space, we only discuss the Drill-Across between a spatial DField and an OLAP cube, and omit the case of the spatio-temporal DField.

Intuitively, the SDField rolls up to the lowest level in the spatial dimension hierarchy of the OLAP cube, and generates a new output SDField containing aggregated values at the level of the non-field cube. This new SDField contains as many summarized values v_i as members g_i (geometries) in the OLAP cube. The last step builds the output cube, using a function denoted $GetMeasures$ that iterates over all of the discrete values 'g_i' and adds the corresponding 'v_i' to the rows of the output OLAP cube that matches 'g_i'.

DEFINITION 22. *Input: An SDField F, a non-field cube C with schema* $\langle Csc, \{\langle SDim, \mathcal{L}_S, \rightarrow_S\rangle\}, M\rangle$ *where SDim is the spatial dimension, an aggregate function* $Agg \in \mathcal{A}$. *Output: a non-field cube NC with schema* $\langle NCsc, \{\langle NSDim, N\mathcal{L}_S, \rightarrow_N S\rangle\}, NM\rangle$ *where* $NM = M \cup \{Fname\}$, $N\mathcal{L}_S$ *contains all levels S such that* $L \rightarrow S$ *where L is the lowest level of SDim in C, and* $\rightarrow_N S$ *is a subset of* $\rightarrow_S$ *whose levels* $\in N\mathcal{L}_S$. *NC is obtained adding the measures resulting from applying* $GetMeasures(Rollup(F, SDim, L, Agg_s))$. □

EXAMPLE 9. *Given the SDField altitude (Fig. 8(a)) and the OLAP cube grapes in Fig. 8(b)), we want to enrich the latter with the average altitude corresponding to its lower level of the spatial hierarchy (district). This new cube can be obtained applying Drill-Across(altitude, grapes, Avg) which: (a) creates a new cube R with the same columns of the* grapes *cube, and adds a new column named* altitude*; (b) applies a Roll-Up over the SDField altitude using the geometries of the districts in* grapes*; (c) iterates over each geometry* g_i*, takes the aggregate value and adds it to the column* altitude *of R for the row corresponding to* g_i. *For example, if the output of the Roll-Up results in an average altitude of 41.4 for the geometry of Turnhout, then the row corresponding to Turnhout in the output SOLAP cube R will have 41.4 for the column altitude. Fig. 8(c) shows the resulting cube.*

5. IMPLEMENTATION AND USE CASES

We have implemented a prototype as a proof-of-concept of the plausibility of our proposal. We now sketch this implementation and present two use cases that show how complex queries can be elegantly expressed hiding the different kinds of spatial data types involved. We use the cube schema *grapeCS* introduced in Example 1. This cube contains two dimensions: dateDim and blockDim. The latter has all the information about vineyards represented as geometries. The measure of grapeCS is vine harvest (in Kg). We also use the instance cube *grapes* (Table 1). Finally, we include three DFields, corresponding to Belgium: the STDField *temperature*, the STDField *precipitation*, and the SDField *altitude*. All of them with raster tessellation.

Data for fields used in our examples have been downloaded from the WorldClim site[2]. For our region of interest (a portion of Belgium), we used *elevation* raster data with a resolution of 5 arc-minutes, obtaining 655 cells, and spatio-temporal *temperature* and *precipitation* raster data with a resolution of 10 arc-minutes. Precipitation and temperature data correspond to monthly values during 2009, resulting in 185 cells for each month. Raster data was downloaded in a generic grid format and imported into a PostgreSQL database equipped with the PostGIS plugin for handling spatial data types. This generates polygons with their associated values. The units for elevation, precipitation, and temperature are meters, milimeters, and Celsius degrees*10, respectively. For the *grapes* cube, we used a real-world map of Belgium containing geographic, demographic and economic information about provinces and districts (represented as polygons). The maps were obtained from the spatial library of the GIS Center[3]. With this geographical information we built the spatial hierarchy dimension described above. The fact table was populated synthetically with 1758 tuples. These data reflects the real distribution of vine harvest over provinces, and the real proportion of blocks according to wine production in districts. On the other hand, dates were generated randomly, although keeping unchanged the harvest time of each kind of vinegrape. We implemented the operators in the Java programming language, and for visualization of results we use OpenJump 1.4[4].

Use Case 1. We want to produce a data cube containing the grape types and the harvest of the blocks (from the SOLAP grapes cube), and their average precipitation during the period that precedes the grape harvest of 2009, i.e., from March to September of 2009. This can be expressed in the algebra we presented in Section 4. We need the following temporal hierarchy: $Date \rightarrow Period; Period \rightarrow All$. Its instance, named TH, is built as follows: (a) Members of level *Date* correspond to 2009; (b) Members of the level *Period* are $period1, period2$ and $period3$; (c) All the dates previous to '01/03/2009' roll-up to $period1$, dates from '01/03/2009' to '01/10/2009' roll-up to $period2$, and the rest of the dates roll-up to $period3$; (d) *period*, $period2$ and $period3$ roll-up

[2] http://www.worldclim.org/current

[3] http://giscenter-sl.isu.edu/other/world/europe/belgium/

[4] http://www.openjump.org/

to '*all*'. For assembling this information in a single data cube, we use the Drill-Across operator. The cube is built as follows:

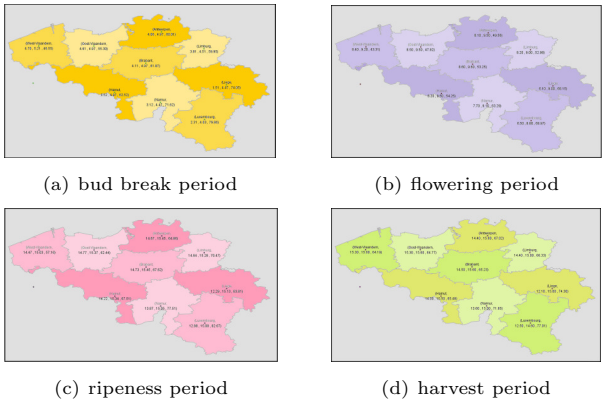
```
1. DrillAcross(
2.   Slice(
3.     Dice(
4.       RollUp(precipitation,TH,Period,Avg),
5.       t>='01/03/2009' AND t<='30/09/2009'),
6.     t,Avg),
7.   Slice(
8.     Dice(grapes,Year(Date)=2009),
9.     dateDim,Avg),
10.  Avg)
```

We explain this query in detail. The RollUp operator (line 4) prepares the aggregation of precipitations in three periods (the second being the one we are interested in). (See Fig. 9(a)). The Dice operator (lines 3 to 5) keeps those values which belong to the chosen period. Fig. 9(b) depicts the temporary STDField produced. The nested Slice operator (line 2 to 6) removes the temporal dimension (after a summarization to the 'All' level is performed) and generates the SDField shown in Fig. 9(c). The Dice operator (line 8) restricts the cube to members rolling up to 2009 in the Date level. Then, the Slice operator (lines 7 to 9) eliminates one of the dimensions not used in the query, preparing the cube for the Drill-Across with an SDField. Finally, a Drill-Across (lines 1 to 10) is applied to link the measures of both worlds, as shown in Fig. 9(d). Notice that, according to Definition 22, the Drill-Across operator generates a non-field cube and the last column corresponds to the measure added from the DField (by applying the GetMeasures() operation).

Use Case 2. We now want to build an OLAP cube to analyze districts in Belgium which present the ideal conditions for the Malbec crop, which are related to the range of temperature, and average precipitation during specific periods (e.g., best conditions for the harvesting period are temperatures between 18 and 22 Celsius degrees, and average precipitations of less than 40 mm). Thus, we build an STDField for analysis including minimum and maximum temperatures, and average precipitation summarized by district, for the following periods: bud break (February-March), flowering (first part of April), ripeness (from the second half of April to the end of August), harvesting (September). To solve this query we use another instance of the same temporal hierarchy of the previous example. This instance, named TH2, is built as follows: (a) We consider dates in 2009; (b) Members of the dimension level Period are p1, p2, p3, p4, p5 and p6; (c) All the dates in January roll-up to p1, dates in February-March roll-up to p2, dates in first part of April roll-up to p3, dates from second part of April to the end of August roll-up to p4, dates in September roll-up to p5, and the dates in October-December roll-up to p6; (d) p1, p2, p3, p4, p5 and 6 roll-up to 'all'. The query reads:

```
1. DrillAcross(
2.   DrillAcross(
3.     RollUp(
4.       RollUp(temperature,TH2,Period,Avg),
5.       blockDim,district,Min),
6.     RollUp(
7.       RollUp(temperature,TH2,Period,Avg),
8.       blockDim,district,Max)
9.   ),
10.  RollUp(
11.    RollUp(precipitation,TH2,Period,Avg),
12.    blockDim,district, Avg))
```

The nested Roll-Up operator (line 4) prepares the aggregation of temperatures in six periods, and the Roll-Up operator (lines 3 to 5) computes the minimum temperature for each disctrict. Analogously, the expression from lines 6 to 8 computes the maximum temperature. Thus, the Drill-Across operator (lines 2 to 9) mixes those values, previously calculated. The expression from lines 10 to 12 calculates the average level of precipitation for each district. Finally, a Drill-Across operator is used for obtaining the three measures together associated to each district.



(a) bud break period (b) flowering period

(c) ripeness period (d) harvest period

Figure 10: Query for Use Case 2(periods p_2, p_3, p_4, p_5)

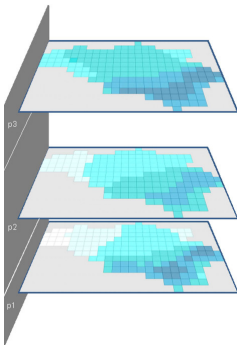
6. CONCLUSION AND FUTURE WORK

In this paper we showed, formally and through a proof-of concept implementation, how spatiotemporal data of different kinds can be seamlessly considered as typical OLAP cubes at the conceptual level, in a way that a user is only exposed to the traditional OLAP operators. We showed that this approach provides the capability of expressing complex queries in an elegant and simple way, hiding the implementation details, and also adding an abstraction layer, not present in any previous proposal in the spatio-temporal OLAP literature. An obvious consequence of this work is that we can now think on an implementation of a generic OLAP tool that can reutilize the enormous amount of existing work in OLAP to implement powerful tools for spatio-temporal data analysis.

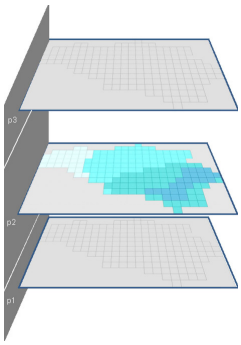
Acknowledgement. This work has been partially funded by the LACCIR Project "Monitoring Protected Areas using an OLAP-enabled Spatio-Temporal Geographic Information System". Alejandro Vaisman has been partially funded by the "Open Semantic Cloud for Brussels (OSCB)" Project, supported by the Brussels Capital Region, Belgium.

7. REFERENCES

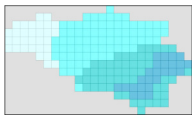
[1] A. Abelló, J. Samos, and F. Saltor. On relationships offering new drill-across possibilities. In *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, DOLAP '02, 2002.



(a) Auxi1 ← RollUp (precipitation, TH, Period, Avg)



(b) Auxi2←Dice(Auxi1, t >= '01/03/2009' and t <= '30/09/2009')



(c) Slice(auxi2, t, Avg)

	blockid	bGeom	provName	grape	gType	harvest	precipitation
1	58498	POLYGON((5.90(	Liege	Kerner	White	7535	83.91
2	59837	POLYGON((3.48(	Oost-Vlaanderen	Kerner	White	7707	59.02
3	59662	POLYGON((5.58(	Limburg	Pinot Blanc	White	7508	68.61
4	58965	POLYGON((3.80(	Oost-Vlaanderen	Kerner	White	6712	62.03
5	60005	POLYGON((4.19(	Brabant	Pinot Gris	White	6105	64.74
6	59953	POLYGON((4.43(	Brabant	Kerner	White	6415	64.60
7	60000	POLYGON((4.45(	Brabant	Chardonnay	White	6486	64.60
8	58620	POLYGON((5.40(	Limburg	Pinot Blanc	White	7355	68.18
9	58922	POLYGON((3.80(	Oost-Vlaanderen	Pinot Gris	White	6706	60.88
10	58668	POLYGON((5.69(	Limburg	Cabernet Sauvignon	Red	7381	65.61
11	59219	POLYGON((4.38(	Brabant	Malbec	Red	6579	64.60
12	59305	POLYGON((4.80(	Antwerpen	Pinot Noir	Red	7951	64.89
13	58426	POLYGON((3.95(	Hainaut	Chardonnay	White	6243	64.46
14	60153	POLYGON((2.75(	West-Vlaanderen	Chardonnay	White	7913	53.28
15	59887	POLYGON((4.95(	Brabant	Kerner	White	7696	65.45
16	58650	POLYGON((5.48(	Limburg	Chardonnay	White	7916	68.18
17	59418	POLYGON((5.28(	Liege	Pinot Gris	White	7270	71.59
18	59338	POLYGON((4.10(	Hainaut	Chardonnay	White	7457	66.31
19	58978	POLYGON((4.35(	Brabant	Kerner	White	7659	67.46
20	58949	POLYGON((3.65(	Oost-Vlaanderen	Pinot Gris	White	6351	60.31
21	58735	POLYGON((5.64(	Limburg	Pinot Gris	White	6357	68.61
22	60104	POLYGON((4.95(	Brabant	Pinot Noir	Red	7644	63.31
23	58391	POLYGON((5.17(	Antwerpen	Pinot Blanc	White	7656	66.60
...	...	...	...	...	...	...	...

(d) Dice(Auxi, t>='01/03/2009' and t<='30/09/2009')

Figure 9: Case 1

[2] R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. In *Proceedings of the Thirteenth International Conference on Data Engineering*, 1997.

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

[4] S. Bimonte and M.-A. Kang. Towards a model for the multidimensional analysis of field data. In *Proceedings of the 14th east European conference on Advances in databases and information systems*, ADBIS'10, pages 58–72, Berlin, Heidelberg, 2010. Springer-Verlag.

[5] L. Gómez, , S. Gómez, and A. Vaisman. Analyzing continuous fields with olap cubes. In *Proceedings of the 14th ACM international workshop on Data Warehousing and OLAP*, DOLAP '11, 2011.

[6] L. Gómez, A. Vaisman, and E. Zimányi. Physical design and implementation of spatial data warehouses supporting continuous fields. In *Proceedings of the 12th international conference on Data warehousing and knowledge discovery*, 2010.

[7] M. Gyssens and L. V. S. Lakshmanan. A foundation for multi-dimensional databases. In *Proceedings of 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 106–115, Athens, Greece, 1997.

[8] C. Hurtado, A. Mendelzon, and A. Vaisman. Maintaining data cubes under dimension updates. In *Proceedings of IEEE/ICDE'99*, 1999.

[9] G. V. Jones. Climate and terroir: Impacts of climate variability and change on wine. *In Fine Wine and Terroir-The Geoscience Perspective*, Geoscience Canada Reprint Series Number 9, Geological Association of Canada, St. John's, Newfoundland, 2006.

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

[11] R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd. Ed.* J.Wiley and Sons, Inc, 2002.

[12] M. P. Kumler. An intensive comparison of Triangulated Irregular Networks (TINs) and Digital Elevation Models (DEMs). *Cartographica*, 31:45, 1994.

[13] H. Ledoux and C. Gold. A voronoi-based map algebra. In A. Riedl, W. Kainz, and G. A. Elmes, editors, *Progress in Spatial Data Handling*, pages 117–131. Springer Berlin Heidelberg, 2006.

[14] J. Mennis and R. Viger. Analyzing time series of satellite imagery using temporal map algebra. In *Proceedings of the ASPRS Annual Conference*, Denver, CO., 2004.

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

[16] J. L. Mennis. Multidimensional map algebra: Design and implementation of a spatio-temporal gis processing language. *T. GIS*, 14(1):1–21, 2010.

[17] 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.

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

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

[20] A. A. Vaisman and E. Zimányi. A multidimensional model representing continuous fields in spatial data warehouses. In *Proc. of ACM-GIS*, 2009.

[21] T. van der Putte and H. Ledoux. Modelling three-dimensional geoscientic datasets with the discrete voronoi diagram. In *Proceedings of 3D GeoInfo 2010*, 2010.

[22] P. Vassiliadis. Modeling multidimensional databases, cubes and cube operations. In *10th International Conference on Scientific and Statistical Database Management, Proceedings, Capri, Italy, July 1-3, 1998 (SSDBM'98)*, pages 53–62, 1998.

[23] M. F. Worboys and M. Duckham. *GIS: A Computing Perspective*. CRC Press, second edition, 2004.