Principles and Model for Web Dataspace

*Zhengtao Liu
School of computer and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing China
School of computer science and engineering, Sanjiang University, Nanjing China
Email: lzt_jh_cn@yahoo.com.cn

Jiandong Wang
School of computer and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing China
Email: aics@nuaa.edu.cn

***Abstract*—Web information integrated management system requires a powerful and versatile data model that is able to represent a highly heterogeneous mix of data such as web pages, XML, deep web, files, etc. It requires access to both structured and unstructured data. Such collections of data have been referred to as dataspace. In order to build a web dataspace support platform, we described some principles. According to these principles, we design architecture for the web dataspace support platform. Based on the RDF model, we present a web dataspace Data Model. This model is able to represent all of the above data in a single, powerful yet simple data model. It can represent unstructured, semi-structured and structured data inside a single model.**

***Index Terms*—Dataspace, Data Model, Web Data, Design Principle**

I. Introduction

Since its inception, the World Wide Web has been dominated by unstructured content, and searching the web has primarily been based on techniques from Information Retrieval. Recently, however, we are witnessing an increase both in the amount of structured data on the web and in the diversity of the structures in which these data are stored. The prime example of such data is the deep web, referring to content on the web that is stored in databases and served by querying HTML forms. More recent examples of structure are a variety of annotation schemes that enable people to add labels to content (pages and images) on the web. The World Wide Web provides a vast source of information of almost all types. However, this information is often scattered among many web servers and hosts, using many different formats.

Dataspace is a recent concept within the databases community. Dataspace provide a powerful abstraction for accessing, understanding, managing, and querying this wealth of data by encompassing multiple data sources and organizing their data over time in an incremental, "pay-as-you-go" fashion. In contrast with other information integration systems, dataspace systems offer best-effort before complete semantic mappings are provided to the system. A key idea of dataspace is that the semantic cohesion of a dataspace is increased over time by different parties providing mappings [1].Dataspace management is not a data integration approach; rather, it is more of a data co-existence approach. The goal of dataspace support platform is to provide base functionality over all data sources, regardless of how integrated they are [2].

The vision of dataspace[3] is that various of the benefits provided by planned, resource-intensive data integration could be obtained at much lower initialisation cost, thereby supporting integration on demand and in changing environments, albeit with a lower initial quality of integration. For this to happen, dataspace would be expected to use techniques that infer relationships between resources, and that improve these relationships in the light of user or developer feedback. As such, a dataspace can be seen as a data integration system that exhibits the following distinguishing features: (i) low/no initialisation cost, and (ii) support for incremental improvement.

The challenges of dataspace discussed in [2] have influenced many research groups of the data management community. However, most effort was put on the mainstream related dataspace research[4,5,6] or on development of personal dataspace systems [7,8]and scientific dataspace. Based on iDM, Blunschi et al. implemented a personal dataspace management system called iMeMex[9,10], which offers some contextual information on query results. jSpace[11] is a prototype of a scientific dataspace support platform which is implemented and deployed to an early core of adopters in the breath gas research domain from which specific use cases are derived.

Based on the concept of dataspace, our work is to build a pay-as-you-go data integration system on the web. We call this system web dataspace. The web dataspace will provide the best-effort query result. The construction of

Manuscript received January 1, 2012; revised May 1, 2012; accepted June 1, 2012.
*Corresponding author.

doi:10.4304/jcp.8.4.998-1005

the system is a pay-as-you-go process, that is to say, with the growing use of and integration, the system will provide more accurate and reasonable results.

This paper presents a web dataspace support platform (WebDSSP). All the web data like web documents, deep web, semi-structured documents, unstructured data, RSS feeds and even data streams can be instantiated in the model of this system. In summary, this paper makes the following contributions:

1. We provide the principles of designing the WebDSSP. According to the design principles, we provide the web dataspace system architecture.
2. Based on the RDF model, we present the Data Model (WebDM) for the WebDSSP. WebDM allows for the unified representation of all web information like web pages, XML, deep web, data streams and RSS feeds inside a single data model.
3. We show that all parts of the WebDM can be computed lazily. Because of this, WebDM is able to support so-called intensional data, i.e. data that is obtained by executing a query or calling a remote service.

This paper is structured as fellows. Section 2 outlines the design principles of the WebDSSP. Section 3 then describes the core architecture of the system. Section 4 and section 5 present the formal definition of the data model. Section 7 reviews related work, Section 8 presents experiments with our initial WebDM implementation in myWebDSSP, and Section 9 concludes the paper.

II. Principles of Web Dataspace

WebDSSP aims to adhere to a small set of guiding principles that we believe are important in enabling applications Subsequent sections describe how these principles are currently reflected in WebDSSP. We note that in all cases, following these principles offers an agenda for a continuous process of improvements.

Manage all data on the web. WebDSSP should contain all of the information relevant to a particular organization regardless of its format and location, and model a rich collection of relationships between data repositories.

There are many data types on the web. The data on the web can by divided into three types. The first type is structure data, such as tables, deep web. Deep web refers to content that lies hidden behind query-able HTML forms. These are pages that are dynamically created in response to HTML-form submissions, using structured data that lies in backend databases. The second data type is semi-structured data, such as XML, E-Mail, etc. The last type is unstructured data, such as pictures, text, documents on the web. The WebDSSP must deal with data in a wide variety of formats accessible through many systems with different interfaces. The WebDSSP is required to support all the data on the web rather than leaving some out.

Apply pay-as-you-go data management principles. It is important to keep in mind that unlike a DBMS, the WebDSSP does not assume completely control over the data in the dataspace. Instead, the WebDSSP allows the data to be managed by the participant. So, the WebDSSP does not require that all the data in the system has semantic in the initial stage. Different data sources in system are coexists.

In theory, the dataspace and the corresponding entity should include all relevant data, but it is often impossible and unnecessary. Because the data itself is changing, new data sources, data items continue to emerge. The system can not be all entities related data included. Data manipulation and user demand are generated step by step. Hence, compared to integrated systems, this data management costs are lower.

Make full use of the existing technology. At present, there has been a lot about web data extraction and use of technical methods. In the system building process, we must fully exploit the existing web data mining methods. For example: we can use the links to some of the data representation to enable them to achieve a seamless transition to the Semantic web network; in data extraction, we make full use of more mature now some deep web extraction, web extraction technologies etc.

Facilitate collaboration. At web-scale, web data access is an ongoing process. In this process, the system will leverage mass collaboration, or the so-called wisdom of crowds, to provide more accurate query results. The component is consists of two parts: explicit user feedback and implicit user feedback. The explicit user feedback provides a lot of questions and let the user to answer or confirm. The implicit user feedback derived from observable user behavior is used as the input to the learning algorithms. Using click-through data, it learns ranked retrieval functions for the web search results.

Share data expediently. Users often desire to share data with others. However, they are faced with a number of disincentives. Data owners are afraid of loss of attribution, of misuse and corruption of their data, and of others not being able to find the data easily. WebDSSP aims to address such concerns.

III. System Architecture

According to the design principles, we provide a WebDSSP system architecture. Figure 1 shows the main components of the WebDSSP. The system is consistent of four layers.

Discovery Component. The goal of this component is to locate data source in the World Wide Web. The web is a complex entity that contains information from a variety of source types and includes an evolving mix of different file types and media. Using search engine technology, we can search for information on the World Wide Web and FTP servers. The information may consist of web pages, images, information and other types of files. The deep web is usually defined as the content on the web not accessible through a search on general search engines. According to survey of the deep web, the query interfaces are often close to the root page of the web site. That is, the depth of a web database is often very small. Motivated by this observation, MetaQuerier[14] develop

a site-based crawler, we use this technique to build our discovery component.

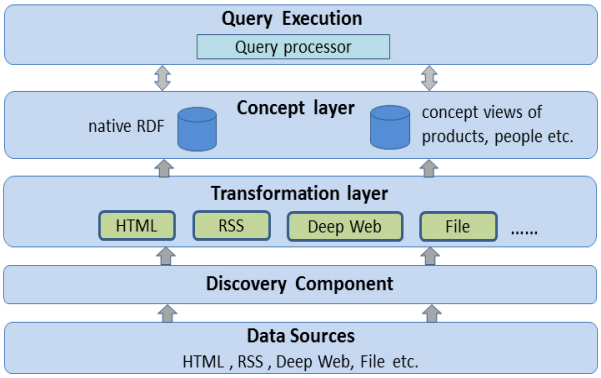

Figure 1. WebDSSP Architecture

Transformation layer. This component provides built-in RDF middleware for transforming non-RDF data into RDF "on the fly". Its goal is to use non-RDF web data sources as input, e.g. (X)HTML web pages, (X)HTML web pages hosting micro formats, and even web services such as those from Google, Flickr etc., and create RDF as output. The implication of this facility is that you can use non-RDF data sources as web dataspace data sources.

Conceptual layer. Dataspace provides a powerful conceptual model based virtualization layer that sits atop heterogeneous data sources. They enable the creation, publication, and general management of structured data items from a pool of disparate data sources.

Query execution. In this component, our design is to provide users a domain category hierarchy, which is similar to the category organization of Yahoo.com, but is automatically formed by the system and users attention. For each domain, a unified interface is provided by the system. When a query is posed, it is reformulated to access the data sources. The reformulation process uses the semantic mappings to determine which data sources are relevant to the query, formulate appropriate sub-queries on the individual data sources, and specify how results from the different sources are combined (e.g., via join or union) to produce answers to the query.

IV. WEB DATASPACE DATA MODEL

This section gives an overview of the implementation of data model which builds the foundation of the web Dataspace System.

In WebDM, all the web data available on a user's desktop is exposed through a set of data views. A data view is a sequence of components that express structured, semi-structured and unstructured pieces of the underlying data. Thus all the web data items, though varied in their representation, are exposed in WebDM uniformly.

The WebDM is based on the *Resource Description Framework (RDF)*. RDF is a standard model for data interchange on the web. RDF has features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed. In the RDF model, the universe to be modeled is a set of *resources*, essentially anything that can have a *universal resource identifier*, URI. The language to describe them is a set of *properties*, technically binary predicates. Descriptions are *statements* very much in the subject-predicate-object structure. Both subject and object can be anonymous objects, known as *blank nodes*. In addition, the RDF specification includes a built-in vocabulary with a normative semantics (RDFS). This vocabulary deals with inheritance of classes and properties, as well as typing, among other features allowing the descriptions of concepts and relationships that can exist for a community of people and software agents, enabling knowledge sharing and reuse.

Now, we present a streamlined formalization of the RDF model along the lines of 12.

DEFINITION 1. Assume there is an infinite set U (RDF URI references); an infinite set $B = \{b_j : j \in N\}$ (Blank nodes); and an infinite set L (RDF literals). A triple $(v_1, v_2, v_3) \in (U \cup B) \times U \times (U \cup B \cup L)$ is called an RDF triple. We often denote by UBL the union of the sets U, B and L.

DEFINITION 2. An RDF graph (just graph from now on) is a set of RDF triples. A subgraph is a subset of a graph.

A graph is ground if it has no blank nodes.

DEFINITION 3. A mapping is a function $\mu : UBL \rightarrow UBL$ preserving URI references and literals (i.e., $\mu(u) = u$ and $\mu(l) = l$ for all $u \in U$ and $l \in L$).

DEFINITION 4. Let G_1, G_2 be RDF graphs. Then G_1 entails G_2 (denoted $G_1 \models G_2$) if and only if an instance of G_2 is a sub graph of G_1. We say that two graphs are equivalent (denoted $G_1 \equiv G_2$) if $G_1 \models G_2$ and $G_2 \models G_1$.

In order to build a web dataspace, we use RDF/RDFS model to describe the data on the web dataspace.

DEFINITION 5. A data view D_i is a 6-tuple $(\theta_i, \kappa_i, \lambda_i, \varphi_i, \upsilon_i, \tau_i)$, where θ_i is a name component, κ_i is a lineage component, λ_i is a schema component, φ_i is a tuples component, υ_i is a group component, and τ_i is a confidence component. We define each component of data view D_i as follows:

θ_i NAME COMPONENT: θ_i is a finite string that represents the name of D_i.

κ_i LINEAGE COMPONENT: κ_i is a lineage function. In WebDM, we focus on "where lineage": the lineage of a tuple identifies the data from which it was derived. Some tuples in our system are derived from data views tuples. The default value of the κ_i is the URL of the D_i.

λ_i SCHEMA COMPONENT: λ_i is the schema of the data view. The schema λ_i may be a relation schema, XML schema or other type schema.

φ_i TUPLES COMPONENT: φ_i is the tuples of the data view. Every tuple is RDF tripe. The Tuples component can be represent a 2-tuple (A,T), where A is the set of all the symbols, T is the set of all the statement in the φ_i. Further, the interpretation of φ_i is a 6-tuples (*DR, DP, IR, IP, IEXT, ICEXT*), where:

DR is the set of resource described by RDF tripe, the resource set contains several types of value, which instance of the value (for example, string or numeric value), the URL corresponding to web resources, also

includes the class concepts and attribute concepts.

DP is the set of attribute concepts, $DP \subseteq DR$.

IR:$A \rightarrow DR$. It interprets the symbol $a(a \in A)$ as a corresponding resource.

IP:$A \rightarrow DP$. It interprets the symbol $a(a \in A)$which describes a attribute as a corresponding attribute.

$IEXT$:$DP \rightarrow DR \times DR$, It interprets attribute as a bivariate vector set, $IEXT(p)=\{(d_i,d_j)|\ d_i,d_j \in DR\}$.

$ICEXT$:$DR \rightarrow \rho(DR)$. It interprets the class concept resource as an instants set. $ICEXT(c)=\{\ (d_i,d_j)|\ d_i,d_j \in$ DR, d is a instant of class $c\}$.

υ_i GROUP COMPONENT: υ_i is a 2-tuple (S,Q), where S is a (possibly empty) set of data views and Q is a (possibly empty) ordered sequence of data views. Further,

The set S and the sequence Q may be finite or infinite. When S is finite, we denote S = $\{V_{s1},...,V_{sm}\}$; when it is infinite, then we denote $S = \{V_{s1},...,V_{sm}\}_{m\rightarrow\infty}$. Likewise, when Q is finite, we denote $Q = <V_{q1},...,V_{qn}>$; when Q is infinite, then we denote $Q = <V_{q1},...,V_{qn}>_{n\rightarrow\infty}$.

$S \cap Q =$, i.e., S and Q are disjoint .

Assume a data view D_i has a non-empty υ_i component. If there exists a data view D_k for which $D_k \in S \cup Q$ holds, we say that D_k is directly related to V_i, i.e., $D_i \rightarrow D_k$. Any given data view may be directly related to zero, one or many other resource views.

If $D_i \rightarrow D_j \rightarrow ... \rightarrow D_k$, we say that data view D_k is indirectly related to D_i.

τ_i CONFIDENCE COMPONENT: τ_i is a confidence value. Its value will change with users using the system.

If any of the components of a data view is empty, we denote its value, where convenient, by the empty n-tuple () or by the empty sequence<>.

In WebDM, all web information available on the World Wide Web is exposed through a set of data views. All web items, though varied in their representation, are exposed in WebDM uniformly.

One important aspect of our model is that the data view is a logical representation of web information. That logical representation does not need to be fully materialized at once and may be computed lazily. The data in the World Wide Web does not have to be imported to our model, but rather represented in it.

V. INSTANTIATING SPECIALIZED DATA MODELS

In the following, we show how to instantiate specialized data models like web page, deep web, files&folders, as well as data streams using WebDM. Due to space constraints, we omit details on how to instantiate relational data. These instantiations will, however, become clear to the reader based on the discussion for the other data models.

A. Web Page

In the WWW, most web pages are HTML files. Now, the famous search engine, Google, can find most of HTML files in the world. Google has one of the largest databases of web pages, including many other types of web documents (blog posts, wiki pages, group discussion threads and document formats (e.g., PDFs, Word or Excel documents, Power Points). Despite the presence of all these formats, Google's popularity ranking often makes pages worth looking at rise near the top of search results. The WWW also provides access to millions of data tables with high-quality content, formatted either in HTML tables, HTML lists, or other structured formats, or stored in on-line data management services. These tables contain data about virtually every domain of interest to mankind. First step, we use the result of Google to handle the web pages. In the future, we will provide a data warehouse to store the web pages, and give a wisdom analysis to it. Then we will provide more accurate answer to the user.

In the Google search result page, each record in the page has a fixed set of properties, such as link name, description of the link page, the link page URL, the link page size, etc. The attributes that appear on each record are defined as the schema of the data view S_{page} = {name: string, description: string, url: string, size: int, unit: string, similar page: string, …}. The sequence of values in the search result page are expressed per record, we denote this sequence T_r. We use Google as the data view name and denoted Nr. In addition, the lineage of the data view is www.google.com and denoted Lr.

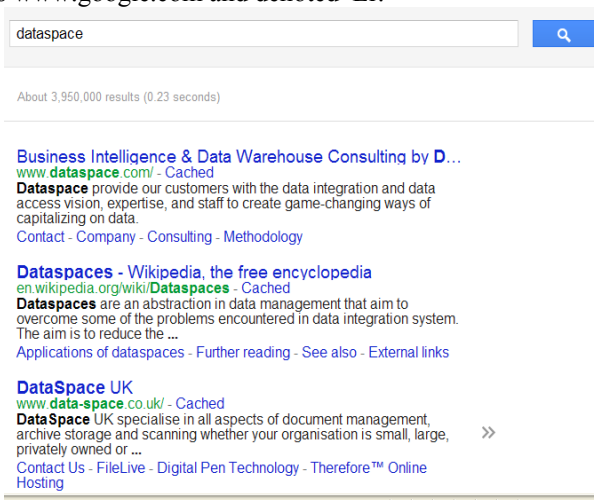

Figure 2 Google Search Result

We define the data view instance of web page as follows:

$D_i^{page} = (\ \theta_i^{page},\ \kappa_i^{page},\ \lambda_i^{page},\ \phi_i^{page}\ )$, *where:*

$\theta_i^{page} = N_r$,

$\kappa_i^{page} = L_r$,

$\lambda_i^{page} = S_{page}$,

$\phi_i^{page} = T_r$.

B. Deep Web

The deep web is usually defined as the content on the Web not accessible through a search on general search engines. This content is sometimes also referred to as the hidden or invisible web.

The Web is a complex entity that contains information from a variety of source types and includes an evolving mix of different file types and media. It is much more than static, self-contained Web pages. In fact, the part of the Web that is not static, and is served dynamically "on

the fly," is far larger than the static documents that many associate with the Web.

The concept of the deep Web is becoming more complex as search engines have found ways to integrate deep Web content into their central search function. This includes everything from airline flights to news to stock quotations to addresses to maps to activities on Facebook accounts. In the screenshot below, notice the various deep Web sources offered by Google, including images, maps, news, video, shopping, scholarly content, blogs, and so on. However, even a search engine as far-reaching as Google provides access to only a very small part of the deep Web.

The deep web represents the most structured data on the web. Although many forms offer search over document collections, the majority of forms offer search into data that is stored in back-end databases and the form queries get translated into queries on the database. Annotation schemes represent the structured data on the web. Using Schema building component, the information maintained about each schema includes the name, data types, relationships between schemas and any available data instances and textual descriptions of each element in the schema.

The result of deep web query may be come from different back-end databases. Each database is defined as a deep web data view that constrains a view instance D_i^{deep} as follows:

$D_i^{deep} = (\theta_i^{deep}, \kappa_i^{deep}, \lambda_i^{deep}, \phi_i^{deep}, \tau_i^{deep})$, *where:*

θ_i^{deep} *, the name of the data source, often is the title of the site that include the back-end database.*

κ_i^{deep} *, the URL of the site if the query data is not to be handled or the a lineage function.*

λ_i^{deep}*, the schema of the database record.*

ϕ_i^{deep}*, the query result records.*

τ_i^{deep}*, the confidence of the data source.*

C. Files&Folders

Using many FTP tools, we can access files in the FTP server on the web. These files contain a lot of information. File systems can be seen as trees in which internal nodes represent non-empty folders and leaves represent files or empty folders. Each node in the tree has a fixed set of properties, such as size, creation time, last modified time, etc. The attributes that appear on each node are defined in a file system-level schema W_{FS} = {size: int, creation time: date, last modified time: date, ...}. The sequence of values that conform to that schema are expressed per node and, for a node n, we denote this sequence T_n. Each node also has a name, denoted N_n. In addition, if n is a file node, then it has an associated content C_n. In WebDM, we consider a 'file' just one out of many possible data views on user data. In order to have WebDM represent the files&folders data model, we define a file data view class, denoted D_i^{file}, that constrains a view instance file to represent a file *f* as follows:

$D_i^{file} = (\theta_i^{file}, \kappa_i^{file}, \lambda_i^{file}, \phi_i^{file})$, *where:*

$\theta_i^{file} = N_n$,

λ_i^{file}*, the url of the files.*

$\kappa_i^{file} = W_{FS}$,

$\phi_i^{file} = C_n$,

Based on this definition we can recursively define the concept of a 'folder'. A folder resource view class, denoted folder, constrains a folder view instance D_i^{folder} to represent a folder F as follows:

$D_i^{folder} = (\theta_i^{folder}, \kappa_i^{folder}, \lambda_i^{folder}, \upsilon_i^{folder})$, *where:*

$\theta_i^{folder} = N_F$,

λ_i^{folder}*, the URL of the files.*

$\kappa_i^{folder} = W_{FS}$,

$\upsilon_i^{folder} = (\{D_1^{child},...,D_m^{child}\},<>)$, $child \in \{file,folder\}$

D. Data Stream

A data stream is an ordered sequence of instances that in many applications of data stream handling can be read only once or a small number of times using limited computing and storage capabilities. Examples of data streams include computer network traffic, phone conversations, ATM transactions, web searches, and sensor data.

In a formal way, a data stream is any ordered pair (S, △)where:

S is a sequence of tuples and

△ is a sequence of positive real time intervals.

Firstly, we define a generic data stream data view class to constrain WebDM to represent a generic data stream data model.

$D_i^{stream} = (\upsilon_i^{stream})$,

Where:

$\upsilon_i^{stream} = (\Phi, <D_1, \ldots D_n>_{n \to \infty})$.

D_i *is a subset of* D_i^{stream}.

Example of a data stream is an RSS/ATOM stream delivering XML messages. An RSS/ATOM stream data view class is defined to restrict a data view $D_i^{rssatom}$ as follows:

$D_i^{rssatom} = (\theta_i^{rssatom}, \kappa_i^{rssatom}, \lambda_i^{rssatom}, \tau_i^{rssatom}, \upsilon_i^{rssatom})$, *where:*

$\theta_i^{rssatom}$*, the name of the RSS/ATOM.*

$\kappa_i^{rssatom}$*, the URL of the site if the RSS/ATOM data is not to be handled or the a lineage function.*

$\lambda_i^{rssatom}$*, the schema of the RSS/ATOM record.*

$\phi_i^{rssatom}$*, theRSS/ATOM records in a time interval.*

$\tau_i^{rssatom}$*, the confidence of the data source.*

$\upsilon_i^{rssatom} = (\Phi, <D_1^{XML}, \ldots D_n^{XML}>_{n \to \infty})$.

D_i^{XML} *is the RSS/ATOM records in a time interval or a finite set.*

E. Linked Data

The term Linked Data refers to a set of best practices for publishing and connecting structured data on the Web. These best practices have been adopted by an increasing number of data providers over the last three years, leading to the creation of a global data space containing billions of assertions - the Web of Data.

Berners-Lee (2006) outlined a set of 'rules' for publishing data on the Web in a way that all published data becomes part of a single global data space:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
4. Include links to other URIs, so that they can discover more things.

A typical case of a large Linked Dataset is DBPedia, which, essentially, makes the content of Wikipedia available in RDF. The importance of DBPedia is not only that it includes Wikipedia data, but also that it incorporates links to *other* datasets on the Web, e.g., to Geonames. By providing those extra links (in terms of RDF triples) applications may exploit the extra (and possibly more precise) knowledge from other datasets when developing an application; by virtue of integrating facts from several datasets, the application may provide a much better user experience.

Each RDF links is defined as a linked data view that constrains a view instance D_i^{linked} as follows:

$D_i^{linked} = (\theta_i^{linked}, \kappa_i^{linked}, \phi_i^{linked}, \tau_i^{linked})$, *where:*

θ_i^{linked}, *the name of the data source, often is the title of the linked data project name.*

κ_i^{linked}, *the URL of the site .*

ϕ_i^{linked}, *the RDF triples.*

τ_i^{linked}, *the confidence of the data source.*

VI. Evaluation

The goal of this experimental evaluation is to show that WebDM can be efficiently implemented in a real WebDSSP. We present results based on an initial implementation of WebDM in myWebDSSP which is a web dataspace prototype system. We execute different classes of queries over webDM and report query response times.

We performed our experiments on an Intel Pentium M 2.6 GHz PC with 2 GB of main memory and an IDE disk of 360 GB. We used MS-Windows7 as its operating system and a volume with NTFS. myWebDSSP is implemented in Java and we used Oracle's hotspot JVM 1.6. The JVM was configured to allocate at most 1G MB of main memory.

We focus our evaluation on response times observed for a set of queries over the web dataset used. Firstly, we acquire the data sets from internet through some tools, such as FTP tools, deep web tools etc. The data sets are shown in Table 1. Figure 2 shows the observed response times for the six queries evaluated. We report all execution times with a warm cache, i.e., each query is executed several times until the deviation on the average execution time becomes small. From this Figure, we can find that the first query will take more time than the others. The reason is that, all components of a resource view may be computed lazily. As we may notice, most queries evaluated (Q1-Q5) are executed in less than 0.5 seconds. The only exception is Q6, a join query that includes information from different subsystems (deep web records, file system) and takes about 1 second. The reason is that, after fetching the data via index accesses, our query processor obtains indirectly related resource views by forward expansion. For Q6, that causes the processing of a large number of intermediate results when compared to the final result size. In order to provide even better response times in such situations, we plan to investigate alternative processing strategies such as cloud computing. Thus, we conclude that for a sample of meaningful queries over real web data the implementation of WebDM in myWebDSSP allows for query processing with interactive response times.

VII. Related Work

Dataspace Research. The most acute information management challenges today stem from organizations relying on a large number of diverse, interrelated data sources, but having no means of managing them in a convenient, integrated, or principled fashion. Franklin, Halevy and Maier have proposed dataspace as a new abstraction for information management. They also lay out a lot of challenges to realizing this system. Dataspace have been identified as one key challenge in information integration[13]. Paper 15 presents a framework for dataspace integration and management system. Recently, web-scale pay-as-you-go information integration for the deep web and Google Base is explored in 16. However, the authors present only a high-level view on architectural issues — the underlying research challenges are not tackled. In contrast to the latter work, our paper is

TABLE I.

WEB DATA SETS

Data Type	Size
Web Pages	2.5G
Files&Folders	2G
Deep web records	500M
Data stream records	525M
Linked data	350M

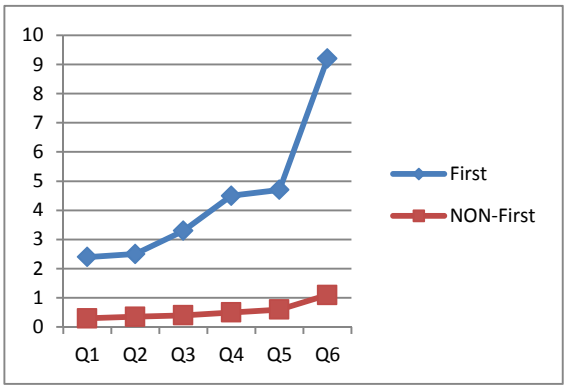

Figure 3. Query response times for queries Q1–Q6 [sec]

the first to provide an actual data model for web dataspace.

Data Model. The relational model has been proposed as a means to provide universal access to data. Object-oriented systems augment relational approaches to represent complex data types. All these previous approaches are based on a schema-first modeling strategy that is not well suited for web information. In contrast, our data model is in line with the goals recently presented in 1, which claims that information systems in the future must be capable of offering a wide range of modeling strategies, such as schema-later or even schema-never. iMeMex[9,10] Personal Dataspace Management System (PDSMS) use iDM as its data model. The iDM model is powerful enough to represent graph structured data, intensional data as well as infinite data streams. Further, the model enables to represent the structural information available inside files. Compared with iDM, our data model is special model to the web information, but iMeMex is a PDSMS. Paper 17 describes the concepts of probabilistic mediated schemas and probabilistic mappings as enabling concepts for Dataspace Support Platforms. Paper 18 introduces the basic principles and rationale of Linked Data and provides detailed guidance for those exploring this emerging area of web technology. Anish Das Sarma, Xin Luna Dong and Alon Y. Halevy argue that to support DSSPs, the system needs to model uncertainty at its core. They describe the concepts of probabilistic mediated schemas and probabilistic mappings as enabling concepts for DSSPs[19].

Web data integration There has been a lot of recent interest in web-data integration systems. The work closest to ours is the Meta-Querier project [20] that investigates the integration of deep-web sources. Their approach focuses on automatically creating a unified interface for accessing multiple deep-web sources in specific domains, i.e., a vertical search site. Our goal is more general – we do not want to create vertical search sites in specific domains, but rather enhance web-search by including content in deep web and other data sources across many domains. Alon etc. propose a new data integration architecture, PAYGO[16], which is inspired by the concept of dataspaces and emphasizes pay-as-you-go data management as means for achieving web-scale data integration.

VIII. Conclusions

Based on the concept of dataspace, we design a web dataspace system. The web dataspace envisions a system that offers useful services on its data without any setup effort and that improves with time in a pay-as-you-go fashion. Firstly, we describe some principles to build this system. According the principles, architecture for this system is designed. In order to integrate all the data on the web, we present a web dataspace model that is based on the RDF model. This model is able to represent all of the above data in a single, powerful yet simple data model. It can represent unstructured, semi-structured and structured data inside a single model.

References

[1] Alon Y. Halevy, Michael J. Franklin, David Maier, “Dataspaces: A New Abstraction for Information Management”, In Proceedings of the DASFAA, pp.1-2, 2006.

[2] Alon Y. Halevy, et al.: Principles of dataspace systems. In: PODS 2006)

[3] M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. SIGMOD Record, 34(4):pp.27–33, 2005.

[4] 3. Dong, X., Halevy, A.: Indexing dataspaces. In: SIGMOD, pp. 43–54,2007.

[5] Jeffery, S.R., Franklin, M.J., Halevy, A.Y.: Pay-as-you-go user feedback for dataspace systems. In: SIGMOD, pp. 847–860 (2008)

[6] Das Sarma, A., Dong, X., Alon Y. Halevy.: Bootstrapping pay-as-you-go data integration systems. In: SIGMOD, pp. 861–874 (2008)

[7] Dittrich, J.P., et al.: iMeMex: escapes from the personal information jungle. In: VLDB. VLDB Endowment, pp. 1306–1309, 2005.

[8] Li, Y., et al.: Research on personal dataspace management. In: IDAR, pp. 7–12(2008)

[9] Jens-Peter Dittrichand, Marcos Antonio Vaz Salles, “iDM: A Unified and Versatile Data Model for Personal Dataspace Management”, In Proceedings of the VLDB, pp.367–378, 2006.

[10] Lukas Blunschi, Jens-Peter Dittrich, Olivier René Girard, “A Dataspace Odyssey: The iMeMex Personal Dataspace Management System(Demo)”, In Proceedings of the CIDR, pp.1-1, 2007.

[11] Ibrahim Elsayed and Peter Brezany. Towards Large-Scale Scienti_c Dataspaces for e-Science Applications. In Proceedings of the Database Systems for Advanced Applications (DASFAA) Conference 2010, 2010.

[12] Claudio Gutierrez, Carlos Hurtado, Alberto Mendelzon, “Formal aspects of querying RDF databases”, In Proceedings of the SWDB, pp.293-307, 2003.

[13] Alon Y. Halevy, Anand Rajaraman, Joann J. Ordille, “Data Integration: The Teenage Years”, In Proceedings of the VLDB, pp.9-16, 2006.

[14] Kein C. Chang, Bin He, Zhen Zhang, “Toward large scale integration: Building a MetaQuerier over databases on the web”, In Proceedings of the CIDR, pp.44-55, 2005.

[15] YuKun Li, XiaoFeng Meng, XiangYu Zhang, “Research on Dataspace”, Journal of Software, vol.19, no.8, pp.2018-2031, 2008.

[16] Jayant Madhavan, Shirley Cohen, Xin Luna Dong, Alon Y. Halevy, “Web-Scale Data Integration: You can afford to Pay as You Go”, In Proceedings of the CIDR, pp.342-350, 2007.

[17] Anish Das Sarma, Xin (Luna) Dong, Alon Y. Halevy, “Data Modeling in Dataspace Support Platforms”, Lecture Notes in Computer Science, no.5600, pp.122-138, 2009.

[18] Tom Heath, Christian Bizer, “Linked Data: Evolving the Web into a Global Data Space”, Synthesis Lectures on the Semantic Web: Theory and Technology, USA, 2011.

[19] Anish Das Sarma, Xin Luna Dong, Alon Y. Halevy, “Uncertainty in Data Integration and Dataspace Support

Platforms", In the Proceedings of Schema Matching and Mapping, pp.75-108,2011.
[20] K. Chang, B. He, and Z. Zhang. toward large scale integration: Building a MetaQuerier over databases on the web. In CIDR, 2005.

Zhengtao Liu male, 1975-07-04 born in Shandong China, received the B.E. degree in computer technology from Beihua University and the M.S. degree in computer application from Nanjing University of Aeronautics and Astronautics in 1997 and 2006, respectively. He is currently pursuing the Ph.D. degree in computer application at Nanjing University of Aeronautics and Astronautics. Research direction: Web data integration, web dataspace etc.

He is the director of the computer application research institute in Sanjiang University. Published paper more than 20 pieces in computer science etc.

Jiandong Wang was born in 1945,Jiangsu China. He is a professor and doctoral supervisor at Nanjing University of Aeronautics and Astronautics. His research interests include data mining, machine learning and knowledge engineering, etc.