What is Spatio-Temporal Data Warehousing?

Alejandro Vaisman[1] and Esteban Zimányi[2]

[1] Universidad de Buenos Aires, University of Hasselt and
Transnational University of Limburg
alejandro.vaisman@uhasselt.be
[2] Université Libre de Bruxelles
ezimanyi@ulb.ac.be

Abstract. In the last years, extending OLAP (On-Line Analytical Processing) systems with spatial and temporal features has attracted the attention of of the GIS (Geographic Information Systems) and database communities. However, there is no a commonly agreed definition of what is a spatio-temporal data warehouse and what functionality such a data warehouse should support. Further, the solutions proposed in the literature vary considerably in the kind of data that can be represented as well as the kind of queries that can be expressed. In this paper we present a conceptual framework for defining spatio-temporal data warehouses using an extensible data type system. We also define a taxonomy of different classes of queries of increasing expressive power, and show how to express such queries using an extension of the tuple relational calculus with aggregated functions.

1 Introduction

Geographic Information Systems (GIS) have been extensively used in various application domains, ranging from economical, ecological, and demographic analysis, to city and route planning [21]. Spatial information in a GIS is typically stored in different so-called *thematic layers* (or *themes*). Information in themes consists of spatial data (i.e., geometric objects) associated to thematic (alphanumeric) information.

OLAP (On-Line Analytical Processing) [7] comprises a set of tools and algorithms that allow efficiently querying multidimensional databases containing large amounts of data, usually called Data Warehouses. In OLAP, data are organized as a set of *dimensions* and *fact tables*. In this multidimensional model, data can be perceived as a *data cube*, where each cell contains measures of interest. OLAP dimensions are further organized in hierarchies that favor the data aggregation process [1]. Several techniques have been developed for query processing, most of them involving some kind of aggregate precomputation.

In spite of the wide corpus of existing work claiming to solve the problem of spatial and spatio-temporal data warehousing and OLAP, there is no clear definition of the meaning of these terms. Moreover, there is no formal notion of "SOLAP query", or "Spatio-temporal OLAP query". Further, existing efforts

do not clearly specify the kinds of queries addressed. As a consequence, comparing proposals or assessing the capabilities of different approaches is difficult. This paper aims at closing that gap in the following way: (a) first, we define a taxonomy of models that integrate OLAP, spatial data, and moving data types; (b) for each of the classes in this taxonomy, we define the queries that they must support; (c) in order to define these classes of queries, starting from the tuple relational calculus extended with aggregate functions, we propose a spatio-temporal calculus supporting moving data types. We show that each extension defines the kinds of queries in each class of the taxonomy.

The remainder of the paper is organized as follows. Section 2 provides a comprehensive background on existing work in GIS-OLAP integration. Section 3 introduces the calculus and conceptual model that we use throughout the paper, as well as presents our running example. Section 4 introduces the taxonomy of data models. Section 5 defines the queries associated to each class in the taxonomy. We conclude and describe future work in Section 6.

2 Related Work

In the last years, the topic of extending OLAP with spatial and temporal features has attracted the attention of the database and GIS communities. In this section we review relevant efforts in this area.

Rivest *et al.* [15] introduced the notion of SOLAP (standing for Spatial OLAP), a paradigm aimed at exploring spatial data by drilling on maps, as it is performed in OLAP with tables and charts. They describe the desirable features and operators a SOLAP system should have. Although they do not present a formal model for this, SOLAP concepts and operators have been implemented in a commercial tool called JMAP[3]. Related to the concept of SOLAP, Shekhar *et al.* [16] introduced MapCube, a visualization tool for spatial data cubes. Given a so-called base map, cartographic preferences, and an aggregation hierarchy, the MapCube operator produces an album of maps that can be navigated via roll-up and drill-down operations.

Several conceptual models have been proposed for spatio-temporal data warehouses. Stefanovic *et al.* [19] classify spatial dimension hierarchies according to their spatial references in: (a) non-geometric; (b) geometric to non-geometric; and (c) fully geometric. Dimensions of type (a) can be treated as any descriptive dimension. In dimensions of types (b) and (c) a geometry is associated to the hierarchy members. Malinowski and Zimányi [9] defined a multidimensional conceptual model, called MultiDim, that copes with spatial and temporal features. The MultiDim model extends the above classification by considering a dimension level as spatial if it is represented as a spatial data type (e.g., point, region), and where spatial levels may be related through topological relationships (e.g., contains, overlaps). In the models above, spatial measures are characterized in two ways, namely: (a) measures representing a geometry, which can be aggregated

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

along the dimensions; (b) numerical measures, calculated using a topological or metric operator. Most proposals support option (a), either as a set of coordinates [15,9], or a set of pointers to geometric objects [19]. Da Silva *et al.* [17] introduced GeoDWFrame, a framework for spatial OLAP, which classifies dimensions in geographic and hybrid, depending on whether they represent only geographic data, or geographic and non-spatial data, respectively. Over this framework, da Silva *et al.* [18] propose GeoMDQL, a query language based on MDX and OGC[4] simple features, for querying spatial data cubes.

It is worth noting that all the above conceptual models follow a *tightly-coupled* approach between the GIS and OLAP components, where the spatial objects are included in the data warehouse. On the contrary, the Piet data model, introduced by Gómez *et al.* [4], follows a *loosely-coupled* approach, where GIS data and data warehouse data are maintained separately, a matching function bounding the two components. Piet supports the notion of *geometric aggregation*, that characterizes a wide range of aggregate queries over regions defined as semi-algebraic sets, addressing four kinds of queries: (a) standard GIS queries; (b) standard OLAP queries; (c) geometric aggregation queries; and (d) integrated GIS-OLAP queries. OLAP-style navigation is also supported in the latter case. Recently, an SQL-like query language was proposed for Piet, denoted Piet-QL. This language, in addition to query types (a) to (d), allows expressing GIS queries filtered by a data cube (i.e., filtered by aggregated data)[5].

Pourabas [13] introduced a conceptual model that uses binding attributes to bridge the gap between spatial databases and a data cube. No implementation of the proposal is discussed. Besides, this approach relies on the assumption that all the cells in the cube contain a value, which is not the usual case in practice. Moreover, the approach also requires modifying the structure of the spatial data.

Traditional data warehouses and OLAP system do not support the evolution of dimension data. *Temporal data warehousing* cope with this issue. Mendelzon and Vaisman [10] proposed a model, denoted TOLAP, and developed a prototype and a Datalog-like query language, based on a temporal star schema. In this model, changes to the structure and/or the instances of the dimension tables are supported, using the concept of transaction and valid time, respectively. Some structural changes also yield different fact table versions. Also, Eder *et al.* [2] propose a data model for temporal OLAP supporting structural changes.

In order to support spatio-temporal data, a data model and associated query language is needed for supporting moving objects, i.e., objects whose geometry evolves over time. This is achieved in Hermes, a system introduced by Pelekis *et al.* [12], and SECONDO [5], a system supporting the model of Güting *et al.* [6]. In spite of their ability to handle spatio-temporal data, neither SECONDO, nor Hermes, are oriented toward addressing the problem of integrating GIS, OLAP, and moving objects. However, in this paper we use many concepts underlying SECONDO, to present our approach. Vega López *et al.* [20] present a comprehensive survey on spatio-temporal aggregation.

[4] Open Geospatial Consortium http://www.opengeospatial.org
[5] A Piet-QL demo can be found at http://piet.exp.dc.uba.ar/pietql.

The work by Orlando *et al.* [11] introduces the concept of *trajectory data warehouses*, aimed at providing the infrastructure needed to deliver advanced reporting capabilities and facilitating the use of mining algorithms on aggregate data. This work is also based on the Hermes system. A relevant feature of this proposal is the treatment given to the ETL (Extraction, Transformation and Loading) process, which transforms the raw location data and loads it to the trajectory data warehouse.

3 Preliminaries

3.1 Extending the Conceptual Model

Throughout the paper we use the following real-world example. The Environmental Control Agency of a country has a collection of water stations measuring the value of polluting substances at regular time intervals. The application has maps describing rivers, water stations, and the political division of the country into provinces and districts.

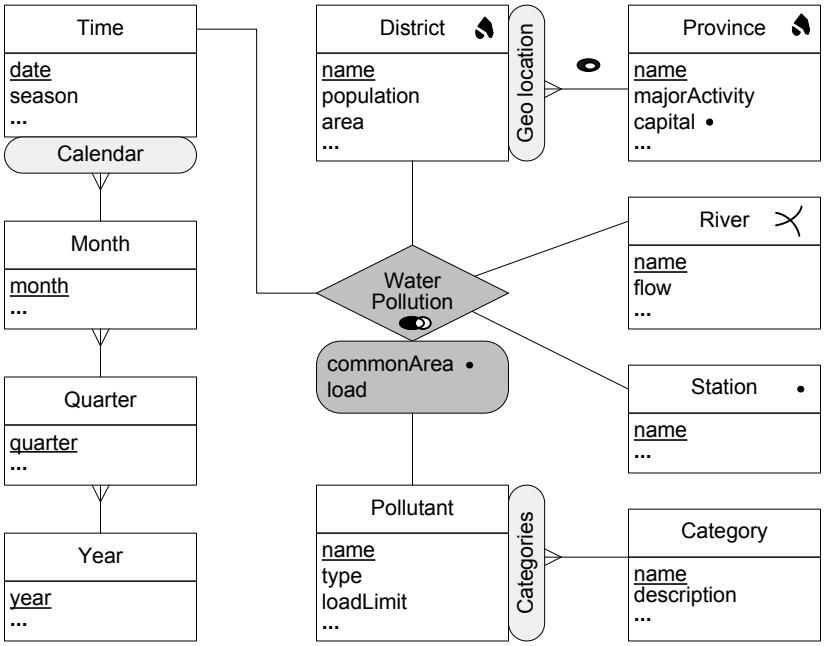

Fig. 1. An example of a spatial data warehouse.

Figure 1 shows the conceptual schema depicting the above scenario using the MultiDim model [9]. There is one fact relationship, WaterPollution, to which

several dimensions are related. The fact relationship WaterPollution has two measures, commonArea and load, and is related to five dimensions: Time, Station, Pollutant, River, and District. Dimensions are composed of levels and hierarchies. For example, while the Station dimension has only one level, the District dimension is composed of two levels, District and Province, with a one-to-many parent-child relationship defined between them.

In the MultiDim model the spatiality of elements is indicated by pictograms. For example, Station, River, and District are spatial levels; they have a geometry represented by a point, a line, and a region, respectively. Similarly, the attribute capital in Province, as well as the measures commonArea in the three fact relationships are spatial. Finally, topological relationships may be represented in fact relationships and in parent-child relationships. For example, the topological relationship in WaterPollution indicates that whenever a water station, a river, and a district are related in an instance of the relationship, they must overlap. Similarly, the topological relationship in the hierarchy of dimension District indicates that a district is covered by its parent province.

To address spatio-temporal scenarios we borrow the data types defined by Güting *et al.* [6]. We refer to this work for the complete definition of the type system and the corresponding operations. There is a set of *base types* which are int, real, bool, string, and an *identifier type* id, which is used for the identifiers of level members. There are also *time types* which are instant and periods, the latter being a set of time intervals. 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. It is allowed that a face lies whith a hole of another face.

Moving types capture the evolution over time of base types and spatial types. Moving types are obtained by applying a constructor moving$(\cdot)$. Hence, a value of type moving(point) is a continuous function f : instant $\rightarrow$ point. Moving types have associated operations that generalize those of the non-temporal types. This is called *lifting*. For example, a distance function with signature moving(point) $\times$ moving(point) $\rightarrow$ moving(real) calculates the distance between two moving points and gives as result a moving real, i.e., a real-valued function of time. Intuitively, the semantics of such lifted operations is that the result is computed at each time instant using the non-lifted operation. Definition 1 summarizes the concepts discussed above.

Definition 1 (Data types). We denote Γ a set of *nontemporal types*, composed of a set of *base types* β, a set of *time types* τ, and a set of *spatial types* ξ. There is also a set of *temporal types* Φ, composed of two sets of temporal types ϕ_β and ϕ_ξ, obtained by applying the moving constructor to elements of β and ξ, respectively.

3.2 Spatio-Temporal Calculus

For addressing the issue of querying data warehouses, we use a relational representation of the MultiDim conceptual model. A dimension level is represented

by a relation of the same name, having an implicit identifier attribute denoted id, an implicit geometry attribute (if the level is spatial), in addition to the other explicitly indicated attributes. The id attribute (e.g., River.id) identifies a particular instance of the dimension. Dimension levels involved in hierarchies (e.g., District) have also an additional attribute containing the identifier of the parent level (e.g., District.province), and there is a referential integrity constraint for such attributes and the corresponding parent (e.g., Province.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 for each measure. There is a referential integrity constraint between the dimension attributes in the fact relationship (e.g., WaterPollution.district) and the identifier of the corresponding dimension (e.g., District.id).

We use a query language based on the tuple relational calculus (e.g., [3]) extended with aggregate functions and variable definitions[6]. We explain this language through an example. Consider the following relations from the data warehouse shown in Fig. 1.

District(id, geometry, districtName, population, area, ... , province)
Province(id, geometry, provinceName, majorActivity, capital, governor, ...).

The following query asks the name and population of districts of the Antwerp province.

$$\{d.\mathsf{districtName}, d.\mathsf{population} \mid \mathsf{District}(d) \wedge \exists p\,(\mathsf{Province}(p) \wedge d.\mathsf{province} = p.\mathsf{id} \wedge p.\mathsf{provinceName} = \text{'Antwerp'})\}$$

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

$$\mathsf{sum}(\{d.\mathsf{population} \mid \mathsf{District}(d) \wedge \exists p\,(\mathsf{Province}(p) \wedge d.\mathsf{province} = p.\mathsf{id} \wedge p.\mathsf{provinceName} = \text{'Antwerp'})\})$$

Notice that however, since the relational calculus is based on sets (i.e., collections with no duplicates), if two districts of the Antwerp province happen to have the same population, they would appear only once in the set to which the sum operator is applied. As in Klug's approach [8], this is solved by using aggregate operators that take as argument a set of tuples (instead of a set of values) and that specify on which column the aggregate operator must be applied. Therefore, the above query is more precisely written as follows.

$$\mathsf{sum}_2(\{d.\mathsf{id}, d.\mathsf{population} \mid \mathsf{District}(d) \wedge \exists p\,(\mathsf{Province}(p) \wedge d.\mathsf{province} = p.\mathsf{id} \wedge p.\mathsf{provinceName} = \text{'Antwerp'})\}$$

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.

[6] Even though languages for OLAP manipulation exist (e.g., [14]), the choice of a relational calculus is motivated by the fact that it applies to the classical relational model, thus providing a clean and elegant way for our purpose of defining different Spatio-temporal OLAP models and languages.

Finally, suppose we want to calculate the total population by province 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. The latter query is written as follows.

$$\{p.\mathsf{name}, \mathsf{totalPop} \mid \mathsf{Province}(p) \wedge \\ \mathsf{totalPop} = \mathsf{sum}_2(\{d.\mathsf{id}, d.\mathsf{population} \mid \mathsf{District}(d) \wedge d.\mathsf{province} = p.\mathsf{id}\}) \wedge \\ \mathsf{totalPop} > 100,000 \}$$

Here, the outer query fixes a particular province p and the inner query collects the population of districts for that province. 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.

Proposition 1. Let us denote $\mathcal{R}_{agg}$ the relational calculus with aggregate functions defined above, over the basic sets of data and time types β and τ, respectively. $\mathcal{R}_{agg}$ has the same expressive power of the relational calculus extended with aggregate functions defined in [8].

The idea of the proof follows from the fact that $\mathcal{R}_{agg}$ is a syntactic variation of Klug's calculus. We show later in the paper how we extend $\mathcal{R}_{agg}$ to support spatial and moving data types in order to define a hierarchy of classes of spatio-temporal queries, starting from the expressive power of $\mathcal{R}_{agg}$.

4 A Taxonomy for Spatio-Temporal OLAP

Existing proposals for spatial data warehousing cover different functional requirements, but, with limited exceptions, there is no clear specification of the kinds of queries these proposals address. This is probably due to the fact that no taxonomy for these systems has been defined so far. When we talk about GIS, we often refer to *static* GIS, i.e., GIS where the geometry of objects does not change over time. On the other hand, when we talk about OLAP or data warehousing, we assume *static* data warehousing, i.e., data warehouses where dimensions do not change over time. Thus, the term SOLAP refers to the interaction between static GIS and static data warehouses. The schema in Fig. 1 is an example of this approach. When time gets into play, things become more involved, and only partial solutions have been provided. On the one hand, different models exist for temporal data warehousing, depending on the approach followed to implement the warehouse. In this paper we define a temporal data warehouse as a warehouse that keeps track of the history of the *instances* of the warehouse dimensions, i.e., we assume there is no structural (schema) changes. The reason for this is that, as far as we know, only academic implementations of fully temporal data warehouses exist.

We define a taxonomy for spatio-temporal OLAP as follows (see Fig. 2). We start by considering four basic classes: Temporal dimensions, OLAP, GIS, and moving data types. As a derived basic class, adding moving data types to GIS

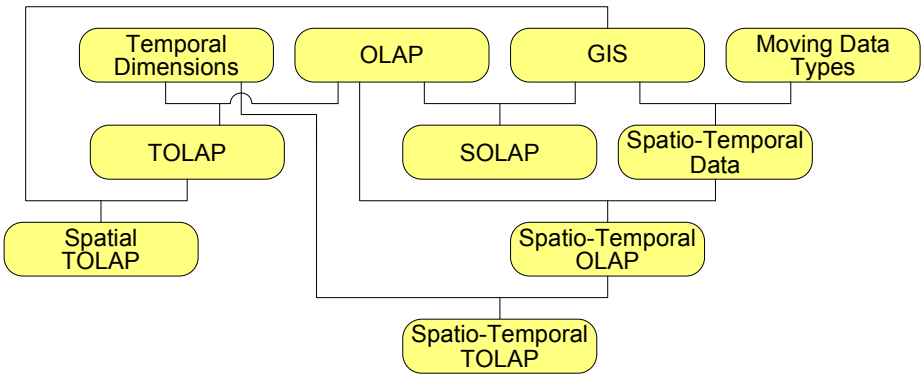

Fig. 2. A taxonomy for spatio-temporal data warehousing.

produces *Spatio-Temporal Data*, typically allowing trajectory analysis in a geographic environment. Providing OLAP with the ability of handling temporal dimensions produces the concept of *Temporal OLAP* (TOLAP). The interaction of OLAP and GIS is denoted *Spatial OLAP* (SOLAP). The interaction between GIS and TOLAP is called *Spatio-Temporal OLAP* (S-TOLAP). Adding OLAP capabilities to spatio-temporal data results in *Spatio-Temporal OLAP* (ST-OLAP). Finally, if the latter supports temporal dimensions we have *Spatio-Temporal TOLAP* (ST-TOLAP).

5 Queries

In this section, we define the kinds of queries that should be supported for each one of the classes in the taxonomy of Fig. 2.

5.1 OLAP and Spatial OLAP Queries

We start by showing examples of OLAP queries.

Q1. For water stations located in districts of the Limburg province and polluting agents of organic category give the maximum load by month.

$$\{s.\mathsf{name}, p.\mathsf{name}, m.\mathsf{month}, \mathsf{maxLoad} \mid \mathsf{Station}(s) \wedge \mathsf{Pollutant}(p) \wedge \mathsf{Month}(m) \wedge \exists c\,(\mathsf{Category}(c) \wedge p.\mathsf{category} = c.\mathsf{id} \wedge c.\mathsf{name} = \text{'Organic'}) \wedge \mathsf{maxLoad} = \mathsf{max}_1(\{w.\mathsf{load} \mid \mathsf{WaterPollution}(w) \wedge w.\mathsf{station} = s.\mathsf{id} \wedge w.\mathsf{pollutant} = p.\mathsf{id} \wedge \exists d, \exists v, \exists t\,(\,\mathsf{District}(d) \wedge \mathsf{Province}(v) \wedge \mathsf{Time}(t) \wedge w.\mathsf{district} = d.\mathsf{id} \wedge d.\mathsf{province} = v.\mathsf{id} \wedge v.\mathsf{name} = \text{'Limburg'} \wedge w.\mathsf{time} = t.\mathsf{id} \wedge t.\mathsf{month} = m.\mathsf{id})\})\}$$

Q2. For each river, give the total number of stations where, for at least one pollutant, the average load in March 2008 was greater than the load limit for this pollutant.

$$\{r.\mathsf{name}, \mathsf{nbStations} \mid \mathsf{River}(r) \wedge \mathsf{nbStations} = \mathsf{count}(\{s.\mathsf{id} \mid \mathsf{Station}(s) \wedge \exists p\,(\mathsf{Pollutant}(p) \wedge \mathsf{avg}_2(\{w.\mathsf{id}, w.\mathsf{load} \mid \mathsf{WaterPollution}(w) \wedge w.\mathsf{river} = r.\mathsf{id} \wedge w.\mathsf{station} = s.\mathsf{id} \wedge w.\mathsf{pollutant} = p.\mathsf{id} \wedge \exists t(\mathsf{Time}(t) \wedge w.\mathsf{time} = t.\mathsf{id} \wedge t.\mathsf{date} \geq 1/3/2008 \wedge t.\mathsf{date} \leq 31/3/2008)\}) > p.\mathsf{loadLimit})\})\}$$

Definition 2 (OLAP queries). Let us call $\mathcal{R}_{agg}$ the relational calculus with aggregate functions defined in Section 3.2. The class of OLAP queries includes all the queries that are expressible by $\mathcal{R}_{agg}$.

We give next some examples of SOLAP queries.

Q3. Total population in the districts within 3Km from the Ghent district that are crossed by the Schelde river.

$$\mathsf{sum}_2(\{d_1.\mathsf{id}, d_1.\mathsf{population} \mid \mathsf{District}(d_1) \wedge \exists d_2, \exists r(\mathsf{District}(d_2) \wedge \mathsf{River}(r) \wedge d_2.\mathsf{name} = \text{'Ghent'} \wedge \mathsf{distance}(d_1.\mathsf{geometry}, d_2.\mathsf{geometry}) < 3 \wedge r.\mathsf{name} = \text{'Schelde'} \wedge \mathsf{intersects}(d_1.\mathsf{geometry}, r.\mathsf{geometry}))\})$$

Note that this query do not use a fact relationship. The function distance verifies that the geometries of the two districts are less than 3Km from each other and the predicate intersects verifies that the district is crossed by the river.

Q4. Stations located over the part of the Schelde river that flows through the Antwerp province, with an average content of nitrates in the last quarter of 2008 above the load limit for that pollutant.

$$\{s.\mathsf{name} \mid \mathsf{Station}(s) \wedge \exists r, \exists p, \exists l, \exists c\,(\mathsf{River}(r) \wedge \mathsf{Province}(p) \wedge \mathsf{Pollutant}(l) \wedge \mathsf{Category}(c) \wedge r.\mathsf{name} = \text{'Schelde'} \wedge p.\mathsf{name} = \text{'Antwerp'} \wedge \mathsf{inside}(s.\mathsf{geometry}, \mathsf{intersection}(r.\mathsf{geometry}, p.\mathsf{geometry})) \wedge l.\mathsf{category} = c.\mathsf{id} \wedge c.\mathsf{name} = \text{'Nitrates'}x \wedge \mathsf{avg}_2(\{w.\mathsf{id}, w.\mathsf{load} \mid \mathsf{WaterPollution}(w) \wedge w.\mathsf{station} = s.\mathsf{id} \wedge w.\mathsf{pollutant} = l.\mathsf{id} \wedge \exists t\,(\mathsf{Time}(t) \wedge w.\mathsf{time} = t.\mathsf{id} \wedge t.\mathsf{date} \geq 1/10/2008 \wedge t.\mathsf{date} \leq 31/12/2008)\}) > l.\mathsf{loadLimit})\}$$

Here, the intersection of the river and the district is computed, and then it is verified that the geometry of the station is located inside this intersection.

Definition 3 (SOLAP queries). Let us call $\mathcal{R}_{agg}^{\xi}$ the language $\mathcal{R}_{agg}$ augmented with spatial types in ξ. The class of SOLAP queries is the class composed of all the queries that can be expressed by $\mathcal{R}_{agg}^{\xi}$.

5.2 Temporal OLAP Queries

The notion of Temporal OLAP (TOLAP) arises when evolution of the dimension instances in the data warehouse is supported, a problem also referred to as *slowly-changing dimensions* [7].

This evolution is captured by using temporal types. In other words, when at least one of the dimensions in the data warehouse includes a type in the set ϕ_β of Definition 1, we say that the warehouse supports the TOLAP model.

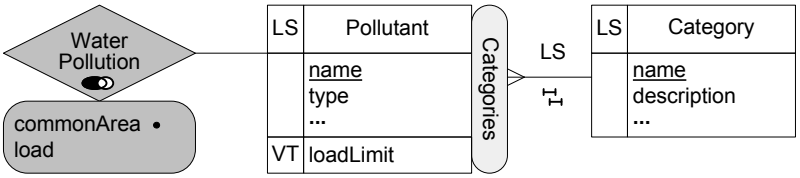

Fig. 3. A temporal dimension Pollutant.

To define TOLAP queries, we modify our running example in Fig. 1 making the dimension Pollutant temporal, as shown in Fig. 3. *Temporal levels* are identified by the LS pictogram. In our example, the level Pollutant is temporal, which means that a new pollutant may start to be monitored from a particular date. Temporal levels have a predefined attribute called lifespan, of type moving(bool), which keeps track of the validity of a member at each instant. *Temporal attributes* are identified by the VT pictogram. In our example, the attribute loadLimit is temporal, meaning that the load limit varies across time.

Temporal attributes are defined over temporal types, for example, moving(real) for the attribute loadLimit. Finally, *temporal parent-child relationships* are indicated by the LS pictogram. In our example, the relationship between Pollutant and Category is temporal, which means that the association of pollutants to categories varies over time, e.g., at a particular date, a category can be split into two. Temporal relationships are also represented by temporal types. For example, the Pollutant level has an attribute category, of type moving(id), which associates, at each time instant, a category to a pollutant.

Notice that the temporal multidimensional model assumes implicit constraints that restrict the lifespan of the instances of temporal levels participating in fact relationships or in temporal parent-child relationships. For example, an instance of the fact relationship WaterPollution relates a time instant t and a pollutant p provided that t is included in the lifespan of p. Similarly, a pollutant p is related to a pollutant category c at instant t provided that t is included in the lifespan of c. We consider the following queries.

Q5. For each province and pollutant category, give the average load of water pollution by quarter.

$$\{p.\mathsf{name}, c.\mathsf{name}, q.\mathsf{quarter}, \mathsf{avgLoad} \mid \mathsf{Province}(p) \wedge \mathsf{Category}(c) \wedge \mathsf{Quarter}(q) \wedge \mathsf{avgLoad} = \mathsf{avg}_2(\{w.\mathsf{id}, w.\mathsf{load} \mid \mathsf{WaterPollution}(w) \wedge \exists d, \exists t, \exists m, \exists l\ (\mathsf{District}(d) \wedge \mathsf{Time}(t) \wedge \mathsf{Month}(m) \wedge \mathsf{Pollutant}(l) \wedge w.\mathsf{district} = d.\mathsf{id} \wedge d.\mathsf{province} = p.\mathsf{id} \wedge w.\mathsf{time} = t.\mathsf{id} \wedge t.\mathsf{month} = m.\mathsf{id} \wedge m.\mathsf{quarter} = q.\mathsf{id} \wedge w.\mathsf{pollutant} = l.\mathsf{id} \wedge \mathsf{val}(\mathsf{initial}(\mathsf{atperiods}(l.\mathsf{category}, t))) = c.\mathsf{id})\})\}$$

In the last line of the above query, since the parent-child relationship is represented by the temporal attribute category, we need to obtain the value of this attribute at the time defined by the instance t of the Time dimension. As the granularity of the Time dimension is day, function atperiods restricts the

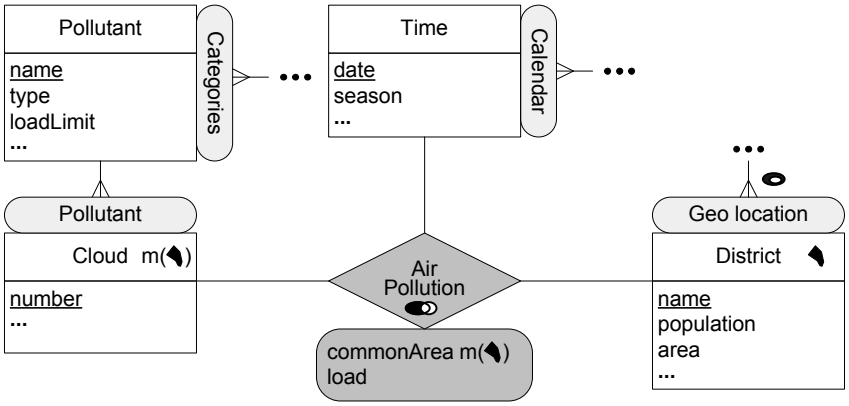

Fig. 4. A fact relationship with a spatio-temporal measure and a spatio-temporal dimension.

temporal attribute to that day, function initial takes the first ⟨instant, value⟩ pair of the function, and function val returns the corresponding value. There is no need to verify that the lifespan of the instances of Pollutant or Category include the time t, since these constraints are implicity kept by the model.

Q6. Calculate by day the number of water stations that, for the pollutant 'lead', had load greater than the maximum value (over its history) of the load limit.

$$\{t.\mathsf{date}, \mathsf{nbStations} \mid \mathsf{Time}(t) \wedge \mathsf{nbStations} = \mathsf{count}_1(\{s.\mathsf{id} \mid \mathsf{Station}(s) \wedge \exists w, \exists p\, (\mathsf{WaterPollution}(w) \wedge \mathsf{Pollutant}(p) \wedge w.\mathsf{time} = t.\mathsf{id} \wedge w.\mathsf{station} = s.\mathsf{id} \wedge w.\mathsf{pollutant} = p.\mathsf{id} \wedge p.\mathsf{pollutant} = \text{'Lead'} \wedge p.\mathsf{load} > \mathsf{val}(\mathsf{initial}(\mathsf{atmax}(p.\mathsf{loadLimit}))))\})\}$$

In the query above, function atmax restricts the temporal attribute to the time instants during which it has its maximum value, function initial takes the ⟨instant, value⟩ pair of the first instant and function val obtains its value.

Definition 4 (TOLAP queries). Let us call $\mathcal{R}_{agg}^{\phi_\beta}$ the language $\mathcal{R}_{agg}$ augmented with the data types in ϕ_β. The class of TOLAP queries is the class composed of all the queries that can be expressed by $\mathcal{R}_{agg}^{\phi_\beta}$.

5.3 Spatio-Temporal OLAP Queries

Spatio-temporal OLAP (ST-OLAP) accounts for the case when the spatial objects evolve over time. For this, we need to consider moving types defined by moving(α) where α is a spatial data type.

In order to define ST-OLAP queries, we add the fact relationship shown in Fig. 4 to our running example in Fig. 1. The Cloud dimension refers to clouds generated by industrial plants. Both the Cloud level and the commonArea measure have a geometry that is a moving region, indicated by the symbol 'm'. Notice that commonArea is a derived measure, i.e., in an instance of the fact

relationship that relates a cloud c, a district d, and a date t, the measure keeps the restriction of the trajectory of the cloud at that date and over that district. This is computed by the expression $\mathsf{at}(\mathsf{atperiods}(c.\mathsf{geometry}, t), d)$. Notice also that the Cloud dimension is related to the Pollutant dimension, while in Fig. 1 the Pollutant dimension participates in the fact relationship.

We give next examples of ST-OLAP queries.

Q7. For each district and polluting cloud, give the duration of time when the cloud passed over the district.

$$\{d.\mathsf{name}, c.\mathsf{number}, \mathsf{dur} \mid \mathsf{District}(d) \wedge \mathsf{Cloud}(c) \wedge \mathsf{dur} = \mathsf{duration}(\mathsf{deftime}(\mathsf{at}(c.\mathsf{geometry}, d.\mathsf{geometry})))\}$$

Function at selects the part of moving geometry that is over the district, function deftime obtains the periods when this happens, and finally function duration calculates the size of the corresponding periods.

Q8. For each district, give by month the total number of persons affected by polluting clouds.

$$\{d.\mathsf{name}, m.\mathsf{month}, \mathsf{totalNo} \mid \mathsf{District}(d) \wedge \mathsf{Month}(m) \wedge \mathsf{totalNo} = \mathsf{area}(\mathsf{union}(\{\mathsf{traversed}(p.\mathsf{commonArea}) \mid \mathsf{AirPollution}(p) \wedge p.\mathsf{district} = d.\mathsf{id} \wedge \exists t\,(\mathsf{Time}(t) \wedge t.\mathsf{month} = m.\mathsf{id})\}))/\mathsf{area}(d.\mathsf{geometry}) \times d.\mathsf{population}\}$$

The inner query selects all facts relating a given district and a day of a given month; then function traversed projects the moving geometry of the commonArea measure over the plane. A union of all the regions thus obtained yields the part of the district affected by polluting clouds during that month, and the area of this region is then computed. Then, assuming a uniform distribution of the population, we divide this by the total area of the district and multiply that by its population.

Definition 5 (ST-OLAP queries). Let us call $\mathcal{R}_{agg}^{\phi_\xi}$ the language $\mathcal{R}_{agg}$ augmented with spatial types in ξ, and moving spatial types in ϕ_ξ. The class of ST-OLAP queries is the class composed of all the queries that can be expressed by $\mathcal{R}_{agg}^{\phi_\xi}$.

Definition 5 captures the model of Orlando *et al.* [11] on trajectory data warehousing, since, actually, such model aims at answering queries like "Total number of trajectories in a square of a grid" and performing OLAP operations.

5.4 Spatial TOLAP Queries

Spatial TOLAP (S-TOLAP) covers the case when in addition to having spatial objects and attributes in the data warehouse, the dimensions are also temporal. As we have done in Sect. 5.2, we modify our running example in Fig. 1 so that the dimension Pollutant is temporal, as shown in Fig. 3.

Q9. For each station located over the Schelde river, give the periods of time during the last quarter of 2008 when the content of nitrates was above the load limit for that pollutant.

$\{s.\mathsf{name}, \mathsf{periods} \mid \mathsf{Station}(s) \wedge \exists r\,(\mathsf{River}(r) \wedge r.\mathsf{name} = \text{'Schelde'} \wedge$
$\mathsf{inside}(s.\mathsf{geometry}, r.\mathsf{geometry}) \wedge \mathsf{periods} = \mathsf{union}_1(\{t.\mathsf{date} \mid \mathsf{Time}(t) \wedge$
$\exists w, \exists p, \exists c\,(\mathsf{WaterPollution}(w) \wedge \mathsf{Pollutant}(p) \wedge \mathsf{Category}(c) \wedge$
$w.\mathsf{station} = s.\mathsf{id} \wedge w.\mathsf{time} = t.\mathsf{id} \wedge w.\mathsf{pollutant} = p.\mathsf{id} \wedge$
$p.\mathsf{category} = c.\mathsf{id} \wedge c.\mathsf{name} = \text{'Nitrates'} \wedge$
$t.\mathsf{date} \geq 1/10/2008 \wedge t.\mathsf{date} \leq 31/12/2008 \wedge$
$w.\mathsf{load} > \mathsf{val}(\mathsf{atinstant}(p.\mathsf{loadLimit}, \mathsf{now})))\})\})\}$

As usual in temporal databases, a distiguished variable, 'now', represents the (moving) current instant. Thus, the above query assumes that the comparison should be made with respect to the *current* value for the limit. Then, the union operator takes a set of dates and construct a minimal set of disjoint periods.

Q10. For each month in 2008, and for each water station in the province of Namur, give the average Biological Oxygen Demand (BOD), if this average is larger than the load limit during the reported month.

$\{m.\mathsf{month}, s.\mathsf{name}, \mathsf{avgBOD} \mid \mathsf{Month}(m) \wedge \mathsf{Station}(s) \wedge \exists q, \exists y, \exists p, \exists l\,($
$\mathsf{Quarter}(q) \wedge \mathsf{Year}(y) \wedge \mathsf{Province}(p) \wedge \mathsf{Pollutant}(l) \wedge m.\mathsf{quarter} = q.\mathsf{id} \wedge$
$q.\mathsf{year} = y.\mathsf{id} \wedge y.\mathsf{year} = 2008 \wedge p.\mathsf{name} = \text{'Namur'} \wedge$
$\mathsf{inside}(s.\mathsf{geometry}, p.\mathsf{geometry}) \wedge l.\mathsf{name} = \text{'BOD'} \wedge$
$\mathsf{avgBOD} = \mathsf{avg}_2(\{w.\mathsf{id}, w.\mathsf{load} \mid \mathsf{WaterPollution}(w) \wedge \exists t\,($
$\mathsf{Time}(t) \wedge w.\mathsf{station} = s.\mathsf{id} \wedge w.\mathsf{time} = t.\mathsf{id} \wedge$
$t.\mathsf{month} = m.\mathsf{id} \wedge w.\mathsf{pollutant} = l.\mathsf{id})\}) \wedge$
$\mathsf{avgBOD} > \mathsf{val}(\mathsf{initial}(\mathsf{atperiods}(l.\mathsf{loadLimit}, m.\mathsf{month}))))\}$

In the last term above, function atperiods restrict the load limit to month m, and then functions initial and val obtain the value of this attribute at the first day of the month. This is compared with the average load of that month in avgBOD.

Definition 6 (Spatial TOLAP queries). Let us call $\mathcal{R}_{agg}^{\xi,\phi_\beta}$ the language $\mathcal{R}_{agg}$ augmented with spatial types in ξ and moving types in ϕ_β. We denote S-TOLAP the class of queries composed of all the queries that can be expressed by $\mathcal{R}_{agg}^{\xi,\phi_\beta}$.

5.5 Spatio-Temporal TOLAP Queries

Spatio-Temporal TOLAP (ST-TOLAP) is the most general case where there are moving geometries and the dimensions vary over time. In our running example in Fig. 1 this amounts to replace the temporal dimension Pollutant as in Fig. 3 and to include the AirPollution fact relationship in Fig. 4. An example of these kinds of queries is the following.

Q11. Total number of days when the Gent district has been under at least one cloud of carbon monoxide (CO) such that the average load in the cloud is larger than the load limit at the time when the cloud appeared.

$\mathsf{duration}(\mathsf{union}(\{t.\mathsf{date} \mid \mathsf{Time}(t) \wedge \exists p, \exists d, \exists c, \exists l\,(\mathsf{AirPollution}(p) \wedge$
$\mathsf{District}(d) \wedge \mathsf{Cloud}(c) \wedge \mathsf{Pollutant}(l) \wedge p.\mathsf{time} = t.\mathsf{id} \wedge$
$p.\mathsf{district} = d.\mathsf{id} \wedge d.\mathsf{name} = \text{'Ghent'} \wedge p.\mathsf{cloud} = c.\mathsf{id} \wedge$
$c.\mathsf{pollutant} = l.\mathsf{id} \wedge l.\mathsf{name} = \text{'CO'} \wedge p.\mathsf{load} >$
$\mathsf{val}(\mathsf{atinstant}(l.\mathsf{loadLimit}, \mathsf{inst}(\mathsf{initial}(\mathsf{at}(c.\mathsf{lifespan}, \mathsf{true})))))\})\})\}$

To obtain the instant when a cloud appeared we use the functions at (to restrict the lifespan of the cloud), initial and inst. Functions atinstant and val return the value of the load limit at the instant when the cloud appeared.

Definition 7 (Spatio-Temporal TOLAP queries). Let us call $\mathcal{R}_{agg}^{\xi,\phi_\xi,\phi_\beta}$ the language $\mathcal{R}_{agg}$ augmented with spatial types in ξ, moving spatial types in ϕ_ξ, and moving types in ϕ_β. We denote ST-TOLAP the class of queries composed of all the queries that can be expressed by $\mathcal{R}_{agg}^{\xi,\phi_\xi,\phi_\beta}$.

6 Conclusion

In this paper, we have defined a conceptual framework that allows characterizing the functionalities that must be supported by spatio-temporal data warehouses. We have shown that such data warehouses result from the combination of GIS and OLAP technologies, further extended with the support of temporal data types. These data types allow to model both, geometries that evolve over time (usually called moving objects), and evolving data warehouse dimensions.

To address the issue of querying spatio-temporal data warehouses, we have defined an extension of the tuple relational calculus with aggregate functions. We defined a taxonomy for spatio-temporal OLAP queries that, starting from the class of traditional OLAP queries, incrementally adds features for defining several classes of queries with increasing expressive power. This is realized by extending the type system underlying the data warehouse and its associated query language. Our taxonomy provides an elegant and uniform way to characterize the features required by spatio-temporal data warehouses and to classify the many different works addressing this issue in the literature.

This work constitutes a first step aiming at defining spatio-temporal data warehouses and therefore many issues remain to be addressed. As we have mentioned above, our framework is defined at a conceptual level and therefore we have omitted any implementation consideration. However, as can be expected, spatio-temporal data warehouses contain huge amounts of data, and therefore optimization issues are of paramount importance. These issues range from appropriate index structures, through pre-aggregation, to efficient query optimization. With respect to the latter issue, our example queries can be expressed in several ways, exploiting either the fact relationship or directly the moving geometries. Although from a formal perspective these alternative queries are equivalent, since they yield the same result, the evaluation time of these queries may vary significatively, depending on the actual population of the data warehouse. Therefore, the translation of our conceptual model into logical and physical models is still another further work.

Acknowledgments. This research has been partially funded by 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.

References

1. Cabibbo, L., Torlone, R.: Querying multidimensional databases. In: Proc. of DBPL. (1997) 253–269
2. Eder, J., Koncilia, C., Morzy, T.: The COMET metamodel for temporal data warehouses. In: Proc. of CAiSE. (2002) 83–99
3. Elmasri, R., Navathe, S.: Fundamentals of Database Systems. Fifth edn. Addison-Wesley (2007)
4. Gómez, L., Haesevoets, S., Kuijpers, B., Vaisman, A.: Spatial aggregation: Data model and implementation. CoRR **abs/0707.4304** (2007)
5. Güting, R.H., de Almeida, V.T., Ansorge, D., Behr, T., Ding, Z., Höse, T., Hoffmann, F., Spiekermann, M., Telle, U.: SECONDO: An extensible DBMS platform for research prototyping and teaching. In: Proc. of ICDE. (2005) 1115–1116
6. Güting, R.H., Schneider, M.: Moving Objects Databases. Morgan Kaufmann (2005)
7. Kimball, R.: The Data Warehouse Toolkit. J. Wiley and Sons, Inc. (1996)
8. Klug, A.: Equivalence of relational algebra and relational calculus query languages having aggregate functions. Journal of the ACM **29**(3) (1982) 699–717
9. Malinowski, E., Zimányi, E.: Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications. Springer (2008)
10. Mendelzon, A., Vaisman, A.: Temporal queries in OLAP. In: Proc. of VLDB. (2000) 242–253
11. Orlando, S., Orsini, R., Raffaetà, A., Roncato, A., Silvestri, C.: Spatio-temporal aggregations in trajectory data warehouses. In: Proc. of DaWaK. (2007) 66–77
12. Pelekis, N., Theodoridis, Y., Vosinakis, S., Panayiotopoulos, T.: Hermes: A framework for location-based data management. In: Proc. of EDBT. (2006) 1130–1134
13. Pourabas, E.: Cooperation with geographic databases. In Raffanelli, M., ed.: Multidimensional Databases. Idea Group (2003) 166–199
14. Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Algebraic and graphic languages for OLAP manipulations. International Journal of Data Warehousing and Mining **4**(1) (2008) 17–46
15. Rivest, S., Bédard, Y., Marchand, P.: Toward better suppport for spatial decision making: Defining the characteristics of spatial on-line analytical processing (SOLAP). Geomatica **55**(4) (2001) 539–555
16. Shekhar, S., Lu, C., Tan, X., Chawla, S., Vatsavai, R.: MapCube: A visualization tool for spatial data warehouses. In Miller, H., Han, J., eds.: Geographic data mining and Knowledge Discovery (GKD). Taylor and Francis (2001) 74–109
17. Silva, J., Times, V.C., Salgado, A.C.: An open source and web based framework for geographic and multidimensional processing. In: Proc. of SAC. (2006) 63–67
18. Silva, J., Castro Vera, A.S., Oliveira, A.G., Fidalgo, R., Salgado, A.C., Times, V.C.: Querying geographical data warehouses with GeoMDQL. In: Proc. of SBBD. (2007) 223–237
19. Stefanovic, N., Han, J., Koperski, K.: Object-based selective materialization for efficient implementation of spatial data cubes. IEEE Transactions on Knowledge and Data Engineering **12**(6) (2000) 938–958
20. Vega López, I.F., Snodgrass, R.T., Moon, B.: Spatiotemporal aggregate computation: A survey. IEEE Transactions on Knowledge and Data Engineering **17**(2) (2005) 744–759
21. Worboys, M.F., Duckham, M.: GIS: A Computing Perspective. Second edn. CRC Press (2004)