From Conceptual Modeling to Logical Representation of Trajectories in DBMS-OR and DW Systems

Bruno de C. Leal[1], Jose Antonio F. de Macedo[1], Valeria C. Times[2],
Marco Antonio Casanova[3], Vania Maria P. Vidal[1], Marcelo Tilio M. de Carvalho[3]

[1] UFC - Universidade Federal do Ceara, Brazil
{brunocleal, jose.macedo, vvidal}@lia.ufc.br
[2] UFPE - Universidade Federal de Pernambuco, Brazil
vct@cin.ufpe.br
[3] PUC-RIO - Pontificia Universidade Catolica do Rio de Janeiro, Brazil
casanova@inf.puc-rio.br, tilio@tecgraf.puc-rio.br

Abstract. With the growth of mobile devices equipped with geographical localization services, it has become economical and technically feasible to capture moving object trajectories in real life. Many interesting applications are being developed based on trajectory analysis. For example, in a traffic management system, traffic jams may be determined by mining movement patterns of groups of cars. In general, trajectory data can be analyzed into two different perspectives: real time and historical. In addition, trajectory applications share a common need, a more structured recording of movement. This allows managing trajectories as first class citizens and attaching whatever semantics the application requires, and developing robust and efficient methods to aggregate a set of trajectories that may support complex analysis. This article extends previous work on the conceptual modeling of trajectories by generalizing the idea of stops and moves and by defining a set of aggregation functions on trajectory data. In addition, this work proposes two modeling approaches, both based on design patterns, for devising trajectory data schemas for relational and multidimensional environments. A real-world case study about truck trajectories is used as a proof of concept. Experimental results showed that these modeling approaches offer the flexibility we are looking for to cope with the potential complexity of trajectory semantics in real time and historical analyses.

Categories and Subject Descriptors: H.2.7 [**Database Management**]: Database Administration

Keywords: Trajectory Data, Semantic Trajectory, Data warehousing

1. INTRODUCTION

With the growth of mobile devices equipped with geographical localization services, it has become economical and technically feasible to capture moving object traces (i.e. trajectories) in real life. Many interesting applications are being developed based on the analysis of moving object trajectories. For example, in a traffic management system, traffic jams may be determined by mining movement patterns of groups of cars. Similarly, in a zoology application, the analysis of a group of bird trajectories can help explaining their migration patterns. In a delivery system, discovering the trucks' movement patterns may be used as input to decision making processes, such as delivery scheduling. From a trajectory data management point of view, these applications require (1) a more structured recording of movement, which allows managing trajectories as first class citizens and attaching whatever semantics the application requires, and (2) methods to model and organize a set of trajectories multidimensionally. From a trajectory analysis point of view, these applications ask for (3) trajectory real time

This work was partly supported by CNPq, under grants 473110/2008-3, 557128/2009-9, 483552/2009-7 and 308247/2009-4, by FAPERJ under grant E-26/170028/2008, by FUNCAP under grant GPF 2151/22, CAPES under grant NF 21/2009 and CAPES student fellowship grant for Bruno Leal.

analysis, and (4) trajectory historical analysis, to support the decision-making process by computing aggregations and by visualizing them according to different perspectives and levels of details. These four requirements are key to facilitating the development of applications relying on trajectory data.

Substantial research has been conducted on providing methods and prototype systems for storing, querying and analyzing moving object trajectories in relational and multidimensional models (e.g. [Güting et al. 2000; Pelekis et al. 2006; Orlando et al. 2007; Orlando et al. 2008; Spaccapietra et al. 2008]). However, none of the proposed approaches provides an adequate solution for managing trajectory data as first class objects in transactional and multidimensional environments. Moreover, only the approach in [Spaccapietra et al. 2008] provides a semantic view of trajectory, which enables applications to associate whatever semantics they want with the trajectories. However, this approach is only applicable to transactional schema. Indeed, no work has been published using trajectories as a semantic objects on multidimensional data modeling.

To fill the gap between the transactional view and the multidimensional view on trajectories, a novel approach has to be developed to enable applications to perform traditional operating queries over semantic trajectories stored in transactional databases and compute aggregations on semantic trajectories stored in multidimensional data warehouses. Thus, a set of aggregation functions on semantic trajectory is required. In addition, a semantic view of trajectories is needed for both transactional and multidimensional schemas. This mandates that trajectories be treated as first class objects in both relational and multidimensional schemas, which is beyond the capability of current spatio-temporal prototypes. Therefore, our contributions are:

—An object-relational meta-schema for semantic trajectories;

—A star meta-schema for semantic trajectories;

—A set of semantic trajectory aggregation functions;

—Two ETL algorithms for populating multidimensional databases from object-relational databases of the operating environment of a corporation by considering trajectories as first class objects with semantic data associated with them.

The rest of this article is organized as follows. Section 2 surveys related work. Section 3 specifies a set of requirements for trajectory modeling and analysis, using an application scenario based on truck fleet management, and informally introduces the definitions used throughout this article. Section 4 describes two modeling approaches that consider trajectories as first class objects, based on the requirements previously listed, whose purposes are: (i) defining a trajectory data type; (ii) designing trajectory data schemas of both transactional and multidimensional environments; (iii) specifying algorithms for populating operational databases and data warehouses with trajectories treated as first class objects; and (iv) designing a set of aggregation functions for enabling real time and historical analysis over traditional and multidimensional DB, respectively. Section 5 details the experiments conducted in a real-world application for managing a delivery truck fleet. Finally, Section 6 concludes the article and highlights future work.

2. RELATED WORK

Spatio-temporal phenomena have been studied by several research communities in the context of several application areas. In the database community, the work of Guting was the first full implementation of a database for coping with moving objects. Based on Guting's and Wolfsonz's work [Wolfson et al. 1998; Güting et al. 2000], Pelekis [Pelekis et al. 2007] devised a framework for representing and querying moving objects over Oracle DBMS platform. The database community has mainly focused on management techniques for moving objects and has largely neglected supporting the concept of trajectories as first class objects.

In addition, data collection of trajectories of moving objects currently consists of the acquisition of their positions in time and space that correspond to sample points (i.e. raw data), represented by triples of the form (x, y, t), where x and y denote the coordinate data and t indicates the time in which such data were collected. Most DBMSs used today in transactional applications perform trajectory analysis by storing and processing queries on the raw data, making it difficult to process queries over trajectories, which should regarded as first class objects.

Several methods have been proposed to improve trajectory data analysis, including the use of spatiotemporal databases and data mining techniques. They can be classified according to the following groups: (i) generation of trajectory patterns (e.g. convergence, encounter, flock and leadership) according to the geometrical properties of trajectories [Laube et al. 2005]; (ii) extraction of clusters of sample points from trajectories, basically using time and space to determine trajectories located in dense regions, trajectories with similar shapes or distances, and trajectories that move between regions during the same time interval [Nanni and Pedreschi 2006; Kuijpers et al. 2006; Giannotti et al. 2007; Pelekis et al. 2007]; (iii) analysis of trajectories from a semantic point of view, trying to add context information [Alvares et al. 2007; Guc et al. 2008; Spaccapietra et al. 2008; Palma et al. 2008; Rocha et al. 2010; Moreno et al. 2010]; (iv) development of architectures, ETL (Extraction, Transformation and Loading) processes, data models and languages for helping in the construction of trajectory data warehouses (TDWs). This last group is the focus of the work described here and, therefore, the rest of this section surveys this group in more detail.

Data Warehouse (DW) technology has helped many organizations improve their decision-making processes and improve their performance. A DW is a multidimensional database that stores subject-oriented, integrated, time-variant and non-volatile data [Kimball and Ross 2002], and is often queried through an OLAP (On-line Analytical Processing) tool that is aimed at multidimensional processing of data extracted from DWs, allowing these data to be analyzed in different perspectives and levels of aggregation [Chaudhuri and Dayal 1997]. There has been a number of proposals in the literature aimed at integrating functionality and characteristics related to trajectory data and data warehouse systems [Orlando et al. 2007; Orlando et al. 2008; Pelekis et al. 2008; Marketos et al. 2008; Leonardi et al. 2009; Baltzer et al. 2008]. The main goal is to enable the storage of trajectory data in a data warehouse and to provide multidimensional processing with the capability of querying these trajectories according to several perspectives of analysis in order to support strategic decision-making processes related to moving objects. This kind of environment has been referred to as Trajectory Data Warehouses (TDWs).

In [Marketos et al. 2008], data originated from GPS sensors were loaded into a trajectory warehouse by using a set of ETL (Extract, Transform and Load) procedures. In addition, mechanisms for performing spatio-temporal aggregations analytical queries over trajectory data are detailed in [Leonardi et al. 2009]. Moreover, based on the trajectory conceptual model proposed in [Spaccapietra et al. 2008], a conceptual model schema for TDW that is aimed at supporting the monitoring of athletes' biochemical data is discussed in [Porto et al. 2010]. However, these biochemical data are not organized into dimensional data structures, through the use of fact and dimension tables and the representation of different levels of aggregation. In fact, traditional data warehouse techniques and current analytical tools do not satisfy the requirements of TDW applications because the representation and aggregation of trajectories imply the need for the modeling and aggregation of varied and interrelated data types, such as geometries, context information and spatiotemporal objects. Consequently, there is still no consensus about how trajectories are modeled multidimensionally, organized in different levels of aggregation and used for answering aggregation queries of OLAP systems.

Even though the advantages of spatiotemporal data warehouses have been known for some time, as far as we know, no commercial business intelligence solutions employ TDWs today. In [Orlando et al. 2007], the first proposal of a DW for trajectories defines an OLAP data cube with both spatial and temporal dimensions, and a set of spatiotemporal hierarchies determined through the use of regular

grids. Both numerical and spatial measures are proposed for this TDW. Some other studies have been conducted to help the development and widespread use of TDWs. However, none of them deals with the semantic aspect of trajectories that are rarely treated as first class objects and, thus, the resulting multidimensional analysis lacks knowledge about the discovered multidimensional patterns.

In this article, two modeling approaches, both based on design patterns, are proposed for designing trajectory data schemas of transactional and multidimensional environments, in which trajectories are considered as first class objects. This enables the creation of reports and other outputs that tend to be much richer than those whose spatiotemporal data are handled as a sample of points with no associated semantics. Also, a case study based on real truck trajectories was developed to illustrate how both real time and historical trajectories are modeled and queried in a way that facilitates analysis and reporting.

3. PRELIMINARIES

Section 3.1 presents an application scenario using trajectory data and discusses specific requirements for trajectory modeling and analysis, which we claim to be necessary for facilitating the development of those applications. Section 3.2 informally introduces basic definitions related to trajectories that will be used throughout this article.

3.1 Application Scenario

Logistic management is an important class of applications that may take advantage of analyzing trajectory data. Particularly, applications that manage truck delivery of goods are of special interest due its importance for the global economy. For this kind of application, truck travels may be used to check whether delivery plans are being correctly executed (real time analysis) or they may analyzed to assess the performance of the delivery system (historical analysis). Indeed, analyzing truck travels is the key to improve knowledge about many questions on logistic management, such as: Are the delivery plans being respected? Taking into account the current state of truck deliveries, when a customer will be visited? Is it possible to reduce the number of truck travels without compromising revenue? Why a delivery was not performed? The answers to such questions are key to support decision-making processes in delivery planning and routing.

In order to analyze the truck travels, hereinafter called simply truck trajectories, this application must record two types of data:

(1) Data about truck trajectories: Each truck is equipped with a controller device, which automatically collects data about the truck location and allows the truck driver to inform certain events that occurred during the travel. Data about truck position is sent at regular intervals to a central server, which stores this data in a conventional database. For example, when the truck stops, the following data is recorded: (a) the kind of stop: delivery or otherwise, (b) when the stop is not a delivery, the activities executed by the truck driver during the stop must be informed, such as: lunching, resting, refueling, emergency, etc. When the truck is traveling, the controller device informs the speed, direction and position of the truck. In addition, data about the streets travelled by is recorded.

(2) Data related to truck trajectories: regions of interest (e.g. district, regions, state, etc) associated with each trajectory; the list of customers visited in each trajectory; traffic conditions associate with the trajectory (to explain delivery delays).

Examples of typical queries executed in this application scenario are:

Q1. What is the location of the truck 1103 and which were the customers it visited?

Q2. What is the number of trajectories and the corresponding average duration, carried out in the City of Sao Paulo in April?

Q3. When customer A will be visited?

3.2 Basic Definitions

Based on the application scenario of Section 3.1, we introduce the definitions on which the rest of the article depends on.

From a conceptual point of view, a trajectory must be perceived by the application or the database designer as an identifiable and manageable object. Even if the trajectory is defined as a first-class object, applications sometimes require relating information to segments of a trajectory, for example, to indicate that, in a time interval along the trajectory, the truck stopped to deliver merchandise to a customer. Thus, a conceptual view of trajectories must provide mechanisms to deal with the trajectory as a first-class object, as well as to decompose the trajectory into meaningful events.

In more detail, a geographical positioning device collects, at regular time intervals, the geographic location of a moving object (e.g. *latitude*, *longitude*), associated with a timestamp. Over time, the device accumulates a set of triples (e.g. *lat*, *long*, *instant*), which represent the movement of the object. However, the sequence of triples collected may in fact represent different trajectories of the moving object. For example, suppose a truck made several trips per day. At the end of the day, the collection of triples captured by the device should be separated into different trajectories, each one representing a different trip. By contrast, suppose that the same set of triples should be used by a truck maintenance application. In this second scenario, the trajectory of the truck should be defined as the sequence of all triples collected throughout the day. Thus, in general, the definition of what is a trajectory depends on the application, that is, the definition of the start and the end points of a trajectory depend on how the application perceives the movement of the object.

Therefore, the designer should provide a function to segment the raw data collected by the positioning device and to generate trajectories, which will then be stored in a database and manipulated by the application. In the literature, the process of segmenting raw trajectory data is known as *semantic construction* of the trajectories (hence the term *semantic trajectories* we adopt). This may also include other tasks, such as the correction of points collected by the positioning device.

To capture such concepts, we introduce a sequence of definitions. A *spatio-temporal point* is a triple $p = (l, g, t)$, where l and g are real numbers that represent latitude and longitude, respectively, and t is a positive real number representing a timestamp.

A *raw spatio-temporal dataset* is a finite sequence of spatio-temporal points $R = ((l_1, g_1, t_1), ..., (l_k, g_k, t_k))$ such that, for all $i \in [1, k)$, $t_i < t_{i+1}$. Note a raw spatio-temporal dataset has no semantics.

Let A be an *event attribute*, taking values from the domain D. An *event* over A is a triple $e = (b, d, a)$, where $b = (l_b, g_b, t_b)$ and $d = (m_d, h_d, u_d)$ are points such that $t_b \leq u_d$, representing the *begin* and *end points* of e, and a is an attribute value from D, representing the *event information*. A *semantic trajectory* over an event attribute A is a quadruple $s = (b, d, R, L)$, where

—$b = (l_b, g_b, t_b)$ and $d = (m_d, h_d, u_d)$ are points such that $t_b \leq u_d$, representing the *begin* and *end points* of s;
—$R = ((l_1, g_1, t_1), ..., (l_k, g_k, t_k))$ is a raw spatio-temporal dataset such that b and d, the begin and end points of s, are the first and the last points of R;
—$L = (e_1, ..., e_n)$ is a finite sequence of events over A such that:
  —for $i = [1, n]$, the begin and end points of e_i are points in R;
  —for $i = [1, n)$, if $d_i = (m_i, h_i, u_i)$ is the end point of e_i and $b_{i+1} = (l_{i+1}, g_{i+1}, t_{i+1})$ is the begin point of e_{i+1}, then $u_i \leq t_{i+1}$.

Note that b and d, the begin and end points of s, are just a definitional convenience, since they can be trivially deduced from R.

To summarize, a semantic trajectory is an application-defined representation of the evolution of the position of a moving object (perceived as a point) during a given time interval. The application may extract (or define) semantic trajectories from a raw spatio-temporal dataset using a function which incorporates the semantics of what a trajectory for the domain should be. The definition of a semantic trajectory includes a sequence of events, which can be referenced and managed individually. The application may in fact treat events as first-class objects that carry additional semantics.

4. MODELING TRAJECTORY DATA

We now investigate solutions to support the application requirements we identified before. We propose two solutions that are driven by different modeling objectives. In one scenario, we imagine designers using trajectories as encapsulated objects without worrying about their internal details. For this scenario, we propose a solution which aims at encapsulating trajectories into a dedicated *TrajectoryType* object type, equipped with methods providing access to trajectory components (events). In a second scenario, we suppose that designers want to associate thematic information with parts of the trajectory object, for instance, to associate a list of the closest points of interest with a trajectory event. We describe a solution for this scenario that focuses on the explicit representation of a trajectory and its components, which makes it possible to link application objects to any of these components.

4.1 Transactional Environment

Extending data models with new data types is the key to provide adequate representation and manipulation of real world complex objects. In the database domain, this technique has been frequently applied in order to reduce the complexity of design and application development. For example, with the creation of new spatial data types, various DBMS were equipped with such types, which greatly facilitated the development of applications dealing with spatial data. The creation of a new data type not only provides comfort to the designer but also enables data management optimizations. Thus, a plausible solution is to define a trajectory data type containing attributes and methods that are useful for instantiating object types and creating attributes of this new type.

Figure 1 shows the proposed trajectory data type. Note that this data type encapsulates two lists: *samplePoints* and *eventList*. The *samplePoints* list represents the sample points collected from the positioning devices and the *eventList* list refers to the semantic segmentation of the trajectory. In fact, according to the semantic trajectory definition (refer to Section 3) the idea of capturing the geometric and semantic facet into a single data type is represented respectively by these two lists. An event has application-dependent information. In this case, the designer should provide a function that maps geometric data to event information. Each event is defined by two spatio-temporal points corresponding to the event begin and end, and a textual information about the event.

The proposed trajectory data type encapsulates a set of useful methods for retrieving geometric data (e.g. sample points), computing measures on trajectory data (e.g., length, duration, speed, etc), and accessing events. Due to space limitations, we do not show all the defined data type methods. Indeed, this data type can be extended with other attributes and methods to meet the requirements considered crucial to the application.

The trajectory data type approach is adequate for applications that only require geometric data and a semantic event view of the trajectories. Whatever semantics the application needs should be explicitly defined by the database designer in the form of attributes of an object type (e.g. moving object), which must contain one attribute of type *TrajectoryType*. Another way to attach semantics is by linking *TrajectoryType* objects to other application objects.

Since much of the information on trajectories is application dependent, it is not adequate to completely solve this problem by adopting only the data type approach. An adequate solution is to expose

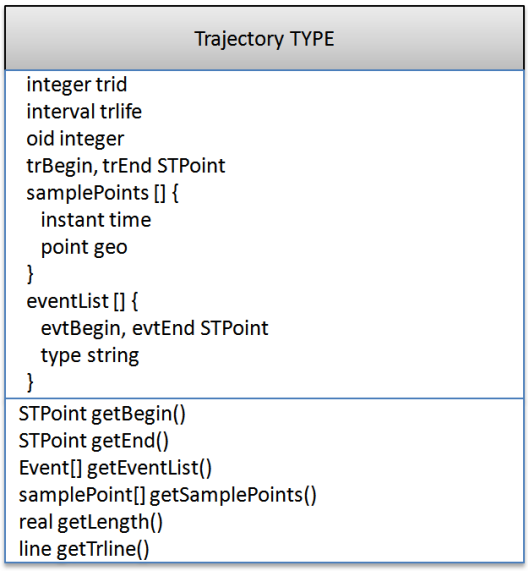


Fig. 1. Trajectory Data Type

TrajectoryType components in such a way that additional application semantics may be linked to these components. This exposure is achieved by decomposing *TrajectoryType* attributes into distinct object types. In addition, we add a new moving object type since a trajectory is always the result of a moving object.

Design patterns are well known artifacts in the software engineering literature. They aim at facilitating and organizing the development process, providing pre-defined solutions to known problems in software design. Unlike the area of software engineering, where design patterns focus on models for software, this article proposes the use of design standards for the construction of database schemas related to moving objects trajectories. Basically, the idea is to provide an off-the-shelf data schema using semantic trajectory concept and points of extension (called hooks in the literature). Extension points can be used by the designer to attach semantic information related to trajectory data.

To meet application requirements related to real time or historical analysis, we propose two design patterns. We assume that transactional schemes are suitable for building applications that perform analysis in real time, and multidimensional schemes are ideal to carry out historical analysis. Therefore, we specify two design patterns, one for instantiating data schemas for transactional environment and another one for creating database schemas in a multidimensional environment.

Figure 2 depicted the proposed design pattern for a transactional schema. This trajectory design pattern holds object types for representing trajectories, their events (a subcomponent of a trajectory) and moving objects. In fact, we assume that moving objects, trajectories and events are the common information needed by database designers of trajectory management applications. In addition to structure trajectory data, this design pattern includes hooks used to connect to application objects. In Figure 2, the hooks are represented as a box with dashed border. Trajectories are linked to a *MovingObjectType* object type that represents the traveling objects covering the trajectories. Hence, a designer can instantiate a data schema from this design pattern by defining new attributes for the object types and substituting hook objects by application object types.

Algorithm 1 populates a transactional schema that instantiates the design pattern presented in Figure 1. It takes the raw spatio-temporal points generated by a positioning device (line 1), and, for each moving object, it creates an instance of MovingObjectType (line 2). Next, it calls a function, provided by the designer, to construct trajectories (line 3). For each trajectory this function constructs,

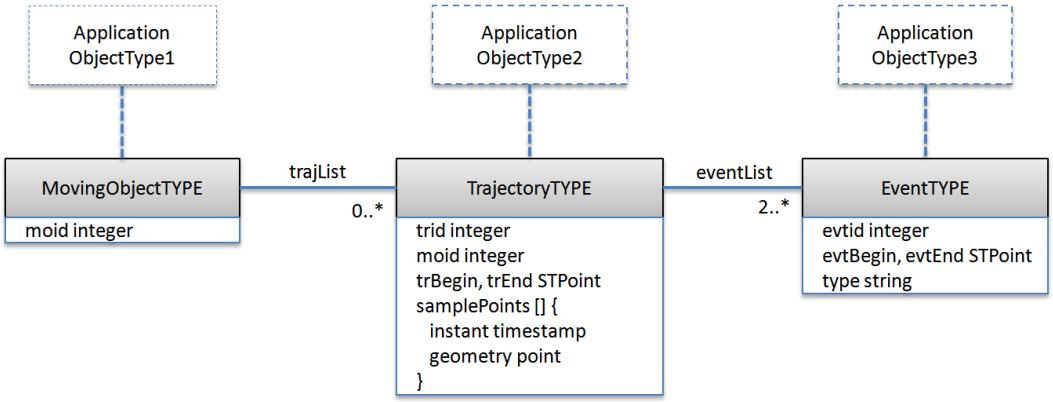


Fig. 2. Trajectory Design Pattern for Transactional Environment

an instance of TrajectoryType is created (line 5). Finally, it uses another function provided by the designer to segment a trajectory into a list of events (line 6).

Algorithm 1 Populate Transactional Schema

1: **for** each distinct *moid* in the Raw Spatio-Temporal Dataset **do**
2: creates an instance *mo* of *MovingObjectType*
3: *mo.trajList* = trajectorySegmentation(lp)
4: **for** each trajectory *ti* ∈ *trajList* **do**
5: creates an instance *tr* of *TrajectoryType*
6: *tr*.eventList = eventSegmentation(ti)
7: **end for**
8: **end for**

4.2 Multidimensional Environment

Multidimensional modeling refers to a set of techniques and concepts used in data warehouse design. The multidimensional model is composed by two main conceptual constructors, called *fact* and *dimension*. Facts are values (in general, numeric) that can be aggregated, and dimensions are groups of hierarchies and descriptors that contextualize the facts.

Figure 3 shows the design pattern for a multidimensional modeling. We use UML notation for this design pattern, and sterotypes are used to denote fact and dimension classes. Dimensions are represented with traced lines, indicating that they are extension points (hooks). Dimensions can be extended with new hierarchies and properties. Facts accept extensions on measures but existing measures must be preserved in order to follow trajectory data type proposal.

In this design pattern, we use *TrajectoryType* and *EventType* as measures of multidimensional models, which define that aggregations will be made over trajectory and event facts. Indeed, these facts represent the geometric and semantic view on trajectories that is coherent with the idea of exposing the trajectory components. One example of geometric aggregation is the union of trajectories that exceed a certain speed threshold and that intersect a given *ad hoc* query window. An example of an aggregation function for events is the average duration of an event of type "Lunch".

Since a trajectory is inherently a spatio-temporal object resulting from traveling moving object, we propose three generic dimensions that are shared by trajectory and event facts: Time, Region (Space), and Moving Object. In addition, the *TrajectoryCategory* and *EventCategory* dimensions are defined to characterize aggregations over trajectory and event facts, respectively. It is worth noting that

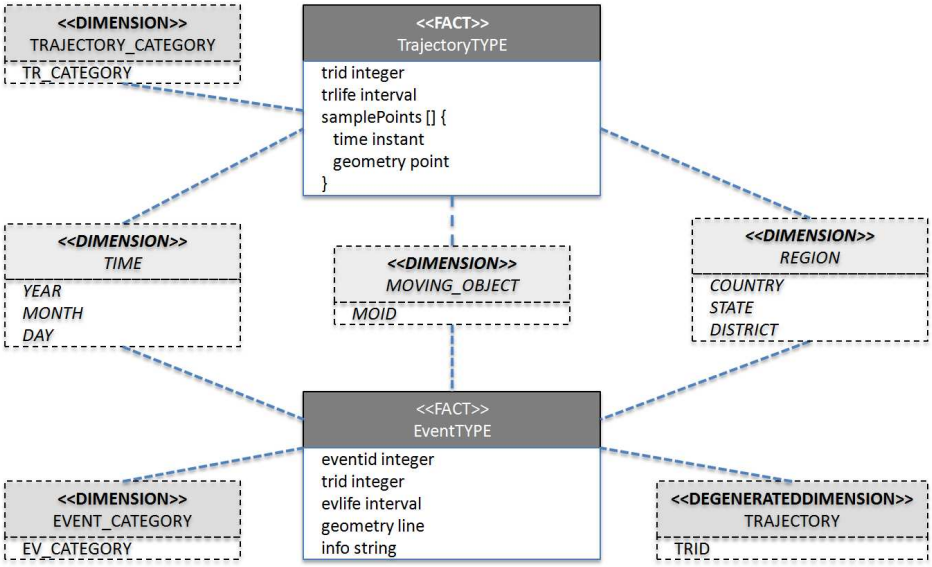


Fig. 3. Trajectory Design Pattern for Multidimensional Environment

EventType facts contain an attribute referring to the trajectory identification, (*trid*), which is used to specify the degenerated Trajectory dimension. We assume that, for some trajectory applications, it is important to apply aggregations to trajectory ids, for instance, to compute the number of events of type "delivery gas" in trajectory id "13052".

Multidimensional modeling should include a process to extract and transform data from the source OLTP systems to feed a data warehouse, called *ETL process*. In our scenario, the first step of the ETL process is to populate all levels from the hierarchy dimensions with values from the *TrajectoryType* and *EventType* attributes stored in the transactional schema. For example, the time dimension must be populated with values ranging over trajectories lifespan, and the *EventCategory* dimension must cover all event categories of interest. For the region dimension, a topological spatial function (e.g. inside, crosses and near by) must be provided by the application designer to establish the spatial relationships between *TrajectoryType* facts and the *Region* dimension. We observe that a region dimension may not have a well-defined hierarchy (e.g. a predefined spatial hierarchy with a 1:N relationship among higher and lower attribute levels), as a trajectory geometry may not be associated with all levels of the region hierarchy. For example, a trajectory may be inside the State of Sao Paulo, but not inside any of its cities. A solution to this problem is to set *null* to all region levels that do not match.

After populating the dimensions, we are able to extract the facts. Algoritm 2 populates the TrajectoryType facts (line 1) and create links to the dimensions (lines 3-12). Lines 5 to 12 create links to the asymmetrical hierarchy [Malinowski and Zimányi 2004] of the Region dimension.

The process for populating the *EventType* facts is very similar. We must apply the same steps, except that we must replace *TrajectoryType* by *EventType* and add one more instruction to link events to values of the *EventCategory* dimension, which must be supplied by a function provided by the designer (similar to the category function for trajectories, shown in line 12 of Algorithm 2).

4.3 Aggregation Functions for Trajectories

From an analytical point of view, it is fundamental to provide summarized information from a large collection of trajectory data. Aggregation functions have been widely used in database and decision-support applications with this purpose. However, the available aggregation functions for transactional

Algorithm 2 Populate Multidimensional Schema

```
1: for each trajectory ti of TrajectoryType table in transactional schema do
2:    copy ti to the trajectory fact in the multidimensional schema
3:    create two links to the lowest level of Time dimension using ti.trbegin and ti.trtEnd properties
4:    create one link to the lowest level of Moving Object dimension using ti.oid;
5:    for each hierarchical level rl of Region dimension ordered in ascending order do
6:       if SpatialTopologicalFunction(ti.geometry,rl.geometry)=true then
7:          create a link to the level rl of Region dimension
8:          break
9:       else
10:         set null to link to the level rl of Region dimension
11:      end if
12:      create one link to the lowest level of Trajectory Category dimension using ti.category() function;
13:   end for
14: end for
```

and multidimensional environments are not appropriate to aggregate complex objects such as trajectories. Therefore, we propose to enhance the analytical processing capabilities of trajectory applications by introducing a new family of analytic functions for trajectory data.

The proposed family of aggregation functions for trajectories reduces a set of trajectories to a single trajectory, to a line, to a time interval, or to a scalar value. However, we do not propose specific aggregation functions, since we believe that it is the responsibility of the designer to identify which functions are of interest to her/his application and to define their semantics. Instead, we introduce a generic signature that helps specifying a large number of trajectory aggregation functions, defined as follows:

$$OP : T \text{ x } F \rightarrow D \text{ where}$$

—OP represents the following aggregation operations: SUM, COUNT, AVERAGE, MIN, and MAX;

—T is a trajectory data set;

—F defines which trajectory feature type is used by the aggregation operation OP. A trajectory feature is any information that can be computed or obtained from a trajectory (for instance, geometry, length, lifespan, set of sample points/events, etc). The possible trajectory feature types are: TRAJECTORY, SPATIAL, TEMPORAL, SET, and NUMERIC.

—D is the output data type

For example, Table I illustrates some functions that have this generic signature. In the examples, the variable t is of type *TrajectoryType* and represents the set of trajectories being aggregated. Trajectory feature values are computed by trajectory functions that must be implemented in the *TrajectoryType* data type. For example, in the second line of Table I, the function $t.events("LUNCH")$ returns a set of events with description "LUNCH".

5. EXPERIMENTAL RESULTS

In this section, we present experiments carried out in the real-world scenario described in Section 3.1. We first instantiated a transactional and multidimensional trajectory schema in a relational database. Next, we implemented algorithms for constructing and populating those schemas. In the sequel, we developed trajectory aggregate functions for both environments. We have also used an open source BI OLAP tool for building data cubes from the trajectory multidimensional schema that was instantiated.

Table I. Examples of Trajectory Aggregation Functions

SIGNATURE	Example	Meaning
COUNT (T, T.ATTR) : NUMERIC	tr_aggr_count (t, t.events("LUNCH"))	returns the number of events of type "LUNCH" in a set of trajectories
AVERAGE (T, T.ATTR) : TEMPORAL	tr_aggr_avg_temporal (t, t.lifespan())	returns the average interval of a set of trajectories
MIN (T, T.ATTR) : TRAJECTORY	tr_aggr_min_spatial (t, t.geometry())	returns the shortest trajectory in space from a set of trajectories
MAX (T, T.ATTR) : SPATIAL	tr_aggr_max_spatial (t, t.geometry())	returns the geometry line from the longest trajectory in space of a set of trajectories

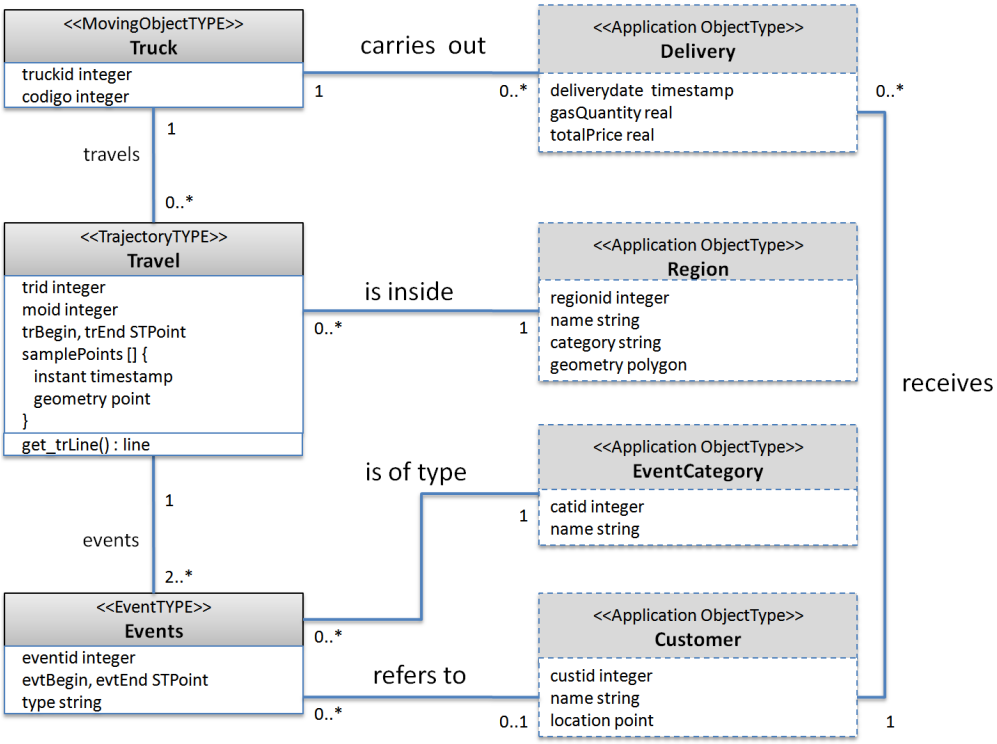


Fig. 4. DBMS Object-Relational schema for the Truck Fleet Control Application

Finally, we performed some queries and analysis in both environments, using the aggregation functions to illustrate the importance of their application.

5.1 Implementation of the trajectory schema in Object-Relational DBMS

Due to the complex nature of trajectory data, we decided to use a DBMS equipped with object-relational features (Oracle). The use of this DBMS helped us define and implement trajectory data types and their corresponding functions. Figure 4 shows the logical schema implemented in Oracle DBMS based on the transactional design pattern described in Section 4.1. Note that, with this design pattern, it is possible to link application thematic information with trajectory data and its events. For example, the Region table is linked to the Travel table, which stores truck trajectories. This relationship is instantiated when a truck trajectory is inside a region. Similarly, application thematic information is associated with events. For example, the Customer table is linked to Events denoting that some events may be related to a customer (e.g. delivery event).

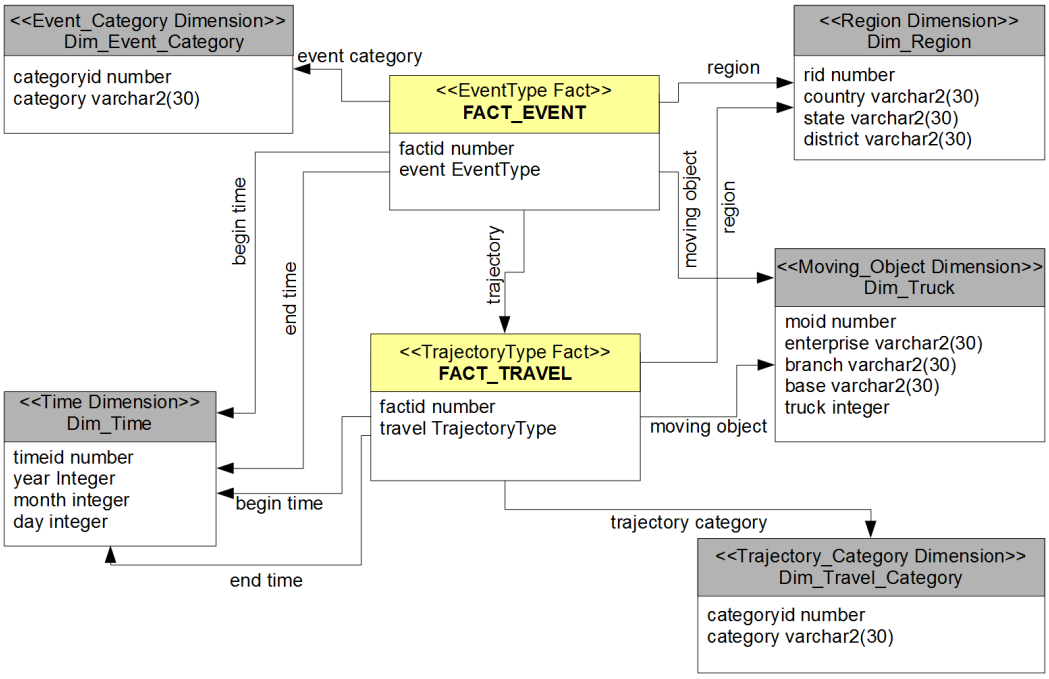


Fig. 5. DW Object-Relational schema for the Truck Fleet Control Application

5.2 Trajectory DW implementation

The multidimensional schema was also implemented in a relational database. However, the proposed multidimensional design pattern is not oriented to any specific data model. Although this design pattern is represented in the object-oriented model, its implementation can be done in whatever target data model. Indeed, the choice of a target model is a decision that concerns the application designer. In our experiments, we decided to use the object-relational model since it facilitates the mapping between the design pattern model and the implementation model.

Figure 5 depicted the logical multidimensional schema generated from the instantiation of our multidimensional design pattern for trajectories. This logical model contains application-defined hierarchies for the *Moving Object* dimension (Dim_Truck) and two links for the *Time* (Dim_Time) dimension, corresponding to the begin and end time for the travel and event facts. The link named *trajectory* between *Event* and *Travel* facts represents the trajectory degenerated dimension, illustrated in Figure 3.

5.3 Aggregation function implementations

We used the generic signature for trajectory aggregation functions to implement some functions regarded as crucial to the application scenario. Since these functions have a parameter that is a complex object (i.e. *TrajectoryType*), we used the *ODCIAggregate* framework built in Oracle to implement them. The creation of user-defined aggregation functions requires the implementation of four routines that are mandatory by the framework interface. Some of the functions that have been created in our experiments are:

—TR_AVG_DURATION - computes the average trajectory duration
—TR_AVG_SPEED - computes the average trajectory speed
—TR_MAX_DURATION (and MIN) - computes the maximum and minimum trajectory duration

```
select tru.codigo, tr_count(t), tr_avg_duration(t), tr_avg_evt_delivery(t)
from trucks tru, table(tru.travels) t, areas a
where a.name = 'OSASCO' and a.category = 'CITY' and
  sdo_inside(t.get_trline(), a.geometry) = 'TRUE' and
  t.trbegin.instant >= to_timestamp('01-01-2010') and
  t.trend.instant <= to_timestamp('30-06-2010')
group by tru.codigo;
```

Fig. 6. Query returning the amount of trips, the average duration and the number of events for each truck within a given time period and spatial region

```
(a)
select  t.get_trline()
from  trucks tru, table(tru.travels) t
where tru.codigo = 11 and
  (t.trbegin.instant, 'DD-MM-YYYY')=(current_date, 'DD-MM-YYYY');

(b)
select  e.evtbegin.geometry
from  trucks tru, table(tru.travels) t, table(t.events) e
where tru.codigo = 11 and
  (t.trbegin.instant, 'DD-MM-YYYY')=(current_date, 'DD-MM-YYYY')
  and e.type = 'Atendimento';
```

Fig. 7. Real time queries for retrieving (a) geometry of a truck trajectory and (b) beginning location for the events of type "Atendimento"

—TR_MAX_LENGTH (and MIN) - computes the maximum and minimum trajectory length
—TR_AVG_QTD_DURATION (LENGTH) - computes the average volume of product delivered per duration (per LENGTH)
—TR_AVG_EVT_DELIVERY - computes the average number of deliveries per trajectory

5.4 Analysis in transactional environment

Transactional environment is used for real-time analysis. In this application scenario, typical queries are to list where trucks are, or which deliveries they have done along the day. For example, we present below two such queries that have been submitted to the transactional schema:

—for each truck, list the amount of travels done in Osasco city, its average trajectory duration and the average number of deliveries (i.e. attendance) (see Figure 6);
—for truck 11, list its route (Figure 7 *a*) and the location of its deliveries that had already been accomplished in the current day (Figure 7 *b*). Figure 8 displays the results using Google Maps.

5.5 Analysis in multidimensional environment

Analysis in a multidimensional environment is useful for discovering patterns of behavior. For example, an interesting analysis is to identify the trucks exceeding the speed limit over a hundred times within "Sudeste" region. In addition, do this verification for trucks that belong to the "Company X" during the first semester (Figura 9). We also present an example of trajectory cube created with Pentaho BI Server[1] to analyse the average speed and average length of moving object trajectories that travelled

[1] http://community.pentaho.com/projects/bi_platform/

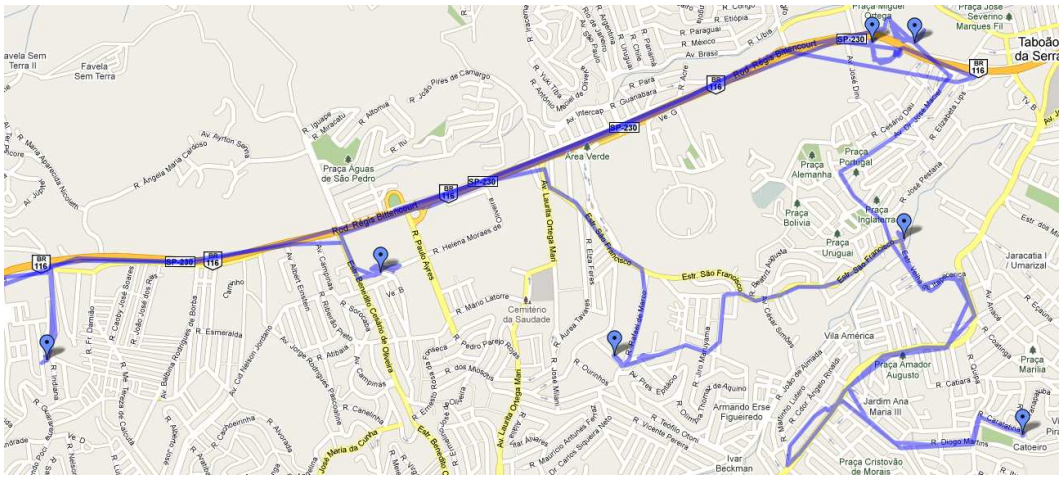


Fig. 8. Query results of Figure 7 plotted on Google Maps

```
select dti.month, dt.truck, tr_avg_speed(ft.travel)
from  fact_travel ft, dim_time dti, dim_truck dt, dim_region dr
where ft.begintime_fk = dti.timeid and dti.year = 2010 and
        dti.month >= 01 and dti.month <= 06 and
    ft.truck_fk = dt.moid and dt.branch = '5' and
    ft.region_fk = dr.rid and r.state = 'SAO_PAULO' and
    dt.truck in (
      select  dt2.truck
      from fact_event fe, dim_time dti2, dim_event_category dty, dim_truck dt2,
           dim_region dr2
      where fe.truck_fk = dt2.moid and dt2.branch = '5' and
       fe.begintime_fk = dti2.timeid and dti2.year = 2010 and
         dti2.month >= 01 and dti2.month <= 06 and
       fe.region_fk = dr.rid and r.state = 'SAO_PAULO' and
       fe.eventcategory_fk=dty.categoryid and dty.category='ExcessoDeVelocidade'
      having count(fe) > 100
      group by dt2.truck )
group by rollup(dti.month, dt.truck);
```

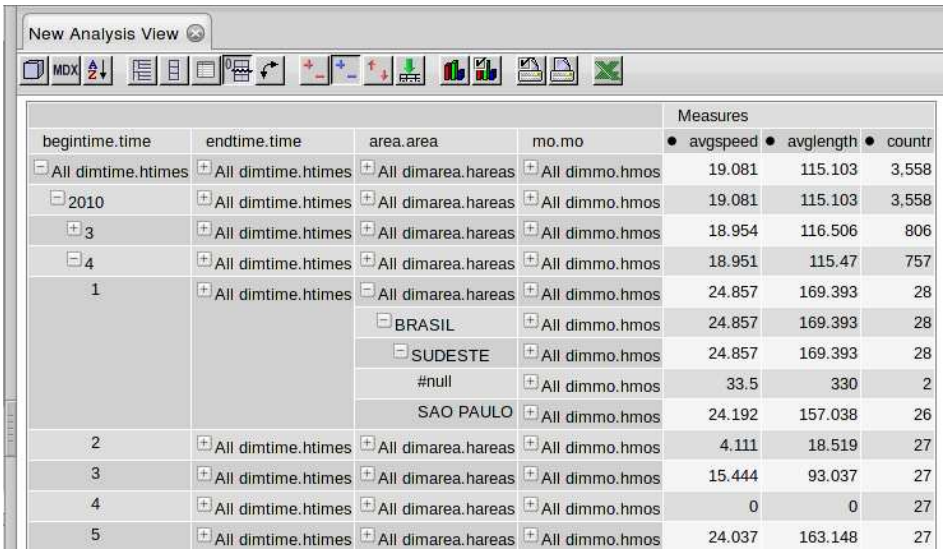
Fig. 9. DW query for recovering the average speed of trucks with more than 100 events of type "Excesso de Velocidade" within a given time interval grouped by month and truck

inside some specific areas (Figure 10).

In summary, the multidimensional environment maintains aggregate information in data cubes, which allows speeding up query execution for trajectory aggregation functions, group-bys, cross-tabs, and subtotals whereas the transactional environment is more adequate to non-aggregate queries that retrieves small number of records at a given time.

6. CONCLUSION

In this article, we presented a set of requirements for trajectory modeling and analysis that allow users to: (i) represent and query trajectories of moving objects as first class objects; and (ii) add semantics from the application domain to moving object trajectories thereby allowing real-time and historical analyses. To meet these requirements, we proposed two design patterns, one for semantic trajectories



				Measures		
begintime.time	endtime.time	area.area	mo.mo	avgspeed	avglength	countr
All dimtime.htimes	All dimtime.htimes	All dimarea.hareas	All dimmo.hmos	19.081	115.103	3,558
2010	All dimtime.htimes	All dimarea.hareas	All dimmo.hmos	19.081	115.103	3,558
3	All dimtime.htimes	All dimarea.hareas	All dimmo.hmos	18.954	116.506	806
4	All dimtime.htimes	All dimarea.hareas	All dimmo.hmos	18.951	115.47	757
1	All dimtime.htimes	All dimarea.hareas	All dimmo.hmos	24.857	169.393	28
		BRASIL	All dimmo.hmos	24.857	169.393	28
		SUDESTE	All dimmo.hmos	24.857	169.393	28
		#null	All dimmo.hmos	33.5	330	2
		SAO PAULO	All dimmo.hmos	24.192	157.038	26
2	All dimtime.htimes	All dimarea.hareas	All dimmo.hmos	4.111	18.519	27
3	All dimtime.htimes	All dimarea.hareas	All dimmo.hmos	15.444	93.037	27
4	All dimtime.htimes	All dimarea.hareas	All dimmo.hmos	0	0	27
5	All dimtime.htimes	All dimarea.hareas	All dimmo.hmos	24.037	163.148	27

Fig. 10. Truck Fleet Control Data Cube

in a transactional object-relational DBMS and another for semantic trajectories in a multidimensional database. We also specified two algorithms to help the construction of trajectory objects from a raw dataset in a transaction environment, as well as to populate multidimensional schemas from the object-relational database. To validate our main contributions, we illustrated our proposals by developing a real world application scenario based on large datasets concerning truck fleet management.

Trajectory management applications require using decision-support tools in both environments to perform both real-time and historical analyses. In addition, specific aggregation functions over moving object trajectories should be developed. This article contributes with an integrated solution to these requirements. To the best of our knowledge, this is the first article in the trajectory data management literature that copes with all these issues together.

As future work, we are interested in performing a more formal and detailed requirement analysis and in creating a complete and optimized application for a long period of tests in a real corporation environment. Another point that requires further investigation concerns the design of aggregation functions for trajectories, which needs a formal and complete study to treat trajectories as complex and first class objects, including aggregation with respect to the semantic information (i.e. events) and to other spatial structures, such as polygon geometries (e.g. areas or regions of interest) and points (e.g. clients or truck fleet bases).

REFERENCES

Alvares, L. O., Bogorny, V., Kuijpers, B., de Macedo, J. A. F., Moelans, B., and Vaisman, A. A model for enriching trajectories with semantic geographical information. In *Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems*. Seattle, WA, USA, pp. 22:1–22:8, 2007.

Baltzer, O., Dehne, F., Hambrusch, S., and Rau-Chaplin, A. Olap for trajectories. In *Proceedings of the 19th International Conference on Database and Expert Systems Applications*. Turin, Italy, pp. 340–347, 2008.

Chaudhuri, S. and Dayal, U. An overview of data warehousing and olap technology. *SIGMOD Rec.* 26 (1): 65–74, 1997.

Giannotti, F., Nanni, M., Pinelli, F., and Pedreschi, D. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Jose, CA, USA, pp. 330–339, 2007.

Guc, B., May, M., Saygin, Y., and Korner, C. Semantic annotation of gps trajectories. In *Proceedings of 11th AGILE International Conference on Geographic Information Science*. Girona, Spain, 2008.

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

Kimball, R. and Ross, M. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (Second Edition)*. Wiley, 2002.

Kuijpers, B., Moelans, B., and Van de Weghe, N. Qualitative polyline similarity testing with applications to query-by-sketch, indexing and classification. In *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems*. Arlington, USA, pp. 11–18, 2006.

Laube, P., Imfeld, S., and Weibel, R. Discovering relative motion patterns in groups of moving point objects. *International Journal of Geographical Information Science* 19 (6): 639–668, 2005.

Leonardi, L., Orlando, S., Raffaetà, A., Roncato, A., and Silvestri, C. Frequent spatio-temporal patterns in trajectory data warehouses. In *Proceedings of the 2009 ACM Symposium on Applied Computing*. Honolulu, USA, pp. 1433–1440, 2009.

Malinowski, E. and Zimányi, E. Representing spatiality in a conceptual multidimensional model. In *Proceedings of the 12th Annual ACM International Workshop on Geographic Information Systems*. Washington, DC, USA, pp. 12–22, 2004.

Marketos, G., Frentzos, E., Ntoutsi, I., Pelekis, N., Raffaetà, A., and Theodoridis, Y. Building real-world trajectory warehouses. In *Proceedings of the Seventh ACM International Workshop on Data Engineering for Wireless and Mobile Access*. Vancouver, Canada, pp. 8–15, 2008.

Moreno, B., Times, V. C., Renso, C., and Bogorny, V. Looking inside the stops of trajectories of moving objects. In *XI Brazilian Symposium on Geoinformatics*. Campos do Jordão, Brazil, pp. 9–20, 2010.

Nanni, M. and Pedreschi, D. Time-focused clustering of trajectories of moving objects. *J. Intell. Inf. Syst.* 27 (3): 267–289, 2006.

Orlando, S., Orsini, R., Raffaeta, A., and Roncato, A. Trajectory data warehouses: Design and implementation issues. *Journal of Computing Science and Engineering* 1 (2): 211–232, 2008.

Orlando, S., Orsini, R., Raffaetà, A., Roncato, A., and Silvestri, C. Spatio-temporal aggregations in trajectory data warehouses. In *Proceedings of International Conference on Data Warehousing and Knowledge Discovery*. pp. 66–77, 2007.

Palma, A. T., Bogorny, V., Kuijpers, B., and Alvares, L. O. A clustering-based approach for discovering interesting places in trajectories. In *Proceedings of the 2008 ACM symposium on Applied computing*. Fortaleza, CE, Brazil, pp. 863–868, 2008.

Pelekis, N., Kopanakis, I., Ntoutsi, I., Marketos, G., and Theodoridis, Y. Mining trajectory databases via a suite of distance operators. In *Proceedings of the IEEE 23rd International Conference on Data Engineering Workshop*. Istanbul, Turkey, pp. 575–584, 2007.

Pelekis, N., Raffaeta, A., Damiani, M. L., Vangenot, C., Marketos, G., Frentzos, E., Ntoutsi, I., and Theodoridis, Y. Towards trajectory data warehouses. In *Mobility, Data Mining and Privacy*. Springer, pp. 189–211, 2008.

Pelekis, N., Theodoridis, Y., Vosinakis, S., and Panayiotopoulos, T. Hermes - a framework for location-based data management. In *Proceedings of 10th International Conference on Extending Database Technology (EDBT)*. Munich, Germany, pp. 1130–1134, 2006.

Porto, F., Moura, A. M. C., Palazzi, D., da Silva, F. C., de Castro, L. E. V., and Cameron, L. C. Towards a scientific database for olympic athletes. In *Challenges in eScience Workshop*. Petropolis, Rio de Janeiro, 2010.

Rocha, J., Oliveira, G., Alvares, L., Bogorny, V., and Times, V. Db-smot: A direction-based spatio-temporal clustering method. In *Proceedings of 5th IEEE International Conference Intelligent Systems*. University of Westminster, London, UK, pp. 114 –119, 2010.

Spaccapietra, S., Parent, C., Damiani, M. L., de Macedo, J. A., Porto, F., and Vangenot, C. A conceptual view on trajectories. *Data & Knowledge Engineering* 65 (1): 126 – 146, 2008.

Wolfson, O., Xu, B., Chamberlain, S., and Jiang, L. Moving objects databases: issues and solutions. In *Scientific and Statistical Database Management, 1998. Proceedings. Tenth International Conference on*. Capri, Italy, pp. 111 –122, 1998.