Spatio-Temporal Data Warehouses and Mobility Data: Current Status and Research Issues

Esteban Zimányi
Department of Computer and Decision Engineering (CoDE)
Université Libre de Bruxelles, Belgium
ezimanyi@ulb.ac.be

***Abstract*—Applications using mobility data are blooming in almost every application domain. Many research efforts have been devoted to developing concepts, models, theories, and tools to properly apprehend mobility data and make it manageable for the benefit of these applications. One particular aspect in this respect concerns mobility data management, i.e., storing mobility data in a data warehouse for their subsuquent analysis. This paper introduces the notion of spatio-temporal data warehouses and shows how such a data warehouse can be defined using an extensible set of data types.**

***Keywords*-spatio-temporal data warehouses; mobility data; OLAP**

I. Introduction

Nowadays, the usage of location-aware devices (e.g., mobile phones, GPS) allows collecting large volumes of trajectory datasets. Effective analysis of such trajectory data imposes new challenges for their management, while raising opportunities for discovering behavioral patterns that can be exploited in applications like location-based services or traffic control management.

Data Warehouses (DW) and Online Analytical Processing (OLAP) have been successfully used for transforming detailed data into valuable knowledge for decision-making purposes. Extending DWs for coping with trajectory data allows us to extract essential knowledge from these trajectories. For example, such a DW can be used for analyzing the variable number of cars, or their average speed, in different urban areas.

Trajectory data in a warehouse must be typically analyzed in conjunction with other data, e.g., to analyze air pollution in a city, or to find out the correlation between areas with heavy traffic and areas with heavy pollution loads. In this case, the data warehouse needs not only to store trajectory data but also air quality data. In light of these needs, in this paper we provide an overall view that integrates trajectory data in a more general data warehousing framework, which we call *spatio-temporal data warehousing* [1].

II. Spatio-Temporal Data Warehouses

Data warehouses (DW) are large data repositories that collect data from several sources, and are used to support the decision-making process. Data warehouses are typically accessed using Online Analytical Processing (OLAP), which enables analysts and executives to gain insight into data through interactive exploration of a wide variety of views. At a conceptual level, data are perceived as a multidimensional cube, where cells contain values (or *measures*), which can be analyzed through several axes (or *dimensions*).

We introduce these notions by an example of a DW for analyzing the air quality in the Italian city of Milano. Stations measure air quality in different points of the city at regular time intervals. Non-spatial data in the DW include the measures taken at the stations, and data about the pollutants that affect air quality. Spatial data feeding the DW are car trajectories as well as maps containing the layout of the road network, the locations of the measuring stations, and the political division of the city into zones and districts.

Figure 1 shows the conceptual schema depicting the above scenario using the MultiDim model [2]. There is a *fact relationship*, AirQuality, with measures load and riskLevel. AirQuality is related to six *dimensions*, e.g., Time, Station, and Road. Dimensions are composed of *levels* and *hierarchies*. For example, the Station dimension has only one level, while the District dimension is composed of two levels, District and Zone, with a one-to-many *parent-child relationship* defined between them.

There are two alternative ways of representing spatial data. In the *discrete view*, the space contains objects (e.g., roads and districts) with a defined location and shape. The *continuous view* sees space as a continuum, holding properties that depend on the location in space. Typical examples are land use and temperature. Furthermore, *spatial relationships* provide a way of representing how spatial objects are related to each other. There are several classes of spatial relationships, for example, *topological* (e.g., adjacent, inside, disjoint), *direction* (e.g., above, below), and *metric* relationships (e.g., distance).

Let us consider first the discrete view of space. In order to provide support for spatial data, a data warehouse must provide spatial data types (such as point, points, line, and region) as part of its data model and its associated query language. Each of these data types comes with a predefined set of operations. For example, a predicate inside can be used to test whether a point is inside a region of whether a

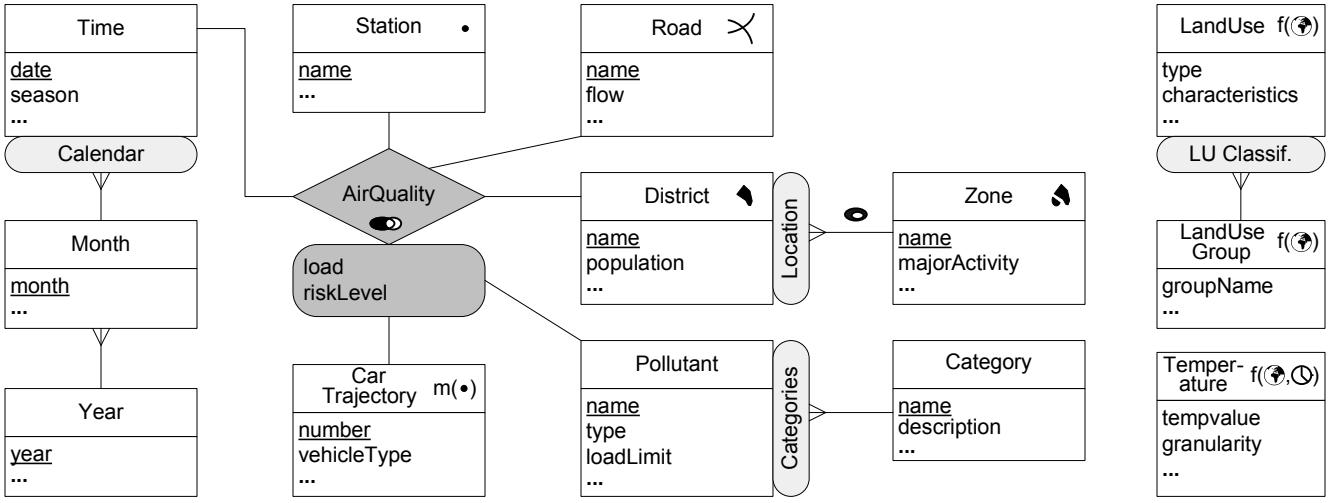

Figure 1. An example of a spatio-temporal data warehouse

region is included into another. Thus, SOLAP (standing for Spatial OLAP), aims at exploring spatial data by drilling on maps, as it is performed in OLAP with tables and charts.

In our example in Figure 1, the pictograms in dimensions Station, Road, and District indicate that these dimensions are spatial, and their geometry correspond to a point, a line, and a region, respectively. In our example, the topological relationship in dimension District indicates that a district is covered by its parent Zone. Similarly, the topological relationship in AirQuality indicates that whenever a station, a road, a district, and a car trajectory are related in an instance of the relationship, they must overlap.

Many real-world applications need to cope with moving objects, i.e., objects whose geometry evolves over time. For example, a *moving point* captures the evolution of objects for which only its position in space is relevant. Examples of moving points are people or cars. A *moving region* captures moving as well as growing or shrinking regions. Examples of moving regions are hurricanes or forest fires. Notice that moving object data amount to spatio-temporal data.

In order to support moving objects, data warehouses need to be extended with moving (or temporal) types. These are obtained by applying a constructor $\mathsf{moving}(\cdot)$ to any base or spatial data types. For example, a value of type moving real is a continuous function $f : \mathsf{instant} \rightarrow \mathsf{real}$, and a value of type $\mathsf{moving}(\mathsf{point})$ is a continuous function $f : \mathsf{instant} \rightarrow \mathsf{point}$.

Let us consider now the continuous view of space. Continuous fields are used for representing phenomena that change continuously in space and/or time. At a conceptual level continuous fields can be represented as a function that assigns to each point of space (and possibly in time) a value of a particular domain (e.g., integer for altitude). Continuous fields are implemented using field types which are obtained by applying a constructor $\mathsf{field}(\cdot)$. For example, a value of type $\mathsf{field}(\mathsf{real})$ (e.g., representing altitude) is a continuous function $f : \mathsf{point} \rightarrow \mathsf{real}$. *Moving types* also apply to field types. For example, a value of type $\mathsf{moving}(\mathsf{field}(\mathsf{real}))$ defines a continuous function $f : \mathsf{instant} \rightarrow (\mathsf{point} \rightarrow \mathsf{real})$ that can be used to represent temperature, which varies on time and space.

Figure 1 shows a field dimension LandUse, which at each point in the space of interest has a value for the use given to land in Milano (e.g., industrial, residential). This is useful for analyzing correlations between emissions and air quality in the city and its surroundings, according to the use of the land. LandUse is further classified into groups, represented by the dimension level LandUseGroup. This level is aimed at providing an aggregate view of the land use. Similarly, Temperature is a *temporal field*, which means that each point in the space of interest has a value of temperature, and this value changes at each time instant. Field levels have a geometry attribute, which keeps track of the value of the field at each point in space (and time, for temporal fields). Note that field dimensions are not connected to a fact relationship, as it is the case for other dimensions. In addition, traditional numerical measures can be calculated from field data. An example is measure riskLevel, which is an indicator (a real value) associated to each instance of the fact relationship, and which is computed from the dimensions Temperature, LandUse, and Time.

III. Querying Spatio-Temporal Data Warehouses

We next discuss how to query a spatio-temporal DW. For this we use a functional first-oder query language, based on the well-known relational calculus with aggregate functions proposed by Klug, which extends the classic relational calculus with aggregate functions. We denote this language $\mathcal{Q}$. Let us show how a $\mathcal{Q}$ query looks like, using our running example.

Query 1. *"Total population by zone provided that it is greater than 10,000".*

$\{z.\mathsf{name}, \mathsf{totalPop} \mid \mathsf{Zone}(z) \wedge$
$\mathsf{totalPop} = \mathsf{sum}(\{d.\mathsf{population} \mid \mathsf{District}(d) \wedge$
$d.\mathsf{Zone} = z\}) \wedge \mathsf{totalPop} > 10000\}$

This query corresponds to an SQL query with the GROUP BY and HAVING clauses. For each zone, the inner query computes the population of all of its districts, and the sum is stored in the variable totalPop.

We show now that an OLAP query is just a relational calculus query with aggregation. Consider a set of *base types*, namely int, real, bool, and string. These types have the usual interpretation, except that their value may be undefined. There are also *time types* which are instant and periods, the latter being a set of time intervals. We denote β the set of base types, and τ the set of time types. These types have an associated set of operators. These include the usual aggregation operators count, sum, avg, etc. It is easy to show that $\mathcal{Q}$ has the same expressive power of the relational calculus extended with aggregate functions.

Definition 1 (OLAP queries)**.** The class of OLAP queries is composed of all the queries that can be expressed by the language $\mathcal{Q}$ defined over the sets of base and time types β and τ. □

Consider now a set of *spatial data types*, namely point, points, line, and region. These types have the usual interpretation, they define a point or a point set over $\mathbb{R}^2$. We denote ξ the set of *spatial types*. These types have an associated set of operations. These include topological predicates (e.g., touches, overlaps), spatial operations (e.g., area, perimeter, distance, direction, center), etc.

As an example of the above, consider the following query, which includes the spatial dimensions Station and Road.

Query 2. *"For air stations located on Via del Mare, give the average values of lead registered in the last quarter of 2010."*

$\{s.\mathsf{name}, \mathsf{avgLead} \mid \mathsf{Station}(s) \wedge \exists r, p\ (\ \mathsf{Road}(r) \wedge \mathsf{Pollutant}(p) \wedge$
$r.\mathsf{name} = \text{'Via del Mare'} \wedge p.\mathsf{name} = \text{'Lead'} \wedge$
$\mathsf{intersects}(r.\mathsf{geometry}, s.\mathsf{geometry}) \wedge \mathsf{avgLead} =$
$\mathsf{avg}(\{a.\mathsf{load} \mid \mathsf{AirQuality}(a) \wedge \exists t\ (\ \mathsf{Time}(t) \wedge$
$a.\mathsf{Station} = s \wedge a.\mathsf{Pollutant} = p \wedge a.\mathsf{Time} = t \wedge$
$t.\mathsf{date} >= 1/10/2010 \wedge t.\mathsf{date} <= 31/12/2010)\})\)\}$

The inner query (an OLAP query) joins the fact relationship AirQuality with the dimensions Station, Pollutant, and Time, for each combination of members of these three dimension levels. Then, for values in the last quarter of 2010, the load is retrieved and the average computed in the variable avgLead. Note that the outer query includes the intersects predicate, which determines if a pair of geometries representing stations and roads intersect.

We define next the class of Spatial OLAP (SOLAP) queries.

Definition 2 (SOLAP queries)**.** Let us call $\mathcal{Q}^{\xi}$ the language $\mathcal{Q}$ extended with spatial types in ξ. The class of SOLAP queries is the class composed of all the queries that can be expressed by $\mathcal{Q}^{\xi}$. □

A *spatial data warehouse* is a data warehouse that supports SOLAP queries.

Consider now the *moving types*. These define a function from the time type instant to a base or spatial type. Note that moving types are partial functions, i.e., they may be undefined for certain periods of time. We denote μ the set of moving types, obtained by applying the moving constructor to elements of β and ξ (the sets of base and spatial types, respectively).

Moving types have an associated set of operations [4]. For example, the projection of a moving point into the plane consists of the points and lines returned by the operations locations and trajectory, respectively. Similarly, the projection of a moving region into the plane consists in a region, which is returned by the operation traversed. Such an operator can be used, e.g., to compute the union of all the regions covered by a moving cloud in a given time interval. Other operations are defined for moving types, e.g., to compute the rate of change for points (e.g., speed, turn, velocity, etc.). Further, moving types have associated operations that generalize those of the non-temporal types. This is called *lifting*. For example, a distance function with signature $\mathsf{moving}(\mathsf{point}) \times \mathsf{moving}(\mathsf{point}) \rightarrow \mathsf{moving}(\mathsf{real})$ calculates the distance between two moving points and gives as result a moving real, i.e., a real-valued function of time. Intuitively, the semantics of such lifted operations is that the result is computed at each time instant using the non-lifted operation.

As an example of the above, consider the following query, which includes the moving point dimension CarTrajectory.

Query 3. *"Total number of trajectories of length larger than 2 km on the Via del Mare, such that at least one station over such road exceeded the limit for Carbon Monoxide".*

$\mathsf{count}(\ \{ct \mid \mathsf{CarTrajectory}(ct) \wedge \exists r, p, a\ (\ \mathsf{Road}(r) \wedge$
$\mathsf{Pollutant}(p) \wedge \mathsf{AirQuality}(a) \wedge a.\mathsf{CarTrajectory} = ct \wedge$
$a.\mathsf{Road} = r \wedge a.\mathsf{Pollutant} = p \wedge$
$r.\mathsf{name} = \text{'Via del Mare'} \wedge$
$p.\mathsf{name} = \text{'Carbon Monoxide'} \wedge a.\mathsf{load} > p.\mathsf{loadLimit} \wedge$
$\mathsf{length}(\mathsf{intersection}(\mathsf{trajectory}(ct), r.\mathsf{geometry})) > 2\})$

In this query we apply the trajectory operation to project the moving car into the plane, and intersect the result with the geometry of the road. After this, we compute the length of this intersection.

We define next the class of Spatio-Temporal OLAP (ST-OLAP) queries.

Definition 3 (ST-OLAP queries)**.** Let us call $\mathcal{Q}^{\xi\mu}$ the language $\mathcal{Q}$ extended with spatial types in ξ and moving types in μ. The class of Spatio-Temporal OLAP queries (ST-OLAP) is composed of all the queries that can be expressed by $\mathcal{Q}^{\xi\mu}$. □

A *spatio-temporal* (or *trajectory*) *data warehouse* is a warehouse that supports ST-OLAP queries.

Consider now the *field types*. These define a function from the spatial type point to a base type. As it is the case for moving types, field types define partial functions, they may be undefined for some extents in space. We denote by ϕ the set of field types obtained by applying the field constructor to the set of base types.

Field types have an associated set of operations [3]. For example, operation at restricts the function to a point or to a point set in the range of the function. Similarly, operations atpoint, atpoints, atline, and atregion restrict the function to a given subset of the space defined by a spatial value. Operators atmin and atmax restrict the function to the points in space when its value is minimal or maximal, respectively. Predicate defspace returns the region over which the the spatial function is defined. Rate of change operations compute how a field changes across space. As for moving types, field types have associated *lifted* operations that generalize those of the base types. The semantics of such lifted operations is that the result is computed at each point in space using the non-lifted operation. Aggregation operators are also uplifted. For instance, an uplifted avg operator combines several fields, yielding a new field where the average is computed at each point in space.

As an example of the above, consider the following query, which includes the spatial field LandUse.

Query 4. *"For Milano zones with at least 20% of industrial land use, give the average load for Carbon Monoxide on February 1st, 2010".*

$$\{z.\mathsf{number}, \mathsf{avgLoad} \mid \mathsf{Zone}(z) \wedge \exists l\ (\mathsf{LandUse}(l) \wedge \mathsf{area}(\mathsf{defspace}(\mathsf{atregion}(\mathsf{at}(l.\mathsf{geometry}, \text{'Industrial'}), z.\mathsf{geometry})))/\mathsf{area}(z.\mathsf{geometry}) >= 0.2 \wedge \mathsf{avgLoad} = \mathsf{avg}(\{a.\mathsf{load} \mid \mathsf{AirQuality}(a) \wedge \exists d, t, p\ (\mathsf{District}(d) \wedge \mathsf{Time}(t) \wedge \mathsf{Pollutant}(p) \wedge a.\mathsf{District} = d \wedge d.\mathsf{Zone} = z \wedge a.\mathsf{Time} = t \wedge t.\mathsf{date} = 1/2/2010 \wedge a.\mathsf{Pollutant} = p \wedge p.\mathsf{name} = \text{'Carbon Monoxide'})\}))\}$$

Here, the LandUse field is restricted to the value 'Industrial' by means of the function at, and then further restricted to the geometry of the zone using the function atregion. The operator defspace then obtains the geometry of the restricted field, the area of this geometry is computed, and finally divided by the total area of the zone. For the zones that satisfy the condition, the average load is then computed into the avgLoad variable.

We define next the class of Spatio-Temporal and Continuous Field OLAP (STCF-OLAP) queries.

Definition 4 (STCF-OLAP queries)**.** Let us call $\mathcal{Q}^{\xi\phi\mu}$ the language $\mathcal{Q}$ extended with spatial types ξ, field types ϕ, and moving types μ (the latter includes moving fields). The class of STCF-OLAP queries contains all the queries expressed by $\mathcal{Q}^{\xi\phi\mu}$. □

A *spatio-temporal and continuous field data warehouse* is a data warehouse that supports STCF-OLAP queries.

IV. Conclusion

Current technological advances made possible the collection of mobility data. As a consequence, there is a huge demand for methods for modeling, managing, and understanding mobility data. In this paper we focused on the management aspect of mobility data, i.e., how to store and query these data into spatio-temporal data warehouses. We have seen that an efficient way to deal with this problem is to extend data warehouses with appropriate data types that cope with the spatial and temporal features.

Many related issues have been omitted, they are discussed in particular in [5]. A first issue concerns *mobility data modeling*, i.e., the process that transforms the raw data obtained from data collection devices (e.g., GPS, Bluetooth, RFID) into the application-dependent concept of trajectory. Another important aspect concerns *mobility data mining*, i.e., extracting useful knowledge out of mobility data. *Mobility data visualization* concerns interactive visualization techniques suitable for analysis of mobility data. A final aspect that we want to mention concerns *mobility data and privacy*. Tracing the movement of people poses big privacy issues, from the viewpoint of both the management of sensitive data and the publishing of personal information.

Acknowledgment

I would like to thank Alejandro Vaisman, with whom many concepts presented in this paper were developed.

References

[1] A. Vaisman and E. Zimányi, "What is spatio-temporal data warehousing?" in *Proceedings of DaWaK 2009*, ser. LNCS. Springer, 2009, no. 5691, pp. 9–23.

[2] E. Malinowski and E. Zimányi, *Advanced data warehouse design: From conventional to spatial and temporal applications*. Springer-Verlag, 2008.

[3] A. Vaisman and E. Zimányi, "A multidimensional model representing continuous fields in spatial data warehouses," in *Proceedings of ACM GIS 2009*. ACM Press, 2009, pp. 168–177.

[4] R. Güting and M. Schneider, *Moving Objects Databases*. Elsevier, 2005.

[5] C. Renso, S. Spaccapietra, and E. Zimányi, Eds., *Mobility Data: Modeling, Management, and Understanding*. Cambridge Press, 2012, forthcoming.