Structuring, indexing, querying and visualizing moving objects in a DBMS

Marco Baars, Peter van Oosterom, Edward Verbree, Ben Gorte
TU Delft
Delft, the Netherlands
m.t.l.baars@student.tudelft.nl, Oosterom@geo.tudelft.nl, e.verbree@geo.tudelft.nl
b.g.h.gorte@lr.tudelft.nl

SUMMARY

Modeling, analyzing, and monitoring the movements of point-objects is becoming more and more important. Database Management Systems make it possible to manage large spatial datasets. In conventional spatial database applications, data is assumed to be less dynamic. Therefore, it is hard to store continuously moving objects. In this research, the available methods investigated with respect to their ability to structure, index, query and visualize moving objects in the context of a Spatio-Temporal Database Management System. A very generic model for moving point-objects is presented together with a number of views suitable for further analysis of the data. For two case studies, some of these methods will be implemented to see how it works and what are the shortcomings of these methods.

KEYWORDS: *Spatio-Temporal, DBMS, structuring, indexing, querying and visualizing*

1. INTRODUCTION

Temporality is an inherent aspect of geo-information and since a number of years it became a topic in several researches. Nowadays more and more applications of spatiotemporal GISs become important. For instance with respect to Cadastral issues (Van Oosterom, 2000), national road databases [Heres, 2000] or the detection of traffic jams (NRC Handelsblad, 2003). In the recent literature, the modelling of moving objects in a Geo-DBMS context is a subject of study (Pfoser and Jensen, 2003). Detection of traffic jams is an example application. Geo-DBMSs make it possible to manage large spatial datasets in databases that can be accessed by multiple users at the same time. These spatial datasets usually contain 2D data, while more and more applications depend on 3D data (Arens, 2003). Moving objects can also be seen as 3D data (x-position, y-position, time) or 4D data (x,y,z,time).

Continuous movement poses new challenges to database technology. In conventional (spatial) databases, data is assumed to remain constant unless it is explicitly modified (Saltenis et al, 1999). This statement causes efficient structuring methods for the data. Also querying these datasets is an issue that has to be taken into account. Examples of queries on moving-object databases are: which object has the highest speed, what's the average distance between two objects, which objects are in a specific area (polygon) in a certain time interval, etc. Answering these queries requires efficient storage and indexing method in case the dataset is large. Visualising the moving objects and answers on the queries is also an important issue.

Two main types of applications based on on moving point-objects can be recognized: the distinction between real time monitoring of moving objects and modelling the data afterwards (post processing). The main question of this research can be described as follows:

> ***How do you structure, index, query and visualise dynamic point clouds in the context of a Spatiotemporal Database Management System?***

In this question, “dynamic point clouds” can be seen as moving objects. This means that only their positions will change in time and the remaining attributes (shape, colour, length, etc) are not relevant (or not considered). The term “Spatiotemporal Database Management System” is more generic than for instance “Moving Objects DBMS”. In fact, a Moving Object DBMS can be seen as one of the Spatiotemporal DBMSs.

After this introduction, this paper starts with some backgrounds of spatiotemporal modelling and indexing. In the third section, a generic model will be introduced for modelling moving objects in a geo-DBMS. After that, two case studies will be described in section 4. The first case deals with moving objects in a database, where the data is post processed and the second one describes real time modelling, indexing, querying and visualizing. The cases (and methods) are implemented and tested, within Oracle 9i Spatial. This paper will be finished in section 5 with conclusions and suggestions for future work.

2. BACKGROUND

In this section, some background of spatiotemporal modelling will be discussed. In 2.1, it starts with a framework, described by Langran in 1992. Parts of this framework deal with modelling and indexing spatiotemporal data. In 2.2, several methods for modelling spatiotemporal data will be described. Paragraph 2.3 deals with indexing methods for spatiotemporal data.

2.1 A framework for spatio-temporal data

According to (Langran, 1992), “five technical requirements will drive the development of a temporal GIS: a conceptual model of spatial change, treatment of aspatial attributes, data processing logistics, a spatiotemporal data access method and efficient algorithms to operate on the spatiotemporal data.”

The first part of this technical framework, the conceptual model, can be described as “the configuration of information, as it will be represented to the computer. It defines the entities, attributes and relationships to be portrayed; it also defines the operations to be performed and the constraints to be enforced. (Langran, 1992)”. Treatment of attributes, that are not spatio-temporal, is outside the focus of this research. The colours, shape, value, name, etc. are assumed to stay constant. The objects, in this research can be represented as points, where other attributes than position or time, do not matter.

The third technical requirement, mentioned by Langran is the “data processing logistics”. This deals with the global partitioning, error control, and updating the stored data. It depends on the purpose of your system: real-time or post-processing the data, the use of a DBMS or not, the collection of only current data or also storing the past spatiotemporal data, archiving (very) historic data, etc. The temporal access methods, the fourth technical requirement, will be described later in this abstract. A large amount of storage and indexing methods that can handle spatiotemporal data is available. Indexing is important to get a faster access of your data. It decreases query-response times.

The last technical requirement, efficient algorithms, is also a very important one. Perhaps it is efficient to save the results of queries that are used very often. An example is the speed of moving objects. In many data structures this attribute is not available in the base table, but it can be derived. For instance in Oracle 9i Spatial, it is possible to define materialized “views”. This is a method that saves the query results of pre-defined queries (see section 3).

2.2 Modelling spatio-temporal data

A trajectory is the path described in space and time by a moving object. Such a trajectory can be represented in different ways. Several researchers describe models to structure these data into a DBMS. Some of these methods will be introduced shortly in this paragraph.

(Vazirgiannis and Wolfson, 2001) describe a model for moving objects on a road network. They show the need for a small and robust set of predicates with high expressive power, suitable for realistic

implementation based on off the shelf DBMS technology. The underlying model they use consists of three parts:

- A map, where each tuple in the relation represents a city block, i.e. the road section between 2 intersections.
- The moving object. The route of a moving object is specified by giving the starting address or (x,y) coordinate, the starting time and the ending address or (x,y) coordinate. The attributes that can be added in this table are things like the driver, color, weight, length, etc.
- The trajectory. This is a relation with the attributes (*sequence#,x,y,t,b*), where the tuple [i,*(x,y), ti,b]* is the i'th intermediate point on the objects route. The attribute *b* is a boolean, which is false, if the i'th tuple is a begin or endpoint of a trajectory.

This model can be used to structure data of moving objects afterwards (post-processing). In this paper we will mainly focus on the most fundamental part of the model, the trajectory (as other information may not always be available).

(Wolfson et al, 1998) describe the MOST-data model. They make a distinction between static attributes and dynamic attributes (for instance the x-position of a moving object). Data in a DBMS are assumed to be constant and not continuously updated. The solution is representing the location as a function of time; it changes as time passes, even without an explicit update. Every dynamic attribute is represented with three attributes (A.value, A.updatetime and A.function). This model can be easily used for objects that move freely in space (like aeroplanes). For objects along a winding route, Wolfson et al extended their solution with two more attributes for every dynamic attribute.

Another aspect is focussed on in the work on the Spatio-Temporal Topological Operator Dimension (Marchand et al, 2003). Marchand et al describe a method to implement spatiotemporal topological operators (like BEFOR, DURING, CROSSES) in multidimensional databases (MDDBs) through a hierarchy of topological operators representing spatial and temporal relationships between instances. In this approach, the trajectory is the primitive.

The last method mentioned here is the Discrete Spatio-Temporal Trajectory Based Moving Objects Database System, introduced by (Meng and Ding, 2003). The main idea is that predicting future trajectories is possible if the moving object submits its moving plans to the information system. In this way, the past, current and future trajectory can be modelled. In this approach, the trajectory is the main primitive.

2.3 Indexing methods

Numerous researches have been developing spatiotemporal access methods (storage and indexing) in order to support efficient spatio-temporal queries. (Mokbel et al, 2003) give an overview of the history of the spatio-temporal indexing methods. The different indexing methods in this article are organised in three parts, the first part is about indexing the past, the second part is about indexing the current time and the last part is about indexing methods that deal with spatiotemporal data from the future.

On the origin of the most indexing methods for spatiotemporal data, is the R-tree. The R-tree indexing method is based on Minimum Bounding Rectangles (MBR) in the 2D-case or Minimum Bounding Boxes (MBB) in the 3D-case. An R-tree index stores this MBR or MBB that encloses each geometry in a spatial dataset. This MBR or MBB is used to reduce the computational complexity in spatial queries and is defined along the axes. The advantage of using an R-tree index is that the irregular sized MBRs or MBBs can fit the objects in the real world, in contrary to the subdivision of space in the quadtree. The disadvantage is that the MBRs or MBBs can be much larger than the objects itself. This increases the load in the exact computation (the second step in solving a query), because more objects need to be processed (Arens, 2003). An example of an R-tree can be found in figure 1.

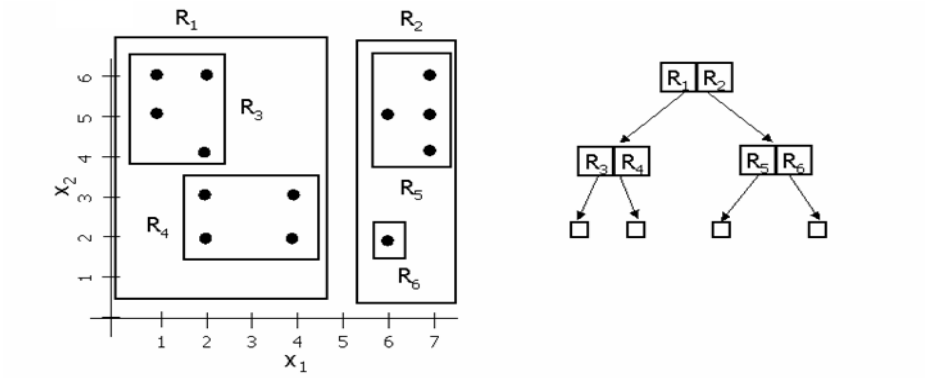

Figure 1 An example of a 2D R-tree

For indexing the past trajectories, the indexing methods are numerous. In (Mokbel et al, 2003), for instance, the RT-tree, the 3D R-tree and the TB-tree are mentioned. For indexing current positions, only a few indexing methods are available. There are not that many methods available that can index the current and the past trajectories as well. Only the 2+3 R-tree and the 2-3 R-tree are described in that article. The main idea is that there are two separate R-trees, one to index the 3-dimensional past (x,y,t) and one for the 2-dimensional current (x,y).

There are also methods available for indexing the current and future predicted trajectories. One of the most important ones is the TPR-tree, which employs the idea of parametric bounding rectangles. The lower bound of the conservative bounding rectangle is set to move with the minimum speed of the enclosed points, while the upper bound is set to move with the maximum speed of the enclosed points. There are variations available on this method, for instance the PR-tree or the VCI R-tree.

In (Pfoser and Jensen, 2003), a method is described that indexes data of objects that move along a network. Their main idea is to index the 3D-space (x,y,t) into two 2-dimensional spaces (linearized network-location, time). Pfoser and Jensen did a performance test with this indexing method. Their experiments show that the lower the complexity of a network, the more likely the mapping approach proves to be beneficial over indexing the data in 3D space.

3. GENERIC MODEL

As can be concluded a number of alternative models for moving point objects be constructed. Two aspects can characterize these models. First, the time dimension is either separate or integrated with the spatial dimension (in such a case a 2D point and time become a 3D point in the spatio-temporal model). Secondly, for a single object the observations are either stored in separate records (with the sampled point in time) or in one record with a polyline attribute (kind of interpolation between the time samples). In the polyline case the time again can be separate or integrated with the spatial point data: 2D spatial polyline with separate attributes for time-stamps or 3D spatiotemporal polyline). For an overview, see figure 2.

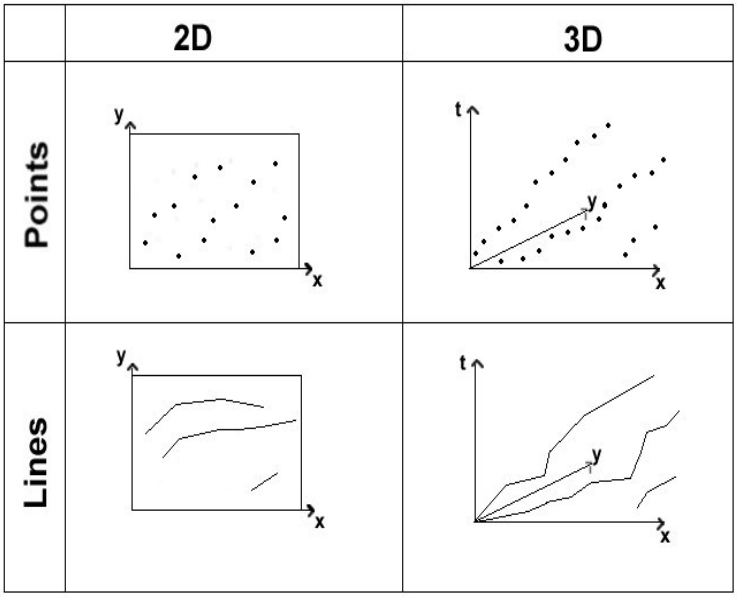

Figure 2. Top row with sampled time: 2D points (time separate) or 3D points (integrated time). Bottom row with interpolated time: 2D polylines (time separate) or 3D polylines (time integrated)

All models can be converted to each other (and could in that sense be considered equivalent) and most likely this can be realized with DBMS views (using spatial operators). In practice they may differ with respect to dynamic behaviour (suitable or not for dynamically growing data in case of real time monitoring) and ease of use during analysis and visualization. This depends on the specific platform used for the implementation (e.g. Informix has time-series data type support, which is then more efficient than a separate record for every moment in time). Anyhow, from the conceptual point of view the models are quite similar and we will use the most elementary to illustrate spatiotemporal modelling. The base table, not assuming any explicit sequence, looks like (in a kind of pseudo SQL):

create table mov_obj(id, t, pos) /* primary key is pair (id, t) */

Based on this base table a number of views can be defined to have easy access to derived attributes such as speed and acceleration (first a view to obtain the next time stamp of a given object):

create view move_obj_succ as

select t1.*, t2.t as next_t

from mov_obj t1, mov_obj t2

where t1.id=t2.id and t2.t=(select min(t) from move_obj where t>t1.t);

create view speed_obj_vw as
select t1.id, t1.t, t1.next_t, t1.pos, dir=diff(t2.pos-t1.pos), speed=distance(t2.pos, t1.pos)/(t2.t-t1.t)
from mov_obj_succ t1, mov_obj_succ t2
where t1.id=t2.id and t2.t=t1.next_t;

create view accel_obj_vw as
select t1.id, t1.t, t1.next_t, t1.pos, t1.speed, accel=(t2.speed-t1.speed)/(t2.t-t1.t)

```
from speed_obj_vw t1, speed_obj_vw t2
where t1.id=t2.id and t2.t=t1.next_t;
```

Using this view one can now derive statistic such as average speed. This can either be grouped by id (of vehicle), position and time. For grouping positions we assume a function (pos_group) to translate an xy(z)-coordinate in an encoding of a position group; e.g. parts of roads. Similar for time we assume a time group function (time_group) to get usable time units (e.g. a block of 15 minutes). One can also make combinations of grouping when computing averages (e.g. by position and time):

```
create view avg_speed_obj_vw as
select id, avg(speed) from speed_obj_vw group by id;

create view avg_speed_pos as
select position=pos_group(pos), avg(speed) from speed_obj_vw group by pos_group(pos);

create view avg_speed_time as
select time=time_group(t, 15), avg(speed) from speed_obj_vw group by time_group(t,15);

create view avg_speed_pos_time as
select position=pos_group(pos), time=time_group(t,15), avg(speed)
from speed_obj_vw group by pos_group(pos), time=time_group(t,15)
```

Of course, one could compute all kinds of other statistics in a similar way; for example the minimum or maximum speed (for the same types of groupings) or compute average, minimum, maximum acceleration (and again grouped by the different options: id, time or position or combinations of these). Quite another type of view may be used to analyse how close the cars are together (as a possible indication of traffic jams. In this case the ordering of the base table data is a bit more difficult compared to ordering on the linear time scale (as the space is two-dimensional). However, we assume that the next car should be found ahead of the current driving direction (and also driving in the same direction).

```
create view dist_objs as
select p1.id1, p2.id, p1.t, p1.pos, p1.speed, distance=length(diff(p1.pos, p2.pos))
from speed_obj_vw p1, speed_obj_vw p2
where p1.t=p2.t and p1.dir = p2.dir and p2.id=
(select closest(id, p1.pos) from speed_obj_vw);
```

All these views may be nice from a functional point of view. However, without the proper storage and indexing the performance may be poor. Important aspects to consider are spatiotemporal clustering (so the physical ordering of the data) and spatiotemporal indexing (efficient selections of the record addresses based on spatiotemporal (range) queries). In Oracle the initial implementation would use an indexed organized table (on the key id, t) in order to obtain ordering of the data based on id and time. Further a 2D r-tree index (on position) or 3D r-tree functional index (on position and time) is the used for initial spatio-temporal indexing.

Analysis may show that it is impossible to answer all (important) queries based on a single physical ordering the base table and in the ultimate situation redundant data storage may be considered. Oracle offer ‘materialized views’ (not part of the SQL92 standard) to implement this in a effective manner. By default a materialized view is only refreshed on demand (by calling a specific Oracle refresh procedure). However, it is possible to specify that the materialized view must be refreshed automatically after the transaction on the base table is committed, for example:

```
create materialized view move_obj_succ_mv1
refresh fast on commit
as select t1.*, t2.t as next_t
  from mov_obj t1, mov_obj t2
  where t1.id=t2.id and t2.t=(select min(t) from move_obj where t>t1.t);
```

Especially in highly dynamic situations (rapid data growth) this is not without problems (as it may not be very efficient to update the materialised view after every transaction). In such a situation it may be more effective (depending on the application) to collect a number of transactions and to refresh the materialized view only periodically, for example (the first time it is refreshed after 45 minutes, then after every 30 minutes):

```
create materialized view move_obj_succ_mv2
refresh start with 'now+45 min' next '30 min'
as select t1.*, t2.t as next_t
  from mov_obj t1, mov_obj t2
  where t1.id=t2.id and t2.t=(select min(t) from move_obj where t>t1.t);
```

Tuning the generic model, by choosing the appropriate storage and index structures and (materialized) views, makes it efficient for a given application, that is, a set of typical queries for a given (static or dynamic data set).

4. CASE STUDIES

Two case studies will be presented: one that can be considered post processing the spatiotemporal data after collection (static) and one that is intended for real rime processing (dynamic). The difference is that the second one requires a dynamic data structure in order to handle the ever-growing data set (large volumes).

The first case is based on a dataset, originally used to test traffic models. The data comes from a range of photos of a highway taken from a helicopter. Every 0.1 second, a picture is made and every car in the photo is tracked. By structuring this dataset in a DBMS, it will be possible to answer questions like "Which cars drive above the speed limit", "Select all the cars that are within five meters of its predecessor", or "Give all the cars in a certain polygon in a certain time period", etc.

A second case will be a dataset of 60 taxis driving in the city of Rotterdam. Structuring these data in a DBMS will make it possible to answer queries like "On time T, on place P, how long do I have to wait for a taxi" or "What is the mean time a taxi cab is empty", etc.

4.1 Traffic data from helicopter observations

In cooperation with groups for Photogrammetry and Remote Sensing and for Traffic Management within Delft University we are populating a database with measurements of highway traffic during circumstances of congestion. The measurements are obtained by automatic analysis of image sequences that are taken with a digital high-resolution camera from a helicopter. A highway section of approximately 500m is monitored during an extended period of time (say 1 hr) with a recording frequency of 10 frames per second. On a crowded or congested highway this may lead to several millions of car observations.

The goal of this effort is to extract parameters for so-called microscopic traffic flow models from the observations by querying the database. Microscopic models are those that take individual car driver's behaviour into account, (including, therefore, variability in behaviour between different drivers), as opposed to macroscopic models, where cars are treated as uniform entities like molecules in a streaming fluid. Since microscopic models are not yet fully understood (they are subject of study in the Traffic Management group), the database system should offer the flexibility to extract a large and partly unforeseen variety of parameters. The parameters should reflect drivers behaviour concerning, for example, acceleration and deceleration in reaction to manoeuvres of the vehicle in front, lane changing behaviour, gap acceptance, etc.

The moving objects database is successfully implemented in an Oracle 9i Spatial, following the principles of the generic model described in section 3. Base tables with the four geometry types from figure 2 are created from the original dataset. These tables are indexed and views with speed, distances and

accelerations are created. With this set of tables and views, some interesting queries can be considered. With aggregation functions and spatial operators, as available in Oracle Spatial, some macroscopic variables (such as intensity, density and average speed) can easily be computed.

An interesting question (on a microscopic point of view) is whether it is possible to calculate and visualize vehicles that do not keep enough distance to their predecessor. The Dutch government formulated in a campaign that every driver has to keep two seconds distance from the vehicle driving in front of him. By creating a rectangle in front of the moving point object with a width of 2 meters and a length of 2 seconds (see figure 3) and by using a spatial operator (SDO_RELATE), the vehicles that do not meet this condition can be calculated.

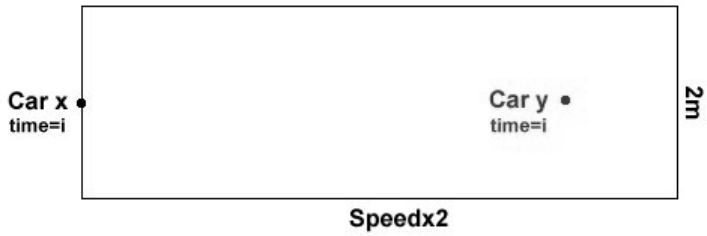

Figure 3. A rectangle in front of car x with width 2m and length 2 seconds at current speed (unit is s * m/s = m, which is correct) where no other vehicles are allowed at time=i.

In 'Cartography, visualization of spatial data' (1996), Kraak and Ormeling discuss the use of dynamic variables opposed to traditional graphic variables, which are used to represent spatial data within individual frames. According to them an appropriate dynamic graphic can be used if the spatial data it represents is dynamic by nature. But as Bertin has stated, the user should be aware of: 'movement only introduces one additional variable, it will be dominant, it will distract all attention from the other (graphical) variables' (Bertin, 1983). MacEachren (1994) has defined the dynamic variables: duration, order, rate of change, frequency, display time and synchronization. These dynamic variables can be seen as additional tools to design an animation.

Knowing this theory, ESRI has provided an extension on ArcGIS to handle spatio-temporal data: the Tracking Analyst (ESRI, 2004). Its functionality is comparable to its predecessor, available as an extension to ArcView 3.x, although the possibilities to serve real-time data to the application are now part of ArcIMS. It is possible to display point and track data (real time and fixed time) by temporal symbology renderers (shape and size), symbolize time by colour (show the aging of data), actions (based on attribute or spatial queries) and highlighting. Besides, the user is in control by the interactive playback manager to start, pause, stop and rewind the animation (for an example, see figure 4). Note that the colours of the objects are based on their id (which does not add much information), therefore another attribute may be more interesting to use as colour attribute basis; e.g. acceleration (red: slow down, green: speed up, and blue: about equal speed)

As with most cartographical visualizations the dynamic map could be used for exploratory data analysis to reveal unknown or difficult to recover information from the data alone. One could state database queries to expose this kind of information, but some more qualitative and descriptive facts like trends, are only to be retrieved by carefully inspection of the animation. For this the dynamic variables should be appropriate and careful used, with the notification of Bertin in mind.

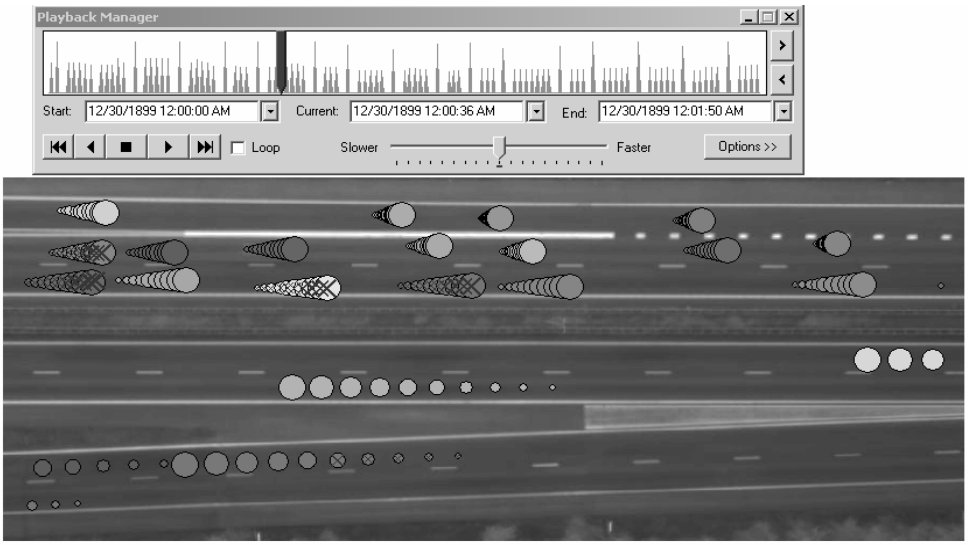

Figure 4. The last 10 timestamps of vehicles (Note crosses if they do not keep enough distance).

4.2 GPS data from moving vehicles

The generic model from section 3 will become more interesting, when real time (dynamic) data is used. The Advise Office for Geo-information and ICT, part of the Dutch Ministry of Traffic and Public Works, has a dataset available for research, where GPS tracking data is stored of a number of taxis, operating near Rotterdam. For a period of two years, data has been captured and stored in files.

This dataset can be simulated as real time data with an application like ArcIMS tracking server. In the future, the generic model will be tested with this real time simulation. Issues like efficient storage and indexing, data consistency and performance will become more important in real time than in the post processing case. Possible queries will be "What are popular places?", "Takes the taxi driver the same route as a route planner?" or "How often drives a taxi driver faster than the maximum speed?"

5. CONCLUSIONS AND FUTURE WORK

The first of the five technical requirements mentioned by (Langran, 1992) is the conceptual model. In this paper, 4 representations of the basic model for moving point-object (trajectories) are described. The fourth technical requirement, the spatio-temporal access method, is also described in this paper. It seems clear that there are numerous methods available to index the past, but for the current time (real time), just a few methods can be mentioned.

A very generic model for spatio-temporal point objects is presented. This model consists of a simple base table (in which data of every application should fit). Together with a number of (materialized) views, this offers a flexible and efficient method to model moving object data in a geo-Database Management System. It is fast and keeps your dataset consistent.

In a case where data was already collected (post processing), the generic model is tested and some queries and visualisations are made. In the future, the generic model will also be tested for a real time dataset. Further, the current cases have been implemented with standard Oracle functionality, such as the 3D R-

tree for spatio-temporal indexing. Alternative, more specific spatio-temporal indices may be implemented in the future.

ACKNOWLEDGMENTS

We would like to thank Dutch Ministry of Traffic and Public Works for making their data available for this research. Further, we would like to thanks our colleagues, Wilko Quak and Theo Thijssen, for their comments on section 3 and specifically on the (materialized) views.

LITERATURE

Arens, C.A., *Maintaining Reality – Modelling 3D Spatial objects in a Geo DBMS using a 3D primitive.* Delft. Master thesis, GIS Technology, TU Delft, 2003.

Bertin, J., *Semiology of graphics*. Madison, Wisc.: University of Wisconsin Press, 1983.

ESRI, http://www.esri.com/software/arcims/tracking_server.html (February 2004)

ESRI, *http://www.esri.com/software/arcgis/arcgisxtensions/trackinganalyst/index.html*, 2004

Heres, L. *Hodochronologics: History and time in the National Road Database.* In: Time in GIS: Issues in spatio-temporal modelling. Delft, Nederlandse Commissie voor Geodesie. P46-56. 2000.

Kraak, M.-J. and Ormeling, F. J., 1996, Cartography, visualization of spatial data, (London: Addison Wesley Longman).

Langran, G. *Time in Geographic Information Systems.* London, Taylor & Francis, 1992.

MacEachren, A.M., 1994, *Visualization in modern cartography: Setting the Agenda*. In Visualization in Modern Cartography (A. M. MacEachren and D. R. F. Taylor, Oxford, UK: Pergamon), pp. 1-12.

Marchand, P., Brisebois, A., Bédard, Y., Edwards, G., *Implementation and evaluation of a hypercube-based method for spatio-temporal exploration and analysis*. In: Journal of the International Society of Photogrammetry and Remote Sensing (ISPRS) theme issue "Advanced techniques for analysis of geo-spatial data" dans la catégorie "multi-scale hierarchies of spatial operators". 2003.

Meng, X., Ding, Z., *DSTTMOD: A Discrete Spatio-Temporal Trajectory Based Moving Object Database System.* DEXA2003, LNCS 2736 (Springer Verlag), p 444-453, Prague, 2003.

Mokbel, M.F., Ghanem, T.M., Aref, W.G., *Spatio-temporal Access methods*, IEEE Data Engineering Bulletin, 26(2) p 40-49, June 2003.

NRC Handelsblad, 9-9-2003, article *Mobiele filemeldingen* (in Dutch)

Oosterom, P. van, *Time in Cadastral Maps*. In: Time in GIS: Issues in spatio-temporal modelling. Delft, Nederlandse Commissie voor Geodesie. P 36-45, 2000.

Pfoser, D and Jensen, C.S., *Indexing of Network-Constrained Moving Objects*, in: Proceedings of the Eleventh International Symposium on Advances in Geographic Information Systems, New Orleans, 8 pages, 2003.

Saltenis, s., Jensen, C.S., Leutenegger, S.T., Lopez, M.A., Aalborg. *Indexing the Positions of Continuously Moving Objects.* A TIMECENTER Technical report. Aalborg, 1999.

Vazirgiannis, M., Wolfson, O., *A spatiotemporal Model and Language for Moving Objects on Road Networks*. In: Proceedingsof the 7th international symposium on advances in Spatial and Temporal Databases, pages 20-35, 2001.

Wolfson, O., Xu, B., Chamberlain, S., Jiang, L., *Moving Object Databases: Issues and Solutions.* Proceedings of the 10th International Conference on Science and Statistical Database Management. Capri, Italy. Pages 111-122. 1998.