A Data Model for Moving Objects Supporting Aggregation

Bart Kuijpers
Hasselt University &
Transnational University of Limburg
bart.kuijpers@uhasselt.be

Alejandro A. Vaisman
Universidad de Chile &
Universidad de Buenos Aires
avaisman@dcc.uchile.cl

Abstract

Moving objects databases (MOD) have been receiving increasing attention from the database community in recent years, mainly due to the wide variety of applications that technology allows nowadays. Trajectories of moving objects like cars or pedestrians, can be reconstructed by means of samples describing the locations of these objects at certain points in time. Although there are many proposals for modeling and querying moving objects, only a small part of them address the problem of aggregation of moving objects data in a GIS (Geographic Information Systems) scenario. In previous work we presented a formal model where the geometric components of the thematic layers in a GIS are represented as an OLAP (On Line Analytical Processing) dimension hierarchy, and introduced the notion of spatial aggregation*. In this paper we extend this proposal in order to address moving object aggregation over a GIS. In this way, complex aggregate queries can be expressed in an elegant fashion. We present the data model, characterize the kinds of queries that may appear in this scenario, and show how these queries can be expressed as an aggregation over the result given by a first order formula expressing constraints over the geometries of the layers.*

1 Introduction

Geographic Information Systems (GIS) have been used extensively in various application domains, ranging from economical, ecological and demographic analysis, to city and route planning [14, 20]. Spatial information in a GIS is typically stored in different so-called *thematic layers* (also called *themes*). Information in themes can be stored in different data structures according to different data models, the most usual ones being the *raster model* and the *vector model*. In a thematic layer, spatial data is typically annotated with classical relational attribute information of (in general) numeric or string type. While spatial data is stored in data structures suitable for these kinds of data, associated attributes are usually stored in conventional relational databases.

OLAP (On Line Analytical Processing) [8] 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 is organized as a set of *dimensions* and *fact tables*. Thus, data is perceived as a *data cube*, where each cell of the cube contains a measure or set of (probably aggregated) measures of interest. OLAP dimensions are further organized in hierarchies that favor the data aggregation process.

Moving objects representation and computing have received a fair share of attention over recent years in the database community [2, 3, 10, 18]. In particular, we are interested in aggregation of moving objects data. There are many applications involving moving objects aggregation, mainly regarding traffic analysis, like truck fleet behavior analysis or commuter traffic in a city. The behavior of all these moving objects is traceable by means of electronic devices.

In [4] we have addressed the problem of aggregating spatial data, and integrating OLAP and GIS environments. In this work, we will extend the model in order to address the aggregation of data about moving objects over a spatial GIS.

1.1 Motivating Example

Throughout this paper we will be working with the following example: on the one hand, we have a layered representation of geographic features of a city. Each layer contains information about neighborhoods (polygons), highways (polylines), streets (polylines), hospitals, schools (points), important stores (points), gas stations (points). There is a river dividing the city into a northern and a southern part. Also, a bounding box determines the portion of the city and its surroundings under consideration. In addition, there is numerical and categorical information stored in a conventional data warehouse. In this data warehouse, there are dimension tables containing information about,

1-4244-0832-6/07/$20.00

for instance, stores, gas stations, schools; there is also a fact table containing economic information based on these dimensions. We also have information about moving objects (pedestrians, bicycles, cars, and so on) represented by means of object identifiers. As we progress in the paper, we will get into more detail on how this information is stored in the different layers, and how it can be integrated into a general GIS-OLAP framework.

1.2 Problem Statement

Assume that moving objects data are captured at a given time interval, with a certain granularity. Thus, the trajectory of a moving object is given by samples composed by a finite number of tuples of the form (Oid, t, x, y), stating that at a certain point in time, namely t, the object Oid was located at coordinates (x, y). The elements in the tuples are given by rational numbers and they are ordered. We do not address here the problem of moving regions, i.e., we consider regions as fixed over time. Let us consider the following setting: we have a database containing the position of six buses at each hour in our example city. Figure 1 shows the estimated trajectories followed by these buses (assuming linear interpolation between two consecutive samples); the figure also shows the neighborhoods the city is divided into, and their different incomes: the shaded regions are the ones with income less than fifteen hundred Euros monthly (in what follows, the "low income region"). Note that: object O_1 remains always within a low income region. Object O_2 starts its trajectory in a high income region, then enters a low-income neighborhood, and then gets out of it again. Objects O_3, O_4 and O_5 are always in high-income neighborhoods, while object O_6 passes through a low-income region, but was not sampled inside it. A typical query that can arise in the spatio-temporal/moving objects scenario described above is "Give me the number of buses per hour in the morning in the Antwerp neighborhoods with a monthly income of less than €1500,00."

Efficiently supporting these kinds of queries requires a solid formal model for spatio-temporal OLAP [17]. Hereto, in this paper we will give a framework which naturally integrates GIS, OLAP and moving objects concepts. Our model builds on the one introduced in [4] for spatial aggregation.

The remainder of the paper is organized as follows. In Section 2 we discuss related work in GIS and moving objects. Section 3 introduces the data model, while Section 4 shows an extensive set of example queries. Section 5 sketches a query evaluation strategy. We conclude in Section 6.

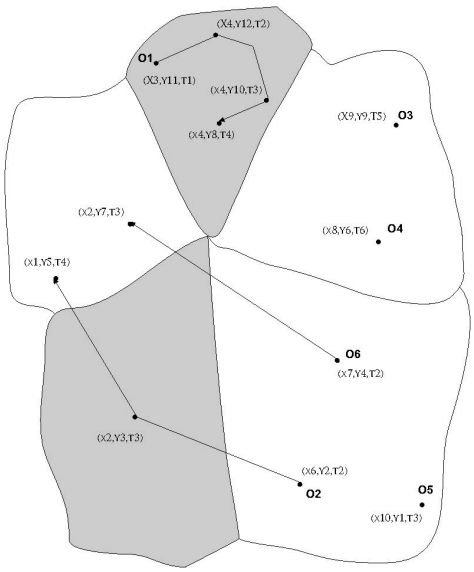

Figure 1. A moving objects example.

2 Background and Related Work

GIS and OLAP Interaction Although some authors have pointed out the benefits of combining GIS and OLAP, not much work has been done in this field. Vega López *et al.* [17] present a comprehensive survey on spatiotemporal aggregation that includes a section on spatial aggregation. Rivest *et al.* [15] introduced the concept of SOLAP (standing for Spatial OLAP), and described the desirable features and operators a SOLAP system should have. However, they did not present a formal model. Han *et al.* [5] use OLAP techniques for materializing selected spatial objects, and proposed a so-called *Spatial Data Cube*. This model only supports aggregation of such spatial objects. Pedersen and Tryfona [12] proposed the pre-aggregation of spatial facts. Other approaches [13, 21] combine OLAP and GIS for querying so-called spatial data warehouses using R-Trees for accessing data in fact tables. The data warehouse is then evaluated in the usual OLAP way.

Moving Objects Many efforts have been made in the field of moving objects databases, specially regarding data modeling an indexing. Güting and Schneider [3] provide a good reference to this large corps of work. Wolfson [19] *et al.* stated a set of capabilities that a moving object database must have, and introduced the DOMINO system (standing for Database fOr MovlNg Objects), that develops those

features on top of existing database management systems (DBMS) [18]. According to the authors, a DBMS supporting moving objects must address the following issues: location modeling, uncertainty, specific spatio-temporal query languages, indexing, and dynamic attributes. Güting *et al.* also proposed a system of abstract data types as extensions to DBMSs to support time-dependant geometries [2]. They defined a set of spatial data types as *point*, *points* (a set of *point* elements), *line* (a set of continuous curves) and *region*, where regions can have holes. They also defined spatial operations and predicates. Meng and Ding [9] proposed a moving objects database model denoted DSTTMOD (standing for Discrete Spatio-Temporal Trajectory Based Moving Objects Database model), where trajectories are used to represent dynamic attributes of moving objects, like past, current, and future location information. Trajectories of moving objects are represented by a set of line segments in the spatio-temporal space, assuming that within a segment, the object moves along a straight line, and the speed of the moving object keeps constant. Moving regions are addressed in [16]. Here, they are represented starting from snapshots of an amorphous region taken at different points in time. Interpolation of the snapshots of the geometries yields so-called *slices* of the moving regions representation. Hornsby and Egenhofer [6] introduced an interesting framework for modeling moving objects, which supports object visualization at different granularities, depending on the sampling time interval. The basic modeling element they consider is a *geospatial lifeline,* which is composed of triples of the form $(Id, location, time)$, where Id is the identifier of the object, $location$ is given by x-y coordinates, and $time$ is the timestamp of the observation (note the similarity with our representation sketched in 1.2). The possible positions of an object between two observation is estimated to be within two inverted half-cones that conform a *lifeline bead*, whose projection over the x-y plane is an ellipse. This model accounts also for the speed of the object and for the addition of intermediate observations.

Aggregate information is still a quite open field, either in GIS or in a moving objects scenario. Meratnia and de By [10] have tackled the topic of aggregation of trajectories. They identify similar trajectories and merge them in a single one, by dividing the area of study into homogeneous *spatial units*; each unit is associated to an integer, representing the number of times any object passes through it. Based on this, they obtain the aggregated trajectories. They claim that their method is insensitive to differences in sequence length and sampling intervals. With goals similar to ours, Papadias *et al.* [11] index historical aggregate information about moving objects. They aim at building a spatio-temporal data warehouse, and give ideas of possible indexing schemes. These schemes allow querying aggregate information about moving objects in a region during a certain time interval. They include pre-aggregate data in the nodes of the tree structures. Their approach could be used by different data models addressing OLAP-like aggregation. However, it is not clear how this proposal could be used for solving queries that involve more than one class of geometries, or involving trajectories (for instance, queries like "number of cars that in Monday morning travelled from Antwerp to Brussels").

3 A Framework for Geometric and Moving Objects Data Aggregation

Our conceptual model is based on the standard OLAP notion of dimension hierarchies and fact tables [8]. However, there are some particularities that deserve a special treatment. Our goal is to integrate in the same conceptual model, geographic, warehousing and moving objects information, in a natural way. These kinds of data may have been produced and stored completely separated from each other. We will model static and dynamic information in a different way. Static information will be represented as a GIS dimension schema [4]. Information about moving objects will be stored in another data structure, as we will see later.

Static Information A *GIS dimension*, consists, as usual, in a dimension schema and dimension instances. Each dimension is composed of a set of hierarchies, each one describing a set of geometries in a thematic layer. Figure 2 shows a GIS dimension schema (there is also the Time dimension, which we will comment later), with three hierarchies, located in three different layers, following our running example: rivers (L_r), schools (L_s), and neighborhoods (L_n) (other layers are represented analogously). We define three sectors, denoted the *Algebraic part*, the *Geometric part*, and the *Classical OLAP* or *Application part*. Typically, each layer will contain a set of binary relations between geometries of a single kind (the latter is not mandatory). For example, an instance of the relationship (*line*,*polyline*) will store the ids of the lines belonging to a polyline.

The finest level in the dimension schema (represented by a node with no incoming edges), called "point", belongs to the *Algebraic part* of the conceptual model. Here, data are represented as infinite sets of points (x, y, l), where l denotes a layer. We assume that the elements in the algebraic part are finitely described by means of linear algebraic equalities and inequalities. In the *Geometric part*, data consists of a finite number of elements of certain geometries. This part is used for solving the geometric part of a query, for instance to find all polygons that compose the shape of a country. Each point in the *Algebraic part* corresponds to one or more elements in the *Geometric part*.

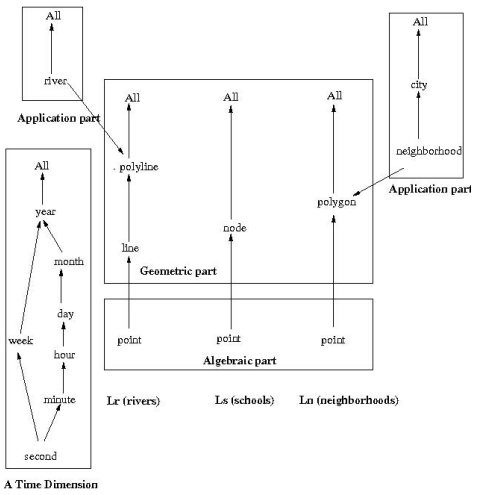

Figure 2. An example of a *GIS* dimension Schema.

Levels in the geometric part are associated with application-dependent concepts, for example, information about states, stored in a relational data warehouse, is associated to polygons, or information about rivers, with polylines. Typically, these concepts are represented as a set of dimension levels or categories, which are part of a hierarchy in the usual OLAP sense. These hierarchies conform the *Application part*.

The Time Dimension Besides the static information representing geometric components (i.e., the GIS), there will be a Time dimension (actually, there could be more than one Time dimensions, supporting, for example, different notions of time). Figure 2 shows a configuration of a Time dimension following the standard OLAP convention. Note that, of course, the application part could contain the time dimension, but, since it is essential for addressing moving objects, we believe that we must consider it as a special kind of dimension.

Example 1 In Figure 2, the level *polygon* in layer L_n is associated with two application-dependent categories, $neighborhood$ and $city$, such that $neighborhood \rightarrow city$ ("$A \rightarrow B$" means that there is a rollup function from level A to level B in the application part). Each category may even have attributes associated, like population, number of schools, and so on. Thus, a geometrically-represented component is associated with an application-dependant concept. There is also an application hierarchy associated to the layer L_r at the level of *polyline*. Notice that since dimension levels are associated to geometries, it is straightforward to associate facts stored in a data warehouse in the application part, in order to aggregate these facts along geometric dimensions, as we will see later. Finally, notice that in the algebraic part, the relationship represented by the edge $(point, polygon)$ associates infinite point sets with polygons. Note that a point may belong to more than one geometry. For instance, this occurs when a point belongs to two adjacent polygons. □

We will now define the data model in a formal way. Let us assume the following sets: a set of layer names $\mathbf{L}$, a set $\mathbf{A}$ of application-dependant attribute names and a set $\mathbf{G}$ of geometry names. Each element a of $\mathbf{A}$ has an associated set of values $dom(\mathrm{a})$. We assume that $\mathbf{G}$ contains at least the following elements (geometries): point, node, line, polyline, polygon and the distinguished element "All". More can be added. Each geometry g of $\mathbf{G}$ has an associated domain $dom(\mathrm{g})$. The domain of Point, $dom(\mathrm{Point})$, for example, is the set of all triples in $\mathbf{R}^2 \times \mathbf{L}$. The domain of All = $\{\mathrm{all}\}$. The domain of all elements G of $\mathbf{G}$, except Point and All, is $\mathrm{G_{id}}$, a set of geometry identifiers.

Additionally, we make the following assumption, which is usual in GIS: elements in a geometry intersect in objects whose extension is at least one dimension lower, i.e., polygons intersect in polylines or points and polylines in points.

Definition 1 (GIS Dimension Schema) Given a layer $L \in \mathbf{L}$, a graph $\mathrm{H}(L)$ is a graph defined as follows: (a) there is a node in $\mathrm{H}(L)$ for each kind of geometry $\mathrm{G} \in \mathbf{G}$ in L; (b) there is an edge from $\mathrm{G_i}$ to $\mathrm{G_j}$ if $\mathrm{G_j}$ is composed by geometries of type $\mathrm{G_i}$ (i.e., the granularity of $\mathrm{G_j}$ is coarser than that of $\mathrm{G_i}$), where $\mathrm{G_i}$ and $\mathrm{G_j} \in \mathbf{G}$; (c) there is a distinguished member *All* that has no outgoing edges; (d) there is exactly one node in $\mathrm{H}(L)$, labeled *point*, that has no incoming edges;

Also, there is a set $\mathcal{A}$ of partial functions Att with signature $\mathbf{A} \rightarrow \mathbf{G} \times \mathbf{L}$ such that, for each attribute $\mathrm{A} \in \mathbf{A}$, $Att(\mathrm{A}) = (\mathrm{G}, \mathrm{L})$ denoting that the attribute belongs to the geometry G in the layer L. The application part is composed by a set of dimension schemas $\mathcal{D}$ defined as in [7], where each dimension $\mathrm{D} \in \mathcal{D}$ is a tuple of the form $(dname, C, \preceq)$, such that $dname$ is the dimension's name, C is a set of dimension levels (in our model these levels are actually the attributes $\mathrm{A} \in \mathbf{A}$ mentioned above), and $\preceq$ is a partial order between levels.

Finally, a *GIS dimension schema* is tuple $\mathrm{G}_{sch} = (\mathcal{H}, \mathcal{A}, \mathcal{D})$ where $\mathcal{H}$ is the finite set $\{\mathrm{H}_1(L_1), \ldots \mathrm{H}_k(L_k)\}$. □

Example 2 In the dimension schema of Figure 2, the hierarchy for the layer containing rivers is: $\mathrm{H}_1(L_r) = (\{\mathrm{point}, \mathrm{line}, \mathrm{polyline}, \mathrm{All}\},$

$\{(\text{point}, \text{line}), (\text{line}, \text{polyline}), (\text{polyline}, \text{All})\})$; also, $At_{\text{G}}(\text{neighborhood}) = (\text{polygon}, L_s)$, $At_{\text{G}}(\text{river}) = (\text{polyline}, L_r)$ and neighborhood $\preceq$ city. Finally, in the application part we have dimensions *Rivers* and *Neighbourhoods* (we omit the schemas for the sake of brevity). Therefore, the GIS dimension schema is: $\text{G}_{sch} = (\{\text{H}_1(L_r), \text{H}_2(L_a), \text{H}_3(L_s)\}, \{At_{\text{G}}(\text{neighborhood}), At_{\text{G}}(\text{river})\}, \{Rivers, Neighbourhoods\})$. □

Definition 2 (GIS Dimension Instance) Let $\text{G}_{sch} = (\mathcal{H}, \mathcal{A}, \mathcal{D})$ be a GIS dimension schema. A *GIS dimension instance* is a tuple $(\text{G}_{sch}, \mathcal{R}, \mathcal{A}_{inst}, \mathcal{D}_{inst})$, where $\mathcal{R}$ is a set of relations $r_{L_i}^{\text{G}_j\text{G}_k}$ in $dom(\text{G}_j) \times dom(\text{G}_k)$, corresponding to each pair of levels such that (G_j, G_k) is in some graph $\text{H}_{G_i}(L_i)$ in G_{sch}. We denote each relation $r_{L_i}^{\text{G}_j\text{G}_k}$ in $\mathcal{R}$, a *rollup* relation.

Also, for each triple $(\text{G}, \text{A}, \text{L}) \in \mathbf{G} \times \mathbf{A} \times \mathbf{L}$ such that $Att(\text{A}) = (\text{G}, \text{L})$ there is a function $\alpha_L^{\text{A,G}} \in \mathcal{A}_{inst}$ with signature $dom(\text{A}) \to dom(\text{G}) \times dom(\text{L})$.

Finally, for each dimension schema $D \in \mathcal{D}$ there is a dimension instance defined as in [7], which is a tuple (D, RUP), where RUP is a set of rollup functions that relate elements in the different dimension levels (intuitively, these rollup functions indicate how the attribute values in the application part can be aggregated). □

Definition 3 (GIS Fact Table) Given a Geometry g in a graph $\text{H}(L)$ of a GIS dimension schema G_{sch} and a list of measures $\text{M} = (M_1, \ldots, M_k)$, a *GIS Fact Table schema* is a tuple $FT = (\text{G}, L, \text{M})$. A tuple $BFT = (\text{point}, L_b, \text{M})$ is denoted a *Base GIS Fact Table schema.* A *GIS Fact Table instance* is a function ft that maps values in $dom(G) \times \mathbf{L}$ to values in $dom(\text{M}_1) \times \cdots \times dom(\text{M}_\text{k})$. A *Base GIS Fact Table* instance maps values in $\mathbf{R}^2 \times \mathbf{L}$ to values in $dom(\text{M}_1) \times \cdots \times dom(\text{M}_\text{k})$. □

Example 3 A fact table containing neighborhood populations across time in our running example will be stored at the polygon level. In this case, the fact table schema would be $(polyId, L_{neighb}, Year, Population)$. If information is stored at the *point* level, for example, temperature data, we would have a base fact table with a schema $(point, L_{temp}, Temperature)$, with instances like $((x_1, y_1), L_{temp}, 25)$. □

Besides the GIS fact tables, there may also be classical fact tables in the application part, defined in terms of the OLAP dimension schemas. For instance, instead of storing the population as in Example 3 (i.e., associated to a polygon identifier), the same information may reside in a data warehouse, with schema $(neighborhood, Year, Population)$.

Definition 4 (Geometric Aggregation [4]) Given a GIS dimension as introduced in Definitions 1 and 2, a *Geometric Aggregation* is an expression of the form

$$\iint_C \delta_C(x, y)\, h(x, y)\, dx\, dy,$$

where $C = \{(x, y) \in \mathbf{R}^2 \mid \varphi(x, y)\}$, $\delta_C(x, y) = 1$ on the two-dimensional parts of C; $\delta_C(x, y)$ is a Dirac delta function on the zero-dimensional parts of C; and $\delta_C(x, y)$ is the product of a Dirac delta function with a combination of Heaviside step functions for the one-dimensional parts of C. Here, φ is a FO-formula in a multi-sorted logic $\mathcal{L}$ over $\mathbf{R}$, $\mathbf{L}$ and $\mathbf{A}$. The vocabulary of $\mathcal{L}$ contains the function names appearing in $\mathcal{F}$ and $\mathcal{A}$, together with the binary functions $+$ and $\times$ on real numbers, the binary predicate $<$. (Details on the use of the Dirac and Heaviside functions are given in [4]). □

Moving Objects We need to integrate the information about moving objects in the former framework. For this, we will consider a distinguished *Moving Object Fact Table* (MOFT), that contains tuples of the form (Oid, t, x, y), where Oid is the identifier of the moving object, t is a time instant, and (x, y) are the coordinates of the object Oid at instant t.

Finally, we will formalize the concept of *trajectory,* and state the difference between a trajectory and a *trajectory sample*.

Definition 5 A *trajectory* T is the graph of a mapping $I \subseteq \mathbf{R} \to \mathbf{R}^2 : t \mapsto \beta(t) = (\beta_x(t), \beta_y(t))$, i.e., $T = \{(t, \beta_x(t), \beta_y(t)) \in \mathbf{R} \times \mathbf{R}^2 \mid t \in I\}$. The *image of the trajectory* T is the image of the mapping β that describes T. The set I is called the *time domain* of T. □

Often, in the literature, conditions are imposed on the nature of the mappings β_x and β_y. For instance, they may be assumed to be continuous, piecewise linear [3], differentiable, or C^∞. For reasons of finite representability, we assume that I is a (possibly unbounded) interval and that β_x and β_y are continuous semi-algebraic functions (i.e., they are given by a combination of polynomial inequalities in x and t and y and t respectively). For example, the set $\{(t, \frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}) \mid 0 \le t \le 1\}$ describes a trajectory on a quarter of a circle. In this example, β_x is given by the formula $x(1 + t^2) = 1 - t^2 \wedge 0 \le t \le 1$.

Definition 6 A *trajectory sample* is a list of time-space points $\langle (t_0, x_0, y_0), (t_1, x_1, y_1), ..., (t_N, x_N, y_N) \rangle$, where $x_i, y_i, t_i \in \mathbf{R}$ for $i = 0, ..., N$ and $t_0 < t_1 < \cdots < t_N$. □

For the sake of finite representability we may assume that the time- space points (t_i, x_i, y_i), have rational coordinates. This will be the case in practice, since these points are typically the result of observations.

If $S = \langle (t_0, x_0, y_0), (t_1, x_1, y_1), ..., (t_N, x_N, y_N) \rangle$ is a sample of a trajectory and $T = \{(t, \beta_x(t), \beta_y(t)) \in \mathbf{R} \times \mathbf{R}^2 \mid t \in I\}$ is a trajectory that is consistent with S and if I is the interval $[t_0, t_N]$ and $x_0 = x_N$, $y_0 = y_N$, then T is called a *closed trajectory*.

A classical model to reconstruct a trajectory from a sample is the *linear-interpolation model* [3], where a unique trajectory is constructed such that it contains the sample and is obtained by assuming that the trajectory is run through at constant lowest speed between any two consecutive sample points. For a sample $S = \langle (t_0, x_0, y_0), (t_1, x_1, y_1), ..., (t_N, x_N, y_N) \rangle$, the trajectory $LIT(S) := \cup_{i=0}^{N-1} \{(t, \frac{(t_{i+1}-t)x_i + (t-t_i)x_{i+1}}{t_{i+1}-t_i}, \frac{(t_{i+1}-t)y_i + (t-t_i)y_{i+1}}{t_{i+1}-t_i}) \mid t_i \leq t \leq t_{i+1})\}$ is called the *linear-interpolation trajectory* of S.

3.1 Spatio-Temporal Aggregate Queries

We are interested in answering aggregate queries involving both, geometric aggregation, and aggregation of moving objects over a spatial GIS. Our goal is to still be able to address spatial queries as in the framework presented in [4], adding the ability to express queries aggregating moving object samples. Let us characterize the different situations that may arise.

1. **Spatial aggregation.** *The fact table is represented by a density function given in the geometric part of the GIS dimension schema.* In this case, we have a typical geometric aggregation. For instance, for queries like "Total population of provinces crossed by a river," where population is given as a density function.

2. **Spatial aggregation & numeric information.** *Numeric information is stored in the application part.* In this case, we also have a geometric aggregation. However, the numeric values can appear in the expression defining the query region C, instead of in the main body of the query. For instance, for queries like "total number of airports with more than one hundred arrivals per day".

3. **Trajectory samples.** *A fact table of moving objects (MOFT) exists, and a query asks for information not involving spatial data.* The fact table will be denoted in what follows $\mathcal{F}_{\mathcal{M}}$. In our running example, the query "Maximum number of buses per hour on Monday morning" only requires to operate over $\mathcal{F}_{\mathcal{M}}$ and the Time dimension.

4. **Trajectory samples & condition over the geometry.** *The query involves a computation over $\mathcal{F}_{\mathcal{M}}$ and spatial information.* For example, "Number of buses per hour in the morning in the neighborhoods with a monthly income of less than €1500,00." We will study this query in detail below. Here, C is a spatio-temporal structure that returns pairs (Oid, t) such that Oid was located at time t in a coordinate satisfying a spatial constraint.
After the region C is determined, any aggregation can be computed over this structure (for instance, the number of buses per hour).

5. **Trajectory samples & spatial aggregation in C.** *The query involves computation over $\mathcal{F}_{\mathcal{M}}$, and the region C includes aggregation.* Here, computation requires the aggregation of some function. For example, "Number of buses per hour in the morning in the neighborhoods where the number of people with a monthly income of less than €1500,00 is larger than 50,000." The query region C now requires a geometric aggregation to be computed, thus becoming a kind of "second order" aggregate query.

6. **Trajectory as a spatial object.** In this case, the trajectory can be treated as a static polyline in a spatial query. For instance, "How many cars are there in the Antwerp neighborhood called Berchem at 9:15 on Monday, Jan 7th, 2006?"

7. **Trajectory query.** Requires knowing the trajectory of the moving object. For example: "Average number of cars that pass through Berchem in the morning". In this case, although two consecutive samples can be taken outside Berchem, a linear interpolation may indicate that the object has passed through that neighborhood.

8. **Trajectory Aggregation.** Asks for an aggregation over a trajectory defined by a moving object.

Definition 7 (Aggregation [1]) Let AGG be the set of aggregate functions, with extension $AGG = \{MIN, MAX, COUNT, SUM, AVG\}$, and r a relation. The *aggregate operation* $\gamma_{f_{A(X)}}(r)$ is the relation $\gamma_{f_{A(X)}}(r) = \{t : t$ is an XA-tuple, $t[X] \in \pi_X(r)$, $t[A] = f_A(\sigma_{X=t[X]}(r))\}$, over XA, s.t. $XA \in schema(r), f \in AGG$, and $f_A(r)$ denotes the aggregation of the values in $t[A], t \in r$, using the function f. □

The semantics of a summable moving objects query $Q(C)$, where C is a relation of the form $C = \{(Oid, t, x, y)\}$ is $Q = \gamma_{AGG\,A(X)}(C)$ where X and A are subsets of the $schema(C)$.

We will now go back to the example study how aggregate queries can be answered in the proposed framework. Let us analyze the query introduced in Section 1.2: "Number of buses per hour in the morning in the Antwerp neighborhoods with a monthly income of less than €1500,00." We will be using the fact table $\mathcal{F}_{\mathcal{M}}{}^{bus}$ (Table 1), that contains the moving objects samples.

We will denote the geometries Pg for polygons, Pl for polylines, Pt for points, and so on. We will use *neighb* as a shorthand for the category neighborhood in the application

Oid	t	(x,y)
O_1	1	(x_1, y_1)
O_1	2	(x_2, y_2)
O_1	3	(x_3, y_3)
O_1	4	(x_4, y_4)
O_2	2	(x_5, y_5)
O_2	3	(x_6, y_6)
O_2	4	(x_7, y_7)
O_3	5	(x_8, y_8)
O_4	6	(x_9, y_9)
O_5	3	(x_1, y_4)
O_6	2	(x_5, y_7)
O_6	3	(x_4, y_9)

Table 1. The M.O. fact table $\mathcal{F}_{\mathcal{M}}{}^{bus}$.

part of the dimension schema. Also, for the sake of readability we will slightly change the notation for the function α : we will assume the layer to which the function refers is implicit by the function's name. For instance, the expression $\alpha^{neighb,Pg}_{L_n}(n) = (p_g, L_n)$ will be written just as $\alpha^{neighb,Pg}_{L_n}(n) = p_g$. (Recall that α is a function with signature $dom(\mathrm{A}) \rightarrow dom(\mathrm{G}) \times dom(\mathrm{L})$). Thus, the region with the required income is expressed as: $C = \{(x, y) \mid (\exists n)(\exists p_g) r^{Pt,Pg}_{L_n}(x, y, p_g) \wedge \alpha^{neighb,Pg}_{L_n}(n) = p_g \wedge n.income < 1,500\}$

The instants corresponding to the morning hours mentioned in the fact tables are obtained through the rollup functions in the Time dimension. We assume in the Time dimension a category denoted *timeOfDay*, rolling up to the dimension category *hour* (i.e., $timeOfDay \rightarrow hour$). We will denote R^j_i a rollup function between two categories i and j. The aggregation of the values in the fact table corresponding only to morning hours is computed with the following expression: $\mathcal{F}_{\mathcal{M}}{}^{bus}_{morning} = \{(Oid, t, x, y) \mid R^{timeOfDay}_{timeId}(t) = \text{“Morning”} \wedge \mathcal{F}_{\mathcal{M}}{}^{bus}(Oid, t, x, y)\}$

Finally, our spatio-temporal region C is expressed: $C = \{(Oid, t) \mid (\exists x)(\exists y)(\exists p_g)(\exists n) n \in neighb \wedge \mathcal{F}_{\mathcal{M}}{}^{bus}_{morning}(Oid, t, x, y) \wedge r^{Pt,Pg}_{L_n}(x, y, p_g) \wedge \alpha^{neighb,Pg}_{L_n}(n) = p_g \wedge n.income < 1,500\}$.

Remark 1 Note that the query result, given the instance of Figure 1 will be $4/3 = 1.333$. This is because O_1 will contribute three times, O_2 will contribute once, and the time span is three hours. □

We could summarize the complete expression in a single one as follows: $C = \{(Oid, t) \mid (\exists x)(\exists y)(\exists p_g), (\exists n) n \in neighb \wedge R^{timeOfDay}_{timeId}(t) = \text{“Morning”} \wedge \mathcal{F}_{\mathcal{M}}{}^{bus}(Oid, t, x, y) \wedge r^{Pt,Pg}_{L_n}(x, y, p_g) \wedge \alpha^{neighb,Pg}_{L_n}(n) = p_g \wedge n.income < 1,500\}$.

Thus, we can see that our *spatial* region C turns, in the spatio-temporal setting, into a set of pairs $(objectId, time)$, which are a key for an object's position in time and space.

4 Examples

In this section we present an extensive set of typical Moving Objects queries, and how they can be expressed in our framework. For each query we will also include its type, according to the characterization we proposed in Section 3.

1. “Give me the number of cars in region “South” of Antwerp on Wednesday morning.” (*Type 4*) We assume there is a layer with city regions. First, obtain the objects satisfying the condition of the query: $C = \{(Oid, t) \mid (\exists x)(\exists y)(\exists p_g)\mathcal{F}_{\mathcal{M}}(Oid, t, x, y) \wedge R^{timeOfDay}_{timeId}(t) = \text{“Morning”} \wedge R^{dayOfWeek}_{timeId}(t) = \text{“Wednesday”} \wedge r^{Pt,Pg}_{L_r}(x, y, p_g) \wedge \alpha^{region,Pg}_{L_r}(\text{“}South\text{”}) = p_g\}$.

We are assuming that cars are only in the regions where they were sampled. Thus, C returns a set of object identifiers. The aggregation (a sum) is performed over this set.

2. “Give me the maximal density of cars on all roads in Antwerp on Monday morning.” (*Type 4*)

This query may have different meanings, namely: (a) count all cars in each street, during the complete morning, and return the road with the highest number of cars per km; (b) take the density for each road in Antwerp at each moment (for the finest granularity) and return moment and street with the highest number of cars per km; (c) take, at each moment, the total number of cars in Antwerp divided by the total length of the road network, returning the moment of highest density. In all cases, the expression for the region C will return a set of triples of the form $(Oid, instant, street)$, capturing objects sampled in the same street more than once. The aggregation will be then applied over these tuples, according to the different interpretations. We have $C = \{(Oid, t, s) \mid (\exists x)(\exists y)(\exists p_l)\mathcal{F}_{\mathcal{M}}(Oid, t, x, y) \wedge R^{timeOfDay}_{timeId}(t) = \text{“Morning”} \wedge R^{dayOfWeek}_{timeId}(t) = \text{“Monday”} \wedge r^{Pt,Pg}_{L_s}(x, y, p_l) \wedge \alpha^{street,Pl}_{L_s}(s) = p_l\}$.

3. “Give me the total number of cars passing completely through cities with a population of more than 50,000 on Wednesday morning.” (*Type 4*)

Here, we have $C = \{(Oid, t) \mid (\exists x)(\exists y)(\exists c)(\exists p_g) R^{timeOfDay}_{timeId}(t) = \text{“Morning”} \wedge R^{dayOfWeek}_{timeId}(t) = \text{“Wednesday”} \wedge \mathcal{F}_{\mathcal{M}}(Oid, t, x, y) \wedge r^{Pt,Pg}_{L_c}(x, y, p_g) \wedge \alpha^{city,Pg}_{L_c}(c) = p_g, \wedge c.pop \geq 50,000 \wedge \neg((\exists x_1)(\exists y_1)(\exists p_{g_1})(\exists t_1)(\exists c_1) c_1.pop < 50,000 \wedge \mathcal{F}_M(Oid, t_1, x_1, y_1) \wedge r^{Pt,Pg}_{L_c}(x_1, y_1, p_{g_1}) \wedge \alpha^{city,Pg}_{L_c}(c_1) = p_{g_1})\}$.

A precise answer to this query requires knowing the trajectory of the object. If the object leaves the city at some instant, then it will not contribute to the result. Above,

we have made a simplification, considering only trajectory samples.

4. "How many cars are there in the Berchem neighborhood at 9:15 on Jan 7th, 2006?." (*Type 6*)

Here, the integration region C can be defined by the points denoting the objects' positions (i.e., in an "static" way): $C = \{(x,y) \mid (\exists Oid)(\exists p_g)\mathcal{F}_{\mathcal{M}}(Oid,t,x,y) \wedge \mathcal{F}(x,y,p_g) \wedge t = \text{"2006-01-07 9:15"} \wedge \alpha_{Ln}^{neighbPg}(\text{"Berchem"}) = p_g\}$.

An alternative version of this query would replace the (x,y) coordinates in the result, by the object Id. Since an object can be at most in one point in the plane at a given instant, both solutions will return the same number of tuples.

5. "Total amount of time spent continuously (i.e., without leaving the city) by cars in Antwerp on January 7th, 2006?." (*Type 7*)

Since a car may have been outside the city without being sampled, we need interpolation in order to give a more precise answer. The query would read: $C = \{(Oid,t) \mid (\exists p_g)(\forall t_1)(\forall x_1)(\forall y_1)(\forall t_2)(\forall x_2)(\forall y_2)(\forall x)(\forall y)$ $R_{timeId}^{day}(t) = \text{"2006-01-07"} \wedge \mathcal{F}_{\mathcal{M}}(Oid,t_1,x_1,y_1) \wedge \mathcal{F}_{\mathcal{M}}(Oid,x_2,y_2,t_2) \wedge t_1 < t_2 \wedge r_{L_c}^{Pt,Pg}(x,y,p_g) \wedge \alpha_{L_c}^{city,Pg}(\text{"Antwerp"}) = p_g \wedge x = \frac{(t_2-t)x_1+(t-t_1)x_2}{t_2-t_1} \wedge y = \frac{(t_2-t)y_1+(t-t_1)y_2}{t_2-t_1}\}$.

6. "Number of cars per hour within a radius of 100m from schools, in the morning." (*Type 7*)

If we do not perform any kind of interpolation, and just consider the points (trajectory sample), the expression for region C reads (treating the query as if it were of *Type 4*): $C = \{(Oid,t) \mid (\exists n)(\exists x)(\exists y)(\exists x_1)(\exists y_1)(\exists sc)\mathcal{F}_{\mathcal{M}}(Oid,t,x,y) \wedge R_{timeId}^{timeOfDay}(t) = \text{"Morning"} \wedge R_{timeId}^{typeOfDay}(t) = \text{"Weekday"} \wedge r_{L_s}^{Pt,Nd}(x,y,n) \wedge \alpha_{L_s}^{school,Nd}(sc) = n \wedge (x-x_1)^2+(y-y_1)^2 \leq 100\}$.

Here, L_s stands for a layer containing schools. The sum of the number of tuples returned, divided with the number of intervals, gives the answer.

Suppose now that an object that was not sampled within a radius of 100m from a school follows a trajectory that passes through that region. This object will not count in the result of the previous expression. Taking the trajectory into account yields the following expression: $C = \{(Oid,t) \mid (\exists n)(\exists t_1)(\exists x_1)(\exists y_1)(\exists t_2)(\exists x_2)(\exists y_2)(\exists x)(\exists y)$ $R_{timeId}^{timeOfDay}(t) = \text{"Morning"} \wedge R_{timeId}^{typeOfDay}(t) = \text{"Weekday"} \wedge \mathcal{F}_{\mathcal{M}}(Oid,t_1,x_1,y_1) \wedge \mathcal{F}_{\mathcal{M}}(Oid,x_2,y_2,t_2) \wedge t_1 < t_2 \wedge r_{L_s}^{Pt,Nd}(x,y,n) \wedge \alpha_{L_s}^{school,Nd}(sc) = n \wedge (x-x_1)^2+(y-y_1)^2 \leq 100^2 \wedge x = \frac{(t_2-t)x_1+(t-t_1)x_2}{t_2-t_1} \wedge y = \frac{(t_2-t)y_1+(t-t_1)y_2}{t_2-t_1}\}$.

7. "Total number of persons waiting for the tram at Groenplaats, by minute and between 8:00 AM and 10:00 AM on weekday mornings." (*Type 4*) Here, we have $C = \{(Oid,t) \mid (\exists bs)(\exists x)(\exists y)(\exists x_1)(\exists y_1)(\exists h)\mathcal{F}_{\mathcal{M}}(Oid,x,y,t) \wedge R_{timeId}^{timeOfDay}(t) = \text{"Morning"} \wedge R_{timeId}^{typeOfDay}(t) = \text{"Weekday"} \wedge R_{timeId}^{hour}(t) = h \wedge hrg \leq 10 \wedge hrg \geq 8 \wedge \alpha_{L_{bus}}^{stop,Nd}(\text{"Groenplats"}) = bs \wedge r_{L_s}^{Pt,Nd}(x_1,y_1,bs) \wedge (x-x_1)^2 + (y-y_1)^2 \leq 16\}$.

We assume that a person is waiting for the tram if she is less than four meters away from the tram stop.

5 Query Evaluation

In a previous paper [4], a spatial aggregate query of the form $Q = \int\int_{\mathbf{R}^2} \delta_C(x,y)\ h(x,y)\ dx\,dy$ was defined as being *Summable* if and only if the condition set C defines a finite set of elements of some geometry and Q can be rewritten as (we will not give the details here) $\sum_{g \in C} h'(g)$. It follows immediately that the queries presented in the previous sections are a variation of summable queries where the integration region is composed of a combination of moving object identifiers, time instants and geometric object identifiers (unless the expression for C contains itself a non-summable query). We also showed that many interesting queries in GIS require computing operations, like intersections or unions, between geometric objects represented in different layers, and proposed to precompute the overlay of such layers.

We implemented the proposal denoted Piet, and proposed a query language, Piet-QL. In short, a Piet-QL query is composed of two parts: the first one contains the spatial (geometric) query (see below), while the second one contains the OLAP query, expressed in an MDX dialect. A pipe acts as a separator of both parts.

In the outgoing implementation of the ideas presented in this paper, we take advantage of the existing model and implementation in order to allow efficient query evaluation of moving objects aggregate queries over complex geometric constraints. We will sketch the idea through an example. A complete description of the process is beyond the scope of this paper. Assume we have a layer with cities, a second one with rivers and a third one with schools. Consider the query "Total number of cars passing through cities crossed by a river, containing at least one store". The geometric part of the query is expressed in Piet-QL as:

```
SELECT layer.usa_rivers,layer.usa_cities,
layer.usa_stores;
FROM PietSchema;
WHERE intersection(layer.usa_rivers,
layer.usa_cities,subplevel.Linestring)
AND(layer.usa_cities)
CONTAINS(layer.usa_cities,
layer.usa_stores, subplevel.Point);
```

Out Piet implementation returns the identifiers of the geometric objects (in this case, the cities), that satisfy the

query. A moving object aggregation query can be added after the separator (i.e., in addition to the OLAP part we will have a "Moving Objects part"). The input to this query will be the object identifiers of the cities that satisfy the geometric query. Since we also have the geometric definition of the cities that correspond to these identifiers, it is easy to intersect these objects with the trajectories. This process will check, for each object, and for each consecutive pair of points in the moving objects fact table, if the intersection between the segment defined by these two points and a city in the answer to the geometric part of the Piet-QL query is not empty. If so, it counts for the aggregation. In the worst case, the whole trajectory must be checked.

6 Conclusion

We have proposed a formal model that integrates Moving Objects, GIS and OLAP in a unified framework. The model naturally extends GISOLAP, the model introduced in [4]. We characterized the aggregation queries that may may appear in this scenario, showed how they can be expressed in our framework and sketched a query evaluation procedure that takes advantage of the overlay precomputation strategy presented in the paper mentioned above. Our immediate future work will be devoted to implement this proposal over the existing GISOLAP implementation.

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 Knowledege Discovery and Delivery, by the Research Foundation Flanders (FWO-Vlaanderen), Research Project G.0344.05 and by Project PICT 21350–Foncyt–Agencia–Argentina.

References

[1] M. Consens and A. Mendelzon. Low complexity aggregation in Graphlog and Datalog. In *Proceedings of the 3rd International Conference on Database Theory, Lecture Notes in Computer Science n.470*, pages 379–394, 1990.

[2] R. H. Güting, M. H. Böhlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and quering moving objects. *ACM Trans. Database Syst.*, 25(1):1–42, 2000.

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

[4] S. Haesevoets, B. Kuijpers, and A. Vaisman. Spatial aggregation: Data model and implementation. In *Submitted for review (*`http://alpha.uhasselt.be/~lucp1265/pdfs/solap.pdf`*)*, 2006.

[5] J. Han, N. Stefanovic, and K. Koperski. Selective materialization: An efficient method for spatial data cube construction. In *Proceedings of PAKDD'98*, pages 144–158, 1998.

[6] K. Hornsby and M. J. Egenhofer. Modeling moving objects over multiple granularities. *Ann. Math. Artif. Intell.*, 36(1-2):177–194, 2002.

[7] C. Hurtado, A. Mendelzon, and A. Vaisman. Maintaining data cubes under dimension updates. In *Proceedings of IEEE/ICDE'99*, pages 346–355, 1999.

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

[9] X. Meng and Z. Ding. Dsttmod: A future trajectory based moving objects database. In *DEXA*, pages 444–453, 2003.

[10] N. Meratnia. Aggregation and comparison of trajectories. In *Proceedings of GIS'02*, Virginia, USA, 2002.

[11] D. Papadias, Y. Tao, J. Zhang, N. Mamoulis, Q. Shen, and J. Sun. Indexing and retrieval of historical aggregate information about moving objects. *IEEE Data Eng. Bull.*, 25(2):10–17, 2002.

[12] T. Pedersen and N. Tryfona. Pre-aggregation in spatial data warehouses. *Proceedings of SSTD'01*, pages 460–480, 2001.

[13] F. Rao, L. Zang, X. Yu, Y. Li, and Y. Chen. Spatial hierarchy and OLAP-favored search in spatial data warehouse. In *Proceedings of DOLAP'03*, pages 48–55, Louisiana, USA, 2003.

[14] P. Rigaux, M. Scholl, and A. Voisard. *Spatial Databases*. Morgan Kaufmann, 2002.

[15] S. Rivest, Y. Bédard, and P. Marchand. Modeling multidimensional spatio-temporal data warehouses in a context of evolving specifications. *Geomatica, 55 (4)*, 2001.

[16] E. Tøssebro and R. H. Güting. Creating representations for continuously moving regions from observations. In *SSTD*, pages 321–344, 2001.

[17] I. Vega López, R. Snodgrass, and B. Moon. Spatiotemporal aggregate computation: A survey. *IEEE Transactions on Knowledge and Data Engineering 17(2)*, 2005.

[18] O. Wolfson, P. Sistla, B. Xu, and S. Chamberlain. Domino: Databases fOr MovINg Objects tracking. In *Proceedings of SIGMOD'99*, pages 547 – 549, 1999.

[19] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving objects databases: Issues and solutions. In *SSDBM*, pages 111–122, 1998.

[20] M. F. Worboys. *GIS: A Computing Perspective*. Taylor&Francis, 1995.

[21] L. Zang, , Y. Li, F. Rao, , X. Yu, and Y. Chen. An approach to enabling spatial OLAP by aggregating on spatial hierarchy. In *Proceedings of DaWak'03*, pages 35–44, Prague, Czech Republic, 2003.