VIETNAM NATIONAL UNIVERSITY
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Vu Danh Viet

TIME MAPPING SYSTEM

MAJOR: Information Technology

HANOI May 2015

VIETNAM NATIONAL UNIVERSITY
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Vu Danh Viet

TIME MAPPING SYSTEM

MAJOR: Information Technology

Supervisor: Assoc. Prof. Nguyen Viet Ha

HANOI May 2015

Declaration of Authorship

"I hereby declare that the work contained in this thesis titled, 'Time mapping system' and the work presented in it are my own and has not been previously submitted for a degree or diploma at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no materials previously published or written by another person except where due reference or acknowledgement is made."

Signed: _____

Date: _____

SUPERVISOR'S APPROVAL

"I hereby approve that the thesis in its current form is ready for committee examination as a requirement for the Bachelor of Information Technology degree at the University of Engineering and Technology."

Signed: _____

Date: _____

ACKNOWLEDGEMENTS

I would like to express my greatest gratitude to Prof. Nguyen Viet Ha and MSc. Vu Quang Dung who have guided me during the period of time that I have researched and completed the graduation thesis.

I am grateful to the teachers of the Faculty of Information Technology, University of Engineering and Technology, Vietnam National University Hanoi, who have taught me in four academic years, supported me all the base knowledge and conditions to finish the graduation thesis.

I would also like to thank all of my colleagues and friends in class K56CLC and Coltech-Toshiba Software collaboration Laboratory of University of Engineering and Technology, Vietnam National University Hanoi.

I greatly appreciate the following organizations: Information Technology Faculty, Coltech-Toshiba Software collaboration laboratory of University of Engineering and Technology, Vietnam National University Hanoi where I received the best conditions to complete the graduation thesis.

Last but not least, I thank my family for being with me wherever and whenever I need them. Their encouragement make this thesis possible.

ABSTRACT

History plays an important role in human life. By study history, we understand people and societies, which lead to improvement in moral understanding. History also help us gain skills that will be used in our life and career.

However, there is not a convenient method to manage historical data, or a way to understand these data easily. The context of historical data includes both location and time. It is necessary to consider both of above factors in order to make a proper conclusion about history truth.

In this thesis, I propose a method for managing map data with additional time dimension by using a graph theory, where nodes are events, and edges are relations between events. I also propose the mechanisms to visualize these data, evaluate their changes when time change for better understanding. The system has been implemented as a framework, provide the mechanism for managing and visualizing time map data.

Keywords: map, time dimension, graph database

Contents

List of Figures

List of Tables

Abbreviations

GIS	**G**eographic **I**nformation **S**ystem
HGIS	**H**istorical **G**eographic **I**nformation **S**ystem
DB	**D**ata**B**ase
SQL	**S**tructured **Q**uery **L**anguage
NoSQL	**N**ot **o**nly **SQL**
REST	**RE**presentational **S**tate **T**ranser
JSON	**J**ava**S**cript **O**bject **N**otation
HTML	**H**yper **T**ext **M**arkup **L**anguage
CRUD	**C**reate **R**etrieve **U**pdate **D**elete
VNU	**V**ietnam **N**ational **U**niversity
UET	**U**niversity of **E**ngineering and **T**echnology

Chapter 1

Introduction

1.1 Motivation

"People live in the present, plan and worry about the future". Why do we need to care about the past? All education system require students study history, but historical knowledge do not help us go to space, discover new materials or cure cancer.

History is useful, but in a way that less tangible, less immediate than other fields. History helps us understand people, societies, improves moral understanding, provides identities and is the essential for good citizenship[2].

However, collecting and managing historical data is quite a difficult task. Over hundred, thousand of years, these data are scattered, lost, missing or altered to the point it is really hard to understand them at first glance. Besides, the truth from today might become wrong the very next day. *"There is no absolute truth in history"*[1]. Each time a new evidence, data is found, we need to reconsider whole problem, because the truth may change dramatically.

> Tu Duc[2], the 4^{th} emperor of Nguyen dynasty, always has bad reputation for being coward, for taking no action to prevent the invasion of the French in 19^{th} century. One of the most

[1] Duong. T. Tran.(2014). "How history was written?". Vietnam Summer School of science.
[2] Tự Đức, Wikipedia

popular evidence supports this idea is that Tu Duc gave an order to Truong Dinh[3], people's hero for leading a guerrilla army in southern Vietnam against the French invasion, to disband the army and surrender the French. However, recently, some secret letters between Nguyen government and Truong Dinh were discovered and showed that: Nguyen government had always supported Truong Dinh and the order was just a distraction, to let the French lower their awareness and think Nguyen dynasty had given up on fighting. That gave both Truong Dinh's and Nguyen government's actions a better change to success.

To come up with a conclusion for a event or a series of events is not simple. Historical data is characterized by two factors: Time and Location. Therefore, historical data basically can be viewed as map data, with additional time information. Furthermore, each event may related to other events. One needs to overview both above factors and event's relations with others to get a proper conclusion. An effective yet simple method for modeling map data with time information is needed. A method that help both people and and historians can easy to understand the data, events and get the knowledge from them.

1.2 Thesis overview

This thesis proposes a method which allows managing map data with time axis based on graph theory. The method is implemented as a framework, where can be used for managing map data with time dimension. In this thesis, I also propose techniques using in visualizing the event and data when we change our time of interest.

In this thesis, my research, the implementation and experimental results are presented in the following chapters:

- Chapter 2 – Background and Related work. This chapter provides background and tools used this project. Beside, some current works relating to the topic are mentioned and discussed.
- Chapter 3 – Data model. In this chapter, the approach for modeling map data with time dimension are presented. The mechanisms for managing these data are also mentioned.

[3] Trương Định, Wikipedia

- Chapter 4 – Visualizing mechanism. This chapter described how to visualize map data with additional time information, so we can look and understand these data.
- Chapter 5 – Experiments. This chapter show experimental results and the discussion on the results.
- Finally Chapter 6 – Conclusion. The conclusion of the project and future work are mentioned in this chapter.

Chapter 2

Background and related works

2.1 Background

2.1.1 The time dimension

Most current GIS system are two-dimensional, which mean they only care about data in two dimensions. Even some advanced system, three-dimensional are just normal 2D GIS with additional mechanism for handling z axis, they still consider z to be special, not equal to (x, y). And none of them are concern about time *(t)* as another dimension.

By providing a method for handling data with both geographical and time dimensional data, such as historical data, we can achieve many applications. One of them, the *Urban environment and population analysis* is very promising. By viewing time dependent data, we can see how environment, population, landscape evolve and changes through time. And finally analyze and understand these changes and come to future planning.

2.1.2 Historical data

History is about events. What/Where/When it happened? Who did it? How did it happen? Events are the core of studying history. These events are described by time, location and description. By gaining those three information, we have precisely described an event.

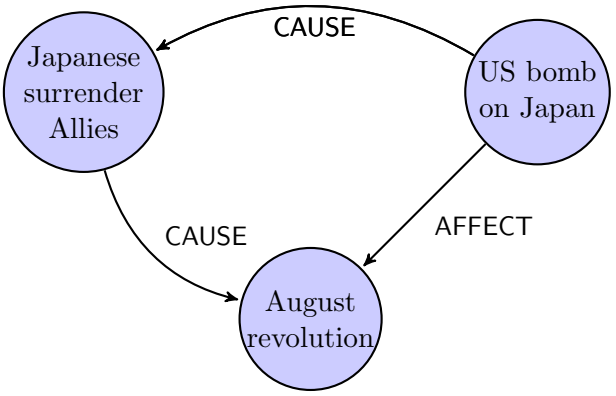

FIGURE 2.1: Graph model for events of August Revolution

Besides the above information, what make history events hard to understand and imagine are the relationships. The relationships of events are the source of precious information, through these relationships we can come up with theories, sometimes conspiracies, and conclusions.

> Let take Vietnam August Revolution[1] as example. The very reason that the event happened and succeeded in August, 1945 is because the Japan had already lost in Wolrd War II. By that time, the Allies[2] had won over almost every battlefield: Europe, Africa, Pacific Ocean, etc. And the crucial event that triggered the surrender of Japan is the Bombings on Hiroshima and Nagasaki[3] which caused Japan to surrendered unconditionally in August 15. The relationship of the revolution and the surrender announcement, the bombings can be shown in figure 2.1.

Therefore this thesis propose the method of modeling the historical data into graph model, where the relationship of events can be easily image. Let we consider the events are nodes, and the relationship are edges, we will have a graph. In the graph, we have nodes relate to each other by relationship, as edges. For the the Revolution, a sample graph can be seen at 2.1.

[1] August revolution, Wikipedia, Online
[2] Allies of World War II, Wikipedia, Online
[3] Bombings on Hiroshima and Nagasaki, Wikipedia, Online

2.1.3 Temporal GIS

Temporal GIS is a GIS system, where spatial data comes with additional temporal information. This make this system the capable of handle map data with time dimension. However, the capabilities of any information system depends on its data models. Data models define data object types, relationships, operations and rules to maintain database integrity [3].

In temporal GIS system, the temporal information is a part of the data objects, and will be used for further analysis by storing or retrieving.

The design of data models is the key to the efficient of system queries. Therefore, a good data models is very important for

2.1.4 Snapshot model

The study for modeling temporal GIS data have been started in 1980s[4]. There have been many approaches has been proposed to model temporal information into spatial data. The most notable, classical ones are Snapshot model (Amstrong, 1988) [1], Space-Time Composite Model (Langran and Chrisman, 1988) [5].

In Snapshot model, the temporal information is associated with a specific time-stamped layers. Which mean every spatial objects we are observing has the same time attributed. As shown in figure 2.2, each layers of data is attached to a specific time. On a layer, i.e T1, all objects will have the time attribute is T1.

In this thesis, I will be using the Snapshot model as the main method to model map data with additional time dimension.

2.1.5 NoSQL

Not Only SQL, or NoSQL, has gained huge attention these years and been used in both many enterprise and personal purpose project.

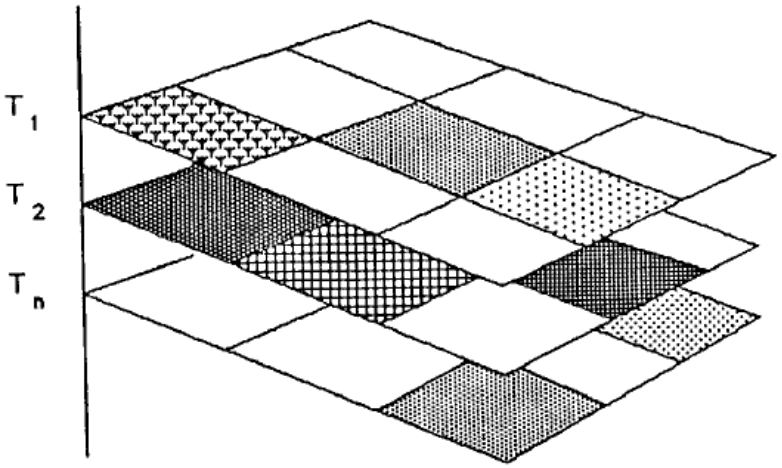

FIGURE 2.2: Temporally stratified layers, snapshot model [1]

NoSQL database provides mechanism to store and retrieve data different than traditional relational database. The principal over the design of NoSQL is: Simple, Scaling and flexible control over data. The structure of data in NoSQL are not tabular based line relational DB but be key-value, graph, document, etc.

Despite having numerous disadvantages over relational database such as relation, atomicity transaction, etc. NoSQL has many advantages in some specific tasks. Two most outstanding of them are[6]:

1 **Faster:** Simpler model data make NoSQL usually faster in queries.

2 **Flexible:** Being schema free, do not have technical requirements, developer can make use of NoSQL the way meet their needs.

2.1.6 Graph Database

Graph Database (GraphDB)[7] is the customized version of NoSQL specialized for graph structure data. GraphDB structure data in nodes and edges, make it is a reasonable choice when data with comes with relationships. This make GraphDB suitable for system and application like social network.

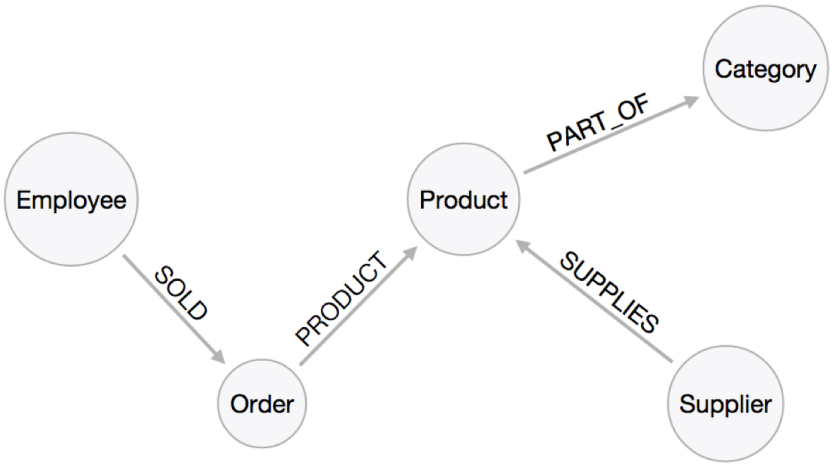

FIGURE 2.3: A sample graph model in graph database, modeling relationship of employees, products, orders, etc.

As shown in [8], GraphDB performed well on data with structural type queries and string search. In this thesis, the searching action on data will mostly on event description, which is text, so using GraphDB may increase performance on queries.

2.2 Related works

2.2.1 Timemap

There have been some similar projects on building a GIS system with additional time dimension. The most popular and maybe the only project for this field is TimeMap[9], a tool allows user to map historical data and archaeological data.

The tools provide a full feature applcation written in JAVA[4]. The system provide operations on map data with consideration on time, includes of:

- A means of recording geographic features which have both a spatial and temporal component

[4] JAVA Programming language

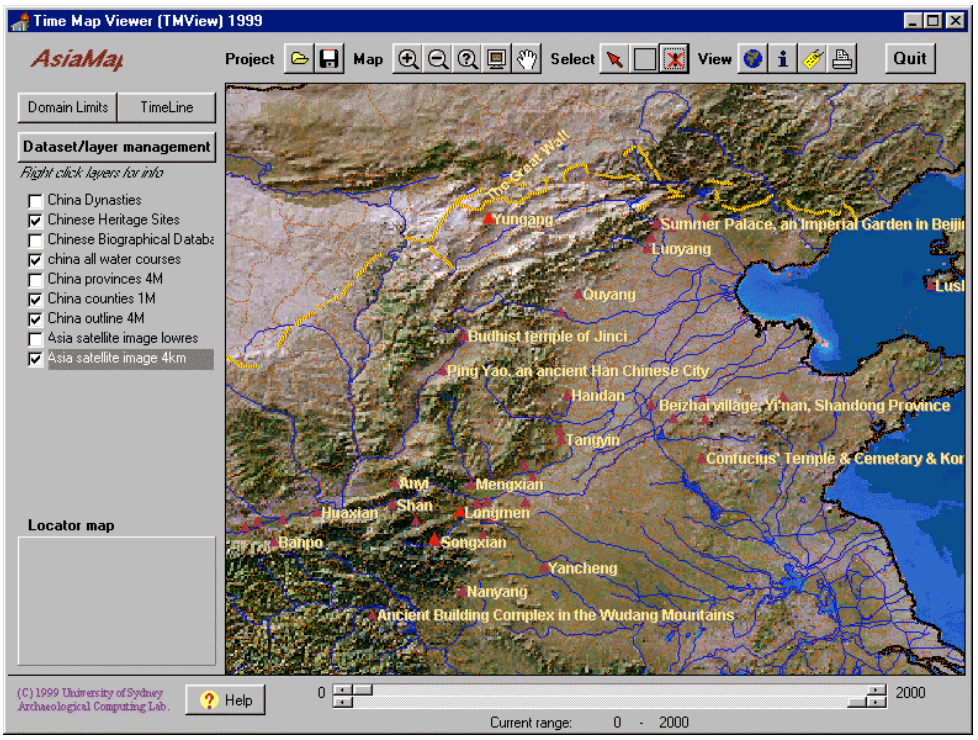

FIGURE 2.4: Timemap Viewer user interface

- Cataloguing and searching on data set
- Display and animation maps through time

The interface of the application is shown on figure 2.4.

However, the project seems to be suspended, outdated. By the time I work on this thesis, I cannot access to the tool, the website, or contact information about the authors.

2.2.2 Reevaluating Vietnam's Nghe-Tinh Soviets (1930-31)

D. D. Testa at[10] had tried to reevaluate Vietnam's Nghe-Tinh Soviets (1930-31) by mapping historical events using Google Maps tools. This project is interesting, because it is one of very rare research on HGIS in Vietnam.

He is a historian and his approach to build the system solely is make use of existing tools. I had some discussions with him and I have known that this project is done quite manually, and took him a lot of effort to manage these data.

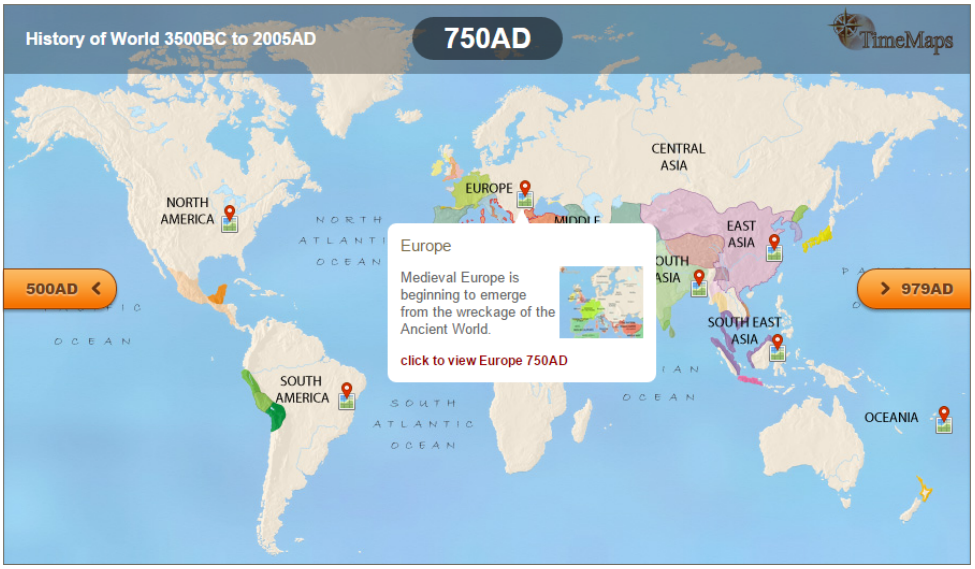

FIGURE 2.5: TimeMaps History Atlas: World History map

This projects is in need of convenient method to managing and visualizing historical data.

2.2.3 TimeMaps History Atlas

The project[11] has a misleading name, easy to confuse with 2.2.1. However, this is a different research project.

The project aims is to provide an easy to understand and follow, history atlas. For example, a map of the world, from 3500BC to 2005AD. As shown in figure 2.5.

The project uses Snapshot model to show the change of time through history, provides a good way for user to interact with visualized data. However, the project is lacking of providing a method for customization and for further research.

2.2.4 Animated Atlas of African History

Project *Animated Atlas of African History 1879-2002*[12] is another Snapshot model adoption, shows the change of Western colonies on Africa, based on specific time layer. As we can see in figure 2.6.

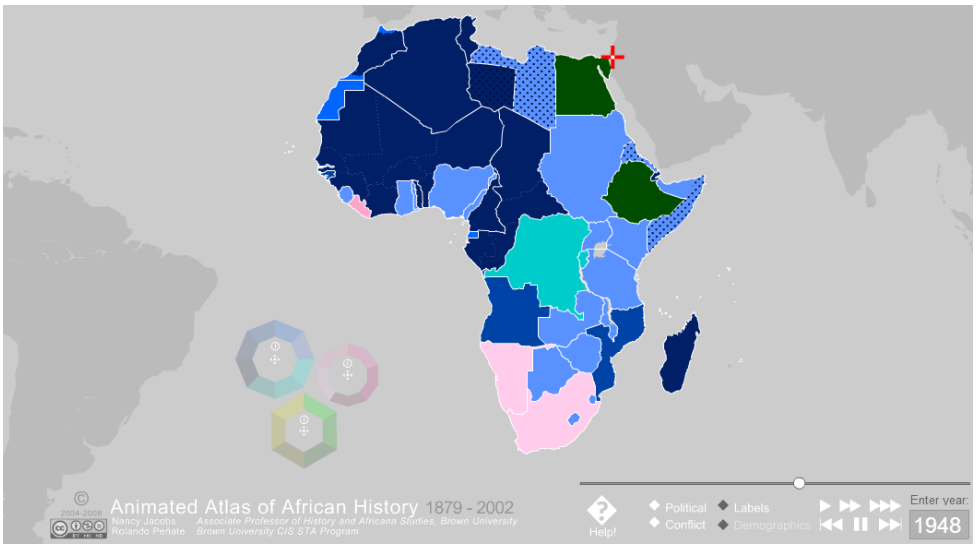

FIGURE 2.6: Animated Atlas of African History user interface

This project provides a control method quite friendly and reasonable, allows changing and viewing the timeline as a flow.

2.3 Tools

2.3.1 Neo4j

Neo4j[5] is a popular GraphDB and has been used by many big project. In this project, Neo4j will be used as the database.

The database provides:

1. **Data model:** Data are modeled as nodes and edges of a graph
2. **Cypher Query Language:** Simple and effective query language for data

[5] Neo4j, "Graph database", Online

2.3.2 REST protocol

The system need an interface for other program or system to use. The interface needs to be standard, simple and popular. Therefore the project use REST[6] protocol for the communication to other system. REST protocol use HTML actions such as GET, POST, PUT, etc. and HTML status code to understand the result of query. The format use for transfer will be JSON for it well structured.

The management actions on data include of: Create, Retrieve, Update, Delete. Those actions will be map to REST protocol as in figure 2.1.

Name	Action	Description
Create data	POST	Data are in body of POST
Retrieve data	GET	Query like id in GET parameters
Update data	PUT	Data in PUT parameters
Delete data	DELETE	Data in DELETE parameters

TABLE 2.1: Actions on data in database

The REST protocol will be used in interaction with Neo4j database. The database has a built-in REST API support.

2.3.3 Openlayers

Openlayers[7] is a powerful library based on Javascript[8], provides the capabilities to visualize map on browser, with many features[13].

Openlayers classified data into many different layers, on top of each other. The tiled layer, vector layer, icon layer, marker layer etc. This classification serves many purposes, include user convenience and software restrictions[14]. Because the layers are independent, when we modify on one layers, it will not affect the others.

[6] RESTful protocol, Wikipedia, Online
[7] Openlayers library
[8] Javascript programming language

FIGURE 2.7: Sample of Openlayers's rendered view about weather in Asia.

Openlayers uses many sources as it tiled layers (map based layer), like Bing Map, Google Map, etc. This provides flexibility.

The library is also great in customization. Styling, positioning, drawing on the map are easy. A sample is shown in figure 2.7. The markers can be styled with different icons, different rotation angle, etc. to show different information about local weather on the map.

Chapter 3

Data Model

3.1 Data structure

In order to model data into graph structure, map data with time information will be treated as event happen in a specific time. These events will be mapped in to nodes and their relationships are edges in the graph.

3.1.1 Events – Nodes

Each event is represented by a node on graph. Nodes have following attributes:

Field name	**Meaning**
id	id of the node
name	name of the event
lon	longitude of the event's location
lat	latitude of the event's location
time	time when event happened
description	description of the event
keywords	keywords featured the event

Table 3.1: Attributes of nodes

The node has its default *id*. Attributes: *name*, *description* and *keywords* are null by default and are optional. Fields: Location (*lon*, *lat*) - define the map position, and *time* of the event are required.

3.1.2 Relationships – Edges

Nodes will be connected by relationships, which is represented by graph's edges. Each edges will has following attributes:

Field name	**Meaning**
id	id of the relationship
type	type of relationship
name	name of the event
from	id of start node
to	id of end node

TABLE 3.2: Attributes of egdes

The graph and its edges are directed[15], because the relationship between nodes are one-way directed (i.e: node A change its attributes and becomes node B but not vice versa). Therefore we need the id of edges's head and tail, as *from* and *to*.

Each edge has default *id* provided by system. *name* and *type* are defined by user. The *type* of relationship depends on relationship of it's *from* node and *to* node. These types will be discusses in 3.3.

3.2 Change of data through time

Let's consider a small map, with a topology of 2 locations, with respectively attributes *aa*, *bb*. With the given map, when we take a snap shot on the map on *time1, time2*, we will have two states. These two states may be identical, slightly different or totally different. The cases are shown in figure 3.1:

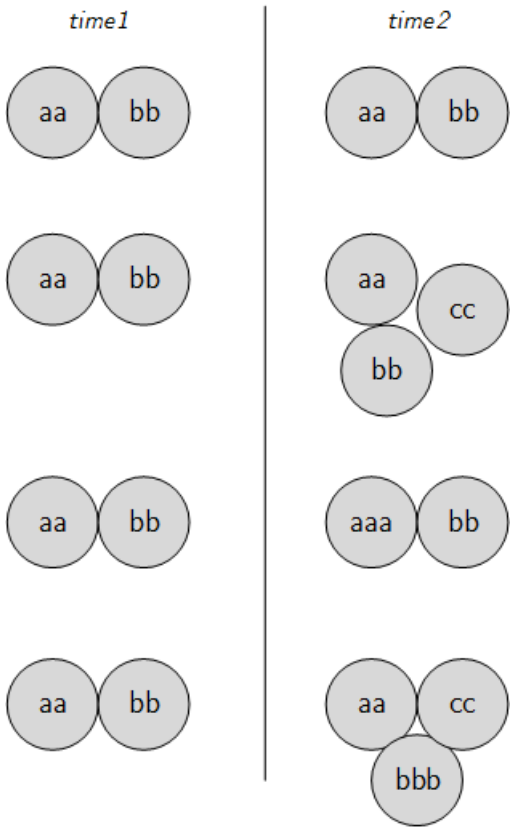

FIGURE 3.1: Change of map when time change from *time1* to *time2*

1. No change: The map stay the same.
2. Topology change: When topology of the map change. I.e: New node shows up, or their positions change.
3. Attributes of locations change: When some attributes of node on the map change.
4. Both topology and attribute change: Both the changes happen.

In order to under successfully understand the map, the system needs con cover all above cases. These changes decides how the map changes on its timeline.

3.3 Type of relationship

The most important thing about a graph is how its nodes are connected by edges. In this system, they are the relationships of object. These relationships are various in different types.

For better understanding, let consider an example:

> Vietnam National University[1] (VNU) has undergone various stages of development: The University of Indochina established on 16 May, 1906; Vietnam National University (November, 1945); the University of Hanoi (June, 1956). And reorganized in December 1993 as Vietnam National University as we know today.
>
> VNU has many universities, institutes, schools members. As its own time line goes on, it's associated with these members. Some of its members are:
>
> - University of Engineering and Technology
> - University of Economics and Business
> - University of Science
> - University of Education
> - University of Social Sciences and Humanities
> - Hanoi University of Science

The history timeline of VNU is described in figure 3.2. As have been shown in the figure, nodes are connected by edges - relationships. These relationships are divided into two groups:

1. *Change_ attribute_ relationships*: where two nodes have a different in attributes, but they describe the same object.

   - CHANGE_NAME: Where object has *name*_1 in event *A* and got *name*_2 in event *B*. (i.e: in figure 3.2, node *N1* change its name from *"University of Indochina"* to *"Vietnam National University"* and become node *N2*).
   - CHANGE_LOCATION: Where object has *location*_1 in event *A* and got *location*_2 in event *B*. (i.e: in figure 3.2, node *N3* change its location fron *"19 Lê Thánh Tông Str."* to *"144 Xuân Thủy Str."* and become node *N4*).

[1] Vietnam National University, Online

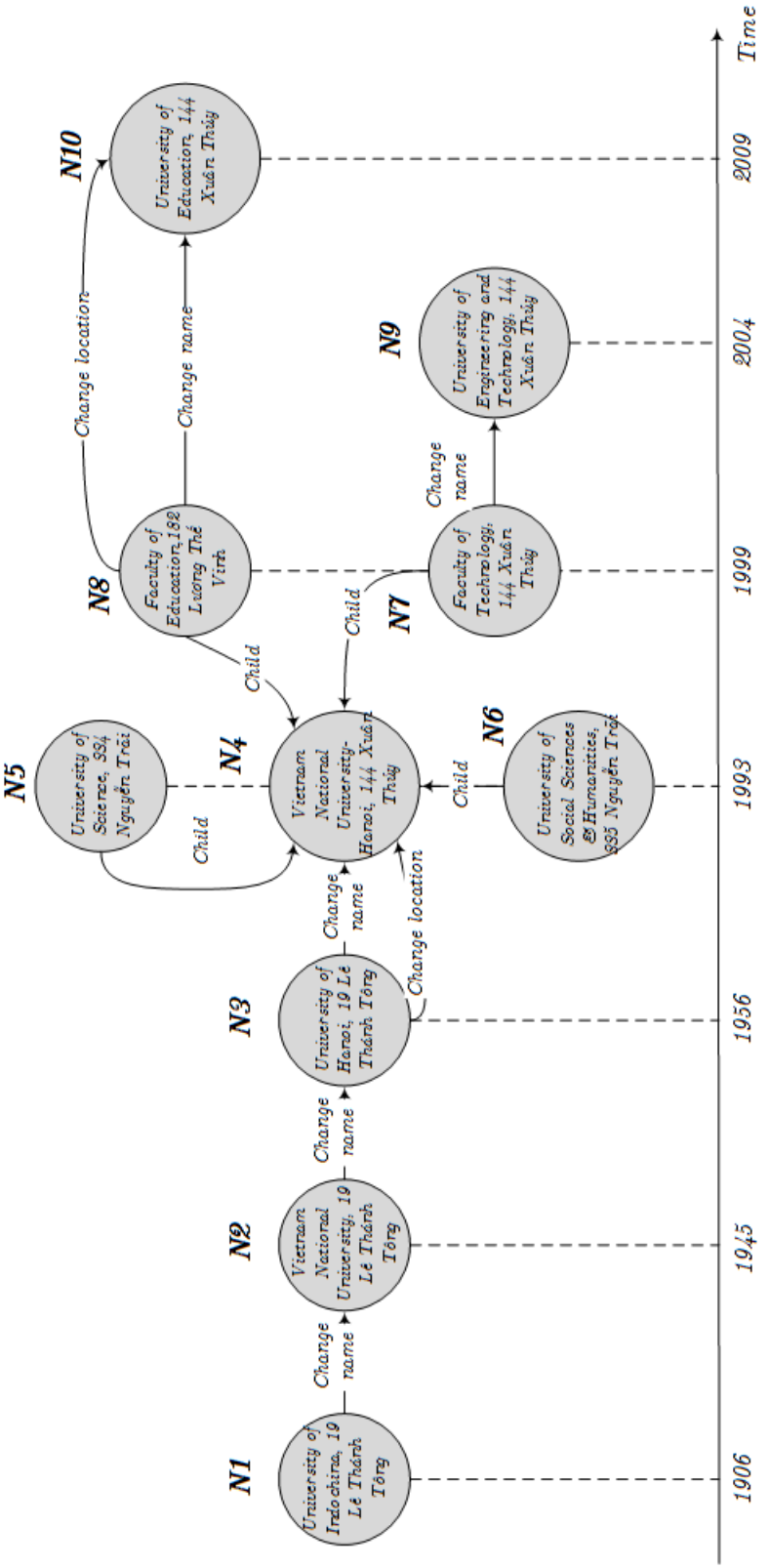

FIGURE 3.2: Timeline of VNU and its member

2. *Connected_ relationships*: where two nodes have connection with each other.

    - CHILD: This is directed relationship, where event A are B's parent. **i.e**: VNU is UET's parent. (i.e: in figure 3.2, node *N5* is the child of node *N4*, because University of Science is a member of VNU).

The first group are the relationships in the case that two nodes of the same thing, change its attributes by time. These nodes can be grouped into a set, with the same label. The second group is the case nodes do not belongs to the same thing, but connected in some way.

3.4 Queries set

Queries are the actions perform on the database to get desired data. These actions determine a database system's value to user[16], because it represent the abilities to provide what user want. In GraphDB, the queries are similar to traditional relational database. They answers questions such as: *What happened in 1999? How many places exists in Hanoi, in 2005?* etc.

This sections discusses the basic queries actions that system provides to user.

3.4.1 Definitions

Before defining the queries for system, we need some definitions.

Definition 3.1. *Object* in the database is the set of nodes: connected to each other, have different attributes, and describe the same thing.

In figure 3.2, Node *N1, N2, N3, N4* are connected, have their attributes changed through time, and describe one common thing, which is what we know as today as VNU.

Definition 3.2. *current_ time*: with given time t, *current_ time* is attribute *time* of a node, describe an object o, with $time \leq t$.

In figure 3.2, given input time $t = 1980$. The *current_ time* of VNU is 1956, because among its nodes, node *N4's time* attribute is the greatest that not greater than t.

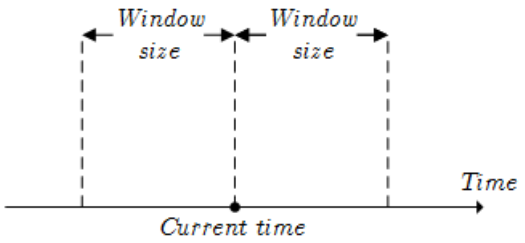

FIGURE 3.3: *window_time* with *current_time* and *window_size*

Definition 3.3. *Window_time*: is a numeric threshold, for the query. Define the range of query *current_time*. The value of *window_time* is *window_size*.

The *window_time* expand our *current_time* from a point in timeline to a range of time, as in figure 3.3. This could lead to conflict when query later, like multiple nodes of the same object satisfy the the greatest value *window_time* attribute)

Definition 3.4. *Current_state*: Set of nodes represents for objects in database, at *current_time*, with *window_time*.

The *current_state* is decides by process in 3.4.2.

Definition 3.5. *Timeline*: Set of nodes and relationships show how an object have changed through time.

Timeline of an object show how it had changed its attributes through time, when and how its change its name, its location, etc. *Timeline* are determined by query in 3.4.3.

3.4.2 Query current view

The most basic action performs on the system is to ask for system state at a specific time.

Input: Let t is the *current_time* for the query, and s is the *window_size*.

Output: The desire output of the query is the *current_state*.

Given *nodes* of one object, function 1 return the nodes satisfy input *t, s*. If there are some node of *nodes* in the *window_time*, the function return them; if there is not, it return the closest node to the window time (the node with greatest value *time* attribute).

Algorithm 1 Get current nodes procedure

```
function GET_CURRENT_NODES(nodes, time, window)
    if exist c_nodes in nodes and node.time in [t − s, t + s] then
        return c_nodes
    else
        return node with greatest node.time in nodes
    end if
end function
```

Function 2 return desired *current_state* of the system. It first fetches all possible nodes in database, iterates through all nodes, find related nodes and decide the appropriate node. Later, and removes the processed nodes from the list and repeat until the list

Algorithm 2 Query *current_state* with *current_time, windo_time*

```
function QUERY_TIME(t, s)
    results := []
    fetch nodes in database
    while empty not nodes do
        fetch first node in nodes
        get object_nodes of node, same object in nodes
        remove object_nodes from nodes
        object_nodes = get_current_nodes(object_nodes, t, s)
        results := results + object_nodes
    end while
    return results
end function
```

A sample process when a query is made is described in example 3.1.

Example 3.1. In figure 3.2, given input time $t = 2002$, *window_size* = 3 years. The *window_time* is [1999, 2005]. One possible process of the *query_time* function is:

1. Initiate result list as []
2. Fetch nodes N_i, i *in* [1..10]
3. Process node *N1*
   - 3.1. Get related nodes *N1, N2, N3, N4*
   - 3.2. There is no related node in *window_time*
   - 3.3. Return the closest node *N4*
   - 3.4. Add *N4* to results: [*N4*]
   - 3.5. Remove *N1, N2, N3, N4* from nodes list
4. Process node *N5*. No related node, add *N5* to results [*N4, N5*]. Remove *N5*.
5. Process node *N6*. No related node, add *N5* to results [*N4, N5, N6*]. Remove *N6*.
6. Process N8, related set [*N8, N10*]. *N8* in *window_time* [1999,2005], add *N8* to result [*N4, N5, N6, N8*]. remove Remove *N8, N10*.
7. Process N9, related set [*N9, N7*]. *N7, N9* in *window_time*, add *N8* to result [*N4, N5, N6, N8, N7, N9*]. remove Remove *N7, N9*.
8. Nodes list are empty. End process.

Therefore the result of query is set of nodes: [*N4, N5, N6, N8, N7, N9*].

3.4.3 Query object timeline

This query return a *timeline* of a given node that we want to know.

Input: Given *node*, *current_time: t*.

Output: Related nodes and relationships.

To know how an object has changed through time, we need to know the informations:

- When did the change happen?
- What has changed? What it is before, after the change?
- How is object at t? What about past, future?

Therefore the needed information to provide the *timeline* is:

1. Set of nodes describe the *timeline*, divided into 3 sets:
    - *Past*: Set of nodes in the past
    - *Current*: Node represent current node
    - *Future*: Set of nodes in the future
2. Set of relationships connected the nodes
3. Current time on the *timeline*

Set *current* are determined by closest node to t. The *future* and *past* sets are determined by comparing *time* attribute to the *current* set.

Example 3.2. In figure 3.2, let t is 1980, *node* is N2. Then, the *timeline* we have is:

1. Nodes
    - *Past*: *N1, N2*
    - *Current*: *N3*
    - *Future*: *N4*
2. Relationships: *R1, R2, R3*
3. Current time: 1980

3.4.4 Query Children

The relationships between nodes are only *change_ attribute_ relationships* type, they can also be *connected_ relationships*. These relationships tell how nodes are connected to each other. In this thesis, I have proposed one relationship CHILD.

Let t is the *current_ time* for the query, node *node* is the node we want to find related nodes and *rel_ type* is the relationships type we want to find. The desire output is the set nodes that related to *node* by type *rel_ type*.

Currently, in this thesis, I have only proposed **one** *connected_ relationship*: CHILD. Therefore from now on, we will assume that *rel_ type* default to be CHILD and do not need to be consider.

In consideration of *current_ time*, when we perform this query, we will want to find the children of a node, we will want to know **current state** of the child, not the time when it started to have the connection.

Example 3.3. In figure 3.2, let t is 2008 and we want to find children of node *N4*, then they will be: *N5, N6 N9, N8*.

In case of *N9*. The direct node has the relationship with *N4* is *N7*. However, the closest node to t is *N9*. Therefore the more appropriate node in this case is *N9*.

Chapter 4

Visualizing mechanism

Based on data model and queries were described in chapter 3, this chapter discusses about the mechanisms for visualizing those data, rendering it to view, and system's user interface.

In this thesis, data will be visualized based on web technology, and interact with user through web browsers. The HTML5 technology provides many promising features for visualize map data[17]. By using Openlayers, the system can make use of these features.

4.1 Map and layers

The rendered map to user are organized into many layers on top of each other. Each layer provide a specific data class we defined.

The base layer of the map is the **tiled layer**. This layer provide the based view, the landscape, the roads, bridges, forests, rivers, lakes, etc. things that usually do not change on the map for a long time. A sample tiled layer us shown on figure 4.1. This layer usually are tiled images, from provided sources. In this thesis, the main source is Bing Map[1].

Marker is an icon on the map. The marker is a representation of a node on the map.

[1] Bing Map, Microsoft

FIGURE 4.1: A sample tiled layer of Hanoi. Source: Bing Map

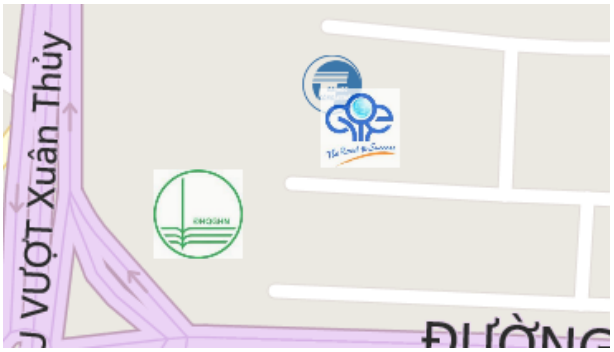

FIGURE 4.2: A sample marker layer, with 3 markers

The **maker layer** is the layer where defined **markers** of object are placed on. Figure 4.2 shows a marker player placed on top of a tiled layer. In the figure, there are 3 markers represent: Vietnam National University, University of Engineering and Technology, University of Economics and Business.

The **pop up** layer is the layer to show details information about each marker. When a marker is clicked, a popup shows up on the marker and show information about the marker. A sample popup is shown in figure 4.3. The marker in figure show the name of marker and its *current_time*.

4.1.1 Controls on map

The system needs to provide user a way to control current configurations. These configurations of the system are: *current_time* and *window_size*. These configuration will be used in render

FIGURE 4.3: A sample popup layer, with a popup shows information about VNU

FIGURE 4.4: The controls system provides: sliders to change current time and window size

phase of data on the map. The controls for user are shown in figure 4.4.

4.2 Visualizing data on map

4.2.1 Node representation

The given data for the visualization on the map is the *current_ state* of the database, the nodes. Each nodes is mapped into a marker on the map. Each marker has following attributes:

1. Location: Follows *lon, lat* of the nodes. The system needs to transform the location to corresponding pixel on the map
2. Icon: The icon representing the node. The icon is decided based on description of the nodes. I.e: If the node describes VNU, then the icon is VNU's logo

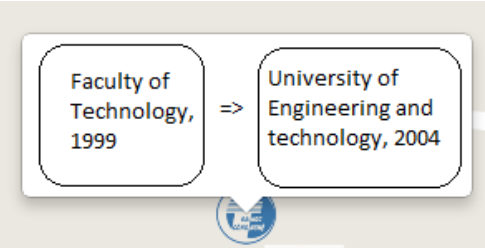

FIGURE 4.5: A sample rendered view of marker in case of overlap marker

4.2.2 Changing time

The current view on the map is rendered base on query on database. With current configuration includes current time and window size, the database query return *current_state*. The nodes from *current_state* are mapped into markers on the map.

When user change the time with input controls, the view to user needs to change. A query with given *current_time, window_size* is performed. The returned *current_state* is used to re-draw the map.

4.2.3 Overlapping content

In some cases when *window_time* are narrow, there might be a chance that two nodes has the same location. I.e: In figure 3.2, if the *window_time* are [1998, 2005], then both node *N7, N9* are in the window. Because both of these nodes has the same *lon, lat* attributes, they will will be mapped into two overlapped makers.

In this case, the markers will be visualized on the map as:

- Show marker of node with greatest *time* attribute
- Show info of nodes in the popup in order of time

The result view is shown in figure 4.5.

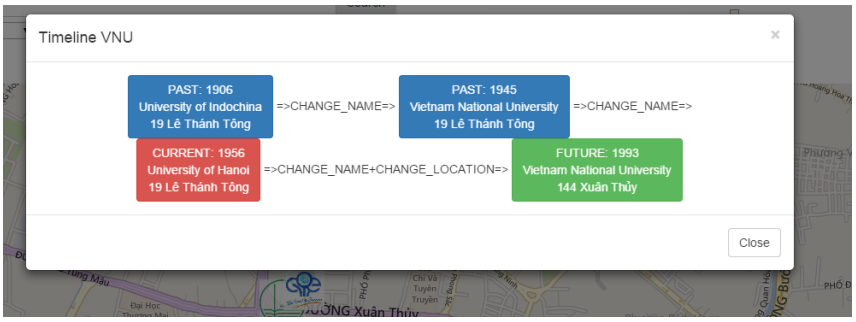

FIGURE 4.6: A sample rendered timeline on map

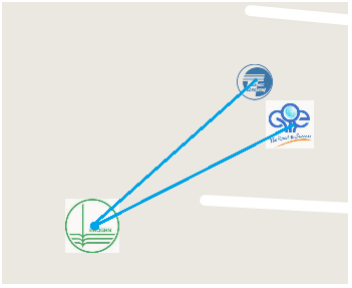

FIGURE 4.7: Children nodes of VNU

4.2.4 Show timeline

When viewing a marker information in popup of a marker, user can ask system to show the *time_line* of corresponding node of the marker. The view based on query on database. The *timeline* shows events happened in the history timeline of the object, and the relationships between the event, i.e: change its name, change its location, etc.

On figure 4.6 is the rendered timeline view on map of VNU, with *current_time* is 1970, and *window_size* is 3 years.

4.2.5 View children

When viewing a marker information in popup of a marker, user can ask system to show the *time_line* of corresponding node of the marker. The view based on query on database. The children nodes are the nodes of *current_time* are currently on the map. On the map, there will be link between the original, the parent node and its children, as shown in figure 4.7.

Chapter 5

Experiments

5.1 Implementations

5.1.1 System architecture

System are divided into 2 parts: Client and Server, the Server side come with Database, as shown in figure 5.1.

Server receives requests, query request data from database and return to client.

The client have two modules. The module **DB handler** is in charge of communicating with Server, send requests, waits for return data. These data are later rendered to view by module **Map view handler**, following mechanisms have been shown in chapter 4.

Database is installed by Neo4j Graph Database, receive query by its built-in REST API and Cypher Language.

Client side are installed with Javascript, based on HTML5.

5.1.2 System specifications

The environment for the experiments specification is shown in table 5.1.

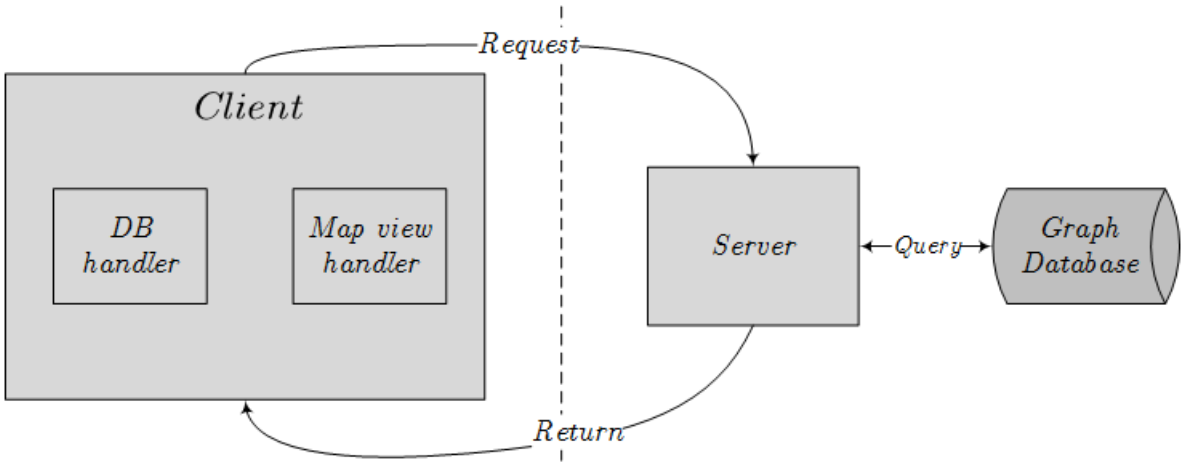

FIGURE 5.1: System architecture

Name	Details
Client	Laptop HP 4530s on Windows 8.1. CPU: 2x2.3GHz. Memory: 4GB.
Server	Dell desktop on Ubuntu 14.10. CPU: 2x2.2GHz. Memory: 3GB.
Webserver	Apache 2
Tools & verions	Openlayers 3.0; Neo4j 2.1.6
Network	Local area network

TABLE 5.1: Experiments environment

5.1.3 Demo data

To run the experiment, the use a demo data on area around Vietnam National University Hanoi at 144 Xuan Thuy Street, Cau Giay District, Hanoi, Vietnam.

The data focus on the history of Vietnam National University and its members in the area. The source of information is from official websites of Vietnam National University Hanoi[1], University of Engineering and Technology[2], University of Economics and Business[3], University of Language and International Study[4], University of Education[5].

[1] Vietnam National University Hanoi
[2] University of Engineering and Technology
[3] University of Economics and Business
[4] University of Language and International Study
[5] University of Education

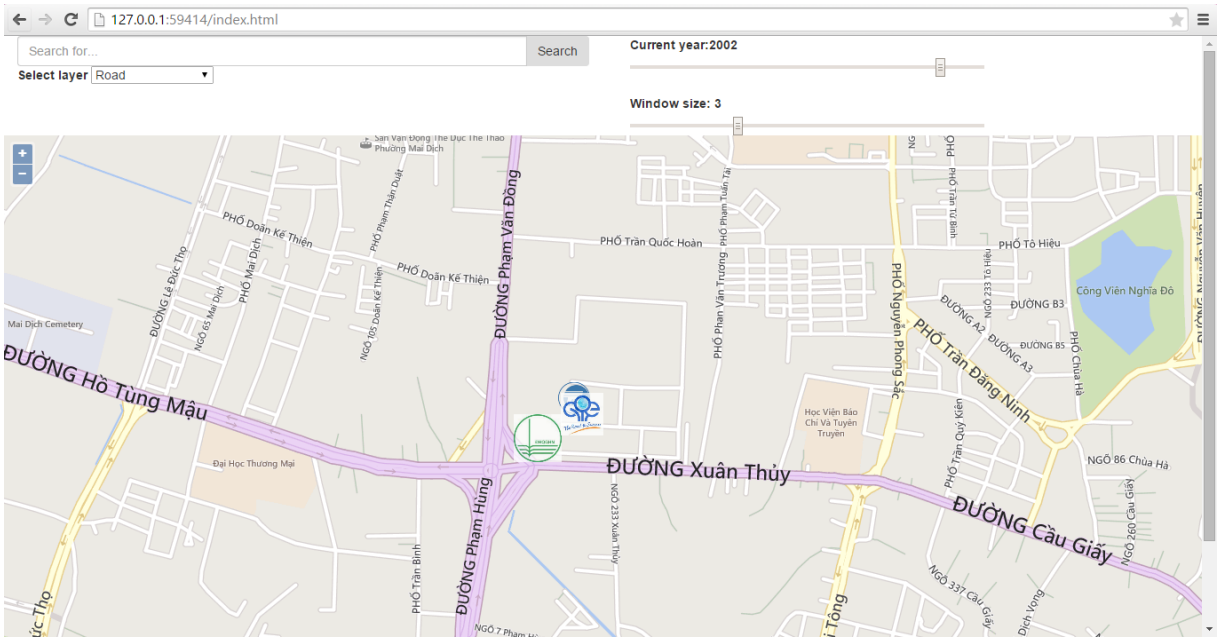

FIGURE 5.2: System user interface

These data provide the information about the timeline of the members and theri relationships with Vietnam National University.

5.1.4 Demonstration

The view of system user interface is shown in figure 5.2. To test the system, we perform the user actions:

1. Change current time
2. Change current window time
3. View timeline
4. View children of a node.

The system has implemented the features for showing the actions 1, 2, 3. The action 4 about view children of a node haven not been finished yet.

5.2 Results and discussions

The experiments results showed it is possible to model map data with additional time information by using graph theory. The relationships of nodes in the graph are capable of modeling the connection of data like the parent-child relations suggested in 3 as the CHILD relationships, or the connected nodes when attributes change through time like the CHANGE_NANE and CHANGE_LOCATION relationships.

The visualization mechanisms of system user interface has provided user a way to interact with the data on map on browsers. User are able to view the change of the area round VNU by changing the time. This can be applied in many real life application, especially for study history.

However, the data is only a small set. The system have not been tested with large data. In that case, the database might shows problem with queries when there are too many nodes and edges. The visualization is still lacking some features like view the children of the nodes.

Chapter 6

Conclusion

6.1 Conclusion

In this thesis I have proposed a method for manage map data with additional time dimension based on graph model. The method provided basic CRUD actions on data in graph databases. I also proposed the mechanisms to visualize these data to user for better understanding. The implementation show the feasibility of proposed method.

However, the project still lacks in implementation and experiments on data. The testing data is small and didn't show the capability of system. The data system work with only is point data, not regional data - which is more popular in map data. The relationships between event is quite simple and have not fully stimulate real life relationships.

6.2 Future works

6.2.1 Work with regional data

Currently, the system only work with point data. In the future, it can be adapt to regional data like provinces, states, lake, etc. To do that, we might use graph database as a abstract layer,

manage a GIS database like PostGIS[1].

6.2.2 Refine data model

The current data model has shown the ability to use graph structure to model map data with time information. However, there is still place for further research, in:

- Add more attribute
- Refine relationships of events: Add more complex relationships
- Add more action on queries, based on added relationships

6.2.3 Relationship weight

Current edges of graph are not weighted. In future, we can add *weight* to these edges. This *weight* will help to decide the most suitable candidate for queries, which provide better results and reduce result set's size. The weight needs to be formalized into range [0,1].

6.2.4 Data

In this thesis, the demo data is quite small, so there might be a problem with the system when data is too large. Therefore in the future, the data set needs to be enlarge

6.2.5 Automatically Determine events' relationship

In the thesis, the relationships of the events are defined by user. These relationships can be automatically determined, based on analyzing the changes in attributes of nodes, at different time. As stated in **??**, the edges can have weight. This weight can also be determined automatically.

- *CHANGE_NAME* This relationship can be detected by comparing the *name* attribute of the two events in *time*1 and *time*2. The weight is the string relevant.

[1] PostGIS

- *CHANGE_ LOCATION* This relationship can be detected by comparing *lon, lat* attribute of the two events in $time1$ and $time2$. The weight can be the distance of the two locations. I.e: $w = \frac{1}{log(\sqrt{(lon_1 - lon_2)^2 + (lat_1 - lat_2)^2})}$. The greater the distance, the less likely two events are related.

- *CHILD* This relationships can be deducted from keywords of the events. If the two events has the same keywords, it may has the parent-child relationship. To decide which one is the father, we can compare the *time* attribute. If there is no other relationship, and the event happen first is the parent, the later is the child.

References

[1] Marc P Armstrong. Temporality in spatial databases. In *Proceedings: GIS/LIS*, volume 88, pages 880–889, 1988.

[2] Peter N Stearns. Why study history? *AMERICAN HISTORICAL ASSOCIATION, available at https://fcps.edu/FairfaxHS/pdfs/summer2013/APWH_2013_2of3.pdf*, 1998.

[3] Christopher John Date, Christopher John Date, and Christopher John Date. *An introduction to database systems*, volume 7. Addison-wesley Reading, Mass., 1986.

[4] May Yuan. Temporal gis and spatio-temporal modeling. In *Proceedings of Third International Conference Workshop on Integrating GIS and Environment Modeling, Santa Fe, NM*, 1996.

[5] Gail Langran and Nicholas R Chrisman. A framework for temporal geographic information. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 25(3):1–14, 1988.

[6] Neal Leavitt. Will nosql databases live up to their promise? *Computer*, 43(2):12–14, 2010.

[7] Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1, 2008.

[8] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. A comparison of a graph database and a relational database: a data provenance perspective. In *Proceedings of the 48th annual Southeast regional conference*, page 42. ACM, 2010.

[9] Ian Johnson. Mapping the fourth dimension: the timemap project. *BAR INTERNATIONAL SERIES*, 750:82–82, 1999.

[10] D. D. Testa. Reevaluating vietnam's nghe-tinh soviets (1930-31). Online, Available: http://www.nghetinhsoviets.org/mapping-the-nghe-tinh-soviets/, 2011.

[11] TimeMaps. Timemaps, world history atlas. Online, Available: http://www.timemaps.com/, 2015.

[12] I. T. G. S. Institute. Animated atlas of african history 1879-2002. Online, Available: http://www.brown.edu/Research/AAAH/index.htm, 2012.

[13] Erik Hazzard. *Openlayers 2.10 beginner's guide*. Packt Publishing Ltd, 2011.

[14] Stan Aronoff. Geographic information systems: a management perspective. 1989.

[15] Jonathan L Gross and Jay Yellen. *Graph theory and its applications*. CRC press, 2005.

[16] Raghu Ramakrishnan and Johannes Gehrke. *Database management systems*. Osborne/McGraw-Hill, 2000.

[17] Mats Taraldsvik. Exploring the future: is html5 the solution for gis applications on the world wide web. *Norvegian University of science and technology, Department of civil and transport technology*, 2011.