International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 8 (2017) pp. 1691-1696

Data Model and Case Study of Seasonal Data in Web GIS

[1]Ing. Jakub Konopásek and [2]Doc. RNDr. Dana Klimešová, CSc.

Czech University of Life Sciences, Prague

Faculty of Economics and Management, Dept. of Information Engineering

Kamýcká 129, 165 21 Praha 6 – Suchdol, Czech Republic

Abstract

This paper describes creation of web GIS application that works with temporal data using contemporary methods and sed in current world. Spatial data that few years ago could have been displayed and used only on local computer and using special software can now be used, analyzed and presented to wide spectrum of users by means of internet. But creation of web applications conforms to its special standards, formats and other requirements. That is specially restricting if we want to use not only spatial data but spatiotemporal data. Adding temporal dimension to data allows user to see changes in time which is often very useful. For example in agriculture knowing how fields were developed and planted during several seasons gives much more useful information then just seeing their current state. In other words, to get important information, we often need user to get access to multiple data sets gathered in several time frames. This requires special type of data model that works effectively with not only data but also with metadata. This paper presents data model for seasonal data often used in agriculture, but can be used for almost any seasonal data. Used methods and tools include data modeling, relational modeling, normalization, UML, metadata structure according to Inspire Directive and ISO19115. Presented model is optimized for use as base for web application. This paper also has description of problems and issues that came up when creating described model including comparison to other available solutions and tools available for creating such application. Paper include simple example in UML. Presented model can be generalized and used with similar types of seasonal spatiotemporal data.

Keywords: GIS, Spatiotemporal data, data modeling, temporal database, web GIS, metadata

INTRODUCTION - Time in webGIS

Integrating time into database, be it with spatial data or other data, can be done with different approaches. Usually time is seen as attribute of an object or time can be seen as another whole dimension of object. Adding time as an attribute of the object in GIS, is easier to implement and it is approach almost always used when working with data structures based on relational data model. Nowadays in web applications relation databases are still used almost everywhere and event GIS tools use classic "table" approach when dealing with data. So integrating time as an attribute is currently most used and most advantageous approach. Integration of time as an attribute can be divided as following:

- relation-level - each change in time of an object creates a new instance of the whole relation with all object that didn't change their attributes duplicated - a significant disadvantage with data redundancy; with even a tiny change we need to create whole new relation
- row-level (n-tuple) - every row has its own attribute with time stamp and with change of attribute of an object, we create a new record for that object with appropriate timestamp; redundancy still present because we duplicate attributes of an object that didn't change
- attribute level - every attribute has its own time stamp - the lowest level of redundancy, but the largest number and complexity of data model and of queries when working with data [1]

Each approach is flawed in some way and all are basically subject to the shortcomings of relational modeling. Redundancy vs complex data structure that is not easy to work with and often takes much more computing resources.

The second main approach to integrate time as another full-blown dimension of described object. There is, however, an obstacle in complexity of the algorithm for working with multidimensional object. Such approach basically can't be used to feed the data in acceptable time to web based GIS. Also this method is still in academic research and tools available for development of specialized webGIS applications require to get data with time as a normal attribute of an object so this approach is not suitable for web development yet. These types of approaches are usually created for specialized applications and specific data which allows for optimized querying and working with large data sets. [2].

For GIS application to work in web environment it is necessary to have quick response time and to use tools and actually methods available in web environment. Adding time as an attribute of object can be done with relational databases that are most used in web application development, but there is need to balance redundancy with efficiency and usability of used data model. Use of spatiotemporal data is currently often discussed topic at conferences concerning geographic information science - e.g. AGILE 2016 - section Spatiotemporal Data Acquisition, Modeling and Analysis. Problems and solutions that comes with designing spatiotemporal relation data model for webGIS are described in more detail later in this article.

Problem of metadata in temporal data

There were many articles describing various spatiotemporal data models in last years but they very rarely deal with importance of metadata for spatiotemporal modeling. Metadata, by definition, are data that define and describe set of a data that we work with. This typically include description of content, time scale, geographic scope, spatial reference, quality, type of representation [3]. Usage of metadata in geographic systems, their structure and usage is covered in multiple different directives and standards (Dublin Core (ISO 15836), ISO19115 / ISO19119 and INSPIRE Directive [4]). The vast majority of data models for temporal data, deals only with the design of the data model for preservation of spatiotemporal data, but in these often highly complex data models, metadata is omitted. Even though these specialized data models often can't work with normally used big GIS server systems and are therefore missing tools and resources of these servers that allow the metadata to be implement. Additionally, if it is a dedicated spatiotemporal model some of metadata (for example time of validity of data) is already included in data model and there can be problem with data redundancy and makes it possible to make conflicting data.

If geographic information system uses one dataset or datasets that are by themselves separate layers, how we store and structure metadata is not that important, because datasets with different metadata are easily separated. However, when we are working with models for temporal data, data from different datasets can be in the same tables but also separated by their changes in time into different tables. In other words structure of storing metadata is also influenced by time. And when working with temporal data, that is history, metadata are often much more important, then when we are simply working with single snapshot in time.

Types of time

There can be three different types of time in temporal databases. Valid time is time that sets validity of object in real time. It is usually single number or time interval (in which case it is usually represented by two database columns- from, to). When end of valid time is not specified, it means that object still exists in current world with attribute values described in that row. If row has end of valid time set, then object currently either no longer exists or some of its attributes were changed. Then there can be transaction time - that is time that object was added or updated in database. Databases that contain both of these times are called bi-temporal databases. Third type of time is user time, which is any other time information that is not used as valid or transaction time [1]. Bi-temporal databases is most standard type of temporal database [5].

WebGIS APPLICATIONS

There is a big difference in how data should be stored and how data are supplied to actual map viewer. In case of displaying vector layers, current internet map viewers need layer to be supplied as one table where each row equals one feature. More advanced map viewers can then be programmed to filter and display features by attributes. Less advanced just allow for all features in layer to be displayed. This means that when working with temporal data we are limited to two main solutions. Simply showing different time frames as separate vector layers, and allow user to switch between them. Or more advanced solution - use filtering function (advanced map viewers have plugins for temporal data that provide this functionality) and supply viewer with table with integrated time at level of row - tables that have attributes that contains valid time. Viewer than can filter features and show only those that correspond to selected timeframe according to values recorded in valid time attribute columns. Timeframe is provided by user typically by directly inputting values or using time slider. As already described above having data in this form is one of standard methods for storing temporal data, but it is often far from optimal when it comes to data redundancy and data management. Also while map viewer can filter features by attributes it has to programmed directly into map viewer application (java script) at users side, which is usually much harder to do then make filtering before sending data to user in server side scripting language (PHP, ASP). There is also limitation of response time of application. Usually we can't send map viewer more than two megabytes of vector layer data, because it would be too unresponsive for internet use. This corresponds to layer with approximately 1000 features in JSON or KML format. Each feature having 15 attributes with keywords and numbers in their values - not long text. And each feature being polygon with in average 8 geographical coordinate points. Although size of file can vary depending on attribute size and other variables it is definitely true we can't easily send thousands features to be processed and filtered at the client side. Because of this we usually store data in different data model and then filter and transform data for viewer application into this form as needed. This also actually dictates most used query and direction we need to optimize data model for web based spatiotemporal application. Apart from this we also need to be able to perform queries concerning history of object and its additional relations and characteristics, all these things are usually displayed in popup, when user clicks on feature on map. This information does not need to be loaded for viewing the map but is usually loaded and presented to user only when feature is clicked.

WebGIS architecture

Every GIS architecture can be separated into three main levels. In GIS on desktop PC one application takes care of everything, typical user can't influence much and even advanced user only has very little control over how data are stored and how application works with them. This however doesn't apply in case of web GIS. Three main tiers of GIS architecture are presentation tier, application logic tier and data tier (Image 1).

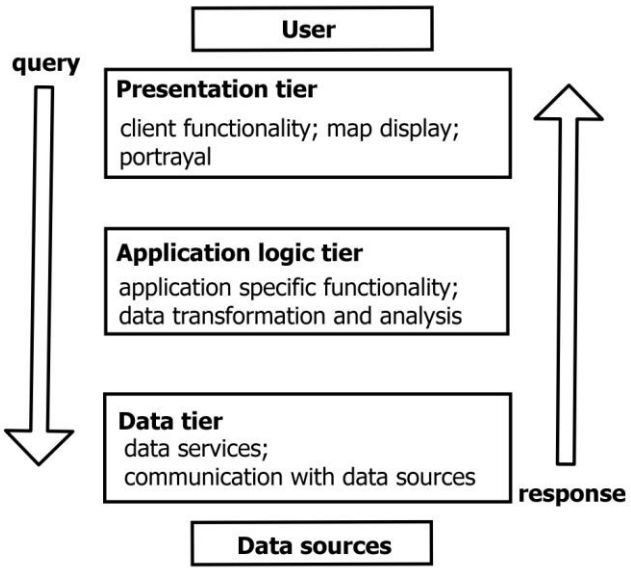

Image 1 - GIS architecture (own processing)

- Presentation tier - this layer basically takes care about everything that user comes into contact with. In web GIS this tier is usually java script library that runs at from different source types including shapefiles, JSON [7], KML, connection to GIS server, WMS, WMF, google maps, openstreet maps.
    - Portrayal layer - accepts geographic data and transforms them into cartographic objects. Processes style definitions and creates layer symbology.
    - Map display layer - cares about drawing of cartographic objects for viewing of user and allows work with them.
    - Client functionality layer - allows additional functions of user
- Application logic tier - this tier takes care of application specific functionality, data transformation and data analysis. This tier can contain multiple modules with varying functionality and uses. For web GIS application this tier can be handled by GIS server if we need specialized GIS functions, but we need specialized GIS server. On the other hand we can use web scripting language to directly generate JSON or KML file from data stored in database. This allows us to use only typical webhosting for websites. On the other hand any specialized function need to be programmed. Comparison on these two options can be found bellow.
- Data tier - tier that takes care of data services and communication with different data sources. Takes care of creating and managing data services, WFS, WMS services, database and file connection and so on. Same as tier before - this tier can be either provided by GIS server or can be replaced by server side scripting language (PHP) and database (MySQL, Oracle).

Specialized web GIS server specifics:

client side. Commercial GIS servers have usually their on web presentation tier application, but can also be used with other viewing application and only forward them data. Openlayers is one of the most used open source presentation tier library [6]. It allows for loading of multiple layers

- Third party software with often big user community and advanced tech support.
- All tiers of GIS application are often connected and developed as a whole and there is little possibility of conflict between them.
- GIS server has to run on special server with additional running costs.
- Almost all setting can be done via user interface. Allows easier setup but limits customization.
- GIS server contains all sorts of premade specialized modules in application logic tier. This allows use of geographic function which would otherwise have to be programmed. But adding new and specialized function can also be much harder.
- Although GIS server allows for use of data from connected database it doesn't allow that much of freedom when customizing it as specialized web application.
- Some advanced GIS servers can have ability to create tiled vector layers or they can at least create raster layer from vector data which can reduce data needed to be sent to user.
- GIS server is massive application that needs much more processing power than simple server side scripting language application which can lead to bigger time lag for user.

GIS on web server specifics:

- Application is made to fit its needs.
- Application tiers are often separate applications made to work together, which can lead to worse compatibility between them.
- Lesser running costs because there is no need for specialized server. Even moderate sized GIS application can be run on simple web hosting with running cost of 1$ per month.

- Can't run advanced analytical and other functions. No available modules that GIS server can provide. Every special function needs to programmed separately. There are limited system resources to process them.
- If done right, application does exactly what we need and can be much more efficient faster. This heavily depends on size of data, data model and transformation required to process data for viewing.
- We can easily manipulate with data. We always have direct access to all data and can have custom CMS system for easier management.
- We can easily share data and communicate with other specialized systems.

GIS data types and their use

Data for use with web GIS can be stored in multiple ways. Simplest is to store data in file. Typical examples are shapfile (shp), JSON (json; geojson) or KML (kml; kmz). JSON and KML are both xml/tag based formats and are easy to generate by script from database. Advantage of heaving data stored in file is that file can be directly used by Openlayers or other presentation tier application without need to process them further. Disadvantage is that data can't be easily updated and worked with and there is limit to how much data can file have.

If we need to have more control over data - if we need to update data often and work with them using filters and specialized functions and we know that there will be large volumes of data (several thousands rows), we need to use database. For web services database needs to be relational database. Most used databases for websites are MySQL and Oracle. By creating special data model specifically suited for data it is possible to make web GIS application much more quicker. It is also possible to filter out unnecessary data and to send to presentation layer only data really requested by user. Database can be used both with GIS server and without. When using it without GIS server, as already mentioned above, there is more freedom when working with data. Geographical data in such database can also directly have connected tables with non-spatial objects. Data model presented in last part of this paper is to be used for small to medium sized GIS applications. It is possible to use it without GIS server to create maps with hundreds of features taken from database of thousands or even tens of thousands without noticeable time lag for user.

TOOLS AND METHODS USED IN CASE STUDY

Integration of time

Due to uniqueness of seasonal data - many of attributes of an object change at the same time. Thanks to that we can integrate time as attribute at row level. But in our case study we build up on it by separating data to several levels to accommodate for attributes that change in different times. This is based on research of Conceptual space-time data model, geo-atom [8] and several other methodologies [9,10,11], notably EDGIS [12].

Other tools and methods used in case study

Case study presented is data model for storing agricultural seasonal data. Presented data model was optimized for use in web application including optimization for commonly used queries and it takes in variable storage and easy use of metadata. Case study data model can be used and generalized for any seasonal data. Case study described bellow allows to keep records about agricultural fields. What kind of plants grow on what field and when. It allows to keep history about who owned these fields, how area of these fields changed. Who worked on these fields and what work was done and when.

In creation of this data model we made use of relational data modeling, data normalization, metadata Dublin Core standard and UML. Case study model was tested using application programmed in PHP, MySQL, HTML, Openlayers3. For viewing, map data are loaded from MySQL by scripting language (PHP), converted into compatible format (JSON) that is then loaded and presented to user by OpenLayers3 java script library. Web application was tested on server of a company, that rents webhosting for normal public use.

CASE STUDY - Data model for seasonal data

This paper presents simple model for use with webGIS application that shows agricultural fields and their development including who owns those fields, what types of plants grow there and what work is done on them. All this information can be subject to change and user can select any point in time and view information about how fields looked like in that point in time. This model was optimized for agricultural data, but can be generalized for other purposes. Important point is that this model is optimized for seasonal data - data sets that come in batches.

When creating, optimizing and testing this data model we basically tried to find compromise between these criteria:

- be fast and allow usability in web services - allow for getting response to display user at least 500 features with their relevant attributes from database table with at least 15000 rows under 0,5s
- be easy to manage - reduce data redundancy as much as possible while not compromising manageability and query time - best query time would of course be one flat table with all data but that would have terrible data redundancy - main part of optimization was comparing speed of data models to allow for previous point while allowing for easiest data manageability
- allow easy storage and management of metadata
- be optimized for taking care of seasonal batches of data as often used in agriculture, but allow for generalization for other purposes

As mentioned above, after choosing best approach, actual optimization was dealing with conflict classic for relational databases - having fast query response time while also have least data redundancy. Proposed data model integrates time at the level of rows in one table [1]. This approach works best when all attributes of an object change at the same time. If this ideal situation happens there is no redundancy as we create new row for each recorded state of an object and as all attributes change, there is no redundancy. This is of course ideal situation and it rarely happens. This approach also stores data in form close to form that most webGIS map scripts need to generate map view. That means it is easy to obtain needed data by simple queries and generate desired output. There are other approaches such as complex time integration at the level of attributes [2] or event based time integration [10] that can be used to store temporal data in relation database. Some of these approaches offer better data management and lower redundancy, then approach we selected. It would be true especially if we didn't deal with seasonal data but with data that change at different times. But they also have much higher demand on query processing time or they can't be easily applied in constrains of relation databases available in web environment and their modification would make them too hard to use. For example EDGIS [12].

In developing of data model we deal with premise that main part of our data set are seasonal data. But there are usually always some data that don't change seasonally and ability to effectively deal with such data, not only makes our model better optimized but also makes it so that it can be easier used for other purposes. We dealt with this problem by expanding method of integrating time on level of row by grouping attributes of features by frequency and dependency of their change. All agricultural seasonal data will almost always change in regular intervals with changing of season. E.g. when data are collected, they are specific to that season and not to many seasons at once and they almost always change with new season.

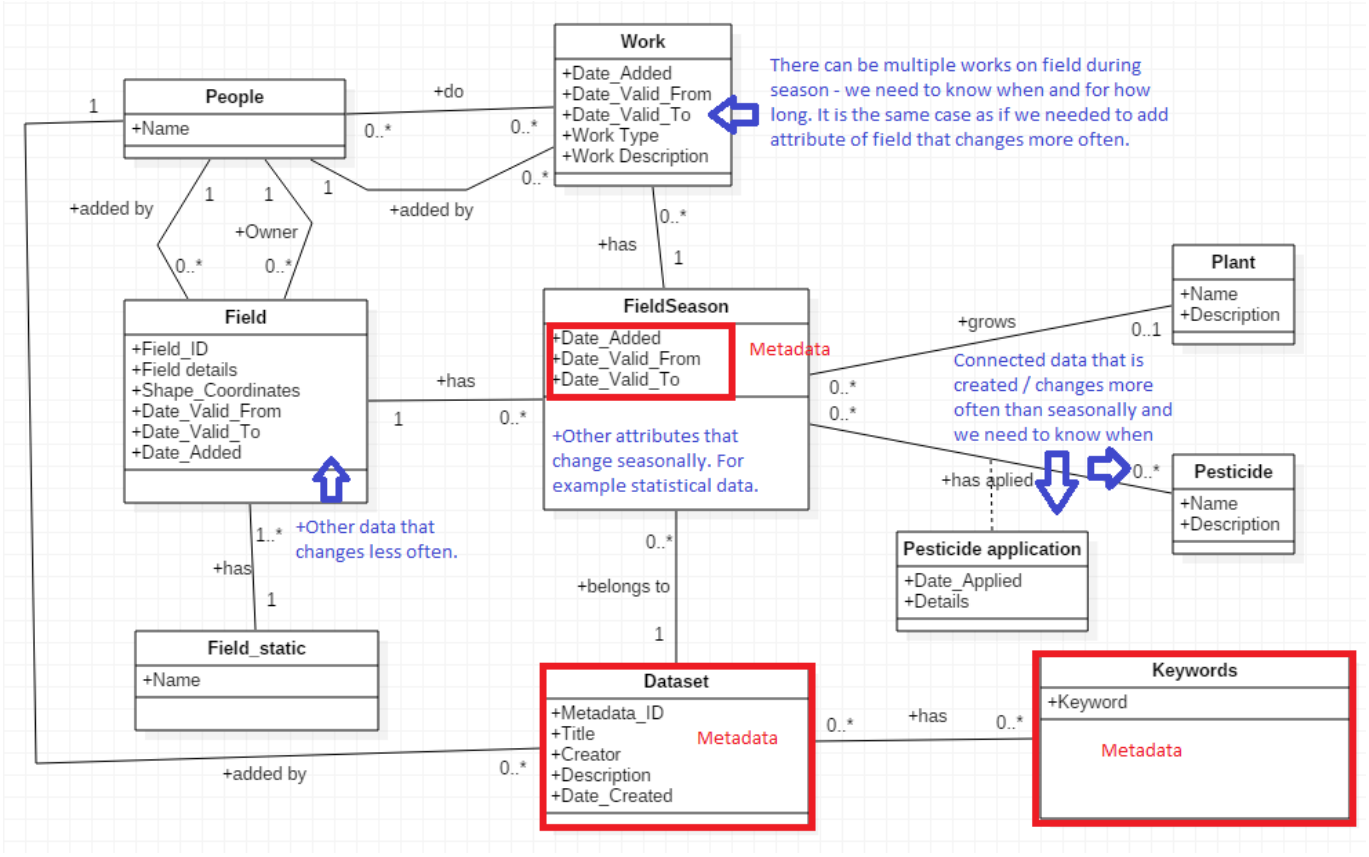

Image 2 - Class diagram (own processing)

In presented model (Image 2, agricultural field is our main focus and geographic feature. It is split into three different classes - future database tables. They are Field_static, Field and FieldSeason. Field_static is class that identifies the feature and contains attributes of the feature that never change or attributes of which we do not need to now their history. Class named Field can contains group of main characteristics of agricultural field that rarely change, but we need to know their history. Example of such can be ground type, text description of agricultural field and so on. Information about owner is also in this group of information, but it is not in form of attribute but as association to class People. Included in this are also geographical coordinates - shape of field, because they also typically do not change that often. Depending on type of data, geographic coordinates could be also moved to FieldSeason class if they should change seasonally, but for agricultural fields borders do not typically change. In presented data model all polygon coordinates are stored as one in single attribute. In case it would be needed to perform advanced analysis, single coordinates can be moved to related

table and stored one by one, as is described by Pultar and collective [12]. For viewing on web it is better to store them in single attribute as it doesn't require additional join and formats that web map viewers read such as JSON and KML contain polygon coordinates in one tag separated by comma anyway. Finally in class FieldSeason can be put group of characteristics of field that change seasonally. Included here can be attributes pertaining to yearly gathered measurements like ground toxicity, level of ground acids and so on and of course type of plant that is grown on that specific field. Type of plant in presented model is represented by connection with class Plant with multiplicity 0..1. To class FieldSeason can also be connected classes with data that changes more often than seasonally if there are any, but more groups are made more complex queries will be needed to gather resulting data. In presented model example is class Work and classes Pesticide and Pesticide application.

Class FieldSeason is also connected class Dataset that basically contains metadata for gathered statistics about fields. Data in class FieldSeason are data that are presumed to be statistics and data collected and added seasonally in batches and contain bulk of data that changes in time. This makes information about to which batch of data what information in FieldSeason belongs more important. For example in situation that whole batch is faulty we would easily be able to find which data we need to correct.

Described model allows for very easy construction of data table for use with web map. Map of fields and their plants in any given time can be constructed with three joins between four tables - Field_static, Field, FieldSeason and Plant. It can also be expected that three of these tables won't have many rows. Only table FieldSeason will have many rows added each season. We can also switch last join to table of Pesticide or Work instead of Plant and let map be styled not by plants but by other value.

Model in this case study has flaw in inability to change attributes of Field during season as association between Field and FieldSeason is one to many. But this allows us to have much simpler join and better efficiency, while only sacrificing possibility, that attributes there will need to be changed during one season.

CONCLUSION

This paper deals with complex problems that come up when designing web GIS application including web server only specifics. Case study in second part then shows practical solution and allows for easy queries on data that are needed for use in web environment. It expands on several well known temporal and spatiotemporal data models and theories [1,5,11]. It adapts and optimizes them for use in web environment and with seasonal data by separating temporal data into several groups by frequency of expected updates. This allows for minimal redundancy of inserted data and good processing speed of commonly used queries. During testing the actual application was able to work with dataset of several tens of thousands entries with response under 0,5s (measured for most used queries - displaying map with all features in set time with simple filtering by attributes and after selecting feature describing changes of that feature in time). Data model can be easily generalized and then optimized for other specific environments and data sets.

REFERENCES

[1] Ott, T. and Swiaczny, F., 2001, Time-integrative geographic information systems, Springer, Berlin, 1sted.

[2] Fan, Y., Yang, J., Zhu, D. and Wei, K., 2010, "A time-based integration method of spatio-temporal data at spatial database level", Mathematical and Computer Modelling, 51(11-12), pp. 1286-1292.

[3] Charvát, K., 2007, Geografická data v informační společnosti, Výzkumný ústav geodetický, topografický a kartografický, Odvětvové informační středisko, Zdiby, 1st ed.

[4] "INSPIRE - Infrastructure for Spatial Information in Europe", 2017, Inspire.gov.cz, available at: http://inspire.gov.cz/dokumenty/smernice (accessed 12 March 2017).

[5] Tang, Y., Ye, X. and Tang, N., 2011, Temporal Information Processing Technology and Its Application, Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 1sted.

[6] "OpenLayers", 2017, Openlayers.org, available at: http://openlayers.org (accessed 12 March 2017).

[7] "GeoJSON", 2017, Geojson.org, available at: http://geojson.org/ (accessed 12 March 2017).

[8] Goodchild, M., Yuan, M. and Cova, T., 2007, "Towards a general theory of geographic representation in GIS", International Journal of Geographical Information Science, 21(3), pp. 239-260.

[9] Combi, C., Keravnou-Papailiou, E., Shahar, Y. and Hunter, J., 2010 , Temporal information systems in medicine, Springer, New York, 1sted.

[10] Peuquet, D. and Duan, N., 1995, "An event-based spatiotemporal data model (ESTDM) for temporal analysis of geographical data", International journal of geographical information systems, 9(1), pp. 7-24.

[11] Ferreira, K. R., Gomes, A., Vieira A. M., and Benincasa D., 2015, " Temporal GIS and Spatiotemporal Data Sources," Proceedings XVI GEOINFO, Campos do Jord˜ao, Brazil, pp. 1–13.

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