A MOBILE BROWSER PROTOTYPE FOR SEMANTIC INFORMATION SYSTEMS

Gergő GOMBOS*, Tamás MATUSZKA**, Balázs PINCZEL*, Gábor RÁCZ*, Attila KISS*, Tamás GAIZER***
*Department of Information Systems, Faculty of Informatics, Eötvös Loránd University, Pázmány Péter sétány 1/C, 1117 Budapest, Hungary, e-mail: ggombos, tomintt, vic, gabee33, kiss@inf.elte.hu
**Inter-University Centre for Telecommunications and Informatics, Debrecen, Hungary
***Régens Zrt, Budapest, Hungary, e-mail: gaizer.tamas@regens.hu

ABSTRACT

Nowadays, companies have to handle large amounts of data every day (e.g. employee data, reports, statements). These data are usually unstructured or semi-structured; they do not have a consistent format, since they come from various sources. As a solution, the Semantic Web provides a common data format, which helps integrate the heterogeneous data. In the industry, in order to be efficient and to remain competitive, it is crucial to have quick, easy, and reliable access to your data from anywhere. In this paper, we introduce a client-server system, which enables browsing of semantic data with smartphones and tablets running Android operating system. The users can access their data using predefined views. These views determine which parts of the data a user can see, and how to present them in the client application. Each user can have different access privileges to different views, so each employee can have a level of access suitable to his status.

Keywords: *Semantic Web, Ontology, Mobile Application, ERP System*

1. INTRODUCTION

During their operation, the companies have to manage a large number of documents, which are usually plain texts, CVs, project reports, websites, e-mails, feeds, blog posts, geographic data, etc. In addition, we can access more and more public knowledge bases on the Internet, for example, encyclopedias, books, dictionaries, databases in field of government, touristic, multimedia content and social networks. Furthermore, there are numerous databases which contain theoretical and experimental results of various scientific experiments in the field of computer science, biology, chemistry, etc. There is a quite complex collection of these kinds of data maintained by the Linked Data Community [3]. This collection contains datasets and ontologies which are at least 1000 lines in length, and which contain links to each other. We could get relevant answers from some databases with the help of search engines using well-defined questions. However, from the perspective of companies, it means much more potential advantages if they can integrate their structured and unstructured data, and they are capable to extract the information from textual contents. The Semantic Web [2] provides a standardized data model and a corresponding query language for this purpose. With these technologies, it is possible to integrate and query data which come from various sources in a standard way. Because of these features, the industrial interest is increasing more and more.

In this paper, we present a system we designed and implemented, which is able to access, filter and display semantic datasets. The system uses a client-server architecture, where the clients can be smartphones, tablets and other devices which are running Android operating system. This is an important feature for the companies, because it is crucial to have quick, easy, and reliable access to their data from anywhere in order to be efficient and to remain competitive. Furthermore, the system is capable of managing multiple users and controlling which parts of the data are accessible for a given user, and how to format them.

The structure of the paper is as follows. In Section 2, we outline the preliminary definitions. Thereafter, we present the architecture of the system in Section 3. In Section 4, we describe the main functions of the system in details. Then, we compare our solution with some similar applications in Section 5. After that, we give two industrial-like examples in Section 6 to demonstrate the usability of our application. Finally, we summarize our experiences with the system in Section 7.

2. PRELIMINARIES

As we mentioned in the introduction, the Semantic Web [2] provides various techniques to manage the data available on the Internet. This section gives insight into the basic concepts of Semantic Web that are necessary for understanding what our system is capable of and how it works. The main technologies that are used in our system are the following: Resource Description Framework (RDF), RDF Schema (RDFS), SPARQL query language, Web Ontology Language (OWL). In the formal discussion we follow the concepts and notations introduced in [11].

The Resource Description Framework is a description language, where the information is represented by RDF triples. Informally an RDF triple consists of a subject, a predicate, and an object; or alternatively it consists of an entity, a property, and the value of that property of the described entity. This representation form is similar to natural language sentences. For example the sentence *'Eötvös Loránd University is located in Budapest.'* can be translated into the triple *(Eötvös Loránd University, location, Budapest)*. Three kinds of terms are distinguished: IRIs represent entities (e.g. *http://dbpedia.org/resource/ELTE*) or relations (e.g. *http://dbpedia.org/ontology/location*); literals can only occur as value of a property; blank nodes are the terms that do not represent real world entities, they just help to construct complex values, for example, mail addresses which consist of multiple parts such as postal code, city, street and number. Below is the formal definition of RDF triples (Definition 2.1).

Definition 2.1. *Let I, B, and L (IRIs, Blank Nodes, Literals) be pairwise disjoint sets. An* RDF triple *is a* $(v_1, v_2, v_3) \in (I \cup B) \times I \times (I \cup B \cup L)$*, where* v_1 *is the subject,* v_2 *is the predicate and* v_3 *is the object. A finite set of RDF triples is called an* RDF graph *or* RDF dataset.

The RDF Schema is a data-modeling vocabulary built on the top of RDF for defining concepts, properties and constraints which are essential for organizing the knowledge represented by triples. The Web Ontology Language also enables us to define concept and property hierarchies, however, it is a computational logic-based language. Therefore logical constraints and rules can be expressed in order to verify the consistency of that knowledge or to make implicit knowledge explicit. The formal definition of an ontology is presented in Definition 2.2, based on [17].

Definition 2.2. *A structure* $\mathcal{O} := (C, \leq_C, P, \sigma)$ *is an* ontology*, where C and P are two disjoint sets. The elements of C and P are called* classes *and* properties*, respectively. A partial order* $\leq_C$ *on C is called class hierarchy and a function* $\sigma : P \to C \times C$ *is a signature of a property. For a property* $p \in P$*, its domain and its range can be defined in the following:* $dom(p) := \pi_1(\sigma(p))$ *and* $range(p) := \pi_2(\sigma(p))$*, where* π *is the projection operation. Let* $c_1, c_2 \in C$ *be two classes; if* $c_1 \leq_C c_2$*, then* c_1 *is a subclass of* c_2 *and* c_2 *is a superclass of* c_1*.*

SPARQL is a query language for retrieving and manipulating RDF data. It is an SQL-like declarative language; the queries are based on pattern matching, where the patterns are in the form of triples, though they can contain variables as well. Most of the keywords and their meanings are the same, such as *SELECT*, *WHERE*, *LIMIT*. However, there are some new keywords in SPARQL, for example, *OPTIONAL* means optional pattern matching, or *FILTER* that defines constraints for the variables. Definition 2.3 gives the abstract syntax of the filter conditions and Definition 2.4 presents the abstract syntax of the SPARQL expressions.

Definition 2.3. *Let V be the set of distinct variables over* $(I \cup B \cup L)$*. The variables are distinguished by a question mark. Let* $?X, ?Y \in V$ *be variables and* $c, d \in (L \cup I)$ *be a literal and an IRI constant, respectively. We define the filter conditions recursively as follows. The* $?X = c$*,* $?X = ?Y$*,* $c = d$*,* $bound(?X)$*,* $isIRI(?X)$*,* $isLiteral(?X)$*, and* $isBlank(?X)$ *are atomic filter conditions. Thereafter, if* R_1, R_2 *are filter conditions, then* $\neg R_1$*,* $R_1 \wedge R_2$ *and* $R_1 \vee R_2$ *are filter conditions as well.*

Definition 2.4. *A SPARQL expression is built up recursively in the following way:*

1. *the triple* $t \in (I \cup V) \times (I \cup V) \times (L \cup I \cup V)$ *is a SPARQL expression,*
2. *if* Q_1, Q_2 *are SPARQL expressions, and R is a filter condition,*
   then Q_1 FILTER R*,* Q_1 UNION Q_2*,* Q_1 OPT Q_2*, and* Q_1 AND Q_2 *are SPARQL expressions as well.*

The discussion of formal semantics of SPARQL is out of the scope of this paper. Set and multiset semantics are described in [11].

3. SYSTEM ARCHITECTURE

As it can be seen on Figure 1, the application uses client-server architecture. All resource-intensive operations are carried out on the server, thus clients of any quality can connect to it, even ones with limited CPU or memory resources, such as smartphones or tablets. The clients only request answers to queries, and display the results. A client can be any devices which are running Android operation system. Such devices are always on hand and they usually have internet connection. The server consists of a middleware system and a local database.

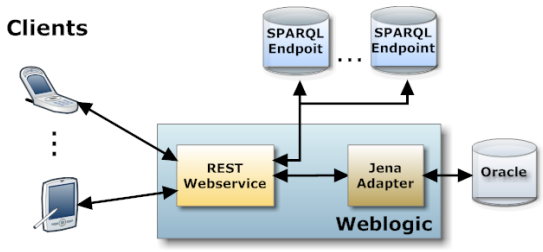

Fig. 1 The architecture of our system

On server-side there is a REST web service which communicates with clients, handles the configuration files, transforms the queries according to the requests, and forwards them to local or remote endpoints. Furthermore, the server uses an Oracle database with Jena Adapter API to store the local models. It is capable the handle large datasets. The results of queries are sent in form XML back to clients. Query results are stored in the local database, and sent to the clients in small parts, which they can handle.

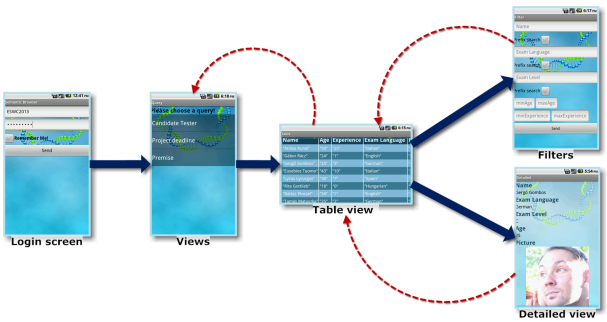

Fig. 2 Screens in the client application, and transitions between them

A typical scenario is the following. The first step is authentication, which is done with a username and a password. Next, the middleware loads the user's configuration file, which contains the views which are accessible by him, so that he can decide which table view he would like to see. The middleware sends the table view's query to either a local or a remote endpoint, depending on where the data for the view is located. The query results are forwarded to the client dynamically. That means only the first few rows are sent, and whenever the user gets close to the bottom of the table while browsing it, more rows are sent to the client.

This way we can prevent the undesired situation, where the client application would hang until a large table loads. If the user filter the table, then he can open the filter dialog and construct the filter conditions using the dynamically built filter form. The desired filter conditions are incorporated to the table view's query, and the middleware resends this modified query to the endpoint, and the results are processed in a similar way. Note that filtering does not require knowledge of the SPARQL query language [14]; the filter options can be selected using a graphical interface, and they are translated to SPARQL by the middleware. If the user selects an object from the table, then the appropriate detailed view (which is also defined in the configuration file) appears. To do this, the middleware needs to send another modified query, but this time the result is small enough to be sent to the client in one step.

4. SYSTEM FUNCTIONALITY

In this section we describe the main functions of the system, both the client and the server-side components. The details of semantic data querying, user account handling, views, filters as well as query rewriting are presented.

4.1. Querying semantic data

One of the many advantages of semantic technology is that we can access data from diverse sources using a unified data model (RDF triples [5]). Furthermore, it provides the SPARQL language, which can be used to query semantic data. These techniques make it possible to create an application which can integrate various data sources under a common user interface. Data sources can be local [9], or one of the numerous publicly available sources, such as DBpedia [4], Geonames [16], etc.

Because the clients of the system can be mobile devices with limited resources (such as memory, or battery life), it is important to preprocess the results of the queries. It means that if the query results are too large, they must be split into smaller fragments, and we have to store these fragments on a server and send them to the clients only if it is necessary.

4.2. Handling user accounts

In industrial environment, it is essential to handle multiple users and to be able to control which parts of the data can be seen by the individual users. Our system authenticates each client with a username and a password. After confirming the identity of a user, the appropriate configuration file will be loaded. A configuration file defines the views that describe the way how a user can see the datasets (see below). Moreover, if a query result has been split into pieces because of its size as we mentioned above, then the fragments are stored separately for each query of each user.

4.3. Views

One of the main features of our system besides data integration is the management of views. Each user has a configuration file that is located on the server-side. These files created based on some general schemes then an administrator can be modifies them. The configuration file defines the data accessible for the user, and the way it is to be presented. There are two types of views: the table and the detailed view.

The table view is suitable to define a table of objects of a given category or class, along with their most important attributes. The filter window is available from this type of view (for further details, see the next section). If an object is selected, the appropriate detailed view will be loaded. The detailed view screen is dynamically built up as the filter screen based on the attribute types. In this view, additional attributes are displayed. Moreover, if one of the attributes describe the location of an image or PDF document, that document could be displayed as well. The semantic representation allows multiple values to an attribute; in the detailed view these values will be represented using lists. Figure 3 illustrates a table view with the corresponding detailed view.

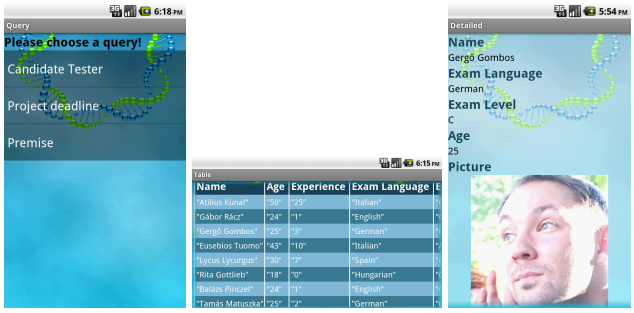

Fig. 3 View selection screen, with examples for table and detailed views

4.4. Filters

From table view screens, a special tab is available, where you can define filter conditions for the main attributes. To do this, the data type of each attribute must be specified in the configuration file. These data types can be one of the followings:

- *Text*: in this case there will be a text area in the filter window. The value of the attribute must contain the given text.
- *Enum*: possible values will be converted to a drop-down list. The possible values are collected automatically by an auxiliary query.
- *Decimal*: you can define an interval with its lower and upper bounds. If one end of the interval is not specified, that means no bounds.
- *Date*: these are filtered using intervals as well. In this case, bounds can be entered using a date selection dialog.
- *Boolean*: determines whether the object has a particular property (e.g. a picture attached to it).

After filtering the table, only those items remain which match the criteria. In Figure 4, you can see two examples for filter windows. The first one illustrate a date selection screen while the second one shows a complex use case with some attributes related to a person, such as name, age or experience level.

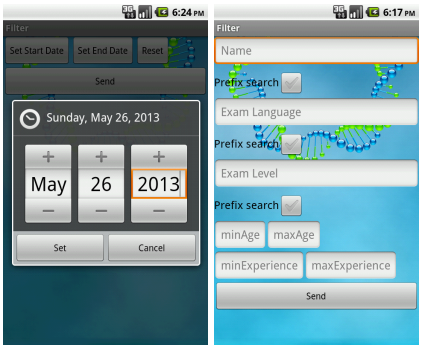

Fig. 4 Filter examples

4.5. Query rewriting

For each table view, its declaration consists of an initial SPARQL query, and additional attribute definitions. That query selects the displayable entities and additional attributes which determine the columns of the table on client side. Every attribute has a URI tag, a name and a type. The system can build the final query from these parameters in the following way:

The names of additional attributes are added to the query as variables after the SELECT keyword. The WHERE clause will contain the corresponding triples based on the attribute declarations, in the form (?entity <URI><name of attribute>) in OPTIONAL groups. In case of filtering, the appropriate groups will contain additional triplets based on the filter conditions. The Figure 5 illustrates the process. On left-hand side, there is a fragment from the configuration file with a table view definition named 'Candidates'. The definition contains the SPARQL query and an attribute description belongs to column named 'name'. When the client application has to build up the table view that initial query will be transformed as the right-hand side box shows it.

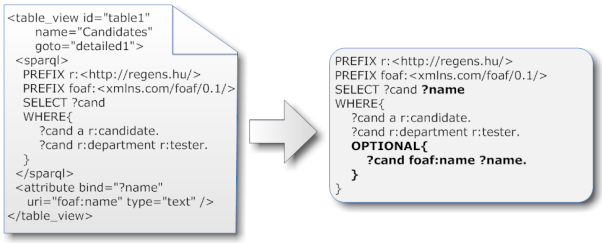

Fig. 5 Building a SPARQL query

Detailed views use the initial query of the corresponding table view, but they define some additional attributes to display in the same way than we do it in case of table views, with an <attribute>tag. In case of detailed views, the SPARQL query will be contain an extra condition to filter data related to the previously chosen entity.

5. RELATED WORK

There are some application which have already used the advantages of semantic technologies. In this section we compare these applications with our system. DBpedia Mobile [1] is a location-sensitive semantic application, which enables its users to browse linked data on their mobile devices by navigating a map. The objects on the map are displayed using icons based on the YAGO categories [10]. By clicking on these icons, we can access DBpedia descriptions and Flickr [8] photos for the selected object. This application shares many common features with our solution: the user can use filters for a search, and the results can be browsed using a detailed view, which displays information about the object using multiple data sources. However, there are differences as well, which stem from the fact that DBpedia Mobile is an application designed exclusively for one particular data set, while our browser is general-purpose, and supports arbitrary data sets. The most significant difference is that our main browsing interface is table-based, not map-based. Moreover, in our application the filter conditions cover every attribute of the data set: for each view, a filter form is generated automatically, where specific filter conditions can be set for each column. In DBPedia Mobile, however, only a limited set of filter conditions are available on the form; to achieve more complex filtering, the users should use the SPARQL query language.

OntoWiki Mobile [7] is an application based on the semantic collaboration platform called OntoWiki. It allows its users to browse, navigate, create and edit semantic data on-the-fly. It focuses on offline editing: users can alter the knowledge base without a data connection, and their modifications will take effect later, when a connection is present.

The idea of displaying RDF data with views comes from Fresnel [13]. It defines a vocabulary, which allows the users to describe the display properties of their data. The vocabulary uses two basic concepts: lenses and formats. Lenses define the parts of the data which the users would like to display, and formats define how the selected data should appear. Selectors for the lenses can be defined using simple restrictions (e.g. filtering objects by their type), but more complex filters can be expressed with the Fresnel Selector Language (FSL) or the SPARQL query language. A visual browser which implements Fresnel is (for example) IsaViz [12]. In our solution we have a different approach: to define views, the user only needs to specify the parts of the data to be displayed, as a SPARQL query. Formatting is done automatically based on the data types of the selected attributes.

6. EXPERIMENTS

As we mentioned previously, the industry's interest towards semantic technologies is growing continuously, thanks to the capability to integrate and query data originating from different sources. In this section, we present two examples, where we can use our application. The first example helps the managers in a company, while the second example illustrates the perspective of truck drivers in a transport company. Since our browser was made for general purposes, the range of use is not limited to these two tasks, but with these two different uses we would like to demonstrate the diversity of our application.

6.1. HR

At large companies, interviewing is typically made by managers. Those who also deal with many other tasks do not sit permanently in the office, so they cannot be informed about the candidates in time. The application we made can be used for a possible industrial use of a company's HR and project management support. Our application allows managers to query the necessary information provided by HR assistants, anytime, anywhere. To help this task, various ontologies have been developed for semantic storage of the data of the employees, trainees, candidates or projects [6] [15].

The views presented above allow for a variety of authorized users to view data at different levels. Returning to the previous example, we can store more data about an applicant, than it is absolutely necessary for the manager. Thus, the data needed for the interview (such as experience and knowledge), can be filtered beforehand, with the help of the configuration file. We can see the process in Figure 2. At a larger company, there are several managers working, each with its own area of responsibility. For example, in the software development department, such areas are designing or testing. It is clear that different areas require different competencies, so we can specify initial filters in the configuration file for these, as well. Figure 3 illustrates the selection of a candidate for the tester position.

6.2. Transportation

Another example shows where our application can be used in transportation. The drivers receive their waybill, the cargo on their truck, then they set off to the destination. The person in office decides what and where to take. In order for the drivers to know where they are to travel, they must ask it from the office worker. It can generate a lot of calls which can be very expensive and make a 24-hour dispatch service necessary. Our system can also be a solution to this problem, because if the person in the office inserted transportation data in a dataset, the drivers would be able to check it anytime on their phones. They could see delivered and the upcoming transports too. Obviously the current transport can be found with him, so these are not relevant. Before they arrive to a certain destination, they could check where to go next, so they could organize their next day. The application exploits the available public datasets as well. Big companies have subsidiaries all over the world. The geographical information of the premises of these subsidiaries can be obtained from DBpedia or GeoNames datasets.

7. CONCLUSION

In our paper, we presented an application, which provides opportunity to make use of semantic technologies in the industry. Our main result is a system that makes it possible to browse semantic data sources in personalized way. The system is simple to configure, yet it is general enough to handle arbitrary semantic datasets. Thanks to its client-server architecture, the resource-intensive operations can be carried out by the server, thus mobile clients with limited resources can use the application as well. Moreover, the client application hides the underlying technological details from the user: they do not have to formulate SPARQL queries, because filtering can be done with the help of easy-to-use forms.

We demonstrated the usability of the system with two enterprise use cases. In the first case, the application helps managers in finding the right applicant for a job. The HR department can record the applicant's data, and only the corresponding manager can access it using a view. In the second use case, truck drivers of a transport company can see their next freight using their mobile phone, or tablet. The application can provide solutions for similar cases, with a suitable configuration file.

In the future, we would like to extend our system to be able to not just only display the data but edit them as well. We also aim that data types of attributes could be extracted from a corresponding ontology. It could simplify the creation and maintenance of configuration files.

ACKNOWLEDGEMENT

This work was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013) and TAMOP-4.2.2.C-11/1/KONV-2012-0001 supported by the European Union, co-financed by the European Social Fund.

REFERENCES

[1] BECKER, C. – BIZER, C.: DBpedia Mobile: A Location-Enabled Linked Data Browser, World 369, No. 1 (pp. 6-7). (2008)

[2] BERNERS-LEE, T. – HENDLER, J. – LASSILA, O.: The semantic web, Scientific American 284, No. 5 (pp. 28-37). (2001)

[3] BIZER, C. – JENTZSCH, A. – CYGANIAK, R.: State of the LOD Cloud (2011). http://www4.wiwiss.fu-berlin.de/lodcloud/state/

[4] BIZER, C. – LEHMANN, J. – KOBILAROV, G. – AUER, S. – BECKER, C. – CYGANIAK, R. – HELLMANN, S.: DBpedia-A crystallization point for the

Web of Data, Web Semantics: Science, Services and Agents on the World Wide Web 7, No. 3 (pp.154-165). (2009)

[5] BRICKLEY, D. – GUHA, R. V.: RDF vocabulary description language 1.0: RDF schema (2004)

[6] DORN, J. – NAZ, T. – PICHLMAIR, M.: Ontology development for human resource management. In: 4th International Conference on Knowledge Management (pp. 109-120). (2007)

[7] ERMILOV, T. – HEINO, N. – TRAMP, S. – AUER, S.: Ontowiki mobile – knowledge management in your pocket. In: The Semantic Web: Research and Applications (pp. 185-199). Springer Berlin Heidelberg. (2011)

[8] FLICKR `http://www.flickr.com`

[9] GOMBOS, G. – MATUSZKA, T. – PINCZEL B. – RÁCZ G. – KISS, A.: VOSD: A General-Purpose Virtual Observatory over Semantic Databases. In: 13th Symposium on Programming Languages and Software Tools (SPLST 2013) (pp. 90-99). (2013)

[10] HOFFART, J. – SUCHANEK, F. M. – BERBERICH, K. – LEWIS-KELHAM, E. – DE MELO, G. – WEIKUM, G.: Yago2: exploring and querying world knowledge in time, space, context, and many languages. In Proceedings of the 20th international conference companion on World wide web (pp. 229-232). ACM. (2011)

[11] PÉREZ, J. – ARENAS, M. – GUTIERREZ, C.: Semantics and Complexity of SPARQL. In The Semantic Web-ISWC 2006 (pp. 30-43). Springer Berlin Heidelberg (2006)

[12] PIETRIGA, E.: Isaviz: a visual environment for browsing and authoring rdf models. In: Eleventh International World Wide Web Conference Developers Day. (2002)

[13] PIETRIGA, E. – BIZER, C. – KARGER, D. – LEE, R.: Fresnel: A browser-independent presentation vocabulary for RDF. In The semantic web-ISWC 2006 (pp. 158-171). Springer Berlin Heidelberg. (2006)

[14] PRUDHOMMEAUX, E. – SEABORNE, A.: SPARQL query language for RDF. W3C recommendation, 15. (2008)

[15] SZEKELY, A.: An Approach to Ontology Development in Human Resources Management. In Proceedings of the 5th International Conference on Virtual Learning (pp. 153-159) (2010)

[16] VATANT, B. – WICK, M.: Geonames ontology (2012)

[17] VOLZ, R. – KLEB, J. – MUELLER, W.: Towards Ontology-based Disambiguation of Geographical Identifiers. In Proceedings of WWW2007. (2007)

Received December 2, 2013, accepted December 22, 2013

BIOGRAPHIES

Gergő Gombos was born on 14.12.1986. In 2012 he graduated from the Department of Information Systems of the Faculty of Informatics at Eötvös Loránd University in Budapest. His master thesis was about the webservice for mobile application that use the semantic web. Since 2012 he is a PhD student at the same Department. His scientific research is focusing on semantic web, cloud and distibuted computing, webservices and federated queries. He is also interested in collecting the social networks and big data.

Tamás Matuszka was born on 25.11.1987. In 2012 he graduated (MSc) with distinction at the department of Information Systems of the Faculty of Eötvös Loránd University in Budapest. Since 2012 he is PhD student at the Department of Information Systems. His scientific research is focusing on Augmented Reality supported by Semantic Web. In addition, he also investigates questions related with the mobile Semantic Web and Augmented Reality.

Balázs Pinczel was born on 23.11.1988. In 2012 he graduated (MSc) with distinction at the department of Information Systems of the Faculty of Eötvös Loránd University in Budapest. Since 2012 he is a PhD student at the Department of Information Systems. His scientific research is primarily focusing on improving the efficiency of Semantic Web technologies (e.g. SPARQL query evaluation). In addition, he is also interested in the area of NoSQL databases and Big Data.

Gábor Rácz was born on 27.02.1989. In 2012 he graduated from the Department of Information Systems of the Faculty of Informatics at Eötvös Loránd University in Budapest. His master thesis was about the visual query languages and the SPARQL query language. Since 2012 he is a PhD student at the same Department. His scientific research is focusing on semantic web and data mining. In addition, he is also interested in natural language processing and social networks.

Attila Kiss was born in 1960. In 1985 he graduated (MSc) as mathematician at Eötvös Loránd University, in Budapest. He defended his PhD in the field of database theory in 1991; his thesis title was Dependencies of Relational Databases. Since 2010 he is working as the head of Information Systems Department at Etvs Lornd University. His scientific research is focusing on database theory and practice, semantic web, big data, graph databases, data mining. In addition, he also investigates questions related with social network analysis.

Tamás Gaizer was born on 1964. In 1986 he graduated (MSc) with distinction at the University of Szeged. Since 1995 he is working as a senior IT consultant at Regens Zrt. His scientific research is focusing on analysis, design and implementation of systems. In addition, he also investigates questions related with business process analysis and modeling.