St-Toolkit: A Framework for Trajectory Data Warehousing

Simone Campora[1], Jose Antonio Fernandes de Macedo[2], Laura Spinsanti[3]

[1]LBD,EPFL Lausanne, CH
[2]LIA, UFC Fortaleza, BR
[3]LBD,EPFL Lausanne CH)
simone.campora@epfl.ch, jose.macedo@lia.ufc.br, laura.spinsanti@epfl.ch

ABSTRACT

Turning a collection of simple time-geography data into mobility knowledgeis a key issue in many research domains, such as social analysis and mobility investigation. Although collecting mobility data has become technologically feasible, turning these huge sets of data into mobility knowledge is still an open issue. Data warehousing is a well-established technique used for analysis of summarized data, but is not yet adequate to support spatio-temporal feature-sets. This paper proposes a framework for improving the design and the implementation of data warehouses that support spatio-temporal concepts on top of relational DBMS. This framework has been used for designing and implementing a trajectory data warehouse (TDW) for analyzing traffic data. Besides, we show that this framework also supports OLAP, SOLAP and STOLAP queries.

INTRODUCTION

Turning a collection of simple time-geography data into mobility knowledge is an open challenge. For instance, let's consider a traffic management scenario where a manager is asked to take some decisions based on the visualization of vehicles trajectories as showed in Figure 1. In this Figure, lines represent trajectories while points represent points of interest (e.g. restaurants, supermarkets, etc). We can conclude that the manager faces a complex task in extracting knowledge from this dataset, which can support his decisions. The challenges are mainly due to:

- The excess of data that is being projected on the screen;
- The lack of semantics for a given trajectory in relation to points of interest;

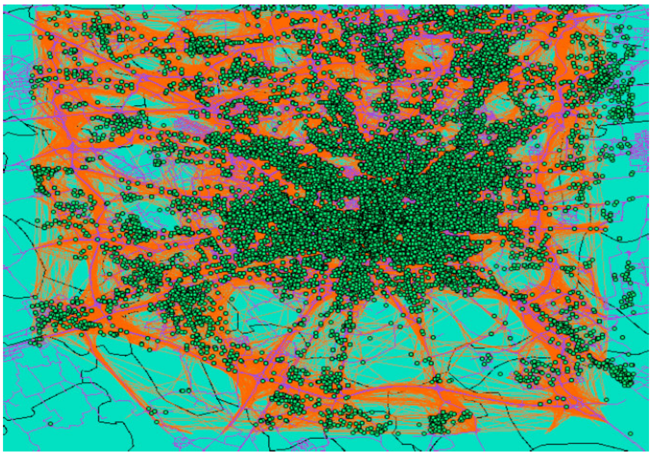


Figure 1:Trajectory and Point of Interest data

In addition to these problems, trajectory data is inherently spatio-temporal, requiring sophisticated methods to cope with spatial and temporal operations on large datasets. Therefore, analysis on this domain will require different analytical query capabilities, which can be classified as thematic, spatial, temporal and spatio-temporal. Here it follows some examples of such queries:

- Thematic: Retrieve the average travel time of December 2009 trajectory groups that includes more than 10% of all the trajectories.

- Spatial: Retrieve the stops that occur near events of type "Restaurant" and that are located within 3km of "Duomo" in Milan.

- Temporal: Retrieve the stops that occur near an area where more than *X* other trajectories are stopping at lunch time or dinner time and that are located within 3km of (45.46,9.18) coordinates.

- Spatio-Temporal: For all summer months, give the number of "Congested Trajectories" in Milan's dense population neighborhoods.

Data Warehouse technology has become *de facto* standard for complex analytical and customized queries with a rapid execution time. However, there are several limitations in current data warehouse tools to cope with spatio-temporal data as found in the literature (Güting *et al*., 2004; Vaisman and Zimányi, 2009).

From our perspective, a trajectory is not only a set of time-geography points but it is a semantic object that has an identity, sub-components (stops, moves and episodes) and several related thematic information. For this reason, in our work, trajectory concept must be treated as a first class object and the resultant TDW is more than a spatio-temporal data repository.

Our contributions are twofold: a semantic model for trajectory data warehouse and a middleware for loading, designing and querying a spatio-temporal data warehouse. The model is based on the conceptual view on trajectories introduced by Spaccapietra *et al.* 2007. A trajectory is a segment of the spatio-temporal path covered by a moving object. The data warehouse design we developed is tailored around Episodes from Zhixian *et al.,* 2010. An episode is a sequence of location-based points that has the same moving dynamic. Episodes constitute a flexible segmentation of the dataset into semantic groups that can be used to model move and stop occurrences. We produced a generic cross-database framework for spatio-temporal data warehousing, basing the implementation on the approach proposed by Malinowski and Zimányi, 2007. This generic middleware has two main functionalities: on one hand it implements a common simple query interface to access the functionality of different proprietary implementation; on the other hand it builds a generic interface to support spatio-temporal concepts in data warehouses.

Furthermore we developed a generic middleware capable of encapsulating the data warehouse product dependent technicalities and building a generic design interface that have been extended to support spatio-temporal concepts in data warehouses.

TRAJECTORY DATA WAREHOUSE SCHEMA

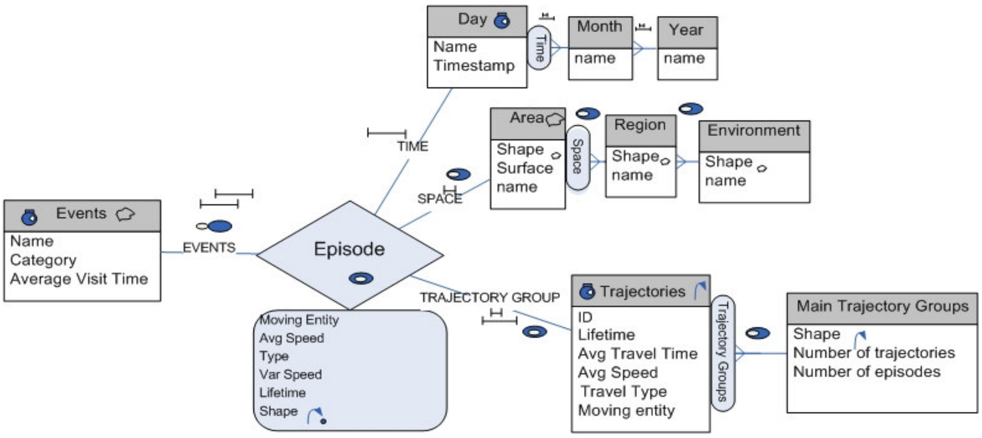


Figure 2: The Proposed Star Schema using MultiDimER notation

Since our scope is to provide a generic solution for designing and implementing TDW, we need to develop a model composed by constructs that allow the specification of a trajectory data warehouse schema. Thus, we have devised a model illustrated in Figure 2. From our perspective and unlike other solutions (e.g. Orlando 2006), it is significantly important that trajectory identity can be preserved.

We adopt the semantic trajectory model, introduced by Spaccapietra *et al.,* 2007, which considers a trajectory as the user defined record of the evolution of the position (perceived as a point) of an object that is moving in space during a given time interval in order to achieve a given goal. Each trajectory can be semantically segmented by defining a sequence of moves and stops. We define the movement primitives focusing the analysis tools on top of the concept of "Episode" introduced by Zhixian *et al.*, 2010 that is a more general unified representation of a trajectory element. This brings flexibility to our model for further applications in which episodes can be calculated in different ways. An important aspect of data warehousing management architectures is the handling of aggregate operators. The most common cited operators that can be found in the literature have been implemented and fully integrated in our generic middleware, see Table 1.

Operator	Type
Trajectory Length	Numeric
Number of Stops	Numeric
Number of Moves	Numeric
Average Speed	Numeric
Stops	Spatial
Moves	Spatial
Interpolated Shape	Spatial
Lifespan	Temporal
Average Travel Time	Temporal
Average stop Time	Temporal
Visited Areas	Spatio-Temporal
Trajectory Clusters	Spatio-Temporal
Most Frequent Set	Spatio-Temporal

Table 1: The Aggregate Operators

ST-TOOLKIT

Our framework proposal is called St-Toolkit[1]and is a generic middleware. It encapsulates the main database and data warehouse components, but also it hides the details of the single data warehouse implementation in order to abstract entities, such as cubes, dimensions, and measures. ST-Toolkit exploits proprietary drivers for OLAP solutions and, when it is possible, it uses vendor-provided APIs in order to boost data warehouse performances. Figure 3 illustrates the main components of ST-Toolkit framework. It is worth noticing that with our framework is possible to implement a data warehouse schema in OLAP servers (e.g. Oracle OLAP and Mondrian), or

[1]Web reference: http://st-toolkit.sourceforge.net/

in relational databases.It is a generic interface from which any proprietary database management system drivers can be mapped in order to become a ST-Toolkit access point for hosting Spatio-Temporal Data Warehouses. From a broad point of view the architecture is taking care of the most common aspects in moving object data management from ETL procedures primitives to the core of the library: the Generic Object Interface for Data Warehousing.

Taking a brief overview on similar architectures, the closest middleware solution that is capable of handling OLAP and SOLAP queries is GeoMondrian. It is an extension of the Mondrian OLAP Server that enables the use of spatial filtering. Mondrian has a dedicated query engine, which translates MDX OLAP queries into simple SQL relational queries, which is introducing further latencies to the query execution-plans.

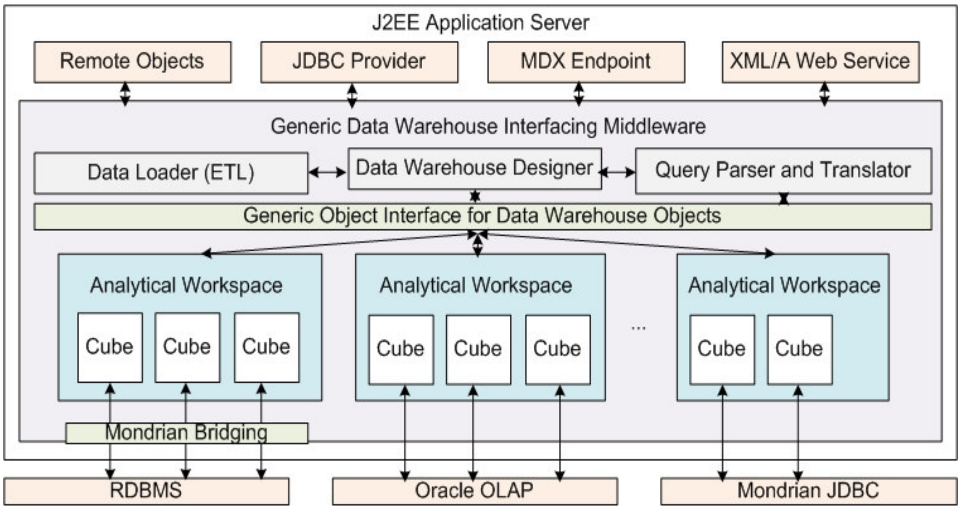


Figure 3: ST-Toolkit Components

From a procedural point of view, Mondrian (and its extension GeoMondrian) needs to look up on an XML description of the data warehouse schema in order to initialize its data connectors. Our solution instead is embedding the schema definition within java objects, allowing the user to gain modularity and making the schema capable of evolutions during execution time.

Instantiating ST DW schema

ST-Toolkit provides an API for creating and querying a DW schema. The java code presented below shows an example of *RelationalTable*creation. First, a database connection is established, then an object representing a relational table *tbl_time* is created, and then the data referring to this table is loaded. A database object is defined as any entity that lives in a persistent state in the database.

For instance a *RelationalTable* object is used to manipulate relational tables; therefore it is extensively used to interface with the data warehouse metadata.

```
        DatabaseConnection co = new DatabaseConnection (...);
    RelationalTabletimeTbl = new RelationalTable("tbl _time");
    timeTbl .loadObject (co );
```

In the Java code below, a dimension is specified with one level in its hierarchy, as follows:

```
        Level monthLevel = new Level("Month", timeTbl , "mon");
        Hierarchy timeH = new Hierarchy("Hier Name",
Hierarchy.LEVEL_BASED ,
                Hierarchy.NON_GEOMETRIC );
timeH .addLevel ( monthLevel );
        Dimension dim = new Dimension("Time", timeH);
```

Semantics

The use of semantics to enrich STOLAP queries it is a fundamental aspect in any knowledge-discovery process. We propose a first step to integrate spatio-temporal semantics using an Event Dimension. It subdivides the data in relation to space-time proximity with application specific events. Those events are spatially and timely correlated with the fact data. Examples of events might be Traffic Jams, Festivals, or simple Points of Interest such as Restaurants or Malls. We want to keep this dimension as much independent from the trajectory dataset as possible. This decision has a twofold objective: it allows using the same trajectory dataset for different application purposes (e.g. traffic management and social analysis) and it avoids heavy computational costs while updating the Event dimension. Encapsulated semantics allows to an easiest and more human readable query definition, as is possible to see from the following examples: *EventCrossingPatterns_1* and *EventVisitingPatterns_1* queries use semantics whereas *EventCrossingPatterns_2* and *EventVisitingPatterns_2* do not use semantics.

- **EventCrossingPatterns_1** (SOLAP Query): Retrieve all the stops that occur near events of type “Restaurant” and that are located within 3km of “Duomo” in Milan.

- **EventCrossingPatterns_2** (SOLAP Query): Retrieve all the stops that occur near an area where more than X other trajectories are stopping at lunch time or dinner time and that are located within 3 km from (45.46,9.18) coordinates.

- **EventVisitingPatterns_1** (STOLAP Query): Give the number of visits of a moving entity for events of type "Food-Shop" where its own trajectory started near a "Residential Area".

- **EventVisitingPatterns_2** (STOLAP Query):Give the number of trajectories that can be contained in spatio-temporal prism which trajectory density is more than X arbitrary value and where any trajectory started near an area where a lot of trajectories start in the morning.

Our design module is highly enriched by the capability of handling context-aware, providing extensibility to application domain feature sets (e.g. POIs).

CASE STUDY

This experimental section is aimed to test and to provide a case study for the trajectory data warehouse architecture. We will briefly describe the possible queries that we can address, showing later with the results on a real dataset of transportation traffic data in Milan, Italy. The dataset we used to test our Trajectory Data Warehouse contains data collected from cars running in Milano provinces, Italy in 2007 (see Table 2 for a dataset description). Please note that tables from 2 and 3 are shown as an informative samples of experimental data that can be extracted using OLAP, SOLAP and STOLAP operators defined for the given data warehouse and they derived from a running instance of ST-Toolkit.

Features	Values
Number of Records	2075213
Number of Trajectories	83134
Number of Stops	464584
Number of Moves	1527495
Number of POIs	39776
POIs Categories	447

Table 2:Dataset Statistical Description

Experiments

EventCrossingPattern_1 query have been explicitly formalized in the previous section, and sub-selecting a base of 1000 randomly chosen trajectory limit (simplified for visualization purpose), have produced as outcome a list of stop points located in Milan, at a distance of meters from a Restaurant location. The graphical geo-localized representation of this result can be seen in Figure 4. As it is possible to denote that the stops are properly located along the principal roads and randomly distributed along Milano (that is consequence of the high density of trajectories that are shown in

Figure 1). EventVisitingPatterns_1 query uses the whole trajectory dataset composed by 83134 trajectories: the top 10 results are shown in Table 3.

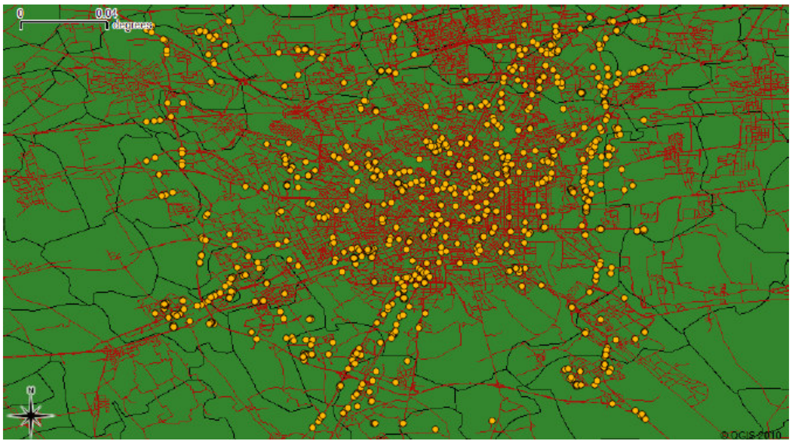


Figure 4:Milan Stops near Restaurants

Number of Visits	Moving Object Entity
64640	89754
56055	78796
52015	70702
49995	79088
47470	82085
46460	62748
38885	43348
37875	86336
33330	47819

Table 3:Moving Objects Visits to Food Shops

The "presence problem"

We run a last test on the spatial aggregation function used to calculate the Presence measure, given a subset of 260 episodes as shown in Figure 5.

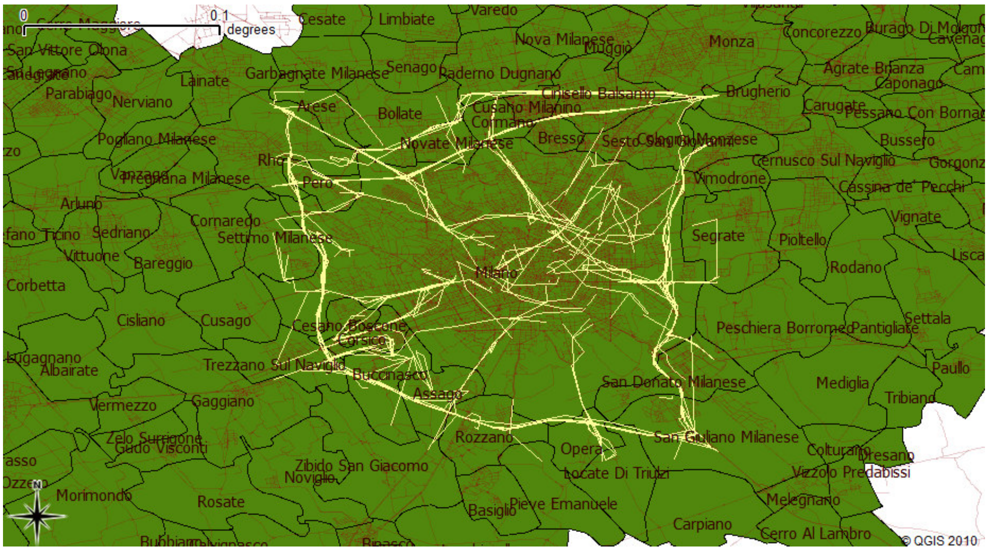
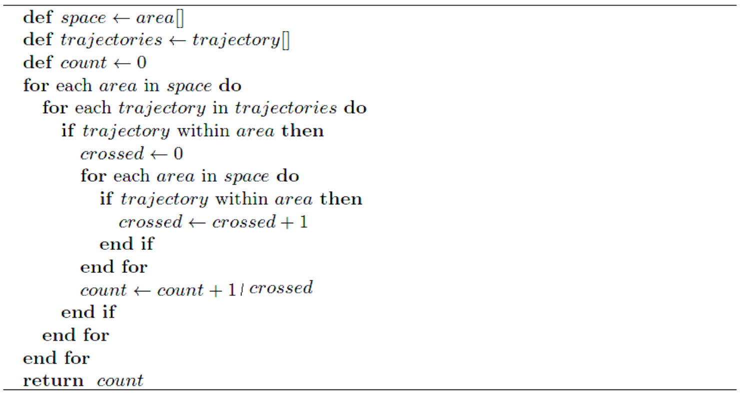


Figure 5:Presence Test Dataset around Milan municipality

Most of the trajectories are traversing multiple areas. The issue encountered while calculating the Presence measure was the so-called double counting problem. It means that rolling-up on multiple areas where different trajectories were crossing those shapes could not have been done with a trivial counting without double counts (e.g. trajectories that were present on two areas were counted twice). Literature calls this the “presence problem”. It is one of the main drawbacks of the TDW model proposed by Orlando *et al.*, 2007. We have developed an algorithm for solving the presence problem as it can be noticed from Figure 6.



```
def $space \leftarrow area[]$
def $trajectories \leftarrow trajectory[]$
def $count \leftarrow 0$
for each $area$ in $space$ do
  for each $trajectory$ in $trajectories$ do
    if $trajectory$ within $area$ then
      $crossed \leftarrow 0$
      for each $area$ in $space$ do
        if $trajectory$ within $area$ then
          $crossed \leftarrow crossed + 1$
        end if
      end for
      $count \leftarrow count + 1/crossed$
    end if
  end for
end for
return $count$
```

Figure 6:Presence Algorithm Counting

This algorithm is quite straightforward: for each fact-data geometries, a basic spatial operator is checking if the object resides in how many regions in the lowest level space division: that becomes the *crossed* value. The inverse of *crossed* is a metadata attribute for each trajectory that can be used to easily weight the spatial count of all the trajectories of a given area without knowing other details on the other regions.

The queries to extract the count are expressed as simple as that:

```
[…]
OlapQuery query = new OlapQuery();
query.addSelection(presenceMeasure, OlapQuery.COLUMNS);
query.addSelection        (spaceDimension.getLevel(“Area”)        ,
OlapQuery.ROWS);
query.setCube(stCube);
query.execute();
```

This is the Olap Query to extract the count of each trajectory slicing on Municipalities, where *presenceMeasure* is a virtual measure that is embedding our simple algorithm as an operator.

Our model uses a fully-geometric spatial dimension that aggregates trajectories belonging to their spatial “presence”. We have made use of spatial functions to aggregate the OLAP measure that operates the count of super aggregates. In this way, despite of trajectories pass throughout more than one area, the roll up counting result of the SOLAP queries, sliced by the three spatial levels of our data warehouse schema,is correct and equal to the 260 trajectories chosen for the sub-set.

RELATED WORK

Among the approaches that have been proposed to model spatio-temporal data warehouses we made use of the one proposed by Malinowski and Zimanyi in 2007. Their mixed approach is coping with the lack of modeling robustness experienced in the literature.

This approach added an object-oriented infrastructure layer to spatial data warehouses experimenting MultiDimER model. Another example of alternative models is the one presented by Stefanovic *et al.* 2000 that adds spatial semantics to the whole multidimensional paradigm. The new dimension types that were defined are Non-geometric spatial dimensions (i.e. if it contains only non-geometric data), Geometric-to-non-geometric spatial dimension and Fully geometric spatial dimension.

Among the first trajectory-enabled frameworks we can recall the data model implemented in Secondo (Güting *et al.,* 2004) and Hermes (introduced by Pelekis *et al.,* 2006). The first is an extensible database platform that allows implementation of open data models. Several papers are presenting solutions to moving objects problems (such as Düntgen *et al.,* 2009 and Güting *et al.*, 2009). Although this architecture has been enriched with an extensive variety of data models, it narrows data management applications to be used only on top of Secondo DBMS. On the other hand, Hermes is a trajectory data management framework that it is explicitly created for spatio-temporal data storage in order to provide modeling and querying facilities for trajectories of dynamic objects by using Oracle DBMS.

Their approaches are both trying to cope with the limitations of data models that can be represented in traditional database systems. Our approach on the contrary is providing an intermediate layer that can be used to model database objects using the best of both worlds: It can be extended with user-defined data models and it can reuse the most deployed database management systems on the market.

One of the major weaknesses in spatio-temporal data warehousing is the lack of taxonomies that defines which limitations and which expressive power are addressed in STOLAP applications. Vaisman and Zimanyi, 2009 defined a complete taxonomy for spatio-temporal OLAP applications, which is describing a perspective proposal of the possible features through spatio-temporal calculus algebra. Following this taxonomy we have subdivided the possible queries of our trajectory data warehouses (TDW) into three main categories: OLAP, Spatial OLAP, Spatio-Temporal OLAP.

The only true implementation of TDW that can be recalled, is the solution proposed by Orlando *et al.* in 2007. Theirmodel is supporting unbounded and unpredictable stream rates of moving object data in order to feed a regular-grid space division.

CONCLUSIONS

In the last years we have seen a continuous evolution of tracking technologies in precision, costs and device support. Collecting mobility data is no more a problem, whereas handling and analyzing these spatio-temporal data is still an open challenge. In this context we have developed a generic cross-platform cross-database framework capable of implementing a common simple query interface to access different product-dependent data sources. Moreover, it contains a generic design interface supporting spatio-temporal concepts in data warehouse.

We have provided an instantiation of a specific trajectory model to explain and support the system. As in many other fields, the main open issue for the future development and improvement still remain the integration of full semantic aspects into the traditional dimension of analysis.

BIBLIOGRAPHY

Düntgen C., T. Behr, and R. H. Güting. Berlinmod: a benchmark for moving object databases. VLDB, 2009

Güting, R.H., T. Behr, V. Almeida, Z. Ding, F. Hoffmann, and M. Spiekermann. Secondo: An extensible dbms architecture and prototype. Technical report, 2004

Güting, R.H., M. H. Bohlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, F. Hagen, F. Hagen, and F. Hagen. A foundation for representing and querying moving objects. ACM Transactions on Database Systems, 2000.

Güting, R.H., A. Braese, T. Behr, and J. Xu. Nearest neighbor search on moving object trajectories in secondo, Advances in Spatial and Temporal Databases 2009

Malinowski, E. and E. Zimanyi. Implementing spatial datawarehouse hierarchies in object-relational dbmss.InICEIS 2007

Orlando, S., R. Orsini, A. Raffaeta, and A. Roncato. Trajectory data warehouses: Design and implementation issues. Journal of Computing Science and Engineering, 2007

Pelekis, N., Y. Theodoridis, S. Vosinakis, and T. Panayiotopoulos. Hermes - a framework for location-based data management. In EDBT 2006

Spaccapietra, S., C. Parent, M. L. Damiani, J. A. d. Macedo, F. Porto, and C. Vangenot. A conceptual view on trajectories. Data& Knowledge Engineering, 2007

Stefanovic, N., J. Han, and K. Koperski. Object-based selective materialization for efficient implementation of spatial data cubes. IEEE, 2000

Vaisman, A. and E. Zimányi. What is spatio-temporal data warehousing? In DAWAK, 2009

Zhixian, Y., C. Parent, S. Spaccapietra, and C. Dipanjan. A hybrid model and computing platform for spatio-semantic trajectories. ESWC 2010