Storing Dynamical Data Using Natural Language Addressing

Krassimira Borislavova Ivanova
University of Telecommunications and Post,
1700, Bulgaria, Sofia, 1 Acad. St. Mladenov str.
e-mail: kras78@mail.bg

Abstract. The present paper refers to the possibilities for storing dynamical data from tracking moving objects. The urgency of this problem is determined primarily by the need for electronic registration, storage and processing of large volumes of semi-structured dynamical data in various fields of practice. The new problem raised, is the way in which the relationships between the data are described. The capabilities of modern relational database management systems do not allow rapid readjustment in accordance with the changes in practice. The emergence of new types of data connections requires readjustment, which takes time and requires a certain level of skill. In this article we consider an approach based on "Natural language addressing", which allows solving these problems and we will offer a conceptual design (prototype) of an extension to existing systems, which facilitates the work, especially with dynamical semi-structured data and implicit relationships between them.

1. Introduction
The present paper focuses mainly on the possibilities for storing data from tracking moving objects. The urgency of this problem is determined primarily by the need for electronic registration, storage and processing of large volumes of semi-structured dynamic data (Big Data) in various fields of practice.

The toll collection systems are based on modern technologies that allow convenient ways to pay and control, without slowing down and impeding the free movement of road vehicles [1], [2], [3], [4].

A more detailed presentation and analysis of toll systems in the European Union is given in [5]. What are important are the problems of switching to work with large dynamic data sets and especially those related to data storage and extraction of patterns in real time. The volume of data on vehicles that are collected and indexed continuously will soon become so large that traditional databases will be overloaded and their work will become extremely slow. It should also be borne in mind that this is dynamic data and it is necessary to know which the new data is, which is old and which is no longer needed. Furthermore, it is never a single vehicle, but thousands that are moving at the same time and all of them must be monitored at the same time.

All this extremely large and dynamic data traffic, which requires traditional databases to be constantly reconstructed and indexed, leads to the fact that at some point they will start using almost all the time for their own self-maintenance or, more likely, the vast majority of data will be backed up to external media and will not be quickly accessible in real time.

When designing and developing automated systems for servicing the activities of vehicle traffic monitoring and electronic reporting of tolls, it is necessary to follow the legal and regulatory requirements governing the main processes - data collection and storage, data retrieval, processing and presenting detailed or summarized results to users, etc. All these requirements are already covered by the existing electronic toll collection systems and should not, with an implemented and working system,
go to the design and implementation of a new one. The correct approach is to expand the capabilities of the existing system with new functionalities that were not available at earlier stages.

This is the leading idea of the current work - to propose an extension that would allow a qualitatively new type of organization of data storage for vehicles, which would supplement the already realized possibilities in the available databases, preserving all available functionalities for data storage of the existing electronic system.

Within the electronic toll collection system, unique identifiers must be used, both for the vehicles themselves and for the electronic devices for registration of their movement. They are accompanied by metadata providing additional data on the vehicle.

All these data for thousands of vehicles, together with the real-time data generated by vehicles’ traffic, give rise to a significant amount of dynamic data that have to be stored, accessed and processed in real time. Naturally, the electronic toll collection systems need appropriate solutions to meet the challenge called Big Data.

The present paper focuses mainly on the possibilities for storing Big Data from tracking moving objects. It is organized as follow: Chapter 2 contains a brief introduction to Natural Language Addressing (NLA) and Chapter 3 describes the functions and experiments with NLA_Trace system. The paper ends with a conclusion.

2. Natural Language Addressing (NLA)
The Natural language addressing (NLA) is the use of letter codes as coordinates (addresses) to access data [6]. In this way, indexing is avoided and high-speed direct data access is available.

For example, let's have the following case:

number of vehicle: **CA2498BX**; owner of the vehicle: **Sakar 3000 OOD**.

In the computer memory this can be stored in a file, for instance, at the address "001042800" and the index pair is: ("CA2498BX", "00084920"), where CA2498BX is a key that identifies the record "Sakar 3000 Ltd."

The data (main text) is stored with key "CA2498BX". To read, we first find the key "CA2498BX" in the indexes and then access the address in memory "001042800" to read the text of the document.

If we assume that the name "CA2498BX" is encoded in the computer memory with eight numbers (letters) and we use the ASCII codes for encoding "CA2498BX", it will look like this - (67, 65, 50, 52, 57, 56, 66, 120) and then we can use this coordinate vector for direct addressing in memory:

("CA2498BX", "67, 65, 50, 52, 57, 56, 66, 120")

We have the same thing written twice above, but this way we can read the address directly. For the user this word means " CA2498BX ", for the computer it will mean "67, 65, 50, 52, 57, 56, 66, 120".

2.1. Benefits of NLA
In order to be able to realize Natural Language Addressing, we do not need anything other than dynamic perfect hash tables.

Hash tables are used in many applications and are attractive because of the constant algorithmic complexity they can achieve. However, collisions are common and can lead to a serious increase in execution time. This means that we cannot use names as hash table keys or as elements. We need a special organization of the file's internal structure and function to transform the name into the location of the computer where the data (text) will be stored without collisions.

This issue is resolved at the file system level. Each file has its own name and each system converts it to an address using distribution tables. This is convenient for information that is relatively long, such as documents, images, etc., because each file occupies at least one cluster. To expand the idea of using files, several types of file internal organization with special additional indexing are established. There is a difference between simple hashing and the development of this idea:
Dynamic perfect hash tables are hash tables that have separate slots for all possible keys of a certain size, and collisions above this size are resolved by subordinate perfect hash tables, i.e. there is a hierarchy of perfect hash tables.

The function that uses the encoding of letters with integers locates unambiguously and there is no way to get collisions. This function can be used recursively for each string character and to build perfect hash tables on many levels and thus have quick access to the data.

For example, the array "67, 65, 50, 52, 57, 56, 66, 120" can be considered as a route to a point in a multidimensional information space and the text "Sakar 3000 Ltd." can be stored at this point.

In other words, our function can be recursive for each character and thus build a hierarchical multilayer set of tables. In the case of "SA2498VH" we will have eight levels.

2.2. Application of Natural Language Addressing

Using the Natural Language Addressing, it is possible to record data directly in accordance with the vehicle identifiers. Moreover, a travel history can be created - which point after which it was - what coordinates the vehicle passed through and, moreover, a distributed, accessible in real time, archive of the passed and probable future points can be maintained. Through the natural language addressing it is possible to compute where the vehicle could be, by very elementary calculations of distances in speed and time. Any future trajectory can be determined at any time. This means that the next position of the vehicle can always be determined and appropriate decisions can be made. For example, if there is a technical problem, such as a torn roof covering of the van, which is registered by the cameras but the driver has not noticed it, and if there is no connection with the driver of the vehicle, a technical team can be sent to meet him at next point of the way and repair the damage. This requires serious intelligent real-time data processing.

3. NLA_Trace system

As noted above, each vehicle is associated with a significant amount of metadata describing different semantic aspects. The availability of standardized languages for describing metadata (XML, SHOE, DAML + OIL, RDF and RDFS, etc.) makes metadata available for analysis and interpretation. Increasing metadata is a phenomenon that requires special attention and analysis. It gives the illusory impression that the electronic system has everything we are interested in, but it has not yet been fully described by metadata and it is only a matter of time before that happens. When this happens, semantic search engines will find the necessary data for us, which must be "sufficient".

The search for effective solutions for working with the natural structures of language and their meaning remains a serious problem. Here we will consider a possible step in solving this problem by proposing a distributed NLA_Trace system for storing data on vehicles on highways and first-class roads in Bulgaria, based on "Natural Language Addressing". The system is based on the NL-ArM access method and the MDNDB™ (“Multi-Domain Numbered Data Base”™) tool system. MDNDB™ is a multi-model database designed to work with distributed multi-space databases, in particular with RDF graphs. MDNDB™ is developed on the base of ArM32, BigArM and NL-ArM access methods and is an instrumental basis of the “INFOS” system [7].

3.1. NLA_Trace data structures

In NLA_Trace, the length of each element (string) can vary from 0 to 1G bytes. There is no limit to the number of strings in the archive. There is no limit to the number of files in the database, as well as their location, including on the Internet.

The main storage idea in NLA_Trace is through triplets of the type:

\[ <\text{Vehicle ID}> <\text{Metadata ID}> <\text{Data}> <\text{CR}> \]

where all three elements are natural language strings.

Metadata identifiers are accepted as layers, and vehicles’ identifiers are accepted as paths valid for all layers. Both elements are not recorded in the archives - they are natural language addresses. Only the
data are stored in appropriate containers located at addresses indicated by the vehicles’ identifiers in layers indicated by the metadata identifiers.

3.2. **NLA_TRACE functions**

NLA_TRACE has two modes of operation: automatic and manual. The main functions are Write and Read.

3.2.1. **Data input.** The input file is in CSV format. Every triple `<Vehicle ID> <Metadata ID> <Data>` occupies one record in the input file. There is no limit to the number of records in the file. The system reads records sequentially from the file and for each of them:

1. Converts `<Vehicle ID>` and `<Metadata ID>` into spatial addresses;
2. Store `<data>` in the container located at the address `<Vehicle ID>` in a layer specified by `<Metadata ID>` in the triple.

After saving of all triplets, two information lines are displayed:
- Total time used to store all instances of the file;
- Average time used to store one instance in tics (milliseconds).

The time used strongly depends on the capabilities of the operating environment and the speed of the hardware.

3.2.2. **Data retrieval.** For data retrieval, NLA_TRACE uses as input a pair file `<Vehicle ID> <Metadata ID> <CR>` (each pair in a separate line) and retrieves the corresponding `<data>` from the archive. If some `<data>` does not exist, the output is empty, i.e. `< >` (one space).

The result is a set of triplets:

```
<Vehicle ID> <Metadata ID> <Data> <CR>
```

occupying one record in the output file.

There is no limit to the number of records in the file. After starting this function, the system reads sequentially from the input file and for each record:

1. Converts `<Vehicle ID>` and `<Metadata ID>` into spatial addresses;
2. Retrieves `<data>` from the container located at the address `<Vehicle ID>` from a layer specified by `<Metadata ID>`.

After finishing the operation, two information lines are shown:
- Total time used to retrieve all instances;
- Average time used to retrieve an instance, in tics (milliseconds).

The time used strongly depends on the capabilities of the operating environment and the speed of the hardware.

3.3. **Program experiments**

The program experiments were performed on the following computer configuration:

- Processor: Intel Core2 Duo T9550 2.66GHz; CPU Launched: 2009, Average CPU Mark: 1810 (PK = 1810);
- Physical Memory: 4.00 GB (MK = 4);
- Hard Disk: 100 GB data partition; 2 GB swap (DK = 100);
- Operating System: 64-bit operating system Windows 7 Ultimate SP1.

The tests assume that moving road vehicles (RVs) are observed following certain routes. Some vehicles are allowed to have more than one passage through a given control point, at different times, as well as several objects to pass simultaneously at the same time, but in different lanes of the road, i.e. even if they are at the same point at the same time, they are in a different position.

To perform experiments with NLA_TRACE, two data sets were prepared, containing 1000 and 10000 instances (triples), respectively, containing vehicle’s identifier, metadata name and metadata value (string). Vehicles’ identifiers are randomly generated. They consist of eight-character vehicles’ numbers
with letters and numbers. The names of metadata (90 six-character: word METADATA and two digits) and values (Bulgarian names - variable-length strings) are also generated at random (Figure 1).

AO7646TP;METADATA54;НИКОЛА БРАНИМИРОВ ИВАНОВ; BP6375XT;METADATA44;ГРИГОР ГАВРИЛОВ ГРАНДЖЕВ; CT2352YX;METADATA54;ЉЪЕЗАР ВЕНЦИСЛАЛОВ ИВАНОВ; EX1803AY;METADATA18;САМИ МОХАМАД АЛЧАЛИАН; 

a) input data

AO7646TP;METADATA54;
BP6375XT;METADATA44;
CT2352YX;METADATA54;
EX1803AY;METADATA18;

b) test data

AO7646TP; METADATA54; НИКОЛА БРАНИМИРОВ ИВАНОВ; 
BP6375XT; METADATA44; ГРИГОР ГАВРИЛОВ ГРАНДЖЕВ; 
CT2352YX; METADATA54; ЛЪЧЕЗАР ВЕНЦИСЛАЛОВ ИВАНОВ; 
EX1803AY; METADATA18; САМИ МОХАМАД АЛЧАЛИАН;

c) extracted data

Figure 1. Sample instances of input (a), test (b) and extracted data (c).

The tests performed showed a recording speed of one instance from 3.6 to 6.2 milliseconds. We should note the time spent on the initial creation of archive structures. In this case, the increase in time is due to the emergence of more types of metadata, the initial registration of which takes time to create the corresponding files (archive layers) and the speed varies from 150 to about 300 instances per second. When extracting, the speed is constant around 700-800 instances per second (Figure 2).

Figure 2. Results for data recording and retrieval - (a) for 1000 vehicles, (b) for 10000 vehicles.

To illustrate the ability to track routes, for the same vehicles, we entered data for passing through three control toll points, and the metadata "toll" was set three times in succession with different values.
To illustrate the possibility of working with a lot of metadata, we introduced three more - "owner", "address" and "branch". The result is displayed in a text file, which is visualized using WordPad (Figure 3a).

Then we added new data for "branch" and "toll" simulating the opening of a new branch of the company, as well as a new route. No data is lost when adding new data (Figure 3b).

4. Conclusion

In this paper, a possible approach for the implementation of a distributed system for storing data on the movement of road vehicles and related metadata and analytical results, based on Natural Language Addressing, was presented. A project (prototype) of vehicles’ data storing system based on Natural Language Addressing (NLA) was considered. Some experiments performed with the system, realized on the basis of an available software implementation for storing data through NLA, were presented.

Analyzing the results of the experiments, we can note that the main conclusions made in [6] regarding NLA are valid here as well.

In all traditional relational databases, continuous reconstructions of the index structures must be done due to the incoming dynamic data. This is a major and extremely serious problem. The volume of data that is collected and indexed continuously soon becomes so large that traditional databases become overloaded and their work becomes extremely slow as they begin to use almost all the time for their own self-maintenance.

NLA does not require such updates. This prevents overloading and slowing down the operation of databases, even with the accumulation of huge arrays of data. The speed of work is constant and independent of the volume of data, i.e. there is a constant algorithmic complexity O(1).

In addition, important advantages of the approach are:

- The reduction of the amount of occupied memory due to the complete absence of additional indexes, absolute addresses and additional files;
- Reduction of processing time due to the complete lack of demand - the data is stored/extracted to/from a direct address;
- The universal presentation of data, both accessible to both humans and the automated system.

The NLA is a fundamentally new approach to database organization that does not replace, but naturally complements, other widely used other types of database management systems. As direction for future work we can point out are the activities for extracting essential data with the help of functions with artificial intelligence and presentation (visualization) of summarized results to the user.
References

[1] Bagoria R, Mittal N, Garg K and Ameria 2015 *A Real Time Toll Rate Determination Using Image Processing & Network Database* IJCS 02(11) 490-493

[2] Popoola S, Popoola O, Oluwaranti A, Atayero A, Badejo J and Misra S 2018 *A Cloud-Based Intelligent Toll Collection System for Smart Cities* In: Bhattacharyya et al. (Eds.): NGCT 2017, CCIS 827 (Springer Nature Singapore Pte Ltd.) 653–663

[3] Soomro S, Javed M and Memon F 2012 *Vehicle number recognition system for automatic Toll tax collection* 978-1-4673-4886-7/12/ (IEEE) 125-129

[4] Persad K, Walton M and Hussain S 2007 *Vehicle/License Plate Identification for Toll Collection Applications* (Center for Transportation Research The University of Texas at Austin, USA) p 38

[5] European Commission 2015 *Study on State of the Art of Electronic Road Tolling* MOVE/D3/2014-259 (EC, DG for Mobility And Transport, DD – Logistics, maritime and land transport and passenger rights) p 113

[6] Markov K, Ivanova K, Vanhoof K, Velychko V and Castellanos J 2015 *Natural Language Addressing* (Hasseit, Kyiv, Madrid, Sofia: ITHEA®) p 315

[7] Markov K, Vanhoof K, Mitov I, Depaire B, Ivanova K, Velychko V and Gladun V 2013 *Intelligent Data Processing Based on Multi-Dimensional Numbered Memory Structures* In: Diagnostic Test Approaches to Machine Learning and Commonsense Reasoning Systems, (IGI Global) 156-184