

# Automatic Generation of Trajectory Data Warehouse Schemas

Nouha Arfaoui and Jalel Akaichi

**Abstract** A mobile object is a spatial object that changes the form and the location permanently over the time. Each displacement creates a trajectory that reflects the evolution of its position in space during a given time interval. It generates, then, a huge amount of trajectory data that are stored into trajectory data warehouse because it is the only tool that can analysis the historical trajectory data. In this work, we focus on the design of trajectory data warehouse schema and we propose automating this task to reduce human intervention since it is done manually and requires good knowledge of the domain. To achieve this goal, firstly, we automate the extraction of trajectory data mart schemas from a moving data base. Then, we merge them to get the trajectory data warehouse schema using a new schema integration methodology that is composed by schema matching and schema mapping.

**Keywords** Trajectory data warehouse schema · Trajectory data mart schema · Schema integration · Animal movement

## 1 Introduction

Thanks to the emergence of location-aware devices, mobile communication systems, GPS, etc., a huge amount of TrD is generated once following the moving object. The latter is defined as a spatial object that changes the form and the location permanently over the time. It is distinguished by spatial components that evolve over time. The movement of such object creates a trajectory that corresponds to “the evolution of the position (perceived as point) of an object that is moving in space during a given time interval in order to achieve a given goal” [1]. Trajectory Data (TrD) is stored

---

N. Arfaoui (✉) · J. Akaichi  
Department of Information Systems, BESTMOD - Institut Suprieur de Gestion,  
King Khalid University, Tunis, Tunisia  
e-mail: arfaoui.nouha@yahoo.fr

J. Akaichi  
e-mail: jalel.akaichi@kku.edu.sa

into Trajectory Data Warehouse (TrDW). The traditional data warehouse [2] and the spatial data warehouse [3] are not created to deal with this kind of data because of its continuous evolution over the time and space.

To ensure a good TrDW, it is important to start by generating its corresponding schema. Many software tools exist, but they describe the schemas manually making this task tedious, error-prone and time-consuming [4].

In this work, we focus on the generation of TrDW schemas, and we opt then for the bottom up approach. It starts by generating automatically the Trajectory data mart schemas (TrDM) from the moving data bases. Then, it automates the generations of the TrDW by merging the TrDW schemas using the new schema integration methodology. The latter generates the mapping rules taking into account the semantic similarities of the elements of the schemas and the existing conflicts. Then, it transforms the generated rules into queries and executes them.

In order to well understand the proposed approach, we take a herd of animals as a moving object. It moves continuously to satisfy its nutritional needs. By this way, it can develop a grazing habit reflecting a form of adaptation and complex dynamic interactions with its environment especially the vegetation. The intensive exploitation of natural resources provokes the rarefaction of species of high value pastoral which influences badly the ecological balance and aggravates the fragility of the environment until a large part of those areas risk irreversible desertification. The different data collected from this movement should be stored into TrDW. This can help later to understand the habits and then to intervene, for example, by changing its future trajectory to protect the vegetation into specific place.

The outline of this work is as following. In Sect. 2, we focus on the state of the art where we present some work related to the TrDW. In Sect. 3, we define the model of the trajectory of a herd. Then, we move to the generation of TrDM from a moving data base. Next, we define the followed steps to build the final schema of TrDW. We finish this work with conclusion and future work.

## 2 Related Works

In the literature, many works have been proposed to deal with TD and/or TrDW. In the following, we present some of them.

Indeed, the authors in [5], introduce the notion of TrD and proposes the TrDW to storage such data to transform the trajectory raws into useful information.

Braz et al. in [6], introduce its model and its corresponding issues.

The authors, in [7], discuss the loading phase of the DW that has to deal with overwhelming streams of trajectory observations.

In [8], a framework is proposed to transform the traditional data cube model into a trajectory warehouse.

The authors, in [9], present a framework for improving the design and the implementation of TrDWs for analyzing traffic data.

In [10], the authors develop an application to receive the stream data set, to store and compute the pre-aggregation values and to present final results in order to reveal the knowledge about the trajectories.

Also, there are different works that propose a real application of TrDW. For example, we can mention, [11] who study the human movements' behavior, [12] who followed the displacement of the seal, [13] who model the trajectory of a mobile hospital, etc.

### 3 Trajectory Data Modeling

In this section, we focus on modeling TrD. We use, then, the Entity-Relationship (ER) model to present the different components that can influence any trajectory as well as their relationships (Fig. 1). Indeed, a herd is a set of animals belonging to

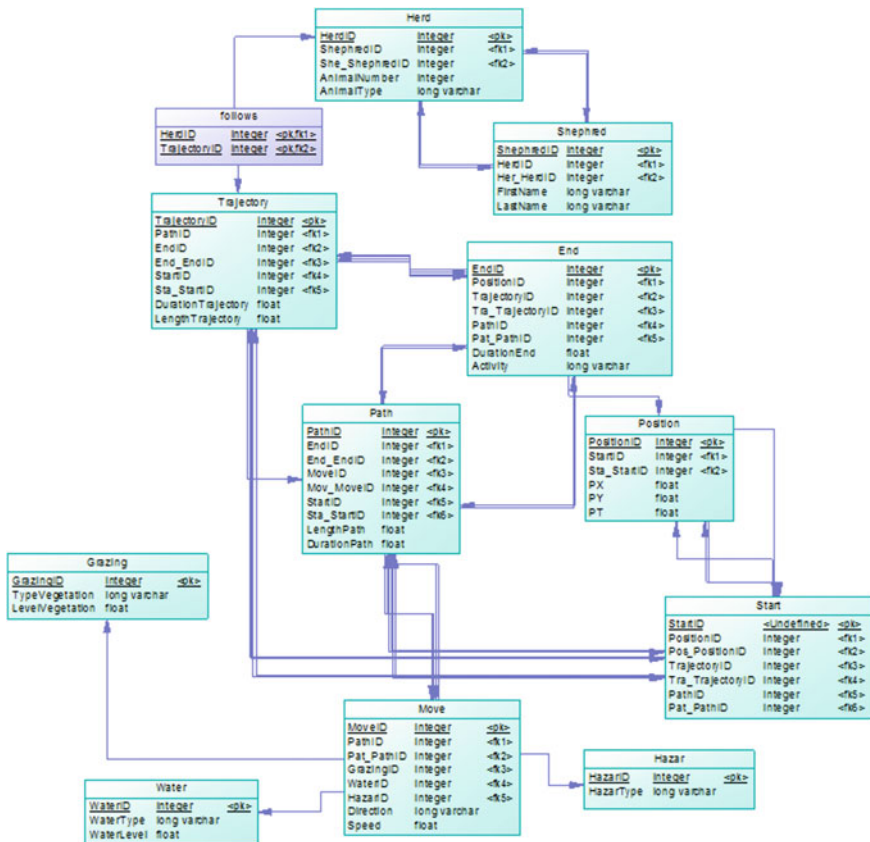


Fig. 1 A moving data base schema

the same type. It can be supervised by a shepherd in the case of domestic animals or not in the case of wild animals. The herd follows one specific trajectory. In this case, a trajectory is considered as a set of paths having different directions, lengths, and duration. So the duration of the trajectory is the sum of the duration of its composed paths. Each path is defined by “start” and “end” and the same thing concerning the trajectory which has a start that represents the start of the first path and an end that represents the end of the last path. Talking about the start, it gives information about the start position thanks to “PX” and “PY” and start time thanks to “PT”. Concerning the end, it gives information about the final position thanks to “PX” and “PY” and the final time that is presented by an interval. The end corresponds to a stop where the animals can have different activities such as eating, drinking, or resting. So that, we have  $TStart_{(N)} = TEnd_{(N-1)} + \text{duration of stop}_{(N-1)}$ .

Between a “start” and an “end”, the herd moves. This movement is in function of the food so it happens generally close to the water especially during the dry seasons and in grazing to allow the animal eating. The animal can find during their displacement some hazards which may be natural (sea, river, lake, mountain or desert) or artificial (road, complex, or railway).

## 4 Generating Trajectory Data Mart Schemas

In order to generate the schema of the Trajectory Data Mart (TrDM), we start with the definition of the multidimensional elements. Then, we extract them from the ER model, and we build the schema of the TrDM. The different steps are detailed in the following.

### 4.1 Potential Multidimensional Elements

- Potential Fact Table: it corresponds to a table with numerical attributes and/or table with n-ary relationship.
- Potentials Measures: they correspond to the numeric attributes. Concerning the aggregation functions, they are specified by the user.
- Potential Dimension Tables: they correspond to the tables that are directly linked to the potential facts extracted in the previous step.
- Potential Attributes: they correspond to the existing elements into the table including the primary key that becomes the primary key of the dimension table and the foreign keys that are used to determine the hierarchies.

### 4.2 Steps for Generating Trajectory Data Mart Schemas

To generate the multidimensional schema, we construct trees that are used as intermediate models. They contain the extracted multidimensional elements. Next, each tree is transformed to multidimensional schema.

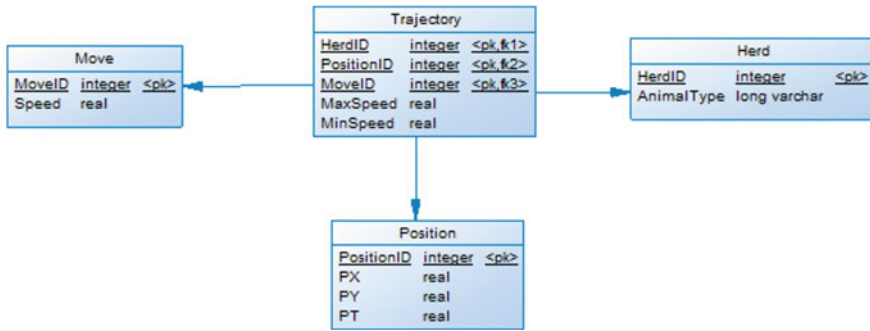


Fig. 2 First example of TrDM schema

**Step 1: Build the trees from ER model**

From the ER model, we extract the entities (Ef) having n-ary relationships with other entities and/or those having numerical attributes. They represent the potential facts. Every “Ef” becomes the root of the tree. The number of trees corresponds to the number of “Ef” entities.

From the ER, we extract the entities (E) that are directly linked to “Ef” corresponding to the potential dimensions. They form the first level of the tree.

For every “E”, we extract the set of entities that are bonded via “-to-one” relationship.

**Step 2: Transform the trees to multidimensional model**

- The fact is created having as name “Trajectory”.
- The existing numeric attributes become the potential measures.
- The root becomes a dimension keeping its attributes.
- The identifier of the root becomes a foreign key in the new fact table.
- The measures are defined by an aggregation functions that are specified by the user.
- The nodes that are directly linked to the roots are transformed to dimensions keeping their attributes and their identifiers.

Figure 2 presents an example of TrDM schema generated from Fig. 1.

**5 Building Trajectory Data Warehouse Schema**

The construction of TrDW schema is about merging the set of TrDM schemas. This task is achieved by using the new schema integration methodology that deals mainly with star schemas. It is composed by schema matching and schema mapping. The schema matching is used to extract the semantically closest elements as well as the conflicts, and presents them as mapping rules. The schema mapping transforms the

generated rules to queries and executes them to merge the schemas regardless their heterogeneity source.

In order to use the schema integration three problems must be solved which are: data model heterogeneity, structural heterogeneity, and semantic heterogeneity.

To overcome the first and the second problems, we propose using the star schema as a common model. Concerning the semantic heterogeneity, it will be solved using the schema matching and mapping.

There are two strategies behind the creation of a global schema using the schema integration, which are “bottom-up” [14] and “top- down” [15]. In order to merge the schema, we opt for the bottom-up and more precisely the binary ladder strategy [16]. It is about merging two schemas each time until having one single schema at the end.

### 5.1 Schema Matching

The schema matching is used to find semantic correspondence between the elements of the two schemas and presents them as mapping rules. The latter will be used to facilitate the merging of schemas. This phase takes as input two or many schemas to get as output set of mapping rules facilitating the merge of the schemas.

The schema matching is composed by the following steps:

- Categorization: It specifies the category of each element. This reduces the risk of error which provides a gain of time.

– Schema 1 (Fig. 2):

Fact: {Trajectory}; FactKeys {HerdID (integer), PositionID (integer), MoveID (integer)}.

Measure: {MaxSpeed (real), MinSpeed (real)}.

Dimension, DimensionKey and Attribute: {Move {MoveID (integer), Speed (real)}, Position{PositionID (integer), PX (real), PY (real), PT (real)}, Herd {HerdID (integer), AnimalType (long varchar)}}.

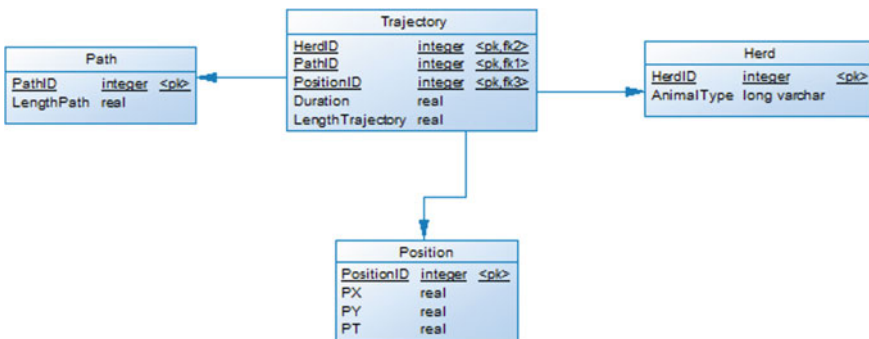


Fig. 3 Second example of TrDM schema

– Schema 2 (Fig. 3):

Fact: {Trajectory}; FactKeys {HerdID (integer), PositionID (integer), PathID (integer)}.

Measure: {Duration (real), LengthTrajectory (real)}.

Dimension, DimensionKey and Attribute: {Path {PathID (integer), Length-Path (real)}, Position{PositionID (integer), PX (real), PY (real), PT (real)}, Herd {HerdID (integer), AnimalType (long varchar)}}.

- Comparison and construction of the matrix of similarity: It is to assign a coefficient to the elements belonging to the same category. To compare the names of schemas elements, we should take into consideration their linguistic matching. For this reason, we suggest the following formula (1) that calculates the degree of similarity of two elements belonging to the same category. It returns “1” if the two elements are similar, and “0” if they are not.

$$DeSim(e1, e2) = DeId(e1, e2) + DeSy(e1, e2) + SeTy(e1, e2) + DePost(e1, e2) + DePre(e1, e2) + DeAbb(e1, e2) \tag{1}$$

With:

- DeId (e1, e2) = 1 if e1 and e2 are identical and 0 else.
- DeSy (e1, e2) = 1 if e1 and e2 are synonymous, and 0 else.
- DeTy (e1, e2) = 1 if e1 and e2 are the same with the existence of typing error, and 0 else.
- DePost (e1, e2) = 1 if one of the two elements is the postfix of the other, and 0 else.
- DePre (e1, e2) = 1 if one of the elements is the prefix of the other, and 0 else.
- DeAbb (e1, e2) = 1 if one of the elements is the abbreviation of the other, 0 else.

At this level, we take, also, into consideration the comparison of the types of the elements. If they are different we need human intervention to specify what should be kept.

In the following, we apply this step to measures, dimensions and attributes because the two schemas have the same fact name “Trajectory” that will be the same in the final schema.

Table 1 presents the similarity matrix used to compare the measures. According to the values of “Max”, there is no similar measures. In such case, the final schema contains all of them.

**Table 1** Measure comparison

Measure	MaxSpeed (real)	MinSpeed (real)	Max
Duration (real)	0	0	0
LengthTrajectory (real)	0	0	0

**Table 2** Dimension tables comparison

Dimension	Position	Move	Herd	Max
Position	1	0	0	1
Path	0	0	0	0
Herd	0	0	1	1

**Table 3** Attributes comparison

Attributes	HerdID (integer)	AnimalType (long varchar)	Max
HerdID (integer)	1	1	1
AnimalType (long varchar)	1	1	1

Table 2 presents the similarity matrix used to compare the dimension tables. “Move” and “herd” have two similar dimension tables. The rest are different.

Table 3 presents the similarity matrix used to compare the attributes belonging to the dimension “Herd”. All the attributes are similar and they have the same types. We keep then for each similar couple, only one. The same thing is done to “herd”.

- Generation of the mapping rules: The rules visualize the conditional relationships between the instances of different categories. They are expressed as: “If Similar (X, Y) then Keep (X or Y) and Save (X, Y)”, with:
  - X and Y: They are the two elements that belong to the same category.
  - Similar ( ): It is a function that specifies if the two inputs are similar or not. It uses the similarity matrix determined in the previous step.
  - Keep ( ): It keeps one of the two elements of the input that will be included in the generated schema.
  - Save ( ): It saves the rule.

An example:

- If Similar (AnimalType (long varchar), AnimalType (long varchar)) Then Keep (AnimalType (long varchar) or AnimalType (long varchar)) and Save (AnimalType (long varchar), AnimalType (long varchar))

The different rules are stored into rules database. But, before making the comparison, it is crucial to start by verifying if there is any rule in the database that contains the two elements “X” and “Y”. If it is not the case, the similarity measure is calculated to determine their similarity degree. If they are similar, the database is updated, else, the next elements is treated.



## 5.2 Schema Mapping

Once the mapping rules are extracted; we move to the next where they are transformed into queries. In the following, we present some of them.

Concerning the fact tables, the two schemas have the same name so the resulting schema contains also the same fact table name.

- Query = “Insert into Schema (idSchema) values (“+schemaId+”)”;
- Query1 = “Insert into Fact (FactName, idSchema) values (‘Trajectory’, “+ schemaId +”)”;

Since the fact tables are similar, we move to compare their measures. If the measures are similar we choose one of them, if not, we add both of them.

- Query2 = “Insert into Measure (MeasureName, MeasureType, idFact) values (‘MaxSpeed’, “+” ‘real’ “+ factId +”)”;

For the dimensions, we compare their names, if they are similar, we move to their attributes. We keep then the similar ones and we add the different. If the dimensions are different we keep them with their attributes without modification.

- Query3 = “Insert into Dimension (DimensionName, DimensionPKName, DimensionPK-Type, idSchema) values (‘Move’, ‘MoveID’, ‘integer’, “+schemaId +”)”;

Figure 4 presents the star schema related to the trajectory of the herd. It is the result of the application of the previous rules.

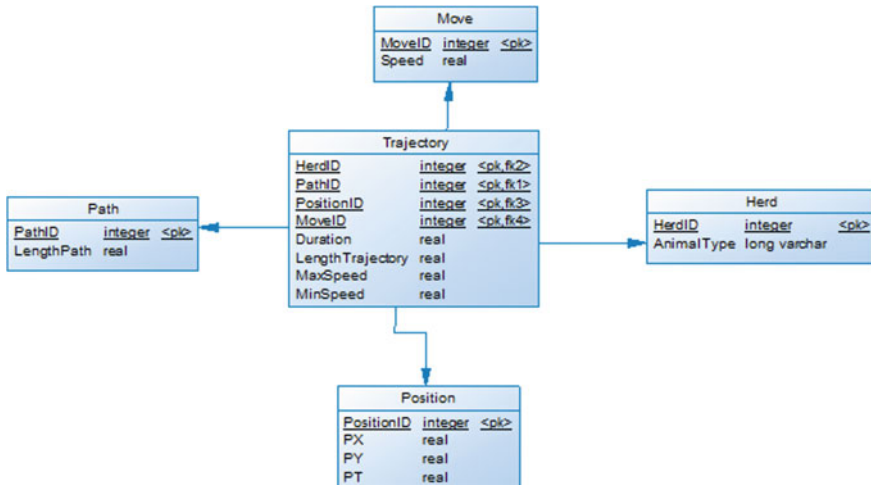


Fig. 4 The TrDW schema

## 6 Conclusion

In this work, we focus on the displacement of the herd that we took as example to study the trajectory of a moving object. The last one is defined as a spatial object that changes the form and the location permanently over the time. During its movement, the herd generates huge amount of TrD. We use the TrDW as a powerful tool to storage this kind of data to transform it later to useful knowledge. The problem at this level concerns the generation of its schema. For this reason, we proposed, in this work, an approach that serves to automatically generate the TrDW schema. Indeed, from the mobile data base, the multidimensional schemas are generated. They correspond to TrDM schemas. The latter are merged to build the final schema of the TrDW using the schema integration methodology that is composed by two steps. The first step extracts the similar elements from two schemas to generate the mapping rules. The second step transforms the rules into queries and executes them to merge the schemas.

As future work, we propose using some data mining techniques to predict the future movement of animals so we will be able to intervene for example by changing the direction of the movement in case where there is a threat to the existing vegetation.

**Acknowledgments** I would like to thank everyone.

## References

1. Spaccapietra, S., Parent, C., Damiani, M.-L., de Macedo, J.A.F., Porto, F., Vangenot, C.: A conceptual view on trajectories. Technical report, Ecole Polytechnique Federal de Lausanne (2007)
2. Inmon, W.H.: Building the Data Warehouse. Wiley (1992)
3. Malinowski, E., Zimnyi, E.: Implementing spatial data warehouse hierarchies in object-relational DBMSs. In: 9th International Conference on Enterprise Information Systems, ICEIS 2007, Funchal, Madeira, Portugal. INSTICC Press (2007)
4. Feki, J., Nabli, A., Ben-Abdallah, H., Gargouri, F.: An Automatic data warehouse conceptual design approach. In: Wang, J. (ed.) Encyclopedia of Data Warehousing and Mining, 2nd edn. (2008)
5. Leonardi, L., Marketos, G., Frentzos, E., Giatrakos, N., Orlando, S., Pelekis, N., Raffael, A., Roncato, A., Silvestri, C., Theodoridis, Y.: T-Warehouse: visual olap analysis on trajectory data. In: 26th International Conference on Data Engineering (ICDE'10), Los Angeles, USA (2010)
6. Braz, F., Orlando, S., Orsini, R., Roncato, A., Silvestri, C.: Approximate aggregations in trajectory data warehouses. In: 23rd International Conference on Data Engineering Workshop, pp. 536–545. IEEE (2007)
7. Orlando, S., Orsini, R., Raffael, A., Roncato, A., Silvestri, C.: Trajectory data warehouses. *J. Comput. Sci. Eng.* **1**(2), 211–232 (2007)
8. Leonardi, L., Marketos, G., Frentzos, E., Giatrakos, N., Orlando, S., Pelekis, N., Raffael, A., Roncato, A., Silvestri, C., Theodoridis, Y.: T-warehouse: visual olap analysis on trajectory data. In: 26th International Conference on Data Engineering (ICDE), pp. 1141–1144. IEEE (2010)

9. Campora, S., de Macedo, J.A.F., Spinsanti, L.: St-toolkit: a framework for trajectory data warehousing. In: 14th AGILE Conference on Geographic Information Science (2011)
10. Braz, F. J., Orlando, S.: Trajectory data warehouses: proposal of design and application to exploit data. In: GeoInfo, pp. 61–72 (2007)
11. Giannotti, F., Trasarti, R.: Mobility, data mining and privacy: the GeoPKDD paradigm. In: 14th SIAM Conference on Mathematics for Industry (MI09), pp. 10–18, San Francisco, California (2009)
12. Sakouhi, T., Akaichi, J., Malki, J., Bouju, A., Wannous, R.: Inference on semantic trajectory data warehouse using an ontological approach. In Foundations of Intelligent Systems, pp. 466–475. Springer International Publishing (2014)
13. Oueslati, W., Akaichi, J.: Trajectory data warehouse modeling based on a Trajectory UML profile: medical example. In: International Work-Conference on Bioinformatics and Biomedical Engineering, pp. 1527–1538 (2014)
14. Batini, C., Lenzerini, M., Navathe, S.B.: A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.* **18**(4), 323–364 (1986)
15. Lenzerini, M.: Data integration: a theoretical perspective. In: Symposium on Principles of Database Systems (PODS02), pp. 233–246. ACM (2002)
16. Batini, C., Lenzerini, M.: A methodology for data schema integration in the entity relationship model. *IEEE Trans. Softw. Eng.* **SE-10**, 650–663 (1984)