Spatiotemporal Data Model for Web GIS

J. Konopásek, O. Gojda, D. Klimešová

***Abstract*—**Current Internet standards provide us many options for processing and manipulation with spatial data but difficulties occur when temporal data are needed to be applied. Traditional relational databases widely used on the Internet are not designed to store temporal data and this fact limits us in developing more enhanced geographic models. This article is comparing several data models and approaches for managing temporal data, their suitability especially for practical usage, e.g. performing difficult queries, analyzing managed data as well as supporting construction of specific data formats needed by different web applications.

***Keywords*—** Spatio-temporal databases, temporal data modelling, time stamping data, web GIS

I. Introduction

In current world most information and data we collect, analyze and use are in some way connected in time and space. In last few years spatial and temporal analysis and prediction became much more desired, needed, developed and used. Databases and data models that store and allow us to use temporal and spatiotemporal data have to gone through many changes and transformations, that allow us to better handle temporal data – use advanced queries and perform more complex data analysis with ease. At least that is true for advanced databases tailored for specific software.

Many of applications and solutions, which had been used by specialized departments and users, recently moved closer to the wide public usage thanks to the Internet. A special term was settled down for GIS applications provided via Internet – WebGIS. It allows the end-users to take advantage of availability of such a system practically anywhere without purchasing and installing specialised GIS software in order to get or manage spatial information [1], [2].

The operating and supporting parts of GIS have to be adapted for the web constraints. Basically there isn't any problem with storing stable data of described objects. For this purpose are used SQL databases which are easily applicable by standard web services. SQL represents standard relational data model which is able to illustrate a single state of the modelled world, called a snapshot. If a database is updated from previous state to the new one means to lose the previous state. There exist many applications which need to have access also to the past and sometimes to the future states of the modelled world. This definitely holds for the most geographic information systems which use spatial data changing in time very often [3].

If we want to implement a WebGIS and present spatiotemporal data on the Internet we have to have an access to specialized application on private servers or we have to revert to using traditional relation databases with SQL queries and with all good and bad that comes with it. This article tries to make best of current data models and trends of temporal data storage within borders set by current Internet standards.

II. Managing Temporal Data

A. Different dimensions of time

While working with temporal databases we can come across different representations, resp. dimensions of time. We can see so called "valid time" present in most temporal databases – valid time is time that sets validity of stored data in our modelled space, resp. space modelled by stored data. Typically we come across valid time set in past or present, but it can also be set in feature. In temporal database we typically use valid time in form of interval or set of intervals (most usually a date when information became valid and date when it expired), but depending on use we can only use single time point.

Tab. I shows an example where are two attributes – the beginning ("valid_from" attribute) of an event and its end ("valid_to" attribute) – setting the event's duration.

event_ID	valid_from	valid_to
001	2011-05-15 13:00:00	2011-05-17 13:00:00
002	2011-07-22 07:15:00	2011-07-28 07:15:00
003	2013-02-14 07:08:00	2013-04-02 07:08:00
004	2010-12-22 07:10:00	2011-01-13 07:10:00
005	2011-02-10 07:09:00	2011-02-12 07:09:00
006	2012-08-21 07:18:00	2012-08-25 07:18:00
...	...	...

Tab. I: Table with event beginning and end explicitly set. [Author]

Following GIS, suitable example of specific usage of such an expression of time is real duration of a road construction. Practically in mentioned example would be stored time intervals related to past events but generally it can be used even for future (planned) events of course.

The other type of valid time is when our stored information is valid for whole single time unit. We can without problem use only single valid time value (unit) – for example day, year or month. The attribute with the same meaning as in the Tab. I can be stored using single time point (beginning of an event) and time unit expressing a period of validity (Tab. II).

event_ID	time_point	time_unit[days]
001	2011-05-15 13:00:00	2
002	2011-07-22 07:15:00	6
003	2013-02-14 07:08:00	47
004	2010-12-22 07:10:00	22
005	2011-02-10 07:09:00	2
006	2012-08-21 07:18:00	4
...	...	...

Tab. II: Table with time point and time unit validity. [Author]

Following the example of road construction, the expression of time using the whole single time unit is more used for planning this type of events (for future), usually in project management branch. Knowing the complexity of planned task and available resources, the duration of each task can be calculated precisely (e.g. amount of days). In real world the tasks can be accomplished sooner or later than was planned and in such a case is more suitable to use the expression of time as in Tab. I as a report of past event duration.

Second time dimension we very often come across in temporal databases is "transaction time". Transaction time tells us not when change occurred to our described objects, but when change occurred for the database management system. That means when insert, delete or update query was performed, changing associated data [4].

Depending on database type of our stored data and type of usage temporal database doesn't have to necessarily have transaction time, but in most databases we at least find timestamp of when the database entry was made.

Last type we can come across is "user-defined time" – this temporal information is basically any other information about described object that is relevant and is expressed by some time value. With exception that its value doesn't influence validity of object or its attributes in time for the database system – that means is not "valid time" or "transaction time" [5].

B. Different types of spatiotemporal databases and their use

Based on whether database contains transaction or valid time, we can differentiate between several basic types of temporal databases.

Snapshot databases contain only user-defined time. They represent only one view, snapshot of a modelled world. Valid time database includes only valid time – that means history of database updates is not kept. Transaction-time databases (rollback databases) keep only transaction time. In practice, we usually use bitemporal databases [5]. Bitemporal databases use both transaction and valid time. Valid time is usually important for analysis of modelled world and even though we may not need transaction time, we include it for ability to search for possible tampering with data – so we can see by whom and when the information about modelled object was altered [6], [7].

The transaction time is mostly stored in separate table in a database. Following example Tab. III shows transaction time for previous one Tab. II. The attribute "modified" records the exact time where the tuple was created or modified. Mostly there is joined a second attribute ("user") storing information about the user who made the modification.

event_ID	modified	user
001	2011-05-10 07:13:20	user0516
002	2011-06-01 12:53:34	user2114
003	2011-07-20 09:25:11	admin066
004	2011-06-29 16:41:09	user6889
005	2011-07-19 18:35:47	admin066
006	2011-07-21 13:18:28	user0006
...	...	...

Tab. III: Table of transaction time of event modification. [Author]

C. Tuple and attribute time stamping

There are three main different approaches for storing temporal data using relation tables. They differ by on which level they integrate factor of time into the database.

First approach integrates factor of time on level of whole relation. It means that with each change in our modelled world we have in a database a whole new instance of the modelled world. Respectively we create whole new snapshot of a table that stores our temporal data. This is the state we usually get our information on the modelled world before any processing. For example from questionnaire or other type of information gathering being repeated two or more times after a period of time [7].

We have all data, concerning time we are interested in, in one place. This gives us ability to easily access information about our modelled situation in different points of time. We can easily view, model and analyze spatial and other attributes of all objects from the same time snapshot.

Problem comes when we want to ask queries on our temporal database that concerns more than one point in time. With SQL there is no easy way to get history of changes of a single object. And depending on data we gather, we can get an enormous amount of redundant data [7], [8].

Second approach integrates factor of time on level of tuples. That means that every row in a temporal database has its own time attribute, be it a valid time or a transaction time or both. This greatly reduces redundancy of data that was a big problem with previous approach. When one object in our database changes, we can simply create new row with new data, instead of creating whole table like we would have to do when integrating factor of time on relation level.

If each row has interval of valid time we can still easily with one query create snapshot – view of desired objects in desired time. By simply asking for data that have valid time interval set so that it encompasses our desired time.

SELECT * FROM temporal_table WHERE valid_time_from < "desired_date" and (valid_time_to > "desired_date" or valid_time_to IS NULL)

But dependent on a change of our data, it may still be a problem to easily get data we want to. If we have information on two different attributes of object in one table and one of them changes we create new row so we can document the change. That means that if we have those two attributes in one relation we have created redundancy for the second attribute that hasn't changed [9].

And of course it becomes harder to get history of changes to that attribute. If on the other hand we know that in every new tuple/row our desired attribute of object has changed we can easily get changes in time by selecting all the rows with the id of said object.

We can remedy this in different ways. First usual approach is to add new attributes dependent on groups of changing attribute. These new attributes gain value when group of attributes they are associated with changes. That helps us more easily build queries, because we have information on what attributes changed in each new tuple.

Id	Object ID	Valid time	Attribute 1	Group 1 change	Attribute 2	Attribute 3	Group 2 change
1001	1	2003	A	1	J	X	1
1002	1	2004	A	0	K	Y	1
1003	1	2005	B	1	K	Y	0

Tab. IV: Temporal object description. [Author]

Tab. IV has two groups of attributes that always changes together. First group is "Attribute 1", second group is "Attribute 2" and "Attribute 3". Thanks to columns documenting change, we can easily get years when every attribute changed in one simple query.

Second variant to help with multiple attributes changing in different time is to split them into different relations. This eliminates redundancy completely, but it adds greatly to complexity of select queries, because we have to join multiple tables in single query even when we only want to gather information about single object in single time [4].

It is clear that, if we are gathering data on objects that we know will change all of its relevant attributes, or we know we won't be interested in working with history of attributes that do not change every time – transforming temporal data into form where changes are logged by time stamping at tuple level is ideal way to manage those data.

In such a case it creates no or minimal redundancy and is able to provide easy way to access temporal data, be it information about history of an object or attributes of multiple objects in time [7].

Third approach integrates factor of time at attribute level. This means, we timestamp new attribute value of an object with its own time parameter and store it as a part of the attribute. This enables us to completely erase data redundancy, because when attribute of an object changes, we only modify existing attribute – add new information with timestamp. But this approach doesn't conform to relation table's standard of normalization.

For storing multiple values resp. multiple tuples of values as one attribute we need to use tool that can work with nested relational model and non-first normal form. Which means it is not possible to normally use it in classic relation database – some more advanced database management systems have the ability to work with nested tables but the more simple databases that are used for website use do not. Instead when are making database that should be usable by web services and we want to minimize redundancy, we have to use tuple based approach and we create different relation for each attribute that can change separately [7], [10].

Temporal Model	***NF***	***Time Stamping***	***Time***	***Features / Query Language***
Tansel's Model	NINF	Attribute	VT	Non homogeneous
Clifford & Croker's Model	NINF	Attribute	VT	HDBM Inhomogeneous
McKenzie's Model	NINF	Attribute	VT & TT	Extension of Snapshot algebra Historical algebra
Gadia's model	NINF	Attribute	VT	Homogeneous HRQUEL
Ben-Zvi's Model		Tuple	VT & TT	Time Relational model, Extension of snapshot algebra
Snodgrass's Model	NINF	Tuple	VT & TT	TQUEL
Lorentzos's Model	1NF	Attribute	VT	Interval Relational Model (IRM), Extended Relational Model (XRM)
Jensen & Snodgrass's Model	1NF	Tuple	VT & TT	Bi-temporal Conceptual Model (BCDM)
Ariav's model	1NF	Tuple	VT & TT	Time stamps are based on time points
Gadia & Yeung's Model		Attribute	VT & TT	Heterogeneous model
TSQL2	1NF	Tuple	VT, TT & User Defined Time	Homogeneous model

Tab. V: Most commonly used temporal data models. [3]

Most commonly used temporal data models are summarized in the Tab. V [3]. Each of them is categorized by their basic type - attribute or tuple time stamped; by their adherence to normal forms - 1NF (first normal form) or N1NF (non-first normal form) and by their support of valid time and transaction time. List of models in the table is not by any means complete. Aside from these models there are of course multiple models

not scientifically described or models that are only slight variations of the models we described in the table.

III. Spatiotemporal Data Management

When working with spatiotemporal data we typically want to do three types of things. Sort, manage and store those data – for that we are typically using a Database Management System. Next we want to analyze those data, refine them and transform them to our specified need – for that we typically use script or program with functions or ability to create algorithms specialized for problem solution we want to perform.

And finally we want to view desired parts of data and of course results to the analysis of those data as needed for end user presentation.

For work with spatial data there are several programs that do all of these things. One of the most used GIS software is ArcGIS Desktop from ESRI [11].

A. ArcGIS database system

ArcGIS includes its own database system, analytic tools and is able to visualize both vector and raster maps in many different ways and prepare them for printing or is able to export them into various formats. Whereas these programs are ideal for analysis of classic GIS data, they usually somewhat lack in ability to manage temporal data.

More complex ability to view temporal data was added to ArcGIS Desktop only in the last version (ArcGIS 10) [12]. These complex programs also usually do not have ability to generate complex customizable web based applications with interactive views of the data. If they do allow some sort of web publication it is most of the time dependent on specialized GIS server – like for example ArcGIS Server (that is separate application to Arcgis Desktop).

If we want to create a complex web application, that can be connected to other databases and systems containing temporal GIS data and we do not want to be dependent on specific GIS server, we have to use separate tools we can modify to store temporal GIS data, to enable easy with those data and to make possible their visualization and publication on web site.

There are several approaches and applications that do so. Many of them are end result of scientific research into temporal databases, GIS data display etc. These applications of course use different DBMS and different tools to create final website view for end-user.

B. Dynamic web application

Most common relational database management system for use with web sites and web site scripting languages are MySQL and Oracle. They both conform to basic relational standards and they both have many functions for working with data, including several spatial functions including testing of spatial relations [13].

To work with database we need server-side scripting language that would allow us to create dynamic web applications. Most popular such language is currently PHP. It has extensive set of functions to communicate with MySQL, Oracle or other relation database and there are many premade web applications enabling us to work with databases and tables through graphic interface (for example PHPMyAdmin) [14].

Scripting language enables us to create dynamic queries on data in our database and generate desired output – be it parts of data we want to export to ArcGIS for analysis or data we want to display to end-user on the websites. For effective interactive display of data at client side, there is basically only one main language used to do so and that is java-script [15], [16].

IV. Creating Web Application Using Data from Temporal Database

To create a website to view spatio-temporal data, we first need to transform those data into data model that enables us to easily update them and place queries that we are interested in. The most basic approach is to create table that integrates factor of time on tuple level and depending on how data change in time allow some sort of redundancy or split the table into more tables with groups of attributes that always change together.

As a part of our research we are currently trying different data models while working with data in our case study. We want to perform testing on speed of different data models. We are also comparing their ability to accommodate queries and ability to easily manage data for use with GIS application that visualize those data for end user.

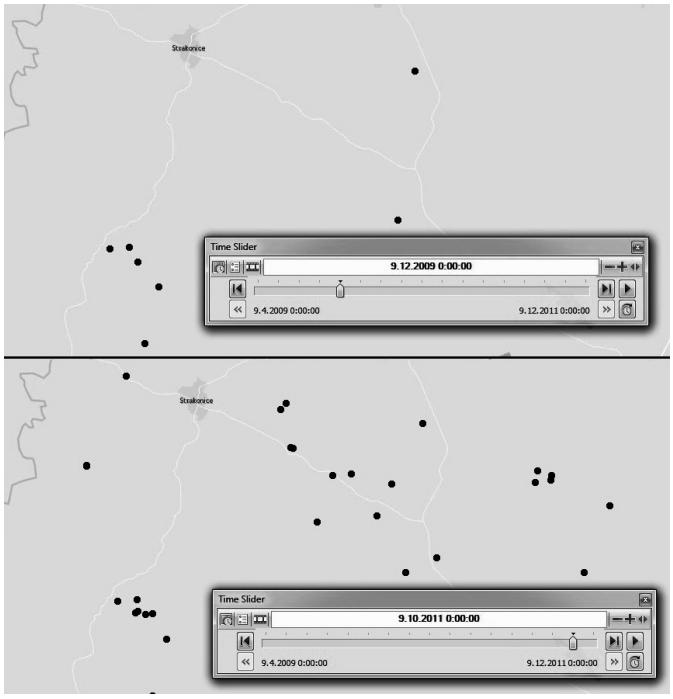

Fig. 1: Time slider in ArcGIS 10. [Author]

From preliminary research we compared various web based GIS applications and we found out that most applications process spatiotemporal data in form of one table with each row time stamped with valid time. Most of those applications then allow user to display objects on the basemap in selected

moment in time. Usually allowing user to turn on and off various layers and adjust viewed time by some form of slider. We can see use of time slider as used for working with temporal data in ArcGIS 10 in Fig. 1.

More advanced applications allow user to specify timeframe and show all objects that have valid time in that timeframe. As for specific format of data that these applications use, it varies from application to application. Most open source applications use Keyhole Markup Language (KML) which is notation of XML that is specifically designed for expressing geographic information.

Other formats include GeoRSS, Google spreadsheets, GeoJSON or some other formats designed for that specific application like ArcGIS shapefiles. Each of those formats has different complexity which allows different usage of additional data. Some offer scheme only for basic spatial coordinates with date, name and descriptions whereas some are much more advanced. Some applications require all temporal data in one file and table while others require them in multiple parts differentiated by layer or time factor.

That allows for different queries to be made at temporal database when creating data for different end user applications. This also requires different level of complexity of server side script that transforms data from temporal database into those formats.

In our case study we are using script written in PHP to get required data from MySQL database. Then we transform those using templates into most common formats used by client side applications and other local GIS programs. Finally we want to compare time that takes to process different queries on different data models as well as time it takes to process results from those queries into different formats as well as take note of different possibilities of combining different data formats used by end-user applications and different data models used for storing data in relational databases.

We are currently converting our case study data to different data models that we modified and used for our case study. For now all of them are based on storing time attribute at tuple level, but we created different levels of data redundancy in data models. Our tests include test of how much has limited amount of data redundancy impact on speed of more complex queries and their ability to allow easy access to important data. Later we want to add and compare more different data models. In preliminary tests with just sample of our case study data (about 150 rows) we got overall good results for data model with bigger redundancy. At such a small number of rows big redundancy allowed, as expected, best speed for simple queries and speeds didn't noticeably slow down even for tasks that needed additional transformations from scripting language or that needed to be separated into several subqueries. We are expecting when performing these tasks, that in data model with lot of redundancy depends on help from scripting language, will probably slow down a lot in tables with lots of data. We hope that it will be interesting to compare performance of data model with big redundancy and data models that conforms to data normalization. We are still at the first phase of our research and we plan to expand both amount of data used and amount of different data models. For comparing results we want to use multiple-criteria decision analysis and compare results with different amount of data, different amount of data redundancy and needs for different types of queries.

V. NAVIGATION APPLICATION

Based on described research in the field of temporal data modelling for web usage we want to apply gained knowledge to develop a web platform combining map, navigation tools and school Information System of Czech University of Life Sciences Prague. This WebGIS will be mainly helpful for new incoming students (1st years) by providing simple, smart and comfortable orientation across the whole university campus.

The main idea was to create an interactive map of the area, especially indoor maps of all the university buildings including relevant data from the integrated school Information System which are related with navigation processes, e.g. classroom schedules, consultation hours of teachers etc. All the mentioned data are available in the Information System but problem is that are changed at least once per semester when classroom schedules switches. This is the dynamic component of planned WebGIS and it was necessary to find a way of transferring data from the Information System's databases to the web based map application.

The core of the developed WebGIS consists of two basic components. First one is the temporal database and the second one is a detailed vector map of the university campus including indoor building plans. The temporal database will be the output of the research described in this article.

The map component will be created combining several methods. As a base map will be used open-source platform OpenLayers. Since the area of our university campus is not perfectly recorded in the map, it will be needed to fix all the inaccuracies manually using portable GPS module.

The indoor plans were already provided to us by the Technical Support Department in form of CAD technical drawings. From those files the relevant layers were extracted and implemented into the basemap. This process is performed in ArcGIS 10 and the resulting layers are exported to KML as a supported format of OpenLayers platform.

There is currently available a pilot version of the map component of the WebGIS including one floor of the Faculty of Economics and Management. All the polygons (classrooms, offices and corridors) were traced over manually but hereafter this process will be done automatically. There are several options of the automation but the most effective one is automatic image classification in ArcGIS.

VI. CONCLUSIONS

We are currently in stage of running case study to test different data models. We have created and are using script in PHP that uses custom templates to process results from SQL queries. This allows us to test speed needed to make queries on

different temporal data models and create different formats of temporal GIS data files. We would like our end result to be methodology for choosing and implementing optimal data model for web based temporal GIS based on specific user requirements. Hereafter we plan to develop a WebGIS following the designed methodology.

ACKNOWLEDGMENT

The paper was supported by the grant project of the Czech University of Live Sciences: Methodology for Designing Relational Database and Web Interface for Dynamic GIS, no. 20131027. This support is very gratefully acknowledged.

REFERENCES

[1] H. Kopáčková, H. Jonášová, I. Mikešová a J. Hejlová, „Gathering of Requirements on WebGIS Development - the Example of Bikeway Mapping Application," in *Recent Researches in Circuits, Systems, Communications and Computers: Proceedings of the 2nd European Conference of Computer Science (ECCS'11),* Stevens Point, WSEAS Press, 2011, pp. 290-295.

[2] J. Komárková, P. Sedlák, M. Novák, A. Musilová a V. Slavíková, „Problems in Usability of Web-Based GIS," in *Proceedings of the International Conference on Applied Computer Science (ACS)*, Athens, WSEAS Press, 2010.

[3] A. Burney, N. Mahmood a K. Ahsan, „TempR-PDM: a conceptual temporal relational model for managing patient data," in *AIKED'10 Proceedings of the 9th WSEAS international conference on Artificial intelligence, knowledge engineering and data bases*, Stevens Point, WSEAS Press, 2010, pp. 237-243.

[4] T. Ott and F. Swiaczny, Time-Integrative Geographic Information Systems, Berlin: Springer Verlag, 2001.

[5] C. Combi, E. Karavnou-Papiliou and Y. Shahar, "Temporal Databases," in *Temporal information systems in medicine*, New York, Springer US, 2010, pp. 45-85.

[6] Y. Tang, X. Ye and N. Tang, Temporal Information Processing Technology, New York: Springer Heidelberg, 2010.

[7] C. E. Atay, "A Comparison of Attribute and Tuple Time Stamped Bitemporal Relational Data Models," in *Proceedings of the International Conference on Applied Computer Science*, Las Vegas, 2010.

[8] P. Revesz, "Temporal Databases," in *Introduction to Databases*, London, Springer-Verlag, 2010, pp. 67-79.

[9] J. R. R. Viqueira and N. A. Lorentzos, "SQL extension for spatio-temporal data," in *The VLDB Journal*, vol. 16(2), p. 179–200, 4 2007.

[10] I. A. Goralwalla, A. U. Tansel and M. T. Özsu, "Experimenting with Temporal Relational Databases," in *Proceedings of the 1995 ACM CIKM International Conference On Information And Knowledge Management*, New York, 1995.

[11] MyTopDozen.com, "MyTopDozen," MyTopDozen.com, 2011. [Online]. Available:http://www.mytopdozen.com/Best_GIS_Software.html. [Accessed 2013].

[12] Esri, "What's new for temporal data in ArcGIS 10", Esri, 2010. [Online]. Available:http://help.arcgis.com/en%20/arcgisdesktop/10.0/help/index.html#//00qp00000018000000.htm. [Accessed 2013].

[13] solid IT, "DB-Engines Ranking of Relational DBMS," solid IT, 2013. [Online]. Available: http://db-engines.com/en/ranking/relational+dbms. [Accessed 2013].

[14] E. Pultar, T. J. Cova, M. Yuan and M. F. Goodchild, "EDGIS: a dynamic GIS based on space time points," in *International Journal of Geographical Information Science*, vol. 24:3, pp. 329-346, 3 2010.

[15] E. Pultar, M. Raubal, T. J. Cova and M. F. Goodchild, "Dynamic GIS Case Studies: Wildfire Evacuation and Volunteered Geographic," in *Transactions in GIS*, vol. 13(s1), p. 85–104, 6 2009.

[16] X. Ye, Z. Peng and H. Guo, "Spatio-Temporal Data Model and Spatio-Temporal Databases," in *Temporal Information Processing Technology and Its Application*, Berlin, Springer Berlin Heidelberg, 2010, pp. 91-112.