

A Survey on Spatio-Temporal Data Warehousing

Leticia Gómez
Instituto Tecnológico de Buenos
Aires

Bart Kuijpers, Bart Moelans,
Alejandro Vaisman
University of Hasselt and
Transnational University of
Limburg, Belgium

Abstract

Geographic Information Systems (GIS) have been extensively used in various application domains, ranging from economical, ecological and demographic analysis, to city and route planning. Nowadays, organizations need sophisticated GIS-based Decision Support System (DSS) to analyze their data with respect to geographic information, represented not only as attribute data, but also in maps. Thus, vendors are increasingly integrating their products, leading to the concept of SOLAP (Spatial OLAP). Also, in the last years, and motivated by the explosive growth in the use of PDA devices, the field of moving object data has been receiving attention from the GIS community. However, not much has been done in providing moving object databases with OLAP functionality. In the first part of this paper we survey the SOLAP literature. We then move to Spatio-Temporal OLAP, in particular addressing the problem of trajectory analysis. We finally provide an in-depth comparative analysis between two proposals introduced in the context of the GeoPKDD EU project: the *Hermes-MDC* system, and *Piet*, a proposal for SOLAP and moving objects, developed at the University of Buenos Aires, Argentina.

Keywords: GIS, OLAP, Data Warehousing, Moving Objects, Trajectories, Aggregation

INTRODUCTION

Geographic Information Systems (GIS) have been extensively used in various application domains, ranging from economical, ecological and demographic analysis, to city and route planning (Rigaux, Scholl, & Voisard, 2001; Worboys, 1995). Spatial information in a GIS is typically stored in different so-called *thematic layers* (also called *themes*). Information in themes can be stored in 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 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. Spatial data in the different thematic layers of a GIS system can be mapped univocally to each other using a common frame of reference, like a coordinate system. These layers can be overlapped or overlaid to obtain an integrated spatial view.

On the other hand, OLAP (On Line Analytical Processing) (Kimball, 1996; Kimball & Ross, 2002) 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*. In the multidimensional model, data can be perceived as a *data cube*, where each cell contains a measure or set of (probably aggregated) measures of interest. As we discuss later, OLAP dimensions are further organized in hierarchies that favor the data aggregation process (Cabibbo & Torlone, 1997). Several techniques and algorithms have been developed for query processing, most of them involving some kind of aggregate precomputation (Harinarayan, Rajaraman, & Ullman, 1996).

The need for OLAP in GIS

Different data models have been proposed for representing objects in a GIS. ESRI (<http://www.esri.com>) first introduced the *Coverage* data model to bind geometric objects to non-spatial attributes that describe them. Later, they extended this model with object-oriented support, in a way that behavior can be defined for geographic features (Zeiler, 1999). The idea of the Coverage data model is also supported by the

Reference Model proposed by the Open Geospatial Consortium (<http://www.opengeospatial.org>). Thus, in spite of the model of choice, there is always the underlying idea of binding geometric objects to objects or attributes stored in (mostly) object-relational databases (Stonebraker & Moore, 1996). In addition, query tools in commercial GIS allow users to overlap several thematic layers in order to locate objects of interest within an area, like schools or fire stations. For this, they use indexing structures based on R-trees (Gutman, 1984). GIS query support sometimes includes aggregation of geographic measures, for example, distances or areas (e.g., representing different geological zones). However, these aggregations are not the only ones that are required, as we discuss below.

Nowadays, organizations need sophisticated GIS-based Decision Support System (DSS) to analyze their data with respect to geographic information, represented not only as attribute data, but also in maps, probably in different thematic layers. In this sense, OLAP and GIS vendors are increasingly integrating their products (see, for instance, Microstrategy and MapInfo integration in <http://www.microstrategy.com/>, and <http://www.mapinfo.com/>). In this sense, aggregate queries are central to DSSs. Classical aggregate OLAP queries (like “total sales of cars in California”), and aggregation combined with complex queries involving geometric components (“total sales in all villages crossed by the Mississippi river and within a radius of 100 km around New Orleans”) must be efficiently supported. Moreover, navigation of the results using typical OLAP operations like roll-up or drill-down is also required. These operations are not supported by commercial GIS in a straightforward way. One of the reasons is that the GIS data models discussed above were developed with “transactional” queries in mind. Thus, the databases storing non-spatial attributes or objects are designed to support those (non-aggregate) kinds of queries. Decision support systems need a different data model, where non-spatial data, probably consolidated from different sectors in an organization, is stored in a data warehouse. Here, numerical data are stored in fact tables built along several dimensions. For instance, if we are interested in the sales of certain products in stores in a given region, we may consider the sales amounts in a fact table over the three dimensions Store, Time and Product. In order to guarantee summarizability (Lenz & Shoshani, 1997), dimensions are organized into aggregation hierarchies. For example, stores can aggregate over cities which in turn can aggregate into regions and countries. Each of these aggregation levels can also hold descriptive attributes like city population, the area of a region, etc. To fulfill the

requirements of integrated GIS-DSS, warehouse data must be linked to geographic data. For instance, a polygon representing a region must be associated to the region identifier in the warehouse. Besides, system integration in commercial GIS is not an easy task. In the current commercial applications, the GIS and OLAP worlds are integrated in an ad-hoc fashion, probably in a different way (and using different data models) each time an implementation is required, even when a data warehouse is available for non-spatial data.

An Introductory Example. We present now a real-world example for illustrating some issues in the spatial warehousing problematic. We selected four layers with geographic and geological features obtained from the National Atlas Website (<http://www.nationalatlas.gov>). These layers contain the following information: states, cities, and rivers in North America, and volcanoes in the northern hemisphere (published by the Global Volcanism Program - GVP). Figure 1 shows a detail of the layers containing cities and rivers in North America, displayed using the graphic interface of the *Piet* implementation we discuss later in the paper. Note the density of the points representing cities (particularly in the eastern region). Rivers are represented as polylines. Figure 2 shows a portion of two overlaid layers containing states (represented as polygons) and volcanoes in the northern hemisphere. There is also non-spatial information stored in a conventional data warehouse. In this data warehouse, dimension tables contain customer, stores and product information, and a fact table contains stores sales across time. Also, numerical and textual information on the geographic components exist (e.g., population, area), stored as usual as attributes of the GIS layers.

In the scenario above, conventional GIS and organizational data can be integrated for decision support analysis. Sales information could be analyzed in the light of geographical features, conveniently *displayed in maps*. This analysis could benefit from the integration of both worlds in a single framework. Even though this integration could be possible with existing technologies, ad-hoc solutions are expensive because, besides requiring lots of complex coding, they are hardly portable. To make things more difficult, ad-hoc solutions require data exchange between GIS and OLAP applications to be performed. This implies that the output of a GIS query must be probably exported as members in dimensions of a data cube, and merged for further analysis. For example, suppose that a business analyst is interested in studying the sales of nautical goods in stores located in cities crossed by rivers. She would first query the GIS, to obtain the cities of interest. She probably

has stored sales in a data cube containing a dimension *Store* or *Geography* with *city* as a dimension level. She would need to “manually” select the cities of interest (i.e., the ones returned by the GIS query) in the cube, to be able to go on with the analysis (in the best case, an ad-hoc customized middleware could help her). Of course, she must repeat this for each query involving a (geographic) dimension in the data cube.

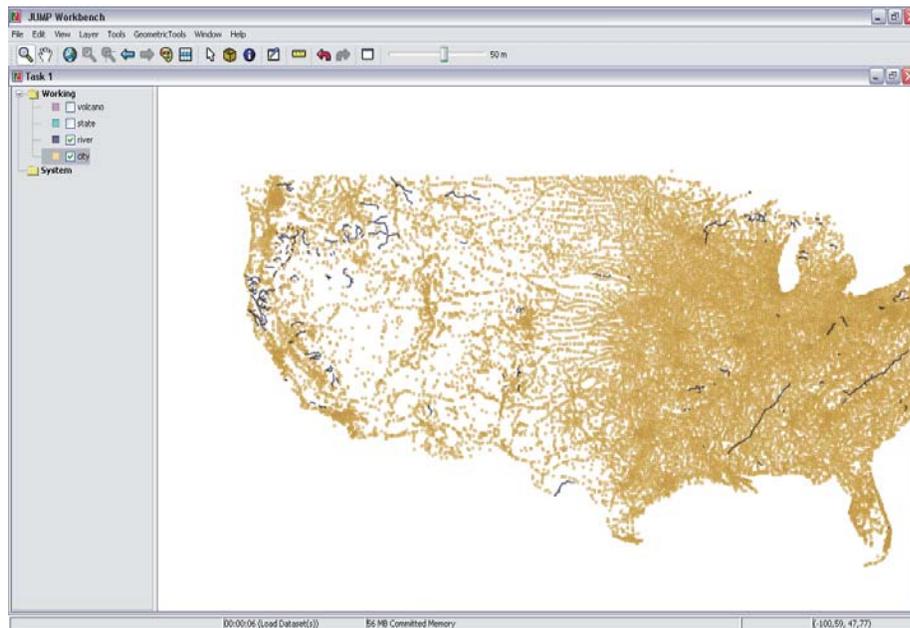


Figure 1. Two overlaid layers containing cities and rivers in North America.

On the contrary, GIS/Data warehousing integration can provide a more natural solution.

The second part of this survey is devoted to spatio-temporal data warehousing and OLAP. 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

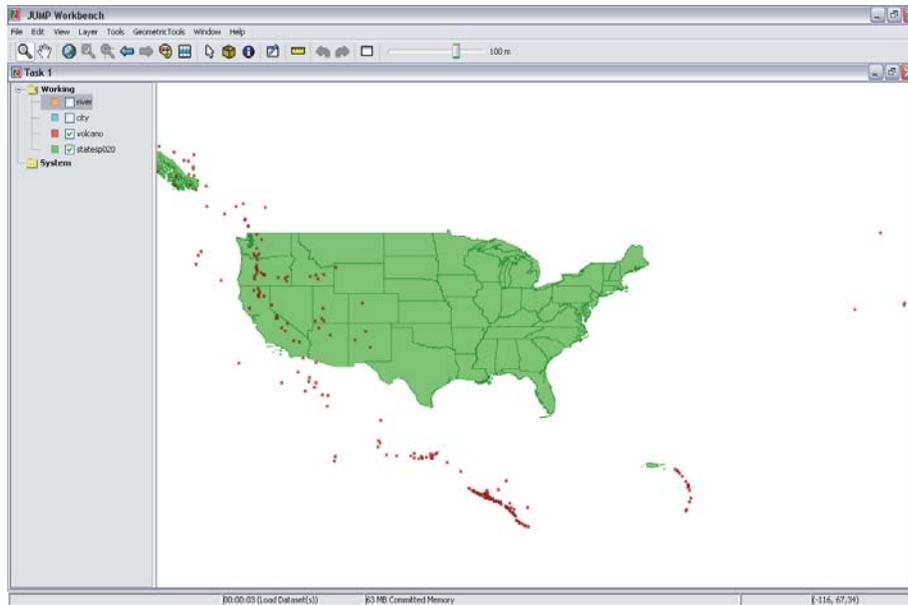


Figure 2. Two overlaid layers containing states in North America and volcanoes in the northern hemisphere.

exist 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. Many interesting applications arise, involving moving objects aggregation, mainly regarding traffic analysis, truck fleet behavior analysis, commuter traffic in a city, passenger traffic in an airport, or shopping behavior in a mall. Building trajectory data warehouses that can integrate with a GIS is an open problem that is starting to attract database researchers. Finally, the MOD setting is appropriate for data mining tasks, and we also comment on this in the paper.

In this paper, we first provide a brief background on GIS, data warehousing and OLAP, and a review of the state-of-the-art in spatial OLAP. After this, we move on to study spatio-temporal data warehousing, OLAP and mining. We then provide a detailed analysis of the *Piet framework*, aimed at integrating GIS, OLAP and moving object data, and conclude with a comparison between this proposal, and the Hermes data cartridge and trajectory data warehouse developed in the context of the GeoPKDD project (Information about the GoePKDD project can be found at <http://www.geopkdd.eu>).

A SHORT BACKGROUND

GIS

In general, information in a GIS application is divided over several *thematic layers*. The information in each layer consists of purely spatial data on the one hand, that is combined with classical alpha-numeric attribute data on the other hand (usually stored in a relational database). Two main data models are used for the representation of the spatial part of the information within one layer, the *vector model* and the *raster model*. The choice of model typically depends on the data source from which the information is imported into the GIS.

The Vector Model. The *vector model* is used the most in current GIS (Kuper & Scholl, 2000). In the vector model, infinite sets of points in space are represented as finite geometric structures, or *geometries*, like, for example, points, polylines and polygons. More concretely, vector data within a layer consists in a finite number of tuples of the form (*geometry, attributes*) where a geometry can be a point, a polyline or a polygon. There are several possible data structures to actually store these geometries (Worboys, 1995).

The Raster Model. In the *raster model*, the space is sampled into pixels or cells, each one having an associated attribute or set of attributes. Usually, these cells form a uniform grid in the plane. For each cell or pixel, the sample value of some function is computed and associated to the cell as an attribute value, *e.g.*, a numeric value or a color. In general, information represented in the raster model is organized into *zones*, where the cells of a zone have the same value for some attribute(s). The raster model has very efficient indexing structures and it is very well-suited to model continuous change but its disadvantages include its size and the cost of computing the zones.

Spatial information in the different thematic layers in a GIS is often joined or overlaid. Queries requiring map overlay are more difficult to compute in the vector model than in the raster model. On the other hand, the vector model offers a concise representation of the data, independent on the resolution. For a uniform treatment of different layers given in the vector or the raster model, in this paper we treat the raster model as a special case of the vector model. Indeed, conceptually, each cell is, and each pixel can be regarded as, a small polygon; also, the attribute value associated to the cell or pixel can be regarded as an attribute in the vector model.

Data Warehousing and OLAP

The importance of data analysis has increased significantly in recent years as organizations in all sectors are required to improve their decision-making processes in order to maintain their competitive advantage. We said before that OLAP (On Line Analytical Processing) (Kimball, 1996; Kimball & Ross, 2002) comprises a set of tools and algorithms that allow efficiently querying databases that contain large amounts of data. These databases, usually designed for read-only access (in general, updating is performed off-line), are denoted data warehouses. Data warehouses are exploited in different ways. OLAP is one of them. OLAP systems are based on a *multidimensional model*, which allows a better understanding of data for analysis purposes and provides better performance for complex analytical queries. The multidimensional model allows viewing data in an n -dimensional space, usually called a *data cube* (Kimball & Ross, 2002). In this cube, each cell contains a measure or set of (probably aggregated) measures of interest. This factual data can be analyzed along dimensions of interest, usually organized in hierarchies (Cabibbo & Torlone, 1997). Three typical ways of OLAP tools implementation exist: MOLAP (standing for multidimensional OLAP), where data is stored in proprietary multidimensional structures, ROLAP (relational OLAP), where data is stored in (object) relational databases, and HOLAP (standing for hybrid OLAP, which provides both solutions. In a ROLAP environment, data is organized as a set of *dimension tables* and *fact tables*, and we assume this organization in the remainder of the paper.

There are a number of OLAP operations that allow exploiting the dimensions and their hierarchies, thus providing an interactive data analysis environment. Warehouse databases are optimized for OLAP operations which, typically, imply data aggregation or de-aggregation along a dimension, called roll-up and drill-down, respectively. Other operations involve selecting parts of a cube (slice and dice) and re-orienting the multidimensional view of data (pivoting). In addition to the basic operations described above, OLAP tools provide a great variety of mathematical, statistical, and financial operators for computing ratios, variances, ranks, etc.

It is an accepted fact that data warehouse (conceptual) design is still an open issue in the field (Rizzi & Golfarelli, 2000). Most of the data models either provide a graphical representation based on the Entity-Relationship (E/R) model or UML notations, or they just provide some formal definitions without user-oriented graphical support. Recently,

Malinowsky and Zimányi (2006) propose the *MultiDim* model. This model is based on the E/R model and provides an intuitive graphical notation. Also recently, Vaisman (Vaisman, 2006a, 2006b) introduced a methodology for requirement elicitation in Decision Support Systems, arguing that methodologies used for OLTP systems are not appropriate for OLAP systems.

Temporal Data Warehouses

The relational data model as proposed by Codd (1970), is not well-suited for handling spatial and/or temporal data. Data evolution over time must be treated in this model, in the same way as ordinary data. This is not enough for applications that require past, present, and/or future data values to be dealt with by the database. In real life such applications abound. Therefore, in the last decades, much research has been done in the field of temporal databases. Snodgrass (1995) describes the design of the TSQL2 Temporal Query Language, an upward compatible extension of SQL-92. The book, written as a result of a Dagstuhl seminar organized in June 1997 by Etzion, Jajodia, and Sripada (1998), contains comprehensive bibliography, glossaries for both temporal database and time granularity concepts, and summaries of work around 1998. The same author (Snodgrass, 1999), in other work, discusses practical research issues on temporal database design and implementation.

Regarding temporal data warehousing and OLAP, Mendelzon and Vaisman (2000, 2003) proposed a model, denoted TOLAP, and developed a prototype and a datalog-like query language, based on a (temporal) star schema. Vaisman, Izquierdo, and Ktenas (2006) also present a Web-based implementation of this model, along with a query language, called TOLAP-QL. Eder, Koncilia, and Morzy (2002) also propose a data model for temporal OLAP supporting structural changes. Although these efforts, little attention has been devoted to the problem of conceptual and logical modeling for temporal data warehouses.

SPATIAL DATA WAREHOUSING AND OLAP

Spatial database systems have been studied for a long time (Buchmann, Günther, Smith, & Wang, 1990; Paredaens, Van Den Bussche, & Gucht, 1994). Rigaux et al. (2001) survey various techniques, such as spatial data models, algorithms, and indexing methods, developed to address specific features of spatial data that are not adequately handled by mainstream DBMS technology.

Although some authors have pointed out the benefits of combining GIS and OLAP, not much work has been done in this field. Vega López, Snodgrass, and Moon (2005) present a comprehensive survey on spatiotemporal aggregation that includes a section on spatial aggregation. Also, Bédard, Rivest, and Proulx (2007) present a review of the efforts for integrating OLAP and GIS. As we explain later, efficient data aggregation is crucial for a system with GIS-OLAP capabilities.

Conceptual Modeling and SOLAP

Rivest, Bédard, and Marchand (2001) introduced the concept of SOLAP (standing for Spatial OLAP), a paradigm aimed at being able to explore spatial data by drilling on maps, in a way analogous to what is performed in OLAP with tables and charts. They describe the desirable features and operators a SOLAP system should have. Although they do not present a formal model for this, SOLAP concepts and operators have been implemented in a commercial tool called JMAP, developed by the Centre for Research in Geomatics and KHEOPS, see <http://www.kheops-tech.com/en/jmap/solap.jsp>. Stefanovic, Han, and Koperski (2000) and Bédard, Merret, and Han (2001), classify spatial dimension hierarchies according to their spatial references in: (a) non-geometric; (b) geometric to non-geometric; and (c) fully geometric. Dimensions of type (a) can be treated as any descriptive dimension (Rivest et al., 2001). In dimensions of types (b) and (c), a geometry is associated to members of the hierarchies. Malinowski and Zimányi (2004) extend this classification to consider that even in the absence of several related spatial levels, a dimension can be considered spatial. Here, a dimension level is *spatial* if it is represented as a spatial data type (e.g., point, region), allowing them to link spatial levels through topological relationships (e.g., contains, overlaps). Thus, a *spatial dimension* is a dimension that contains at least one spatial hierarchy. A critical point in spatial dimension modeling is the problem of multiple-dependencies, meaning that an element in one level can be related to more than one element in a level above it in the hierarchy. Jensen, Kligys, Pedersen, and Timko (2004) address this issue, and propose a multidimensional data model for mobile services, i.e., services that deliver content to users, depending on their location. This model supports different kinds of dimension hierarchies, most remarkably multiple hierarchies in the same dimension, i.e., multiple aggregation paths. Full and partial containment hierarchies are also supported. However, the model does not consider the geometry, limiting the set of queries that can be

addressed. This means that spatial dimensions are standard dimensions referring to some geographical element (like cities or roads).

Malinowski and Zimányi (2006) also propose a model supporting multiple aggregation paths.

Pourabbas (2003) introduces a conceptual model that uses binding attributes to bridge the gap between spatial databases and a data cube. The approach relies on the assumption that all the cells in the cube contain a value, which is not the usual case in practice, as the author expresses. Also, the approach requires modifying the structure of the spatial data to support the model. No implementation is presented.

Shekhar, Lu, Tan, Chawla, & Vatsavai (2001) introduced MapCube, a visualization tool for spatial data cubes. MapCube is an operator that, given a so-called base map, cartographic preferences and an aggregation hierarchy, produces an album of maps that can be navigated via roll-up and drill-down operations.

Spatial Measures. Measures are characterized in two ways in the literature, namely: (a) measures representing a geometry, which can be aggregated along the dimensions; (b) a numerical value, using a topological or metric operator. Most proposals support option (a), either as a set of coordinates (Bédard et al., 2001; Rivest et al., 2001; Malinowski & Zimányi, 2004; Bimonte, Tchounikine, & Miquel, 2005), or a set of pointers to geometric objects (Stefanovic et al., 2000). Bimonte et al. (Bimonte et al., 2005) define measures as complex objects (a measure is thus an object containing several attributes). Malinowski and Zimányi (2004) follow a similar approach, but defining measures as attributes of an n-ary fact relationship between dimensions.

Damiani and Spaccapietra (2006) propose MuSD, a model allowing defining spatial measures at different granularities. Here, a spatial measure can represent the location of a fact at multiple levels of (spatial) granularity. Also, an algebra of SOLAP operators is proposed.

Spatial Aggregation

In light of the discussion above, it should be clear that aggregation is a crucial issue in spatial OLAP. Moreover, there is not yet a consensus about a complete set of aggregate operators for spatial OLAP. We now discuss the classic approaches to spatial aggregation.

Han et al. (1998) use OLAP techniques for materializing selected spatial objects, and proposed a so-called *Spatial Data Cube*, and the set of operations that can be performed on this data cube. The model only supports aggregation of spatial objects.

Pedersen and Tryfona (2001) propose the pre-aggregation of spatial facts. First, they pre-process these facts, computing their disjoint parts in order to be able to aggregate them later. This pre-aggregation works if the spatial properties of the objects are *distributive* over some aggregate function. Again, the spatial measures are geometric objects. Given that this proposal ignores the geometries, queries like “total population of cities crossed by a river” are not supported. The paper does not address forms other than polygons, although the authors claim that other more complex forms are supported by the method, and the authors do not report experimental results.

With a different approach, Rao, Zhang, Yu, Li, and Chen (2003), and Zhang, Li, Rao, Yu, Chen, and Liu (2003) 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 exploited in the usual OLAP way. Thus, they take advantage of OLAP hierarchies for locating information in the R-tree which indexes the fact table.

Although the measures here are not only spatial objects, the proposal also ignores the geometric part of the model, limiting the scope of the queries that can be addressed. It is assumed that some fact table, containing the identifiers of spatial objects exists. Finally, these objects happen to be points, which is quite unrealistic in a GIS environment, where different types of objects appear in the different layers.

Some interesting techniques have been recently introduced to address the data aggregation problem. These techniques are based on the combined use of (R-tree-based) indexes, materialization (or pre-aggregation) of aggregate measures, and computational geometry algorithms.

Papadias, Tao, Kalnis, and Zhang (2002) introduce the *Aggregation R-tree* (aR-tree), combining indexing with pre-aggregation. The aR-tree is an R-tree that annotates each MBR (Minimal Bounding Rectangle) with the value of the aggregate function for all the objects that are enclosed by it. They extend this proposal in order to handle historic information (see the section on moving object data below), denoting this extension aRB-tree (Papadias, Tao, Zhang, Mamoulis, Shen, and Sun, 2002). The approach basically consists in two kinds of indexes: a host index, which is an R-tree with the summarized information, and a B-tree containing time-varying aggregate data. In the most general case, each region has a B-tree associated, with the historical information of the measures of interest in the region. This is a very efficient solution for some kinds of queries, for example, window aggregate queries (i.e., for the computation of the aggregate measure of the regions which intersect a spatio-temporal window). In addition, the

method is very effective when a query is posed over a query region whose intersection with the objects in a map must be computed on-the-fly, and these objects are totally enclosed in the query region. However, problems may appear when leaf entries partially overlap the query window. In this case, the result must be estimated, or the actual results computed using the base tables. In fact, Tao, Kollios, Considine, Li, and Papadias (2004), show that the aRB-tree can suffer from the distinct counting problem, if the object remains in the same region for several timestamps.

SPATIO-TEMPORAL DATA WAREHOUSING, OLAP AND MINING

The field of moving objects databases has been extensively studied in the last ten years, mainly regarding data modeling and indexing. Güting and Schneider (2005) provide a good reference to this large corpus of work. Moving objects, carrying location-aware devices, produce *trajectory* data in the form of a sample of (O_{id}, x, y, t) -tuples, that contain object identifier and time-space information. In this survey, we will focus on the problem of building trajectory data warehouses and exploiting them through OLAP and data mining techniques.

Modeling

Generally speaking, in order to support trajectory data a spatio-temporal data cube should allow analysis along (a) temporal dimensions; (b) spatial dimensions at different levels of granularity (point, cell, road); (c) thematic dimensions, containing, for instance, demographic data. In this sense, hierarchies must take into account the fact that an element may rollup to more than one in an upper level. For instance, a road can probably cross more than one cell, thus, there is no function from a level *cell* to a level *road*. It is worth noticing that some proposals deal with this problem defining complex relationships (e.g., containment) in the dimension hierarchies (Jensen et al., 2004), which in general, lead to approximations. The *Piet framework*, discussed below, defines different GIS dimensions for different kinds of geometries. In the former example, the query language would handle the problem of finding out the cells that intersect the road.

Wolfson, Sistla, Xu, and Chamberlain (1999) define a set of capabilities that a moving object database must have, and introduce the DOMINO system, that develops those features on top of existing database management systems (DBMS). Hornsby and Egenhofer (2002) introduce a framework for modeling moving objects,

which supports viewing objects 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 $\langle Id, location, time \rangle$, where *Id* is the identifier of the object, *location* is given by x-y coordinates, and *time* is the timestamp of the observation. The possible positions of an object between two observations is estimated to be within two inverted half-cones that conform a *lifeline bead*, whose projection over the x-y plane is an ellipse.

Particular interest has received the topic of moving objects on road networks. Van de Weghe et al. propose a qualitative trajectory calculus for objects in a GIS (Weghe, Cohn, Tré, & Maeyer, 2005), based on the assumption that in a GIS scenario, qualitative information is necessary (and, in general, more useful than quantitative information). For mining trajectories in road networks, Brakatsoulas, Pfoser, and Tryfona (2004) propose to enrich trajectories of moving objects with information about the relationships between trajectories (e.g., *intersect*, *meets*), and between a trajectory and the GIS environment (*stay within*, *bypass*, *leave*). They also proposed a mining language denoted SML (for Spatial Mining Language). This language is oriented to traffic networks, and it is not clear how it could be extended to other scenarios. Moreover, all information on moving objects must be processed (on the contrary, we use semantic information to reduce, if possible, the amount of data to be considered).

Also in the framework of road traffic mining, Gonzalez, Han, Li, Myslinska, and Sondag (2007) use a partitioning approach for obtaining interesting driving and speed patterns from large sets of traffic data. They compute frequent path-segments at the area level with a support relative to the traffic in the area (i.e., a kind of adaptative support), and propose an algorithm to automatically partition a road network and build a hierarchy of areas.

The work of Lee, Han, and Whang (2007) is aimed at discovering common sub-trajectories, using a partitioning strategy which divides a trajectory into a set of line segments, and then groups similar line segments together into a cluster.

Like in the case of spatial OLAP (and multidimensional databases, in general), from the conceptual modeling point of view, there has not been much interest from the database community. Malinowski and Zimányi (2006) propose a model to provide a graphical representation, based on the Entity/Relationship model, and on UML. To the best of our knowledge, the Piet data model discussed in detail below, is the first attempt to provide a formal framework for integrating spatio-temporal data with OLAP and data warehousing.

Adding Semantic Information to Trajectory Data

Techniques that add semantic information to trajectory data have been recently proposed. Mouza and Rigaux (2005) present a model where trajectories are represented by a sequence of moves. They propose a query language based on regular expressions, aimed at obtaining so-called *mobility patterns*. Note that this language, as well as the proposals commented above, *does not relate trajectories with the GIS environment*, which limits the types of queries that can be addressed. Along the same lines, Damiani, Macedo, Parent, Porto, and Spaccapietra (2007) introduced the concept of *stops and moves*, in order to enrich trajectories with semantically annotated data. Below, we will give more details about the stops and moves paradigm. Giannotti, Nanni, Pinelli, and Pedreschi (2007) studied trajectory pattern mining, based on so-called Temporally Annotated Sequences (*TAS*), an extension of sequential patterns, where a temporal annotation between two nodes is defined. In this way, the sequence $\langle s_1, 2, s_2 \rangle$ defines a pattern that starts at s_1 and after two seconds arrives at s_2 . In other words, a *trajectory pattern* is a set of trajectories that visit the same sequence of places with similar travel times between each of them. They also propose three different mining methods. They also introduce the concept of Region of Interest (RoI). In the paper, the authors focused on *computing the RoIs dynamically from the trajectories*. Along similar lines, Gómez, Kuijpers, and Vaisman (2008a,2008b) present a model and implementation where trajectories are replaced by sequences of *stops and moves*, also following the ideas of Alvares et al. (2007). This work differs from the one of Giannotti et al. (2007) in several ways: first, the authors work with stops and moves instead of pre-defined regions of interest. This allows identifying which of the RoIs are really relevant to a trajectory. Second, the stops and moves are used to “encode” or compress a trajectory, which, in many practical situations turns out to be enough to identify interesting sequences very efficiently. A third difference is that in this proposal, the user defines the places of interest of an application *in advance*, and then they compute the stops and moves to perform trajectory mining. Finally, the approach of Gómez et al. (2007) allows integration between trajectories and geographic data, an issue mentioned albeit not addressed in (Giannotti et al., 2007).

Trajectory Similarity and Aggregation

The problem of trajectory similarity and aggregation in moving object databases is a new topic in the spatio-temporal database literature.

Existing work focuses on the spatial notion of similarity, sometimes borrowing from the time-series analysis field. This is the approach followed by Pelekis, Kopanakis, Ntoutsis, Marketos, Andrienko, and Theodoridis (2007), and Pelekis, Kopanakis, Ntoutsis, Marketos, and Theodoridis (2007), who introduce a framework consisting of a set of distance operators based on parameters of trajectories like speed and direction, and propose distance operators based on this. Frentzos, Gratsias, and Theodoridis (2007) propose an approximation method for supporting the k-most-similar-trajectory search using R-tree structures.

Data aggregation is still quite an open field, either in GIS or in a moving objects scenario. Meratnia and de By (2002) study trajectory aggregation by identifying similar trajectories, merging them in a single one, and dividing the area under study into homogeneous *spatial units*. We commented above the work of Papadias et al (2002), about indexing of historical aggregate information about moving objects. Kuijpers and Vaisman (2007) presented a taxonomy of aggregate queries on moving object data. The model and query language we present here covers the different types of aggregation queries in this taxonomy.

The Hermes System and the GeoPKDD Trajectory Data Warehouse

Among a limited number of proposals for trajectory data warehouses, the work by Orlando, Orsini, Raffaetà, Roncato, and Silvestri (2007) is worth an in-depth discussion. They build a trajectory data warehouse aimed at providing the infrastructure needed to deliver advanced reporting capabilities and facilitating the application of mining algorithms on aggregate data, for the GeoPKDD project (see <http://www.geopkdd.eu>). Since this project is based in the *Hermes architecture*, we first give a brief overview of the Hermes system (Pelekis, N., Theodoridis, Y., Vosinakis, S., and Panayiotopoulos, T., 2006; Pelekis & Theodoridis, 2006).

The Hermes System for Location-Based Services. Hermes provides the functionality needed for handling two-dimensional objects that change location, shape and size, through three kinds of data types: (a) static base data types (b) static temporal data types; (c) static spatial types; (d) moving data types. Data of type (a) are the standard DBMS data types (integer, real, etc.). Data of type (b) are based on the so-called TAU temporal object model (Kakoudakis, 1996), and provide Hermes with temporal object-relational capabilities, through a library denoted TAU-TLL (Pelekis, 2002). The new temporal data types supported (extending the ODMG data model) are Timepoint, Period, and

Temporal Element. The spatial data types (c) are provided by the Oracle Spatial library. The object type defined in Oracle, and used by Hermes, is called *Sdo_Geometry*. The *Moving* data type (d) encapsulates semantics and functionality of different data types: moving point, linestring, circle, rectangle, polygon, and moving collection. Below these types, a class hierarchy is defined. The basic type is *moving point*, defined as a sequence of different types of simple functions. It is based on the sliced representation proposed by Güting, Böhlen, Jensen, Lorentzos, Schneider, and M. & Vazirgiannis (2000). Here, a temporal development of a moving object is decomposed in slices such that, between each slice, a simple function is defined. The idea is to decompose the definition of each moving type into several definitions, one for each function. The composition of these sub-definitions defines a moving type. This way, a *unit_function* models the case where a user is located at a point (x_i, y_i) and moves, with initial velocity v and acceleration a or a linear or circular arc route. A flag indicates the type of movement. The point (x_e, y_e) is the end point of the movement. The unit function along with the *period* object type, conforms the *moving point* data type, which is the basis for the other types. For instance, the type *moving circle* is formed by the function *unit_moving_circle* plus the *period* data type. In turn, the former is composed of three *unit_moving_point* objects. Details on these data types can be found in (Pelekis & Theodoridis, 2006). The objects belonging to the moving type are provided with a set of operations: (a) topological and distance predicates, like *within_distance*; (b) temporal functions, like *add_unit* (adds a new unit of movement), and *at_instant* (returns the union of the projection of a moving object at a time instant); (c) distance and direction operators (for instance, the distance between two moving objects); (d) set relationships (like *intersection*). Also, numeric operations on objects are supported, like area or length. In consequence, it would be easy to compute, for instance, the area of an object at a given time instant.

The Hermes architecture can be described as follows: the basic components are the TAU-TLL library, the Oracle spatial cartridge and the Hermes-MDC (Moving Data Cartridge), which includes the moving data types. PL-SQL statements, which are compiled and stored in binary form, use those cartridges and data types. Thus, the PL-SQL statements are available for interacting with Oracle 10g data structures. Applications written, for instance, in Java, can consume these data. The types of queries supported by Hermes are: (a) queries on stationary objects, like: point, range, distance-based, topological, and nearest-neighbor queries; (b) queries on moving reference objects (distance-

based and similarity queries); (c) Join queries; (d) queries involving unary operators (traveled distance, speed).

Actually, given that Hermes consists, essentially, in a set of data types, an application designer can define a database schema that uses these data types, and take advantage of their functionality. For example, to describe the movement of a toxic cloud, one could define a relation:

Cloud (id:integer, name: varchar, shape: moving_polygon)

Then, an application programmer could write code that uses these data structures, to find, for instance, when did the cloud arrived to California. Obviously, formally, the expressive power is provided by the data types, because there is no language associated to Hermes. Instead, a host language (Java, PL-SQL, both), is used.

Having introduced Hermes, we are ready to discuss the spatio-temporal data warehousing architecture for GeoPKDD

The GeoPKDD Trajectory Data Warehouse. Figure 3, taken from Damiani, Vangenot, Frenzos, Marketos, Theodoridis, Veryklos, and Raffaeta (2007), depicts the GeoPKDD trajectory warehouse architecture (TDW). Initially, location data is captured, and handled by a so-called trajectory stream manager, which builds trajectories from these data (e.g., splitting the raw data according to some criteria, providing a trajectory identifier, among other functionalities). These trajectories are stored in a relational table, denoted *RelTrajectories*, and then loaded into a moving object database (MOD), which is in turn managed by the Hermes system introduced above. Basically, the MOD includes a relation *MODTrajectories* with schema (O_{id} , trajectory_{id}, trajectory), where *trajectory* is of type *Moving Point*. Actually, although this may appear redundant, since trajectories are stored twice, this is required to be able to work with the the moving point data type. Finally, an ETL (Extraction, Transformation, and Loading) process, feeds the TDW. Queries to this warehouse can integrate geographical data. Below, we give details of the TDW.

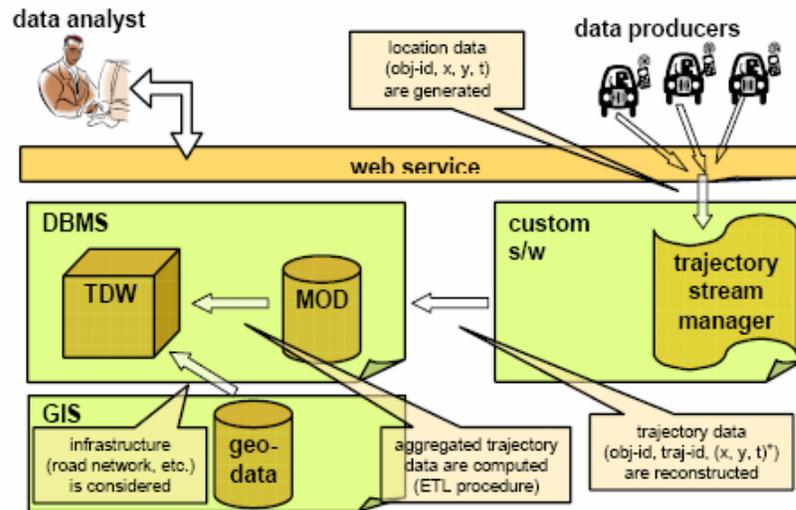


Figure 3. The trajectory warehouse architecture.

The trajectory data warehouse model proposed upon the criteria commented above, is based on the classic star schema. It contains a standard temporal dimension, and two spatial dimensions. The former ranges over equally sized time intervals, which are aggregated according to larger intervals as we move up in the dimension hierarchy (e.g., the interval $[60,120]$ aggregates over the interval $[0,120]$). The spatial dimensions, denoted DimX and DimY, range over equally sized spatial intervals (x,y, respectively), defining the cells where measures are recorded. A fact table containing references to the dimensions and (some of) the measures commented above, exists, in this case, presence, crossX, crossY, and crossT, where, for instance, crossX is the number of distinct trajectories crossing the spatial border between two cells along the horizontal axis. Roll-up and drill-down are performed aggregating measures over the cells, at different granularities (for instance, combining two or more cells). The key of this fact table is composed of the keys of the dimensions, namely dimX_id, dimY_id, dimT_id. We remark that the actual implementation has a slightly different form than this model, although for presentation clarity, in what follows we base ourselves on this structure. The interested reader can see Marketos, G., Frentzos, E., Ntousi, I., Pelekis, N., Raffaeta, A., & Theodoridis, Y. (2008) for the actual implementation. It is important to note that *no trajectory information* is recorded in the TDW. This information lies only in the MOD, and can be used for

querying, along with the information in the TDW, in order to obtain higher level information. In addition, modeling and storing trajectories is performed with *Hermes*.

A relevant feature of the TDW proposal is the treatment given to the *ETL process*, that transform the raw location data and loads it to the trajectory data warehouse. This process design is aimed at minimizing the amount of memory needed to load and transform raw data into trajectory data. For this, Orlando et al. (2007) define a model supporting two equivalent forms of trajectory representation: (a) the standard (O_{id}, x, y, t) ; (b) an alternative representation where coordinates in the trajectory database are replaced by cell identifiers that cover the (x, y) points. In this case, the tuples in the trajectory database are of the form $(O_{id}, Cell_{id}, t)$. In addition, other information of interest could be recorded, like, for instance, signal strength. As we explained above, the *raw location data* (usually arriving as a continuous data stream) is transformed into *trajectory data*, splitting the former according to certain assumptions like, for instance, if a large time gap between two consecutive sampled positions, a new trajectory identifier is created starting from the latter position.

The TDW introduced at the beginning of this section, is based on requirements that the authors defined for the dimensions and measures. We now discuss the *rationale* for this design. For the *dimensions*, they include temporal, spatial and thematic dimensions. The choice for the *measures* impact on the design and query evaluation processes. Typical measures include: (a) number of trajectories in a cell; (b) number of trajectories entering and/or leaving a cell, denoted *presence* ($C_{x,y}.presence$); (c) number of objects in a cell in a certain interval; (d) distance covered by trajectories in the cell, and the time spent in a cell; (e) velocity of trajectories in the cell. OLAP operations require aggregation of these measures over the set of cells. The problem of *double counting* arises for some of these measures, like (a) above. This problem appears not only during *aggregation* of the base data during a roll-up operation, but *also in the loading phase*. For example, suppose we have three consecutive observations o_1, o_2 and o_3 ; further, o_1 and o_3 fall in the same cell, but o_2 does not. When o_3 arrives, the system stores a duplicate for $C_{x,y}.presence$ (recall data is assumed to come as a continuous input stream). The *presence* measure deserved an in-depth treatment in (Orlando et al., 2007), where the problem of multiple counting was addressed, and some strategies for approximating the

results of computing the pre-aggregated facts were proposed. For instance, linear interpolation is used to prevent omitting in the result the cells crossed by a trajectory but such that no sampling occurred within them. Finally, two alternative functions for computing the aggregate *presence* are defined and compared against each other: one *algebraic*, and one *distributive*. The authors address the problem of double counting during aggregation, borrowing from statistical methods. For example, knowing the values of presence for two cells, $C_{x,y}$ and $C_{x+1,y}$, and defining a new cell, $C_{x',y'} = C_{x,y} \cup C_{x+1,y}$, the aggregate presence over the new cell, will be:

$$C_{x',y'}.presence = C_{x,y}.presence + C_{x+1,y}.presence - C_{x,y}.crossX$$

where $C_{x,y}.crossX$ is the number of distinct trajectories crossing the spatial border between $C_{x,y}$ and $C_{x+1,y}$.

Some example queries are provided in Orlando et al. (2007), and the two *presence* functions implemented (i.e., distributive and algebraic). It is reported that *algebraic presence* is more difficult to implement because it requires the combination of several aggregate functions and using non/standard SQL operations. The experiments reported showed that the distributive function (*sum*) quickly reaches large errors when the roll-up granularity increases. The *algebraic* method resulted to be more accurate.

With respect to querying the TDW, and from the point of view of the expressive power of the proposal, considerations here are similar to the ones we made when discussing *Hermes*. The data types provide the functionality, and clients can consume them. Of course, this allows any external data to participate in any query. However, again, the formal model is embedded in the data types, and the TDW can be regarded as an application where queries are built on top of the former. This is reflected in the fact that the warehouse contains only aggregated information, and the MOD contains the moving point type. The following is an example of a query over the MOD, showing a temporal intersection, taken from the TDW demo website.

SELECT

```

m.trajectory.at_period(tau_tll.d_period_sec(tau_tll.
D_Timepoint_Sec(2006,11,24,7,45,0),
tau_tll.D_Timepoint_Sec(2006,11,24,7,52,0))).to_stri
ng() as trajectory
FROM modtrajectories m where m.obj_id=1 and
m.traj_id=87

```

Here we can see that the table in the FROM clause is MODTrajectories, which includes the moving point data type. These kinds of queries could also use the fact tables that contain aggregate data. Dimensions and fact tables could also be analyzed using any OLAP viewer.

THE PIET FRAMEWORK

An approach different from the ones discussed previously in this paper was presented in the Piet framework (<http://piet.exp.dc.uba.ar/piet>). The Piet data model was introduced by Escribano, Gomez, Kuijpers, and Vaisman (2007) and Gómez, Haesevoets, Kuijpers, and Vaisman (2007). The core idea is the integration of spatial, spatio-temporal, and non-spatial data, in a single framework, oriented to solve many of the problems discussed in Section “Data Warehousing and OLAP”.

Dimensions

The model defines a *GIS dimension* as composed of a set of graphs, each one describing a set of geometries in a thematic layer. A GIS dimension is considered, as usual in databases, as composed of a schema and instances. Figure 4 shows the schema of a GIS dimension: the bottom level of each hierarchy, denoted the *Algebraic part*, contains the infinite points in a layer, and could be described by means of linear algebraic equalities and inequalities (Paredaens, Kuper, & Libkin, 2000). Above this part there is the *Geometric part*, which stores the identifiers of the geometric elements of the GIS, and is used to solve the geometric part of a query. Each point in the Algebraic part may correspond to one or more elements in the Geometric part (e.g., if more than one polylines intersect with each other). Thus, at the *GIS dimension instance* level we will have *rollup relations* (denoted $r_L^{geom1 \rightarrow geom2}$). For instance, $r_{L_{city}}^{Point \rightarrow Pg}(x, y, pg_1)$ says that, in a layer L_{city} a point (x,y) corresponds to a polygon identified by pg_1 in the Geometric part. In spite of this, the authors propose a mechanism to

precompute the overlaid layers in the map, that turns these relations back into rollup function, i.e., where a point (x,y) will correspond to *exactly* one geometry identifier. Finally, there is the *OLAP part* for storing non-spatial data. This part contains the conventional OLAP structures, as defined in (Hurtado, Mendelzon, & Vaisman, 1999). The levels in the geometric part are associated to the OLAP part via a function, denoted $\alpha_{L,D}^{dimLevel \rightarrow geom}$. For instance, $\alpha_{L_r, Rivers}^{riverId \rightarrow g_r}$ associates information about a river in the OLAP part (*riverId*) in a dimension *Rivers*, to the identifier of a polyline (g_r) in a layer denoted L_r , which represents rivers in the Geometric part.

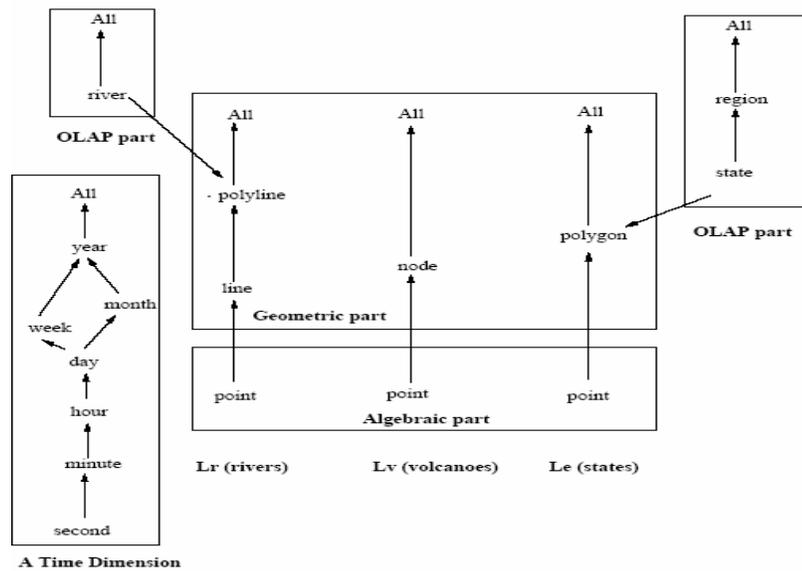


Figure 4. An example of a *GIS* dimension Schema

Example 1. Figure 4 shows a *GIS* dimension schema, where we defined three layers, for rivers, volcanoes, and states, respectively. The schema is composed of three graphs; the graph for rivers, for instance, contains edges saying that a point (x,y) in the algebraic part relates to line identifiers in the geometric part, and that in the same portion of the dimension, lines relate to polyline identifiers. In the OLAP part we have two dimensions, representing districts and rivers, associated to the corresponding graphs, as the figure shows. For example, a river identifier at the bottom layer of the dimension representing rivers in the OLAP part, is mapped to the polyline level in the geometric part in the graph representing the structure of the rivers layer.

Figure 5 shows a portion of a GIS dimension instance for the rivers layer L_r in the dimension schema of Figure 4. We can see that an instance of a GIS dimension in the OLAP part is associated (via the function) to the polyline pl_1 which corresponds to the Colorado river. For clarity, we only show four different points at the *point* level $(x_1, y_1) \dots (x_4, y_4)$. There is a relation $r_{L_r}^{point, line}$ containing the association of points to the lines in the *line* level. Analogously, there is also a relation $r_{L_r}^{line, polyline}$ between the line and polyline levels, in the same layer. \square

Time in the OLAP part will be represented by a *Time* dimension (actually, there could be more than one Time dimension, supporting, for example, different notions of time). As it is well-known in OLAP, this dimension can have different configurations that depend on the application at hand.

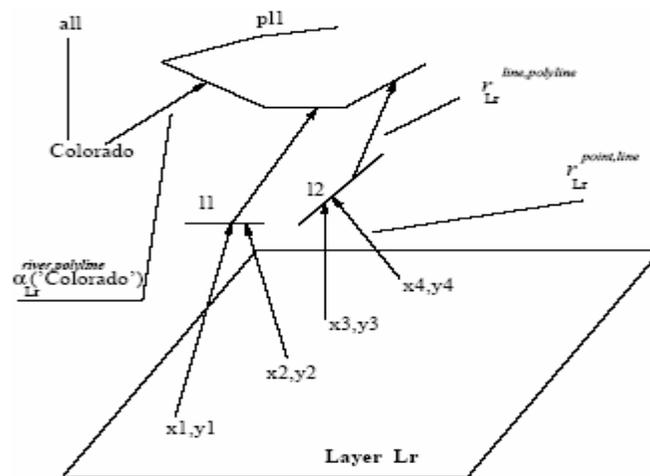


Figure 5. A GIS dimension instance for Figure 4.

Measures and Facts

A key point in the Piet model is the way it accounts for measures and fact tables. Most of the proposals discussed above consider *spatial* measures, and apply OLAP operators over them. Piet is capable of working in this way, operating over the GIS dimensions (the authors define the concept of *spatial aggregation* for this), but also of using facts defined in the OLAP part, to support spatial DSS queries, like the ones commented in the introductory section of this paper. Thus, elements in the geometric part are associated with *facts*, each fact being

quantified by one or more *measures*, not necessarily a numeric value. The following example gives the intuition of a so-called GIS fact table. For details, we refer the reader to (Gómez et al., 2007).

Example 2. Consider a fact table containing state populations. Also assume that this information will be stored at the polygon level. In this case, the fact table schema would be $(polyId, L_s, population)$ where $polyId$ is the polygon identifier, L_s represents the states layer, and population is the measure. If information about, for example, temperature data, is stored at the *point* level, we would have a base fact table with schema $(point, L_e, temperature)$, with instances of the form $(x_1, y_1, L_e, 25)$. Note that temporal information could be also stored in these fact tables, by simply adding the *time* dimension to the fact table. This would allow storing temperature information across time. \square

Example 2 shows that, basically, a GIS fact table is a standard OLAP fact table where one of the dimensions is composed of geometric objects in a layer. Classical fact tables in the OLAP part, defined in terms of the OLAP dimension schemas can also exist. For instance, instead of storing the population associated to a polygon identifier, this information may reside in a data warehouse, with schema $(state, population)$.

Geometric Aggregation

Based on the data model described above, the notion of *geometric aggregation* was defined. However, in general, geometric aggregation queries are hard to evaluate, because they require the computation of a double integral representing the area where some condition is satisfied. Thus, Piet addresses a class of queries denoted *summable*, of the form: $\sum_{g \in S} h(g)$, where h is a function (represented, for instance, by a fact table), and the sum is performed over all the identifiers of the objects that satisfy a condition. For example, the query “total population of the cities crossed by the Colorado River would read:

$$Q \equiv \sum_{gid \in C} ft_{pop}(gid, L_c).$$

$$C = \{gid \mid (\exists x)(\exists y)(\exists pl_1)(\exists c \in dom(Ci))$$

$$(\alpha_{L_r, Rivers}^{Ri \rightarrow Pl}('Colorado') = (pl_1) \wedge r_{L_r}^{Pt \rightarrow Pl}(x, y, pl_1) \wedge$$

$$\alpha_{L_c, Districts}^{Ci \rightarrow Pg}(c) = gid \wedge r_{L_c}^{Pt \rightarrow Pg}(x, y, gid))\}.$$

The meaning of the query is: $\alpha_{L_r, Rivers}^{Ri \rightarrow Pl}$ ('Colorado') maps the identifier of the Colorado river to a polyline in layer L_r (representing rivers). The relation $r_{L_r}^{Pt \rightarrow Pl}(x, y, pl_1)$ contains the mapping between the points and the polylines representing the rivers that satisfy the condition. The other functions are analogous. Thus, the identifiers of the geometric elements that satisfy both conditions can be retrieved, and the sum of ft_{pop} (which represents the population associated to a polygon g_{id}) over these objects can be performed.

Discussion

Piet supports four kinds of queries: (a) *Pure Geometric Queries*, like “Districts crossed by at least one river”; (b) *Geometric Aggregation Queries*, like “List each region, along with the total number of rivers that crossed it”, or “For each region show the total length of the part of rivers which intersects it, only for regions with at least an area under cereal cultivation equal or higher than 1000 Km^2 ”; (c) *Geometric Aggregation Queries limited to a query region*, like “List each region with the total number of rivers that crossed it, considering only the part of the river that lies within the query region”; (d) *GISOLAP Queries* which integrate GIS and OLAP in a very natural way. A query of this kind is, for example, “Unit Sales, Store Cost and Store Sales for products and promotion media offered by stores only in provinces crossed by rivers”. These queries can be expressed in the *Piet-QL* language (Gómez, Vaisman, & Zich, 2008). Piet-QL also allows to place constraints over a data cube, including pre-aggregated facts into the WHERE clause. (A functional demo of Piet-QL, and some example queries can be found at <http://piet.exp.dc.uba.ar/pietql>). A typical example of a Piet-QL query is:

“Names of cities in provinces crossed by the Dyle river, in Belgium, such that the cities had sales greater than 5000 units.”

Variables in Piet-QL range over elements in the thematic layers. Thus, in the **FROM** clause below, the expression `bel_city lc1` means that `lc1` will be instantiated with all the polygons representing cities in the layer `bel_city`.

```

SELECT GIS lc1.name
FROM bel_city lc1, bel_prov lp2, bel_river lr2
WHERE contains(lp2,lc1) AND
intersects(lp2,lr2) AND lr2.name="Dyle"
AND lc1 IN(
    SELECT CUBE
    filter([Store].[Store City].Members,
    [Measures].[Unit Sales]>5000)
    FROM [Sales])
AND lp2 IN(
    SELECT CUBE
    filter([Store].[Store City].Members,
    [Measures].[Unit Sales]>0)
    FROM [Sales]))

```

Piet-QL supports the following kinds of queries: (a) pure GIS queries; (b) pure OLAP queries; (c) GIS queries filtered with aggregation (i.e., filtered using a data cube); (d) OLAP queries filtered using a geometric or geographic condition. The query above corresponds to class (c).

If we consider the classification proposed by Pelekis et.al., (2004), attribute, point, range, distance-based, nearest neighbor and topological queries are supported by Piet-QL (i.e., geometric queries). Note that these queries could be used to build the other ones, that include aggregation and OLAP capabilities.

Overlay Pre-computation in Piet. Many interesting queries in GIS require computing intersections, unions, etc., of objects that are in different layers. Hereto, their overlay has to be computed. For the summable queries defined above, on-the-fly computation of the sets "C" containing all those cities in the example, would be costly, mainly because most of the time we will need to go down to the Algebraic part of the system, and compute the intersection between the geometries (e.g., states and rivers, cities and airports). In addition to the typical R-tree-based techniques commented in previous sections, Piet implements a different strategy for materialization, consisting in three steps: (a) partitioning each layer in sub-geometries, according to the carrier lines defined by the geometries in each layer (see below); this allows detecting which geographic regions are common to the layers

involved; (b) pre-computing the overlay operation; (c) evaluating the queries using the layer containing all the pre-computed sub-geometries

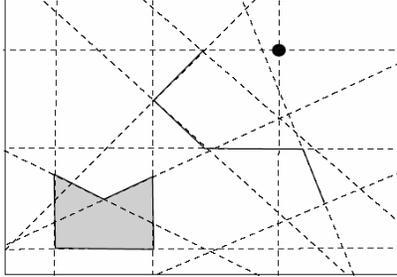


Figure 6. The carrier sets of a point, a polyline and a polygon are the dotted lines.

The carrier set of a layer induces a partition of the plane into open convex polygons, open line segments and points. Thus, the rollup relations r will turn into functions (given that no two points can map to the same open convex polygon). Given CL and a bounding box, we denote the *convex polygonization of L* , the set of open convex polygons, open line segments and points, induced by CL , that are strictly inside the bounding box. Given two layers L_1 and L_2 , and their carrier sets C_{L_1} and C_{L_2} , the *common sub-polygonization of L_1 according to L_2* , denoted $CSP(L_1, L_2)$ is a refinement of the convex polygonization of L_1 , computed by partitioning each open convex polygon and each open line segment in it along the carriers of C_{L_2} .

This can be generalized for more than two layers. Figure 6 illustrates the carrier sets of a point, a polyline and a polygon.

Experimental evaluation showed that overlay pre-computation (i.e., pre-computing the common sub-polygonization) in general can perform better than R-trees, and also be competitive with aR-trees, except when the query region must be computed in running time, because computing the intersection between the query region and the common sub-polygonization, turns out to be expensive in some situations (Escribano et al., 2007).

EXTENDING THE PIET FRAMEWORK TO TRAJECTORY DATA

Moving objects are integrated in the framework presented above, by means of a distinguished fact table that we denote *Moving Object Fact Table* (MOFT).

Let us introduce an example. Figure 7 (left) shows a (very) simplified map of Paris, containing two hotels, denoted Hotel 1 and Hotel 2 (H1 and H2 from here on), the Louvre and the Eiffel tower. We consider three moving objects, O1, O2 and O3. Object O1 goes from H1 to the Louvre, the Eiffel tower, spends just a few minutes there, and returns to the hotel. Object O2 goes from H2 to the Louvre, the Eiffel tower, (spending a couple of hours visiting each place), and returns to the hotel. Object O3 leaves H2 to the Eiffel tower, visits the place, and returns to H2. Figure 7 (center) shows part of these trajectory samples. All points of the same trajectory are temporally ordered and stored together (i.e., the raw trajectories table is sorted by O_{id} and t). In what follows, we will use the object identifier as the trajectory identifier, unless specified, although it is usual to generate a trajectory identifier in a pre-processing step. This is motivated by the fact that, in general, trajectories are given as continuous data streams, that need to be partitioned according to a certain criteria, for example, when a minimum amount of time without movement occurs. In that case, O_{id} will not be a trajectory identifier any longer.

In this scenario, for instance, a GIS user may be interested in queries like “number of persons going from H1 to the Louvre and then to the Eiffel tower (stopping to visit both places) in the same day”. Also, a data mining analyst may want to identify interesting patterns in the trajectory data using association rule mining or sequential patterns algorithms, like “people do not visit two museums in the same day”. Complex queries that aggregate non-spatial information, and also involve GIS and moving object data, must also be addressed. For instance, “total sales in museum shops, for museums located on the left bank of the Seine, such that people visit them before going to the Eiffel Tower in the same day”.

A moving object fact table (MOFT for short, see the table in the center of Figure 7), contains a finite number of identified trajectories.

Definition 1 formalizes this.

Definition 1 (Moving Object Fact Table) Given a finite set T of trajectories, a *Moving Object Fact Table* (MOFT) for T is a relation with schema $\langle O_{id}, T, X, Y \rangle$, where O_{id} is the identifier of the moving object, T represents time instants, and X and Y represent the spatial coordinates of the objects. An instance M of the above schema contains a finite number of tuples of the form $\langle O_{id}, t, x, y \rangle$ that represent the position (x, y) of the object O_{id} at instant t , for the trajectories in T . \square

In practice, the MOFTs can contain huge amounts of data. For instance, suppose a GPS takes observations of daily movements of one thousand people, every ten seconds, during one month. This gives a MOFT of $1000 \cdot 360 \cdot 24 \cdot 30 = 259,200,000$ records. In this scenario, querying trajectory data may become extremely expensive. Note that a MOFT only provides the position of objects at a given instant. Sometimes we are not interested in such level of detail, but we look for more aggregated information instead. For example, we may want to know how many people go from a hotel to a museum on weekdays. Or, we can even want to perform data mining tasks like inferring trajectory patterns that are hidden in the MOFT. These tasks require semantic information, not present in the MOFT. In the best case, obtaining this information from that table will be expensive, because it would imply a join between this table and the spatial data.

As we commented above, the notion of stops and moves was recently introduced. Intuitively, if a moving object spends a sufficient amount of time in a certain geographic place (which we denote a *place of interest of an application*, PoI for short), this place is considered a *stop* of the object's trajectory. In-between stops, a trajectory has *moves*. Gómez, Kuijpers, & Vaisman (2008b) present an in-depth study on how moving object data analysis can benefit from replacing raw trajectory data by a sequence of *stops and moves*. The authors propose to use the notion of *stops and moves* in order to obtain a concise MOFT, that can represent the trajectory in terms of places of interest, characterized as *stops*. This table cannot replace the whole information provided by the MOFT, but allows to quickly obtain information of interest without accessing the complete data set. In this sense, this concise MOFT, which we will denote SM-MOFT, behaves like a summarized materialized view of the MOFT. The SM-MOFT will contain the object identifier, the identifier of the geometries representing the Stops, and the interval $[t_s, t_f]$ of the stop duration. Obviously, we do not need to store the information about the moves,

which remains implicit, because we know that between two stops there could only be a move. Definition 2 formalizes the above.

Definition 2 (SM-MOFT) Let the set $\mathcal{P}_A = \{C_1 = (R_{C_1}, \Delta_{C_1}), \dots, C_N = (R_{C_N}, \Delta_{C_N})\}$ be the PoIs of an application, and let M be a MOFT. The *SM-MOFT* M^{sm} of M with respect to \mathcal{P}_A consists of the tuples $(O_{id}, g_{id}, t_s, t_f)$ such that (a) O_{id} is the identifier of a trajectory in M ; (b) g_{id} is the identifier of the geometry of a PoI $C_k = (R_{C_k}, \Delta_{C_k})$ of \mathcal{P}_A , such that the trajectory with identifier O_{id} in M has a stop in this PoI during the time interval $[t_s, t_f]$. This interval is called the *stop interval* of this stop. \square

The table in Figure 7 (right) shows the SM-MOFT for our example of the beginning of this section.

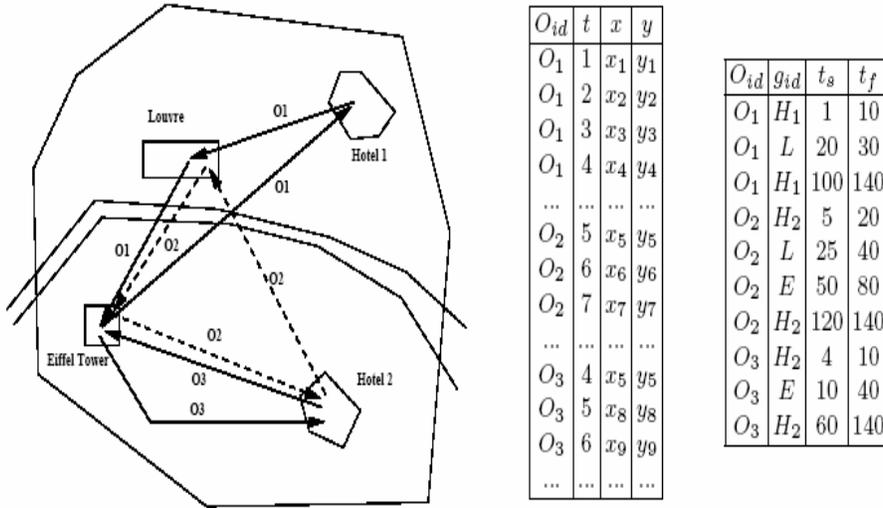


Figure 7. Three trajectories (left), the MOFT (center), and the SM-MOFT (right)

Spatio-Temporal Aggregation in Piet

The approach for spatio-temporal aggregation in Piet, differs from other proposals we discussed in this paper. Gómez et al. (2008b) define

a query language, denoted RE-SPaM, based on regular expressions, and aimed at obtaining sequential patterns in trajectory databases. Aggregation is performed on top of this language, applying aggregate operators to the sequences that are in the query result. Association rule analysis is also supported by this approach. Four key features characterize RE-SPaM:

1. Items are not only composed of identifiers, but are also *complex objects, composed of attributes that can be organized in hierarchies*. This allows adding *OLAP capabilities to the language* in a very natural way.
2. The support of rollup functions allows performing *mining at different levels of aggregation*. Thus, complex sequential patterns can be found, at different granularity levels.
3. It can be proved that RE-SPaM is actually a subset of the first-order language introduced in Section “Geometric Aggregation” extended to support moving objects. We denoted this language L_{mo} .
4. As a consequence of the above, not only semantic trajectories are supported, but also, if necessary, one can *go back to the base data, in order to support any kind of queries*, for instance, most of the ten queries in the benchmark proposed by Theodoridis (2003). In fact, aggregation is not considered in such benchmark.

We do not give the formal definition of the language, but we give the idea through a couple of examples.

We begin with a query not including aggregation, using only semantic trajectories (i.e., the SM-MOFT): “Trajectories going from a hotel to a tourist attraction, stopping at the latter, and ending at a hotel (maybe it had stopped at several other places)”. The query simply reads in RE-SPaM:

H.T.? .H.

Adding conditions over the labels of the elements in the language, like in: “Trajectories going from the Eiffel tower to the Hilton hotel”, we would have:

T[name=“Eiffel”].H[name=“Hilton”].

The conditions could be extended with functions (e.g., rollup functions, topological functions), and variables. Basically, the language is defined as follows.

Definition 3 (R.E. for Stops and Moves) A regular expression on stops and moves, denoted RE-SPaM is an expression generated by the grammar

$$E \quad dim|dim[cond]|(E)^*|(E)^+|E.E| |?$$

where $dim \ D$ (a set of dimension names in the OLAP part), ϵ is the symbol representing the empty expression, “.” means concatenation, and $cond$ represents a condition that can be expressed in L_{mo} . The term “?” is a wildcard meaning “any sequence of any number of dim ”. \square

The semantics of the language is the following: for each trajectory T in an SM-MOFT such that there is a sub-trajectory of T that matches the expression, the query returns the O_{id} of T . Aggregate functions can be applied over this result. An example including aggregation is the query: “Total number of trajectories from a Hilton hotel to a tourist attraction, stopping at a museum,” which reads in RE-SPaM:

$$COUNT(H[name="Hilton"].? .M.? .T)$$

As another example, including a rollup function, the query “Total number of trajectories that went from a Hilton hotel to the Louvre, in the morning” is expressed in RE-SPaM:

$$COUNT(H[name = "Hilton"].?.M[name = "Louvre" \wedge \int_{Time}^{timeId \rightarrow TimeOfDay}(t_s) = "morning"])$$

In these queries, the conditions are evaluated over the *current* nodes (the node the parser is currently evaluating). Also, t_s is a special variable representing the starting point of the time interval of the node of the automaton -see (Gómez, Kuijpers, & Vaisman, 2008b) for details), that is being visited when evaluating the expression. The next query illustrates the full power of the language, since it includes geometric and temporal conditions that show how all elements in the model interact. Note that in the query, the SM-MOFT is not enough, and we need to go to the geometry. However, for many useful queries and patterns, much simpler expressions will suffice. The query is:

“Total number of trajectories going from a tourist attraction to a museum in the 19th district of Paris in the morning,” and in RE-SPaM reads:

$$\begin{aligned} & \text{COUNT}(T.?.M[f_{Time}^{timeId \rightarrow TimeOfDay}(I.t_s) = \text{“morning”} \wedge \\ & (\exists g_{id}) (\exists x) (\exists y) (\exists O_{id}) (\exists (t_1)) (\exists p) (\exists pg) (\exists d) (M(O_{id}, t_1, x, y) \wedge \\ & \alpha_{L_p, Museum}^{m_{id} \rightarrow Pg}(p) = g_{id} \wedge r_{L_p}^{point \rightarrow Pg}(x, y, g_{id}) \wedge \alpha_{L_d, Distr}^{distr \rightarrow Pg}(d) = pg \wedge \\ & pg.number = 19 \wedge f_{L_{dist}}^{point \rightarrow Pg}(x, y) = pg]) \end{aligned}$$

Let us explain this expression. The function $\alpha_{L_p, Museum}^{m_{id} \rightarrow Pg}(p) = g_{id}$, maps the id of the PoI (i.e., a museum) in the extension of the *current* node (p), to the polygon representing it in the geographic part (g_{id}). The rollout $r_{L_p}^{point \rightarrow Pg}(x, y, g_{id})$ identifies the x, y coordinates corresponding to g_{id} . The function $\alpha_{L_d, Distr}^{distr \rightarrow Pg}(d) = pg$ has the meaning already explained, i.e., it maps a district identifier d in the *Distr* dimension to a polygon identifier in layer L_d . The equality $f_{L_{dist}}^{point, Pg}(x, y) = pg$ checks that the point of the trajectory belongs to the 19th district. M is the MOFT containing the trajectory samples.

CONCLUSION: DISCUSSING THE PROPOSALS

We will focus on comparing the two proposals (introduced in the context of the *GeoPKDD* project) that, in our opinion, more comprehensively address the issue of spatio-temporal OLAP: *Hermes/TDW*, and *Piet*. The other proposals discussed in sections “Spatial Data Warehousing and OLAP” and “Spatio-Temporal Data Warehousing, OLAP and Mining” address different parts of the problem, but no spatio-temporal OLAP as a whole. We show below that, even though there exists some degree of overlapping, both approaches tackle different parts of the SOLAP problem. We remark that the analysis will be performed in terms of the capabilities to fulfill SOLAP requirements.

Hermes does not specifically address *SOLAP support*. It is left open (although not explicitly stated) as a possible application of the general framework, but no formal model supports spatial data aggregation, because *Hermes* has not been designed as a model for *spatio-temporal decision support*. On the other hand, *Piet* focuses on GIS-OLAP integration, and is oriented specifically toward *aggregate queries* and

spatial decision-support, although, as showed, standard spatial queries are also supported.

Integrating geographic data and warehouse data is not built into the Hermes model, while Piet handles this integration through the “ α ” function. As of the moment when this paper is being written, this binding must be performed “manually”, and a tool for automatically matching geometric elements in the GIS layers to non-spatial objects in the warehouse, is being implemented for Piet. On the Hermes side, integration of an external warehouse would require defining, in an ad-hoc fashion, how geographic objects will be mapped to warehouse objects.

Being conceived as a SOLAP system, the Piet formal data model also integrates naturally into the SOLAP framework the problem of modeling and analyzing trajectory data, either using the whole trajectory data (i.e., the MOFT), or the semantic trajectory represented via the SM-MOFT.

Probably the strongest feature of the Hermes/TDW proposal is the analysis and implementations of the ETL process for *trajectory data* analysis. On the other hand, Piet does not have similar automatic data loading machinery, and assumes that data has already been loaded into a trajectory file.

In the TDW approach, *only aggregate measures* are loaded into a fact table, and dimensions conform cells in a three-dimensional space (x,y,t) . The main achievement, in this sense, is the treatment of *double counting* for some of the measures. Trajectory data is stored in the moving object database (MOD), and it is used to extract higher level knowledge that may also be used to feed the TDW. Therefore, the TDW could be considered an application, based on on the traditional star schema, developed over the underlying architecture. Instead, in the Piet/RE-SPaM (Piet’s regular query language for trajectories) approach, the MOFT and SM-MOFT *do not store aggregated measures* (actually, the “facts” here are represented by the existence of the trajectory in the database, in a sense, a kind of boolean measure), but just the base trajectories (or the “semantic” trajectories, in the SM-MOFT). In fact, the MOFT is, basically, the *RELTrajectories* table in the TDW approach, which suggests that both approaches may complement each other in this sense. Aggregation over the ‘cells’ hierarchy could be supported by RE-SPaM, although the language is

mainly oriented to trajectory pattern mining. Further, aggregation is performed over trajectories that satisfy a certain pattern - see (Gómez, Kuijpers, & Vaisman, 2008a) for details on the different aggregate operators and their arguments. In summary, implementing aggregation over cells in Piet, in the way proposed in GeoPKDD would not be trivial.

The Hermes-MDC framework supports moving objects that can change shape or position over time, while *Piet assumes that the regions and geometric objects, in general, are static*, and that traceable objects (e.g., representing pedestrians, buses, cars) move through the geographic space. In other words, Piet does not provide temporal support for the GIS part of the model, only for the moving objects whose trajectories are being analyzed.

Finally, all Piet software components are open source: the database, postgres, and its GIS extension posGIS (<http://postgis.refrations.net>), the Mondrian OLAP server (<http://www.mondrian.sourceforge.net>), and Java. On the other hand, Hermes is built as an extension of Oracle 10g.

Table 1 summarizes the similarities and differences between Hermes, the TDW, and Piet/ RE-SPaM.

Acknowledgements. This research has been partially funded by the European Union under the FP6-IST-FET program, Project n. FP6-14915, “GeoPKDD: Geographic Privacy-Aware Knowledge Discovery and Delivery”, (www.geopkdd.eu) and by the Research Foundation Flanders (FWO-Vlaanderen), Research Project G.0344.05, and the Argentinian National Scientific Agency, project PICT 2004 11-21.350.

	Hermes	Trajectory DW	Piet	RE-SPaM
GIS-OLAP integration	No	Through Hermes-MDC	Yes	Yes
SOLAP Formal model	No	N/A	Yes	N/A
Fact table	N/A (can be defined ad-hoc)	Pre-aggregated measures	External, defined in the OLAP part	MOFT,SM MOFT (no pre-aggregation)
Query Language	PL-SQL	PL-SQL	Piet-QL	RE-SPaM
Support for spatial aggregation	Ad-hoc	Yes	Yes	Yes
Support for querying an external DW	Ad-hoc	Ad-hoc	Built-in	Built-in
Spatial Queries	Point,Range, distance, nearest-neighbor	Point,Range, distance, nearest-neighbor	Point,Range, distance, nearest-neighbor	Point,Range, distance, nearest-neighbor
Mining capabilities	No	Through external functionality	No	Yes
Roll-up and drill-down over spatial objects	No	Yes	Through Piet-QL	No
Support of changing objects	Yes	Through Hermes-MDC	No	No
Temporal support	Yes	Through Hermes-MDC	No	Only for moving points
ETL Support and tools	N/A	Yes	No	No
Semantic trajectory support	N/A	Through external functionality	N/A	Built-in
Open Source Architecture	No	No	Yes	Yes
Needs non-standard data libraries for querying?	Yes	Yes	No	No

Table 1. Comparing Hermes, TDW, Piet and RE-SpaM.

REFERENCES

- Alvares, L. O., Bogorny, V., Kuijpers, B., Macedo, J. A. F. de, Moelans, B., & Vaisman, A. (2007). A model for enriching trajectories with semantic geographical information. In *ACM-GIS 2007*.
- Bédard, Y., Merret, T., & Han, J. (2001). Fundamentals of spatial data warehousing for geographic knowledge discovery. In *Geographic data mining and knowledge discovery* (p. 53 – 73). Taylor & Francis.
- Bédard, Y., Rivest, S., & Proulx, M.-J. (2007). Spatial online analytical processing (SOLAP): Concepts, architectures, and solutions from a geomatics engineering perspective. In *Data warehouses and OLAP: Concepts, architectures and solutions* (p. 298 – 319). IGI Global.
- Bimonte, S., Tchounikine, A., & Miquel, M. (2005). Towards a spatial multidimensional model. In *DOLAP* (p. 39-46).
- Brakatsoulas, S., Pfoser, D., & Tryfona, N. (2004). Pre-aggregation in spatial data warehouses. In *Proceedings of IDEAS'04* (p. 68-77). Washington D.C, USA.
- Buchmann, A. P., Günther, O., Smith, T. R., & Wang, Y.-F. (1990). Design and implementation of large spatial databases. In *First symposium (SSD)* (Vol. 409). Springer.
- Cabibbo, L., & Torlone, R. (1997). Querying multidimensional databases. In *Database programming languages* (Vol. 1369, p. 319–335). Springer.
- Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6), 377–387.
- Damiani, M., & Spaccapietra, S. (2006). Spatial data warehouse modelling. In *Processing and managing complex data for decision support* (p. 21 – 27). Idea Group.
- Damiani, M. L., Macedo, J. A. F. de, Parent, C., Porto, F., & Spaccapietra, S. (2007). A conceptual view of trajectories. *Technical Report, Ecole Polytechnique Federal de Lausanne, April 2007*.

Damiani, M. L., Vangenot, C., Frentzos, E., Marketos, G., Theodoridis, Y., Veryklos, V., Raffaeta, A. (2007). Design of the trajectory warehouse architecture. *Technical Report D1.3, GeoPKDD project*.

Eder, J., Koncilia, C., & Morzy, T. (2002). The COMET metamodel for temporal data warehouses. In *Caise* (p. 83-99).

Escribano, A., Gomez, L., Kuijpers, B., & Vaisman, A. A. (2007). Piet: a GIS-OLAO implementation. In *ACM 10th international workshop on data warehousing and OLAP (DOLAP)* (p. 73–80). ACM.

Etzion, O., Jajodia, S., & Sripada, S. M. (Eds.). (1998). *Temporal databases: Research and practice*. Springer.

Frentzos, E., Gratsias, K., & Theodoridis, Y. (2007). Index-based most similar trajectory search. In *ICDE* (p. 816-825).

Giannotti, F., Nanni, M., Pinelli, F., & Pedreschi, D. (2007). Trajectory pattern mining. In *KDD* (p. 330-339). San Jose, California, USA.

Gómez, L., Haesevoets, S., Kuijpers, B., & Vaisman, A. A. (2007). Spatial aggregation: Data model and implementation. *CoRR*, *abs/0707.4304*.

Gómez, L., Kuijpers, B., & Vaisman, A. A. (2008a). Aggregation languages for moving object and places of interest. In *SAC 2008 - ASIIS track*.

Gómez, L., Kuijpers, B., & Vaisman, A. A. (2008b). Querying and mining trajectory databases using places of interest. *Annals of Information Systems (in press)*.

Gómez, L., Vaisman, A., & Zich, S. (2008). Piet-QL: a query language for GIS-OLAP integration. In *ACM/GIS 2008 (to appear)*.

Gonzalez, H., Han, J., Li, X., Myslinska, M., & Sondag, J. P. (2007). Adaptive fastest path computation on a road network: A traffic mining approach. In *VLDB* (p. 794-805).

Güting, R. H., & Schneider, M. (2005). *Moving objects databases*. Morgan Kaufman.

Güting, R. H., Böhlen, M., Jensen, C., Lorentzos, N., Schneider, M. & Vazirgiannis, M. (2000). A foundation for representing and querying moving objects. In *ACM Transactions Database Systems*, 25(1), (p. 1-42

Gutman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proceedings of SIGMOD'84* (p. 47-57).

Han, J., Stefanovic, N., & Koperski, K. (1998). Selective materialization: An efficient method for spatial data cube construction. In *Research and development in knowledge discovery and data mining (PAKDD)* (Vol. 1394, p. 144-158). Springer.

Harinarayan, V., Rajaraman, A., & Ullman, J. D. (1996). Implementing data cubes efficiently. In *SIGMOD conference* (p. 205–216). ACM Press.

Hornsby, K., & Egenhofer, M. J. (2002). Modeling moving objects over multiple granularities. *Ann. Math. Artif. Intell.*, 36(1-2), 177-194.

Hurtado, C., Mendelzon, A., & Vaisman, A. (1999). Maintaining data cubes under dimension updates. In *Proceedings of ICDE* (p. 346-355).

Jensen, C. S., Kligys, A., Pedersen, T. B., & Timko, I. (2004). Multidimensional data modeling for location-based services. *VLDB*, 13(1), 1–21.

Kakoudakis, I. (1996). The TAU Temporal Object Model. *M.Ph. Thesis*. UMIST, Department of Computation.

Kimball, R. (1996). *The data warehouse toolkit*. J.Wiley and Sons, Inc.

Kimball, R., & Ross, M. (2002). *The data warehouse toolkit: The complete guide to dimensional modeling, 2nd. ed.* J.Wiley and Sons, Inc.

Kuijpers, B., & Vaisman, A. (2007). A data model for moving objects supporting aggregation. In *Proceedings of the first international workshop on spatio-temporal data mining (STDM)*. Istanbul, Turkey.

Lee, J.-G., Han, J., & Whang, K.-Y. (2007). Trajectory clustering: a partition-and-group framework. In *SIGMOD conference* (p. 593-604).

- Lenz, H., & Shoshani, A. (1997). Summarizability in olap and statistical data bases. In *Ninth international conference on scientific and statistical database management, proceedings, august 11-13, 1997, olympia, washington, usa* (p. 132-143). IEEE Computer Society.
- López, I. F. V., Snodgrass, R., & Moon, B. (2005). Spatiotemporal aggregate computation: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 17(2), 271–286.
- Malinowski, E., & Zimányi, E. (2004). Representing spatiality in a conceptual multidimensional model. In *GIS* (p. 12-22).
- Malinowski, E., & Zimányi, E. (2006). Hierarchies in a multidimensional model: From conceptual modeling to logical representation. *Data Knowledge Engineering*, 59(2), (p.348–377).
- Marketos, G., Frentzos, E., Ntousi, I., Pelekis, N., Raffaeta, A., & Theodoridis, Y. (2008). Building real-world trajectory data warehouses. In *MobiDE'08* (in press).
- Mendelzon, A. O., & Vaisman, A. A. (2000). Temporal queries in OLAP. In *VLDB* (p. 242-253).
- Mendelzon, A. O., & Vaisman, A. A. (2003). Time in multidimensional databases. In *Multidimensional databases* (p. 166-199).
- Meratnia, N., & By, R. A. de. (2002). Aggregation and comparison of trajectories. In *ACM-GIS* (p. 49-54).
- Mouza, C., & Rigaux, P. (2005). Mobility patterns. *Geoinformatica*, 9(23), 297-319.
- Orlando, S., Orsini, R., Raffaetà, A., Roncato, A., & Silvestri, C. (2007). Spatio-temporal aggregations in trajectory data warehouses. In *DaWaK* (p. 66-77).
- Papadias, D., Tao, Y., Kalnis, P., & Zhang, J. (2002). Indexing spatio-temporal data warehouses. In *International conference on data engineering (ICDE)* (p. 166–175). IEEE Computer Society.

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

Paredaens, J., Bussche, J. Van Den, & Gucht, D. V. (1994). Towards a theory of spatial database queries. In *Thirteenth ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, (PODS)* (p. 279–288). ACM Press.

Paredaens, J., Kuper, G., & Libkin, L. (Eds.). (2000). *Constraint databases*. Springer-Verlag.

Pedersen, T. B., & Tryfona, N. (2001). Pre-aggregation in spatial data warehouses. In *Advances in spatial and temporal databases (SSTD)* (p. 460–480).

Pelekis, N. (2002). Stau: A spatio-temporal extension to ORACLE DBMS. *Ph.D Thesis, UMIST Department of Computation*.

Pelekis, N., Theodoulidis, B., Kopanakis, Y., & Theodoridis, Y. (2004). Literature Review of Spatio-Temporal Database Models. In *The Knowledge Engineering Review journal*, 19(3) (p. 235-274).

Pelekis, N., & Theodoridis, Y. (2006). Boosting location-based services with a moving object database engine. In *Mobide* (p. 3-10).

Pelekis, N., Theodoridis, Y., Vosinakis, S., & Panayiotopoulos, T. (2006). Hermes - a framework for location-based data management. In *EDBT* (p. 1130-1134).

Pelekis, N., Kopanakis, I., Ntoutsis, ., Marketos, G., Andrienko, G., & Theodoridis, Y. (2007). Similarity Search in Trajectory Databases. In *Proceedings of the 14th IEEE International Symposium on Temporal Representation and Reasoning (TIME)* (p. 129-140) ACM Press.

Pelekis, N., Kopanakis, I., Ntoutsis, ., Marketos, G., & Theodoridis, Y. (2007). Mining Trajectory Databases via a Suite of Distance Operators. In *Proceedings of the IEEE/ ICDE workshop on Spatio-Temporal Data Mining (STDM)* (p. 575-584)

- Pourabbas, E. (2003). Cooperation with geographic databases. In *Multidimensional databases: Problems and solutions* (p. 393–432). Idea group.
- Rao, F., Zhang, L., Yu, X., Li, Y., & Chen, Y. (2003). Spatial hierarchy and OLAP-favored search in spatial data warehouse. In *ACM sixth international workshop on data warehousing and OLAP (DOLAP)* (p. 48–55). ACM.
- Rigaux, P., Scholl, M., & Voisard, A. (2001). *Spatial databases: With application to GIS*. Morgan Kaufmann.
- Rivest, S., Bédard, Y., & Marchand, P. (2001). Towards better support for spatial decision making: Defining the characteristics of spatial on-line analytical processing (SOLAP). *Geomatica*, 55(4), 539–555.
- Rizzi, S., & Golfarelli, M. (2000). Date warehouse design. In *ICEIS* (p. IS 39-42).
- Shekhar, S., Lu, C.-T., Tan, X., Chawla, S., & Vatsavai, R. R. (2001). Map cube: A visualization tool for spatial data warehouses. In *Geographic data mining and knowledge discovery* (p. 73 – 108). Taylor and Francis.
- Snodgrass, R. T. (Ed.). (1995). *The TSQL2 temporal query language*. Kluwer.
- Snodgrass, R. T. (Ed.). (1999). *Developing time-oriented database applications in SQL*. Morgan Kaufmann.
- Stefanovic, N., Han, J., & Koperski, K. (2000). Object-based selective materialization for efficient implementation of spatial data cubes. *IEEE Trans. Knowl. Data Eng.*, 12(6), 938-958.
- Stonebraker, M., & Moore, D. (1996). *Object-relational DBMSs: The next great wave*. Morgan Kaufmann.
- Tao, Y., Kollios, G., Considine, J., Li, F., & Papadias, D. (2004). Spatio-temporal aggregation using sketches. In *ICDE* (p. 214-226).
- Theodoridis, Y. (2003). Ten benchmark database queries for location-based services. *Comput. J.*, 46(6), 713-725.

Vaisman, A. (2006a). Data quality-based requirements elicitation for decision support systems. In *Data warehouses and olap: Concepts, architectures and solutions, idea group*.

Vaisman, A. (2006b). Requirements elicitation for decision support systems: a data quality approach. In *Proceedings of ICEIS'06, volume 3*. Cyprus.

Vaisman, A., Izquierdo, A., Ktenas, M.(2006). Web-enabled Temporal OLAP. In *Proceedings of LA_WEB '06*, 220-229, Puebla, Mexico.

Weghe, N. V. de, Cohn, A., Tré, G. D., & Maeyer, P. D. (2005). A qualitative trajectory calculus as a basis for representing moving objects in geographical information systems. *Control and Cybernetics*

Wolfson, O., Sistla, P., Xu, B., & Chamberlain, S. (1999). Domino: Databases fOr MovINg Objects tracking. In *Proceedings of SIGMOD* (p. 547 - 549).

Worboys, M. F. (1995). *Gis: A computing perspective*. Taylor&Francis.

Zeiler, M. (1999). *Modeling our world: The ESRI guide to geodatabase design*. ESRI Press.

Zhang, L., Li, Y., Rao, F., Yu, X., Chen, Y., & Liu, D. (2003). An approach to enabling spatial OLAP by aggregating on spatial hierarchy. In *Data warehousing and knowledge discovery (DaWak)* (Vol. 2737, p. 35–44). Springer.