Ontology-based Virtual Query System for the SemanticLIFE Digital Memory Project: Concepts, Designs and Implementation

Hanh Huu Hoang, A Min Tjoa, and Tho Manh Nguyen

***Abstract*— Ever increasing capacities of contemporary storage devices inspire the vision to accumulate (personal) information without the need to delete old data over a long time-span. Hence the target of the SemanticLIFE project is to create a personal information management system for a human lifetime data. One of the most important characteristics of the system is its dedication to retrieve information and data in a very efficient way by adopting user demands regarding the reduction of ambiguities aiming at a user-friendly and yet powerful enough system with a satisfactory query performance. In this paper, we introduce the concepts and a design of the query system of SemanticLIFE, the Virtual Query System, which uses emerging Semantic Web technologies to fulfill necessary requirements.**

***Index Terms*—Digital memories, Information retrieval, Intelligent systems, Ontology-based query, Virtual query.**

I. Introduction

TOWARDS the goals of Personal information storage and retrieval of all one's media throughout a lifetime, researchers consider continuous archival and retrieval of all media relating to personal experiences. People are capturing and storing an ever-increasing amount of information about them, including emails, contacts, appointments, web browsing histories, pictures, documents, recordings, phone calls, chat session and so on. The challenged issues are how to extract useful knowledge from this rich library of information; how to use this knowledge effectively, and how to effectively present memories and knowledge to different kinds of users [8].

Manuscript received October 16, 2005. This work has been generously supported by ASEA-UNINET and the Austrian National Bank within the framework of the project "Anwendung von Semantic-Web-Konzepten für betriebliche Informationsmanagementsysteme" (Application of Semantic-Web-Concepts for Business Intelligence Information Systems - Project No. 11284).

H. H. Hoang is a PhD student at the Institute of Software Technology and Interactive System, Vienna University of Technology, Favoritenstrasse 9-11/188, A-1040 Vienna, Austria (phone: +43 (1) 58801-18861; fax: +43 (1) 58801-18899; e-mail: hanh@ifs.tuwien.ac.at).

A. M. Tjoa, is full professor, head of the Institute of Software Technology and Interactive System, Vienna University of Technology, Favoritenstrasse 9-11/188, A-1040 Vienna, Austria (e-mail: tjoa@ifs.tuwien.ac.at).

T. M. Nguyen is a Post-Doctoral researcher at the Institute of Software Technology and Interactive System, Vienna University of Technology, Favoritenstrasse 9-11/188, A-1040 Vienna, Austria (e-mail: tho@ifs.tuwien.ac.at).

The SemanticLIFE project [1] is an effort to come a step closer to Vanevar Bush's vision of Memex [3] by providing a general semantic Personal Information Management (PIM) system. As proposed by Tim Berners-Lee the realization of the Semantic Web would narrow the gap towards the implementation of Memex-like systems by the use of domain specific ontologies and their reuse [2].

The SemanticLIFE system deals with a large range of data sources including communication and personal data as mentioned above, and may include a whole range of data from additional sources such as temperature, geographic location, medical records. And then it integrates this wide variety of data sources and stores them in an ontology-based repository.

The SemanticLIFE user is supported in issuing imprecise queries to retrieve the rich semantic information from his/her historical personal data. For example, consider scientists, who work in a specific domain. They might be interested to get into contact with other researchers in the scientific community that (1) share the same interests or have similar problems (2) are publishing in similar conferences and (3) were recently active in the specific field of research (4) and speak a common language. The result of such a query could be the web pages and email addresses of the researchers coming into question.

Most often when developing similar PIM systems, people focus on back-end issues, i.e. capturing as much data sources as possible, and then integrate and store them in huge repositories. For this purpose it is necessary to map the ontologies of the various data sources into a common ontology of the system. Users are confronted with the lack of knowledge concerning the stored information inside the system, and they would formulate ambiguous requests, so that many barriers have to be overcome before the system could deliver the demanded results.

This paper is aiming at a description of the innovative features of our "Virtual Query System" design, its principles and implementation solutions. This query system is based on a front-end approach allowing the user to retrieve information from huge ontology-based repositories in an efficient way. The conception of this query system which is primarily based on the reduction of semantic ambiguities of user query specifications at the very early stage of the retrieving process. This approach integrates many research efforts from the area of Semantic Web, query refinement, semantic query caching

for RDF data, inference, ontology mapping, and user interaction.

The remainder of this paper is organized as follows: a range of projects is currently addressing issues similar to the SemanticLIFE project are briefly presented in section II, section III presents a overview of the SemanticLIFE system and outlines the functions of its components. Section IV describes the relevant query issues of the SemanticLIFE system and presents the overall query architecture. Details of the Virtual Query System design are pointed out in sections V and VI. Section VII presents our ongoing implementation results. The paper is concluded with a sketch of the intended future work.

II. Related Work

In domain of digital memories systems, in the mean time numerous projects are available. Basically, the efforts fall into the major category: Personal Information Management.

As far as PIM systems are concerned, a few tools based upon the Semantic Web technology are available. To give some examples: *MyLifeBits* is a lifetime store of everything [4]. It captures a lifetime's worth of articles, books, cards, CDs, letters, memos, papers, photos, pictures, presentations, home movies, videotaped lectures, and voice recordings and stored them digitally. So far, however, it does not focus on semantically structuring or exploiting the resulting pool of data beyond mere retrieval.

Haystack is an information management system is built in the Eclipse plugin environment [5]. It is a platform for creating, organizing and visualizing information using ontologies. The primary goal is to provide high quality retrieval. RDF based data models enable the user to define new object attributes that help them categorizing and retrieving information, and creating new relationships between objects.

HP Labs provides a framework called *e-Person*, a personal information infrastructure that enables groups of users to organize and share information [17]. A basic principle of the e-Person infrastructure design is to use the RDF language to represent all stored information.

Some other tools like *LifeStreams* [6], *Edutella* [7] and *Semagix Freedom* [26] cover certain aspects of an individual's life but not all of it or not in a complete semantic manner. In these systems a limited support for features like automatic feeding from different data sources, metadata extraction from information chunks, manual annotation and more importantly processing of Life Events is noticed.

Current Information Retrieval systems, PIMs and especially data integration systems in the Semantic Web area mostly pay attention on a back-end solution approach: trying to gather data from different data sources, integrate them by means of a central meta-information repository, or unified ontology. High-profiled examples herefore are the INDUS Framework [16] and the Unicorn Workbench [27].

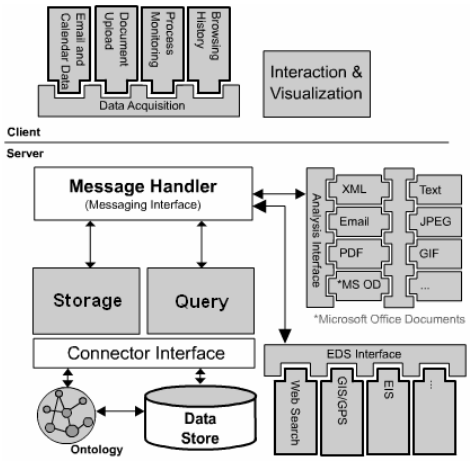

Fig. 1. Architecture overview of the SemanticLIFE system.

III. The SemanticLIFE System

A. Brief Overview

The SemanticLIFE system is based on a highly modular architecture. It relies on plug-in mechanisms in order to guarantee flexibility and extensibility. Communication within the system is based on a message-oriented design with the advantage of its loosely coupled characteristics [1].

An overview of the system architecture is depicted in Fig. 1. Data with user annotation is fed into the system using a number of dedicated data acquisition modules. The data objects are passed on by the message handler to the analysis module. This module contains a number of specific analysis plug-ins providing semantic mark-up by applying a bunch of feature extraction methods and indexing techniques in a cascaded manner. The semi-structured and semantically enriched information objects are forwarded to the storage module for an ontologically structured storage. In the SemanticLIFE system, data sources are stored in forms of RDF triples with their ontologies and metadata. This repository is called a *metastore*.

A set of query processing and information visualization tools provides the means for information exploration and report generation.

The analysis module and metadata extraction capabilities make associations among the lifetime items/objects and as well as lifetime events based on user annotation, user profile and the system ontologies. This makes the system data is easily and effectively traceable. As the process of loading new data sources is time consuming if the system has a tight coupling with its components, a light weight messaging based solution forms a very promising alternative where new modules can easily be plugged into the system.

B. State-of-the-art Implementation

SemanticLIFE still is work in progress. The data acquisition modules monitor and feed to the system data about emails, outlook data (calendar, contacts, tasks and notes), web browsing sessions, programs run on the system, files in the file

system. In the future we will focus also on other data sources. We now turn towards many of the more interesting issues involved in exploiting the large amounts of data we have gathered. Among the tasks we have done: storing mentioned data feeds in a *metastore* using an effective RDF and ontology storage with an effective index strategy; a query mechanism to parse user requests into ontology-based RDF queries and retrieve the data from the metastore. We also have some positive results on user annotation, analysis for object associations detecting and visualization module.

IV. Information Querying in SemanticLIFE System

For SemanticLIFE metastore, it is uneasy to define highly structured queries (e.g., in a language like SQL of relarional or object-oriented databases) when a multitude of information systems are addressed or the information is only semi-structured. Hence the system must be capable to support user-queries which are formulated with an "imprecise search" terminology by automatically transforming them into more specific queries.

If the data is already stored in a semantically enriched manner (RDF and ontology), it will be possible to provide more powerful imprecise searches, that go far beyond "simple" full-text indices. Here, the term "imprecise" has two meanings: firstly, the query processing system has to satisfy imprecisely defined user information needs. Secondly, the target of the user-query is well specified but there are ambiguities in processing the query because of the heterogeneity of the different sources of data. Therefore, the system solves these problems during query generation, by exploring the system database and ontology repository and then generate the specific queries. A part of the query module uses system metadata and global ontology to provide the user a better understanding of the system's data inside. The query module helps the user creating the clear requests, takes a further needed analysis and executes them in an effective manner.

The next is query post-processing stage: The initial query results require processes of analysis, ranking and aggregation to derive more precise results matched to user's requests and preferences. Analysis and ranking would rely on specific calculations in terms of semantic distances and reasoning steps to give more concrete and accurate results. To perform these tasks, the system refers to a set of rules and user preferences.

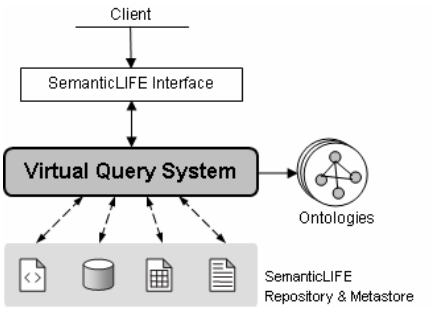

Fig. 2. The overview of information querying in SemanticLIFE through the Virtual Query System.

After analysis and ranking of the query results, the system aggregates the results. The results can be presented to the user through presentation layer, that prepares the search result.

The querying process in SemanticLIFE depicted in Fig. 2 is supported by our Virtual Query System. This mediator system is not only capable to deal with the discussed issues above but also reduces the imprecision of user requests by offering the user an overview of the relevant stored information. By this way, the user will significantly specify more precise queries on information and data stored in the system.

V. The Virtual Query System

A. Goals

Formulating non-ambiguous queries is often a too demanding task to users as they do not have the overview on the semantics of data stored in the system. The goal of the Virtual Query System (VQS) is to overcome this problem by providing an ontology-based virtual information view of the data available in the system. If the user can "understand" what is inside of the system he/she can very clearly specify queries on the "real" data stored in the system repositories. In addition, based on a common ontology mapped from local ontologies of the data sources and the internal analysis (inference, detecting ambiguity and fuzziness of user queries and so on), the VQS refines the user's queries and creates "real" sub queries against data sources in the metastore. Furthermore, relied on user's experiences, reflected in user ontology or user profile, the VQS recommends the more related results to the user.

B. Design

The Virtual Query System consists of six modules/parts to deal with the challenging task of a complete Semantic Web query system. Details of the VQS design are depicted in the Fig. 3 and its modules design descriptions are mentioned as following.

1) Meta-data sources

This important part of the VQS is the module containing the Meta-data sources. This module acts as a "virtual" data source to be offered to the user. It enables the user to 'understand' the meaning of the data sources stored and to specify more precise queries. The VQS collects metadata from data sources in the metadata repository of the SemanticLIFE system. An analysis process and statistical computation are carried out on these metadata sources to get the semantic information. Then the processed information is stored in this module and will be delivered. The features of VQS are very similar to those of a recommender system. Furthermore, this part is also referred as an "image" of the system database in further query processing.

2) The Ontology Repository

The second part to be described is the ontology repository which builds up the core of the VQS. The ontology repository contains the ontologies used in the VQS system such as the Global ontology or inference ontology. Following [8] an

ontology-driven approach to data integration relies on the alignment of the concepts of a global ontology that describe the domain, with the concepts of the ontologies that describe the data in the local databases. Once the alignment between the global ontology and each of the local ontologies is established, users can potentially query hundreds of databases using a single query that hides the underlying heterogeneities.

3) Sub-Queries Formulation

Sub-queries formulation is another essential part of the VQS. From the original virtual query of the user, this part parses it into the sub queries ($Q_{i \mid i=1...n}$ in the Fig. 3) based on the global ontology for specific data sources. This module does not only transform the virtual query to sub-queries for specific data sources but additionally perform inference on the user's request in order to create more possible sub-queries afterward basing on the global ontology as well. After this process, the 'real' queries for each of the data sources will be generated based on a specific RDF query langue

4) The VQS Services

The VQS services consist of an ontology mapping engine, a semantic query caching mechanism, and an ontology-based query refinement.

a) Ontology Mapping

The reason why we need an ontology mapping mechanism instead of creating a new ontology is that mapping of existing ontologies is relatively easier to perform because of the smaller community involved in the process [14]. This is one of the main arguments for having an ontology mapping service in the VQS. This service is the mechanism to map local ontologies into a global one. This service deals with new data sources added with their respective ontologies, so that these ontologies are mapped or integrated to the global ontology.

b) Query Caching

This service improves the performance of the VQS by caching the queries in a period of time. We distinguish two kinds of caching mechanisms: query caching and result caching. The first addresses the process of generating sub-queries; and the second covers the caching of query results. Both caching types will use the semantic query caching methodology proposed in [11] and similarity-based query caching methods [17]. Furthermore, some statistical evaluation are used to take into account most frequent queries which ultimately leads to the generation of query templates to be offered as pre-defined queries in the query user interface.

c) Query Refinement

Query refinement is another important service for our ontology-based query processing. This is the interactive way for the VQS dealing with user's ambiguous queries, which is based on incrementally and interactively (step-by-step) tailoring a query to the current information needs of a user, whereas these needs are implicitly and on-line elicited by analyzing the user's behavior during the searching process

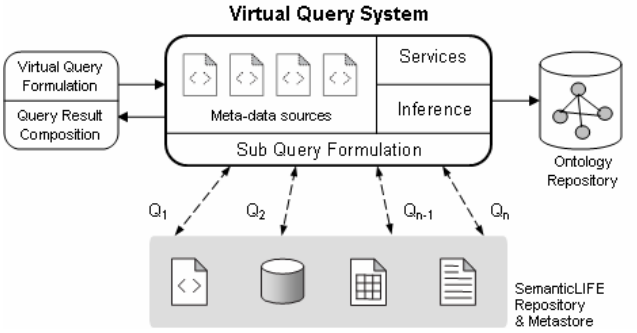

Fig. 3. The architecture an components details of the Virtual Query System. The Q_i (i = 1..n) are real queries used to query real system metastore

[10]. This query refinement service of the VQS is a semi-automated process: in the refinement process, the user is provided with a ranked list of refinements, which leads to a decrease of some of these ambiguities. In another hand, by exploiting the user's profile, the ontology background, and as well as user's annotation on data, this VQS service supports finding "similar" results that can help a user to satisfy his information need.

5) Ontology-based Inference

The inference service provides a basis for the deduction process on the relationships (rules) of concepts of the ontologies specified. Inference tasks are performed on the foundation of the ontologies and the data described by them. This service helps the system to analyze and evaluate the user's virtual queries in the process of generating sub-queries based-on the inference ontology. Inferencing is also helpful for other tasks of the VQS such as results aggregation.

6) The Virtual Query User Interface

a) Virtual Query Formulation

This graphical user interface delivers an overview of the system's data and its schema to the user and helps the user to specify virtual queries. A set of pre-defined queries, substantiated by query templates, is offered to the user. If these templates do not match the demands, the user can use a query-by-example tool alternatively to write the virtual queries.

b) Query Result Composition

Query result composition is the part which performs the integration and aggregation of the sub-query results and show to the user. It also can help the user navigating the system by asking new queries based on the results.

VI. WORKFLOW OF THE VIRTUAL QUERY SYSTEM

The workflow of the VQS is illustrated in the Fig. 4. The dashed arrows denote the internal activities; normal arrows are used for the interaction between the user and the VQS. The external query queue is used for receiving queries from the user and to return the results. Inside of the system, an internal query queue will be used for receiving processed queries of the

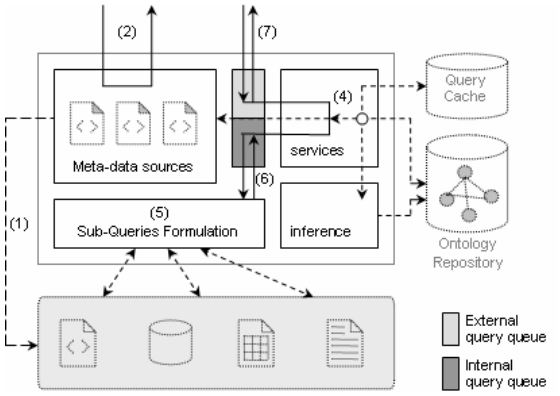

Fig. 4. The workflow of the Virtual Query System and Components communication inside. The numbered could be understood as sequenced actions of the system.

sub-queries and to deliver results.

The following steps are performed in a chronological order: At the very early stage when the VQS is installed into the whole system of SemanticLIFE, action (1) will be performed. This action collects metadata of the data sources from the metastore, performs necessary statistical computation and stores the results into the Meta-data sources. Parallel, the global ontology is mapped from local ontologies using the VQS ontology mapping service.

As also depicted in Fig. 2, the user directly retrieves the information by specifying his/her queries to the VQS with help of the query interface. The VQS gets user input through the virtual queries. Using this interface, the user can be confronted with intentional information of the metadata sources, so that we can avoid to a certain degree ambiguous requests (2).

In the next step, the formulated query, so-called virtual query, is sent to the VQS (3) and from this stage, the virtual query is evaluated on the foundation of the ontology-based services of the VQS (4). The VQS query caching service is now in charge of the subsequent processing of the virtual query: firstly, it looks up in to the *query cache*, if there is a match, the result is quickly returned to the user. In the second case if the virtual query does not match, an analysis and evaluation process will be undertaken (4) and the virtual query with the analyzed information will be sent to the Sub-queries formulation (5). As a result, sub-queries for each specific real data source will be generated.

Finally, query results will be passed to the user ((6) and (7)). They are aggregated in the Query Result Composition part before the delivery to him/her in a suitable form.

VII. The VQS Implementation Issues

In this section, we discuss the essential issues of the work in progress of our VQS implementation using the current Semantic Web technologies or frameworks.

The VQS is intently designed to acts as an independent plug-in framework. At first, it satisfies the SemanticLIFE software architecture. Later, it would be changed to become adaptable for any similar systems or operated as a separated framework or service. Among important things the VQS must get the user requests parameters in a correct manner. This is the task of the *virtual queries*. The virtual query is coded in XML in client-side modules or agents with helps of user ontology and the system ontology. Fig. 5 illustrates a sample of a virtual query. The `type` attribute is used for specifying query types specification such as *schema*, *data*. The part `<params>` contains parameters with the metadata properties and their values as well as attributes. Recommended data sources are specified in `<sources>` and the constraints are put in `<relations>` part with respective operators. The query result will be returned in XML format (in `<resultformat>` tag). The VQS enables to return results in forms of text, XML, RDF, or query answer objects.

We can now turn towards many of the more interesting issues involved in implementation solutions for the VQS

1. The *Meta-data Sources* module is developed from the scratch. This has been done with help of ontologies, and RDF data accessing methods. As the SemanticLIFE metastore is large scaled RDF storage, so that the RDF data accessing is required; and Jena [22] is our solution.

2. The supposed ontology language, which is used for the VQS ontologies, is chosen among languages being used for the SemanticLIFE system (RDFS [24] and OWL [25]). However, OWL is preferable because of its features which support advanced operations on ontology and inference. The choice for the ontology framework API is also Jena with flexible features.

3. About ontology mapping issue, MAFRA [13], a mapping framework for distributed ontologies in the Semantic Web, and PROMPT [15] are preferred to be the framework alternatives for ontology mapping service of the VQS

4. The *Sub-queries Formulation* module is also developed from scratch with help of the system ontology, RDF query languages, and the VQS services. Sub queries are generated from the user's virtual query after some analysis operations. This is a tough and most important task for the VQS because it needs as much the accuracy of user demands as possible. This module and the VQS services can deal with the ambiguity of terminology used in the user's requests. For example, the input

```
<query type="data">
  <params>
    <param show="1" name="p1:emailReceivedDate">
     2005-11-05</param>
    <param show="0" name="p2:emailReceivedDate">
     2005-11-15</param>
    <param show="1" name="p3:emailMbox">
     mailto:hanh@ifs.tuwien.ac.at</param>
    <param show="1" name="p4:calBody">RIVF</param>
  </params>
  <sources>
    <source name="email">Email</source>
    <source name="calendar">Calendar</source>
  </sources>
  <relations>
    <relation id="1" param="p1" source="email">dt:gt</relation>
    <relation id="2" param="p2" source="email">dt:lt</relation>
    <relation id="3" param="p3" source="email"/>
    <relation id="4" param="p4" source="calendar">str:match
    </relation>
  </relations>
  <resultformat>xml</resultformat>
</query>
```

Fig. 5. A virtual query coded in a XML document.

"AM" causes confusion of time terminology and name of a continent (America). It also utilizes user profile/ontology to generate the queries with specific criteria for data sources.

As far as concerning the RDF query languages, SPARQL [21] and iTQL [19] appears as promising candidates for a standardized RDF query language. iTQL is being used for the query prototype because it is closely related to the current storage solution – Kowari [19]. SPARQL provides facilities to extract information in the form of URIs, blank nodes, plain and typed literals, extract RDF subgraphs and construct new RDF graphs based on information in the queried graphs [17]. Because of these reasons, SPARQL will be used in coding the generated sub-queries in the next build.

5. There are some high-profiled frameworks for *ontology-based inferencing* module such as RACER [12] or Pellet [20]. However, there is no complete survey for the comparison of these frameworks so far. The proposed framework is selected based on our experience. RACER and Pellet are considered to be used for inferencing tasks.

6. Finally, the client-side application is discussed in two aspects: user interface design and user profile/modeling. The query user interface is another part developed from scratch. This part consists of two sub-modules; the first is the *virtual query formulation*. This sub-module helps the user creating virtual queries through a friendly query interface or a set of query templates. The second sub-module is the *query result composition* aiming at the aggregation of the results and their representation according to the user's demand. It show the recommended results to the user based on his/her profile and ontology.

VIII. Conclusion and Future Work

In this paper we have presented the Virtual Query System, an approach of building a complete Semantic Web query system based on a front-end approach. Besides applying existing Semantic Web technologies known in the area such as ontology mapping mechanisms, user annotation and semantic query caching for RDF data, we have designed and implemented a query system which aims at a significant complexity reduction in formulating semantic meaningful queries and at the same time aims at a considerable reduction of the number of ambiguous user queries.

The VQS is work in progress. Prototypes are developed to test the various aspects and design alternatives. The most important parts are developed to obtain a proof of concept of the architecture described.

As the next step, the issues of realizing a virtual query language such as semantic query caching, ontology-based query refinement, query optimization and user modeling will be discussed consequently in the details in context of the VQS.

Acknowledgment

The authors are very indebted to the all members of the SemanticLIFE-team who have been working since 2004 in this huge project.

References

[1] H.H. Hoang, A. M. Tjoa, T.M Nguyen, et al. "SemanticLIFE – a Framework for Managing Information of a Human Lifetime," In *Proc. 6th Int. Conf. Information Integration and Web-based Applications and Services*, Jakarta, Indonesia, Sep. 2004.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. "The Semantic Web," Scientific American Magazine. May 2001.

[3] V. Bush, "As we may think." *The Atlantic*, Monthly, Vol. 176 (1), Jul. 1945, pp. 101–108.

[4] J. Gemmel, G. Bell, R. Lueder, S. Drucker, C. Wong, "MyLifeBits: Fulfilling the Memex Vision," In *Proc. ACM Multimedia '02, ACM Press*, pp. 235-238.

[5] D. Huynh, D. Karger, D. Quan, "Haystack: A Platform for Creating, Organizing and Visualizing Information Using RDF," presented at Semantic Web Workshop, 2002.

[6] E. Freeman, D. Gelernter, "LifeStreams: A storage model for Personal Data," ACM SIGMOD Record, 1996.

[7] W. Nejdl, B. Wolf, S. Staab, J. Tane, et al, "EDUTELLA: Searching an Annotating Resources within an RDF-based P2P Networks," In *Proc. 11th Int. WWW Conf.*, Hawaii, 2002.

[8] Andrew Fitzgibbon, Ehud Reiter, "Memories for life", Managing information over a human lifetime, Grand Challenges in Computing workshop, UK Computing Research Committee's, 2003.

[9] I. F. Cruz, W. Sunna, and A. Chaudhry, "Ontology Alignment for Real-World Applications," DG.O National Conference on Digital Government Research, 2004.

[10] N. Stojanovic, R. Studer, L. Stojanovic, "An Approach for Step-By-Step Query Refinement in the Ontology-Based Information Retrieval," In *Proc. IEEE Int. Conf. Web Intelligence* (WI'04) 2004, pp. 36-43.

[11] P. Godfrey, J. Gryz, "Answering Queries by Semantic Caches." In *Proc. 10th DEXA*, Florence, Italy, Aug. 1999.

[12] V. Haarslev, R. Möller, "RACER: A Core Inference Engine for the Semantic Web." In *Pro. .2nd Int. Workshop on Evaluation of Ontology-based Tools*, .Florida, USA, Oct. 2003.

[13] A. Maedche, B. Motik, et al. "MAFRA - An Ontology Mapping Framework in the Semantic Web." In *Proc. Workshop on Knowledge Transformation*, Lyon, France, Jul. 2002.

[14] A. Maedche, A., S. Staab, "Semi-automatic Engineering of Ontologies from Texts." In *Proc. 12th Int. Conf. Software Engineering and Knowledge Engineering*, Chicago, IL, USA, Jul. 2000, pp. 231–239.

[15] N.F. Noy, M.A. Musen, "The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping." In *Int. J. Human-Computer Studies*, Vol. 59 (6), Academic Press Inc, MN, USA, 2003, pp. 983-1024.

[16] J. Pathak, et al, "INDUS: A System for Information Integration and Knowledge Acquisition from Autonomous, Distributed, and Semantically Heterogeneous Data Sources," In *Proc. 13th Intl. Conf. Intelligent System for Molecular Biology*, Michigan, Jun. 2005.

[17] D. Reynolds, (2004, 01, 08), "e-Person: Personal Information Infrastructure," Available: http://www.hpl.hp.com/semweb/eperson.htm

[18] H. Stuckenschmidt, "Similarity-Based Query Caching." In *Proc. 6th Intl. Conf. Flexible Query Answering Systems*, Lyon, France, 2004.

[19] D. Wood, P. Gearon, T. Adams, "Kowari: A Platform for Semantic Web Storage and Analysis", In *Proc. 14th Int. WWW Conference*, Edinburgh, Scotland, UK, May 2005.

[20] B. Parsia, E. Sirin, et al. "Pellet OWL Reasoner." The MINDSWAP Group, 2003. Available: http://www.mindswap.org/2003/pellet/

[21] E. Prud'hommeaux, A. Seaborne. (2005, 07, 21) "SPARQL Query Language for RDF." *W3C Working Draft*. Available: http://www.w3.org/TR/rdf-sparql-query/

[22] HP Labs Semantic Web Programme. (2005, 09, 02). "Jena – A Semantic Web Framework for Java." Available: http://jena.sourceforge.net/.

[23] Resource Description Framework (RDF). (2005, 09, 02). Available: http://www.w3.org/RDF/

[24] RDF Vocabulary Description Language: RDF Schema (RDFS). (2005, 09, 05). Available: http://www.w3.org/TR/rdf-schema/

[25] Web Ontology Language (OWL). (2005, 09, 02). Available: http://www.w3.org/2004/OWL/

[26] Semagix Freedom. (2004, 04, 13). Available: http://www.semagix.com

[27] Unicorn System. (2005, 05, 10). Available: http://www.unicorn.com