A Conceptual Data Model for Trajectory Data Mining

Vania Bogorny[1], Carlos Alberto Heuser[2], and Luis Otavio Alvares[2]

[1] Depto de Informatica e Estatistica – Universidade Federal de Santa Catarina
Campus Universitario C.P. 476, Florianopolis, Brazil
vania@inf.ufsc.br
[2] Instituto de Informatica – Universidade Federal do Rio Grande do Sul
Av. Bento Goncalves 9500, Porto Alegre, Brazil
{heuser,alvares}@inf.ufrgs.br

Abstract. Data mining has become very popular in the last years, and it is well known that data preprocessing is the most effort and time consuming step in the discovery process. In part, it is because database designers do not think about data mining during the conceptual design of a database, therefore data are not prepared for mining. This problem increases for spatio-temporal data generated by mobile devices, which involve both space and time. In this paper we propose a novel solution to reduce the gap between databases and data mining in the domain of trajectories of moving objects, aiming to reduce the effort for data preprocessing. We propose a general framework for modeling trajectory patterns during the conceptual design of a database. The proposed framework is a result of several works including different data mining case studies and experiments performed by the authors on trajectory data modeling and trajectory data mining. It has been validated with a data mining query language implemented in PostGIS, that allows the user to create, instantiate and query trajectory data and trajectory patterns.

Keywords: conceptual model, data mining, trajectory data, trajectory patterns.

1 Introduction

Since its origin, database design has the purpose of modeling data for operational purposes only. However, with the globalization and information dissemination the need to extract more interesting information and knowledge from large amounts of data by far exceeds operational database models. This is specially true for trajectories of moving objects, which is a new kind of spatio-temporal data generated by mobile devices, that in the last few years have become very popular in daily life.

Knowledge Discovery in Databases (KDD), which is the technology to extract interesting and previously unknown patterns from data, gives support for strategic and decision making purposes, and it is an advanced step towards intelligent data analysis. Database designers, however, have not yet realized the need of thinking about data mining when designing a database. While on the one side

S.I. Fabrikant et al. (Eds.): GIScience 2010, LNCS 6292, pp. 1–15, 2010.

database designers focus on operational functionalities only, on the other side the data analysts have a lot of work to preprocess large databases for data mining. This results in the *gap* that exists between databases and data mining, having several consequences. The most relevant is the preprocessing phase for knowledge discovery, where data have to be filtered, cleaned, and transformed for data mining algorithms. It has been stated that data preprocessing for data mining consumes between 60% and 80% of the whole efford required in the KDD process [1] for conventional data. This problem increases even further when dealing with spatial and spatio-temporal data, which are the focus of this paper.

Several attempts have been made to overcome the limitations of database systems over data mining tasks [2]. One well known approach is the development of data mining query languages for temporal data [3] and spatial data [4]. Most of these attempts propose extensions of SQL that might be difficult to achieve, and after the years have not been adopted by commercial DBMSs [5]. Another problem is that most of these works focus on the mining step itself, without addressing the most important and time consuming task in the discovery process: *data preprocessing* [1] [5] and *pattern post-processing*. The problems with SQL and commercial systems motivated interest in using other database languages. For instance, [6] proposed an algebra to integrate data and patterns.

Another problem for several data mining tasks is that data have to be preprocessed and transformed into different granularities, otherwise no patterns may be discovered [7]. This is specially true for trajectories of moving objects [8]. Both data preprocessing for data mining and granularity transformation are arduous tasks from the user's perspective, and these problems could be significantly reduced if data mining was foreseen during the conceptual database design.

In previous work we have shown that data mining should consider not only the data, but the schema of the data as well [9]. Conceptual schemas represent all relationships among the data that are well known by the database designer, specially those which represent integrity constraints. These relationships are uninteresting for knowledge discovery, since they lead to the discovery of well known relationships among data. We have shown in [9] that all relationships with cardinality constraints *one..one* or *one..many* represented in the conceptual schema will generate association patterns that are well known and uninteresting. In [10] we have shown that the conceptual schema can be used to both visualize and represent spatio-temporal patterns of trajectories of moving objects. These examples show that the conceptual schema of a database should be considered during the KDD process.

Recently, Spaccapietra [11] proposed the first conceptual model for trajectories of moving objects. This model has been defined in the context of the European Project GeoPKDD [12], that has its focus on developing the science of trajectories of moving objects, from conceptual modeling, to databases, data warehouses and privacy-preserving data mining. This model is called *stops and moves*, and has been adopted by several approaches as a standard for *semantic trajectory data analysis*, as for instance, [13] [10] [14] [15] [16] [8].

In this paper we propose a novel solution to reduce the problem of data preprocessing for data mining, by reasoning and thinking about data mining during the

database conceptual design. We strongly believe that it is time to consider data mining already during the conceptual modeling phase of a database. Therefore, we propose a conceptual framework for modeling trajectory data and trajectory patterns. The model foresees three main types of trajectory patterns, previously defined in [8]: *frequent patterns, sequential patterns*, and *association rules*. The proposed framework is an extension of the conceptual model for trajectories proposed by [11], in order to support *semantic trajectory pattern mining*.

The proposed data mining modeling framework is inspired on experiments performed on real trajectory data, by two of the authors, in previous works as, for instance, [13] [16] [17] [8] [15]. From this set of results, which cannot be detailed because of space limitations, we propose a conceptual framework for trajectory conceptual modeling and mining, which can be used as a design pattern for semantic trajectory conceptual modeling.

The reminder of the paper is organized as follows: Section 2 introduces the basic concepts on trajectory conceptual modeling and trajectory data mining. Section 3 presents the proposed framework, which is evaluated with query examples in Section 4. Section 5 concludes the paper and suggests directions of future research.

2 Trajectory Conceptual Modeling and Mining

Several data models have been proposed for efficiently querying moving objects [18,19,20]. In [18] the main focus relies on the geometric properties of trajectories, while [20] considered semantics and background geographic information. In [20] a semantic model for trajectories has been proposed as well as relationships of trajectories with geographic and environment information. This model, however, is restricted to a specific application domain, and trajectory relationships are related to vehicles and roads. In [19] trajectories are modeled over road networks. This kind of approach is limited to the road network, while in several applications there exits no network, such as animal migration, recreational activities in a park, the behavior of people in a shopping center, etc.

SECONDO and HERMEs are spatio-temporal DBMS prototypes in which a lot of effort has been made for representing trajectories of moving objects, creating new data types and operations to manipulate the spatio-temporal properties of trajectories. However, very little has been done for trajectories from the application point of view. To overcome these problems, a new concept has been introduced for reasoning over trajectories from a semantic point of view. This concept is called *stops* and *moves* [11].

2.1 The Model of Stops and Moves of Trajectories

Spaccapietra [11] has introduced the first conceptual model for trajectories, which is an evolution of the spatio-temporal well known model $MADS$ [21] to support trajectories. This model is application independent, and the user may add the semantics he is interested in, according to his application scenario. The semantic issue is very important because trajectory data are normally available

as sample points, without any semantics. Figure 1 shows an example of a trajectory sample (1) and a semantic trajectory (2). The *semantic trajectory* has associated the important places where the moving object has visited (the stops).

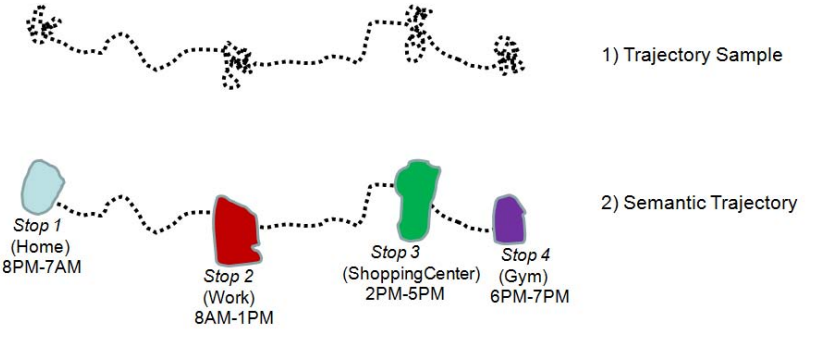

Fig. 1. Trajectory Sample (1) and Semantic Trajectory (2)

Considering trajectories from a semantic point of view, Figure 2 shows the model proposed by Spaccapietra, where a trajectory is the user defined record of the evolution of the position of an object that is moving in space during a given time interval, in order to achieve a given goal. In this model, a *Trajectory* belongs to an object (TravellingOT = Travelling Object Type). A trajectory is composed by a set of *stops* and *moves*. A *stop* (BES) is an important part of the trajectory from the application point of view where the moving object has stayed for a minimal amount of time. A *move* is a part of the trajectory between two consecutive stops. These concepts have been formally defined in [13].

BES represents the *Begin* and *End* of a trajectory or a *Stop*. Both stops and moves are related to a spatial object type (SpatialOT1 and SpatialOT2)), standardized by the Open GIS Consortium as *spatial feature type*.

A spatial feature type represents any spatial object in a set of spatial objects, in which a stop occurs. For instance, in a bird migration scenario, a stop can be at a specific country where birds are feeding. In a tourism application, a stop can be in a shopping center, a restaurant or a monument. In a traffic management application, a stop can be a traffic light, a roundabout, a parking place. Figure 1 shows an example of a trajectory sample (1) and a semantic trajectory (2), where the *semantic trajectory* has four stops: at home, at work, at a shopping center and at a gym. The moving points between the stops are the *moves*.

2.2 Semantic Trajectory Pattern Mining: Basic Concepts

In this section we describe the three data mining tasks supported by our model, considering the concept of stops and moves: trajectory frequent patterns, trajectory sequential patterns and trajectory association rules. These tasks are already supported and implemented in the first semantic trajectory data mining query

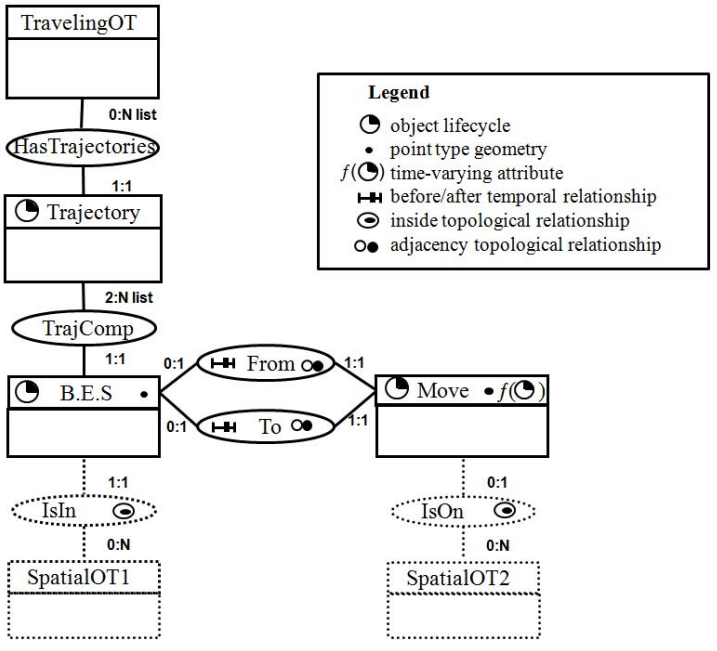

Fig. 2. The Model of stops and moves of trajectories [11]

language, named ST-DMQL, which we have introduced in [8]. First we define what is a semantic trajectory.

Definition 1. *Semantic Trajectory. A* Semantic Trajectory S *is a finite sequence* $\langle I_1, I_2, ..., I_n \rangle$ *where* I_k *is a stop or a move.*

Mining Trajectory *Frequent* Patterns. Frequent semantic trajectory patterns represent a set of items (stops or moves) that occur in a set of trajectories with support higher than a user defined minimum support (*minSup*).

Definition 2. *Support. Let* T *be a set of semantic trajectories. The support of a set of items* $X = \{x_1, x_2, ..., x_n\}$ *with respect to* T *is defined as the fraction of the trajectories in* T *that contain* X*, and is denoted as* $s(X)$*.*

A trajectory frequent pattern is defined as follows:

Definition 3. *Trajectory Frequent Pattern. Let* T *be a set of semantic trajectories. A set of items* $X = \{x_1, x_2, \ldots, x_n\}$ *is a trajectory frequent pattern with respect to* T *and minSup if* $s(X) \geq minSup$*.*

An example of trajectory frequent pattern is, for instance: $\{ReligiousPlace_{[weekend]}, Restaurant_{[weekend]}\}$ (s=0.07). This pattern expresses that 7% (support) of the trajectories stop at both religious places and restaurants on weekends. This pattern is a set with two items: $\{\text{ReligiousPlace}_{[weekend]}, \text{Restaurant}_{[weekend]}\}$. ReligiousPlace and Restaurant correspond to the spatial part of each item, while *weekend*

represents the time dimension of the item, i.e., when the moving objects stayed at these places. More details about *items* are presented in section 3.1.

Mining Trajectory *Sequential* Patterns. The sequential pattern mining technique differs from frequent patterns in the order as the items occur in a trajectory. In sequential patterns the order as the items occur plays the essential role. It corresponds to the relative order between items in the trajectories, but not necessarily the absolute order of the items. A Sequential trajectory pattern is a sequence of movements between items that have support higher than minimum support.

Definition 4. *Trajectory Sequential Pattern. Let T be a set of semantic trajectories. A sequence of items $X = \langle x_1, x_2, \ldots, x_n \rangle$, ordered in time, is a trajectory sequential pattern with respect to T and $minSup$ if $s(X) \geq minSup$.*

An example of sequential pattern is: $Work_{[morning]}$, $ShoppingCenter_{[afternoon]}$, $Gym_{[afternoon]}$ *(s=0.08%).* This pattern expresses that these three items occur in this temporal relative order in 8% of the trajectories.

Mining Trajectory Association Rules. Given an implication of the form $X \Rightarrow Y$ (s) (c), where X and Y are disjoint sets of items, the support s of the rule $X \Rightarrow Y$ is given as $s(X \cup Y)$. The confidence c is given as $s(X \cup Y)/s(X)$.

Definition 5. *Trajectory Association Rule. Let T be a set of semantic trajectories and X and Y be disjoint sets of items. An implication of the form $X \Rightarrow Y$ is a trajectory association rule with respect to T, $minSup$ and $minConf$ if $s(X \cup Y) \geq minSup$ and $s(X \cup Y)/s(X) \geq minConf$.*

Trajectory association rules represent associations among items. Considering our example of semantic trajectory shown in Figure 1 (2) and considering that a minimum number of trajectories would go from *home* to *work* and from there to a *Gym*, we could have an association rule such as:

$$Home_{[night]}\, Work_{[morning]} \Rightarrow Gym_{[afternoon]}\ \text{(s=0.10) (c=0.50)}$$

This rule expresses that in 10% of the trajectories the moving object has stayed at home at night, at a working place in the morning, and at a gym in the afternoon. Furthermore, among the trajectories that stop at home at night and at a working place in the morning, 50% also stop at a gym in the afternoon.

3 From Trajectory Conceptual Modeling to Data Mining

The model proposed in this paper is an extension of the model of stops and moves proposed by Spaccapietra to deal with trajectories from a semantic point of view. Therefore, our model is for mining *semantic trajectory patterns.* Figure 3 shows the proposed framework, represented as a UML class diagram. We represent the spatial attributes by the attribute *the_geom*, instead of stereotypes, and the time dimension by attributes. The classes in dark gray represent the data mining concepts, which are the main contributions of this paper. Including these concepts,

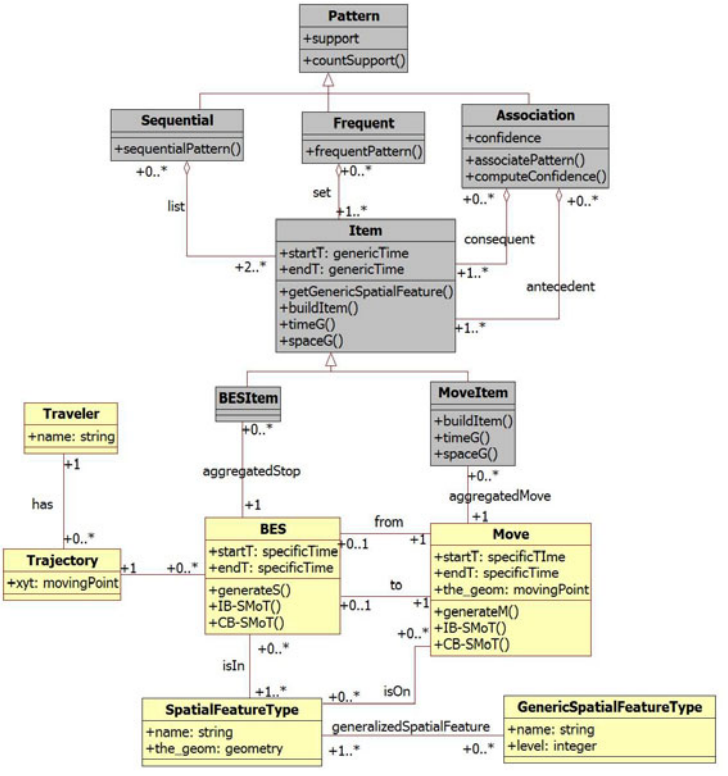

Fig. 3. A framework for Modeling Trajectory Data and Trajectory Patterns

that will be explained later in this section, we first extend the concept of *stops and moves* with two methods to automatically instantiate stops and moves, IB-SMoT and CB-SMoT, implemented and validated respectively in [13] and [16]. The method *generateS* computes stops based on one of the previous methods, while *generateM* generates the moves. These methods are defined, respectively, in the classes *BES* (Begin and End Stop) and *Move*. The model can easily be extended to support different methods to instantiate either stops or moves.

It is important to notice that in the original model of stops and moves the BES entity includes the begin and the end of the trajectory, while in our framework this entity corresponds to stops only. Therefore, the minimal cardinality between the entities Trajectory and BES has been changed from two to zero.

The attributes *startT* and *endT* represent the time in which Stops (BES) and Moves started and ended. They represent a specific time (e.g. 10:01). These attributes will be automatically instantiated by the methods that compute stops or moves (IB-SMoT and CB-SMoT). Both methods are supported by the first Semantic Trajectory Data Mining Query Language (ST-DMQL) [8], implemented in PostGIS (spatial database extension for PostgreSQL) and Weka[22], in order to instantiate the proposed model.

Patterns can be extracted from both stops and moves, but not directly. Very rarely patterns will be obtained directly from the data without aggregation. For instance, it is almost impossible that two or more moving objects would stop at *IBIS Hotel* exactly at the same time interval (e.g. 8:01 to 9:17). Patterns have to be extracted from data at aggregated levels. Therefore, we introduce the concept of *item*, which has been originally introduced in [8].

An *item* is the element that will be considered in the mining process, i.e., it is the element that will compose the pattern, which in trajectories is an aggregated stop or move. We call this element *item* because this term is well known in the classical data mining literature. In the context of trajectories, an *item* represents a set of information. While in transactional data mining an item is a single attribute like *milk* or *bread*, a trajectory item is a complex object that contains information about both *space* and *time*.

Stops and moves are defined and instantiated at the lowest granularity level, which we call as *feature instance* granularity. For data mining, these instances have to be generalized in order to discover patterns. Therefore, the classes *BESItem* and *MoveItem* specialize the *Item* and are respectivelly associated to one and only one stop or move.

Every *item* has the *time* information represented by the attributes *startT* and *endT*, and space information. The time information is a generalized time such as *weekend, weekday, month, year* and so on, while the time information at *BES* and *Moves* is a specific time, like 10:30. The *space* information of the item is the name and the type of the spatial feature related to the stop. The method *buildItem* will build the item, aggregating both space and time information using the methods *spaceG* and *timeG*, respectivelly. These three methods are redefined for the *MoveItem*, since an item composed by a move is a bit different from a stop. A *MoveItem* is build on two stops. These methods are detailed in Section 3.1.

The method *getGenericSpatialFeature* generalizes stops or moves related to spatial features through concept hierarchies. It will get from the class *GenericSpatialFeatureType* the respective level of a concept hierarchy to be considered for data mining.

The proposed model supports the three patterns described before: frequent patterns, sequential patterns, and association rules. The patterns have the attribute *support*, which is inherited by all three types of patterns, each of which is a subclass of pattern. Patterns are always extracted from items, therefore, patterns are aggregations of items, but with different meanings. A *sequential pattern* is a *list* of items ordered in time. A *frequent pattern* is a *set* of items, i.e., a set of stops or moves at any order. A *trajectory association rule* is an association pattern that is composed by a set of items that correspond to the *antecedent* of the rule and a set of disjoint items corresponding to the *consequent* of the same rule. The association pattern has an extra attribute, the *confidence* of the rule.

Notice in the model that for trajectory conceptual modeling the user may easily add new attributes, methods, or spatial feature types to stops (BES), according to the application requirements. The concepts corresponding to the pattern tasks, however, remain the same, i.e., for any application domain that is modeled using the extended model of stops and moves, the user will be able

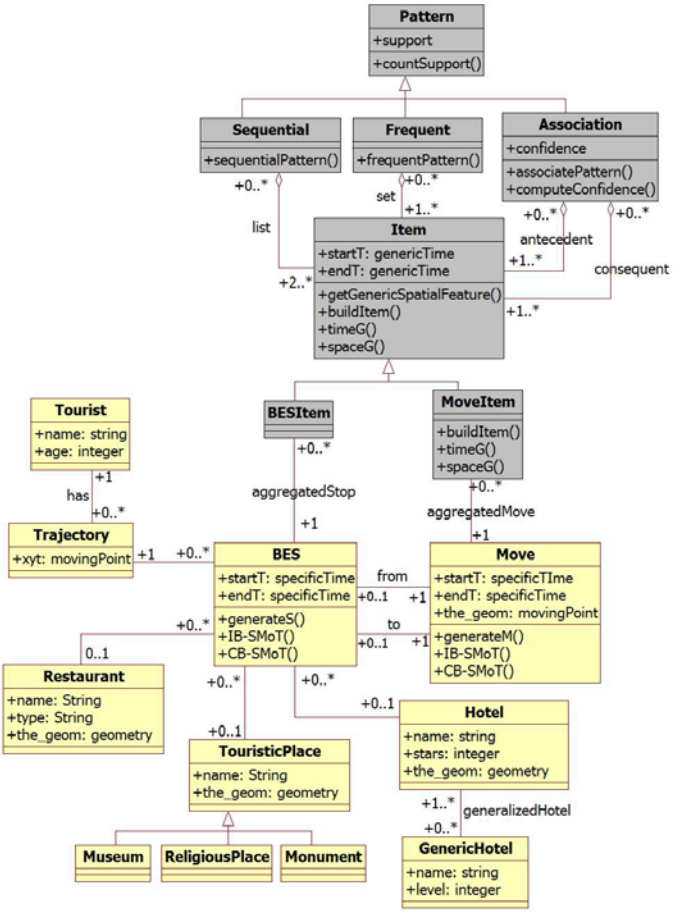

Fig. 4. A conceptual model for a trajectory tourism application

to extract frequent patterns, sequential patterns and association rules from trajectories without having to change the model.

Figure 4 shows an example of an instantiation of the proposed framework for the context of a tourism application. The travelers are tourists that have one or more trajectories. Every trajectory has zero or more stops (BES) that are interconnected by moves. In this example, a stop (BES) of a trajectory can occur at a hotel (Spatial Feature 1), restaurant (Spatial Feature 2), or a touristic place (Spatial Feature 3), which can be of type *museum*, *monument* or *religiousplace*. Hotel has a concept hierarchy associated (GenericHotel), which can be used for mining stops that occur at hotel at different granularities, such as 1 star, 2 stars, 3 stars and so on. Notice that any other relevant spatial feature that could generate a stop of a trajectory could be associated to *BES*, such as airport, shopping centers, and so on.

As can be seen in Figure 4, the main contribution of the proposed model is that for data mining, independently of the spatial features related to BES in the conceptual schema, the way as the item is build and the patterns are modeled remains standard. It is a *design pattern* for semantic trajectory pattern modeling.

3.1 Defining an Item

The *item* is the attribute that will be considered in the mining step, i.e., it is an element that will compose the pattern. In the context of trajectories, an *item* can be defined as: (i) *Name* - the name of the spatial feature where the moving object has passed and stayed for a minimal amount of time (e.g. Hotel, IbisHotel); (ii) *NameStart* - the name of the stop (spatial feature) with the time that the moving object has entered in the place (e.g. Hotel[morning], LouvreMuseum[10:00-12:00]); (iii) *NameEnd* - the name of the stop (spatial feature) and the time in which the moving object has left the place (e.g. Hotel[afternoon], LouvreMuseum[14:00-16:00]); and (iv) *NameStartEnd* - the name of the stop with the time period in which the moving object has respectively arrived and left the place (e.g. Hotel[morning][afternoon], LouvreMuseum[10:00-12:00][14:00-16:00]).

The item is build with the method *buildItem* defined in the class *Item*. It receives as an argument the *itemType*, which defines which space and time information will be considered in the item. According to the application, the time may be aggregated in different levels. In a traffic management application, it might be interesting to define rush hour at intervals like [7:00-9:00], [12:00-14:00], and [17:00-19:00]. In a tourism application, it might be interesting to aggregate time in [morning], [afternoon], and [evening]. The same is true for the space dimension. For instance, one may want to extract patterns at instance level (e.g. Ibis Hotel) or at feature type level (e.g. Hotel). In the following section we describe how the methods *spaceG* and *timeG*, defined in the class *Item*, work.

3.2 Time and Space Granularity (timeG and spaceG)

The time granularity is defined by the method *timeG*. This method generalizes a timestamp into different granularities, according to the user's interest. More specifically, it converts a timestamp (e.g. 08:12)into a semantic discretized *label* as [07:00-09:00].

The method *timeG* is very powerful because it allows the user to discretize the time dimension in several granularities. Please see [8] for details.

The language ST-DMQL that implements *timeG* provides some pre-defined time granularities, which are: WEEKEND-WEEKDAY, YEAR, MONTH, SEASON, and DAY-OF-THE-WEEK. For these specific granularities the user can use the parameter *pre_defined_TG* for the function *timeG*. This will automatically transform the time dimension of either stops or moves in the specified granularity. If the user is interested in specific time intervals, then he can decide for the parameter *user_defined_TG*, where he can specify a given time interval and choose a label, like for instance, *14:00-18:00* as *afternoon*.

The *space* granularity of either stops or moves is managed by the method *spaceG*. Two granularity levels can be automatically generated, without the need of any specification of background knowledge like concept hierarchies or ontologies. These granularities are called *feature instance* (e.g. Centrum Hotel) and *feature type* (e.g. Hotel). With the method *spaceG* the user can, for instance, set as default the granularity *type* or *instance*, and specify intermediate granularity levels defined in a concept hierarchy for some specific spatial feature types. Granularity levels different from *feature type* and *feature instance* are obtained from the class *GenericSpatialFeatureType*, with the method *getGenericSpatialFeatureType*, which has as a parameter the name of the generic spatial feature to be retrieved(e.g. Hotel). To better understand the *spaceG* function please see [8].

4 Evaluating the Proposed Model

To evaluate the proposed model we use the semantic trajectory data mining query language [8] implemented in PostGIS and Weka to both instantiate and query trajectory patterns. To facilitate the comprehension of the examples presented in this Section, we show how the conceptual model can be seen in a logic level. The relations of stops (that correspond to BES in the conceptual model) and moves have the following schema:

```
STOP (Tid integer, Sid integer, SFTname string,
     SFid integer, startT timestamp, endT timestamp)

MOVE (Tid integer, Mid integer, SFT1name string,
      SF1id integer, SFT2name string, SF2id integer,
      startT timestamp, endT timestamp, the_move geometry)
```

The attribute *SFTname* corresponds to the name of the spatial feature type and the attribute *SFid* is the identification of the instance (e.g. Ibis) of the spatial feature type (e.g. Hotel) in which the moving object has stopped. The attributes *SFT1name* and *SF1id* correspond to the stop in which a move starts, while the attributes *SFT2name* and *SF2id* represent the stop where the move finishes.

The relation which stores frequent patterns and sequential patterns has the following schema:

```
frequentPattern/
sequentialPattern (Pid integer, pattern itemSetType, support real)
```

The attributes are respectively the identifier of the pattern, which is a sequential number, the pattern itself, which we represent and store as a *nested relation* [23], and the support of the pattern. The type *itemsettype*, which represents a nested relation, is defined by the following structure:

```
itemSetType (SFT1name string, SF1id integer, SFT2name string,
  SF2id integer, startT string, endT string)
```

The pattern schema is similar to the structure of the move relation, because it stores both patterns of stops and patterns of moves. The main difference is

that instances of a pattern represent aggregated information of stops and moves, and not the instances of stops and moves.

When the pattern is a move pattern, then *SFT1name, SFT1id* and *SFTname2, SFT2id* are two stops that form a move pattern. If the pattern is generated from stops, then *SFTname2, SFT2id* will be null.

Concerning trajectory association rules, the association pattern relation has the following structure:

```
associatePattern (Pid integer, antecedent itemSetType,
 consequent itemSetType, support real, confidence real)
```

The attributes are respectively the identifier of the rule, the antecedent of the rule, the consequent, the support, and the confidence. In this type of pattern both antecedent and consequent are stored as nested relations, and therefore both have the structure of the *itemSetType*. These structure is what allows quering patterns just as data, as will be shown in Section 4.2.

4.1 Instantiating Patterns

Considering the context of a tourism application presented in Figure 4, let us suppose that the user wants to know the types of places most frequently visited by tourists on weekdays and weekends. This question can be answered by the following simple query to extract frequent patterns:

```
SELECT frequentPattern (itemType=NameStart, timeG=WEEKEND-WEEKDAY,
                        spaceG=[type,GenericHotel=1], minsup=0.15)
  FROM stop
```

In this query, the item that composes the pattern will contain the name of the stop and the time in which the stop started. This is expressed by the parameter *itemType=NameStart*. The time granularity (timeG) is weekday and weekend, while the granularity of space will be at the feature type level for all stops, except for those that occur at hotels. For hotels, a concept hierarchy *GenericHotel* is used at the granularity level 1, and therefore touristic stops at hotels will be aggregated by types like, for instance, 2StarsHotel, 3StarsHotel, etc...

The query will generate patterns over stops that appear in at least 15% of the trajectories, and the output of this query will be frequent patterns as, for instance:

$\{4StarsHotel_{[weekend]}, Museum_{[weekend]}, Restaurant_{[weekend]}\}$ (s=0.16)

Now let us suppose that the user wants to know the sequences of moves that occur most frequently in the morning and in the evening. This question can be answered by the following query:

```
SELECT sequentialPattern (itemType=NameEnd, timeG=[8:00-12:00 AS morning,
                    18:00-23:00 AS evening],spaceG=instance, minsup=0.03)
  FROM move
```

In this query a move has to occur in at least 3% of the trajectories in order to generate a pattern. The pattern will contain the name of the move and the time

that the move finishes, and time will be aggregated in morning and evening. In this data mining query there is no aggregation on the name of the stop, i.e., the patterns will contain the places where the moves happened. A sequential pattern that will be generated by this query may be the following:

$$\{IbisHotel\text{-}NotreDame_{[morning]},\ EiffelTower\text{-}IbisHotel_{[evening]}\}\ (s{=}0.04)$$

Considering now a traffic management application (the conceptual model is not shown because of space limitation), where traffic jams (very low speed regions) are the important places in trajectories. The stops have been instantiated with the method CB-SMoT and the user, for instance, may ask a question like: which are the relationships among stops (traffic jams) in a city at different time periods during rush hours? This question may be answered by the following query:

```
SELECT associatePattern(itemType=NameStart, spaceG=instance,
                        timeG=[07:00-08:00, 08:01-09:00,
                        16:00-17:00,17:01-18:00,18:01-19:00],
                        minsup=0.1, minconf=0.40)  FROM stop
```

This query will generate association rules that have at least 10% support and at least 40% confidence. Each *item* in the rule, either in the antecedent or the consequent, will contain the name of the stop and the time period in which the stop starts (itemType=NameStart). The stop granularity will be at the feature instance level for all stops (spaceG=instance), and the time are specific hours within the rush hours (timeG). From this query the user may obtain association rules like, for instance:

$$AvGrandArmee_{[07:00-08:00]} \Rightarrow AvChampsElisees_{[08:01-09:00]}\ (s{=}0.06)\ (c{=}0.40)$$

4.2 Querying Trajectory Patterns

Since the patterns are stored as database relations, any query can be performed. Any filter or constraint may be applied directly over the patterns. For instance, considering the tourism application, let us suppose that the user is interested in association patterns which have *weekend* as the time dimension in the antecedent of the rule. A query like the following can be performed:

```
SELECT *
  FROM associatePattern
 WHERE antecedent.startT='weekend' or antecedent.endT='weekend'
```

The power of storing the discovered patterns for further analysis allows complex queries, where the geometry of either stops or moves can be used to filter the patterns. For instance, suppose the user is interested in how many moves that cross the bridge Pont Neuf are contained in sequential patterns. This query will join three relations: sequential patterns, moves, and bridge, as follows:

```
SELECT count(m.*)
  FROM sequentialPattern s, bridge b, move m
 WHERE s.pattern.SFT1name=m.SFT1name AND s.pattern.SF1id=m.SF1id AND
       s.pattern.SFT2name=m.SFT2name AND s.pattern.SF2id=m.SF2id AND
       b.name='Pont Neuf' AND intersects(m.the_geom,b.the_geom)
```

5 Conclusions

In this paper we presented a general framework for conceptually modeling trajectory patterns. The proposed model is an extension of the conceptual model proposed by Spaccapietra for modeling trajectories of moving objects from a semantic point of view. We extend this model to support semantic trajectory patterns, that are extracted from aggregated stops and moves of trajectories. The model provides to the user semantic trajectory sequential patterns, trajectory association rules and trajectory frequent patterns. The proposed model can be implemented using the ST-DMQL, a semantic trajectory data mining query language developed for defining, mining and querying trajectory patterns at different space and time granularity levels. The proposed model significantly reduces the preprocessing tasks for data mining, which normally are the most effort and time consuming tasks. Furthermore, it facilitates pattern filtering in postprocessing. The proposed framework is a result of several works including different data mining case studies and experiments performed by two of the authors on trajectory data modeling and trajectory data mining. As future works we are investigating the extension of the model to support trajectory clustering and trajectory classification.

Acknowledgements

This work was supported by the Brazilian agencies CAPES (Prodoc Program), FAPESC (CP005/2009) and CNPq (550891/2007-2, 573871/2008-6 and 307588/2008-4).

References

1. Pyle, D.: Data Preparation for Data Mining. Morgan Kaufmann, San Francisco (1999)
2. Wang, H., Zaniolo, C.: Atlas: A native extension of sql for data mining. In: Barbará, D., Kamath, C. (eds.) SDM. SIAM, Philadelphia (2003)
3. Chen, C.X., Kong, J., Zaniolo, C.: Design and implementation of a temporal extension of sql. In: Dayal, U., Ramamritham, K., Vijayaraman, T.M. (eds.) ICDE, pp. 689–691. IEEE Computer Society, Los Alamitos (2003)
4. Malerba, D., Appice, A., Ceci, M.: A data mining query language for knowledge discovery in a geographical information system. In: Meo, R., Lanzi, P.L., Klemettinen, M. (eds.) Database Support for Data Mining Applications. LNCS (LNAI), vol. 2682, pp. 95–116. Springer, Heidelberg (2004)
5. Boulicaut, J.F., Masson, C.: Data mining query languages. In: Maimon, O., Rokach, L. (eds.) The Data Mining and Knowledge Discovery Handbook, pp. 715–727. Springer, Heidelberg (2005)
6. Calders, T., Lakshmanan, L.V.S., Ng, R.T., Paredaens, J.: Expressive power of an algebra for data mining. ACM Transactions on Database Systems 31(4), 1169–1214 (2006)
7. Han, J.: Mining knowledge at multiple concept levels. In: CIKM, pp. 19–24. ACM Press, New York (1995)
8. Bogorny, V., Kuijpers, B., Alvares, L.O.: St-dmql: a semantic trajectory data mining query language. International Journal of Geographical Information Science 23(10), 1245–1276 (2009)

9. Bogorny, V., Kuijpers, B., Alvares, L.O.: Reducing uninteresting spatial association rules in geographic databases using background knowledge: a summary of results. International Journal of Geographical Information Science 22, 361–386 (2008)
10. Alvares, L.O., Bogorny, V., de Macedo, J.F., Moelans, B., Spaccapietra, S.: Dynamic modeling of trajectory patterns using data mining and reverse engineering. In: Twenty-Sixth International Conference on Conceptual Modeling - ER2007 - Tutorials, Posters, Panels and Industrial Contributions, November 2007. CRPIT, vol. 83, pp. 149–154 (2007)
11. Spaccapietra, S., Parent, C., Damiani, M.L., de Macedo, J.A., Porto, F., Vangenot, C.: A conceptual view on trajectories. Data and Knowledge Engineering 65(1), 126–146 (2008)
12. GeoPKDD, P (2006), `http://wwww.geopkdd.eu`
13. Alvares, L.O., Bogorny, V., Kuijpers, B., de Macedo, J.A.F., Moelans, B., Vaisman, A.: A model for enriching trajectories with semantic geographical information. In: ACM-GIS, pp. 162–169. ACM Press, New York (2007)
14. Baglioni, M., de Macêdo, J.A.F., Renso, C., Wachowicz, M.: An ontology-based approach for the semantic modeling and reasoning on trajectories. In: ER Workshops, pp. 344–353 (2008)
15. Bogorny, V., Wachowicz, M.: A framework for context-aware trajectory data mining. In: Cao, L., Yu, P.S., Zhang, C., Zhang, H. (eds.) Data Mining for Business Applications, pp. 225–240. Springer, Heidelberg (2008)
16. Palma, A.T., Bogorny, V., Kuijpers, B., Alvares, L.O.: A clustering-based approach for discovering interesting places in trajectories. In: ACMSAC, pp. 863–868. ACM Press, New York (2008)
17. Alvares, L.O., Oliveira, G., Heuser, C.A., Bogorny, V.: A framework for trajectory data preprocessing for data mining. In: International Conference on Software Engineering and Knowledge Engineering, pp. 698–702 (2009)
18. Wolfson, O., Xu, B., Chamberlain, S., Jiang, L.: Moving objects databases: Issues and solutions. In: Rafanelli, M., Jarke, M. (eds.) SSDBM, pp. 111–122. IEEE Computer Society, Los Alamitos (1998)
19. Güting, R.H., de Almeida, V.T., Ding, Z.: Modeling and querying moving objects in networks. VLDB Journal 15(2), 165–190 (2006)
20. Brakatsoulas, S., Pfoser, D., Tryfona, N.: Modeling, storing, and mining moving object databases. In: IDEAS, pp. 68–77. IEEE Computer Society, Los Alamitos (2004)
21. Parent, C., Spaccapietra, S., Zimanyi, E.: Conceptual Modeling for Traditional and Spatio-Temporal Applications – The MADS Approach. Springer, Heidelberg (2006)
22. Frank, E., Hall, M.A., Holmes, G., Kirkby, R., Pfahringer, B., Witten, I.H., Trigg, L.: Weka - a machine learning workbench for data mining. In: Maimon, O., Rokach, L. (eds.) The Data Mining and Knowledge Discovery Handbook, pp. 1305–1314. Springer, Heidelberg (2005)
23. Abiteboul, S., Schek, H.-J., Fischer, P.C. (eds.): NF2 1987. LNCS, vol. 361. Springer, Heidelberg (1989)