International Journal of Computer Science and Network (IJCSN)
Volume 1, Issue 6, December 2012 www.ijcsn.org ISSN 2277-5420

Web Service Integration Using Cloud Data Store

[1]Asfia Mubeen, [2]Mohd Murtuza Ahmed Khan , [3]Sana Mubeen Zubedi

**[1] Cse, Jntu, Lords institute of engineering and technology
Hyderabad, Andhra Pradesh 28, India**

**[2] Cse, Jntu, Lords institute of engineering and technology
Hyderabad, Andhra Pradesh 28, India**

**[3] Cse, Jntu, Lords institute of engineering and technology
Hyderabad, Andhra Pradesh 28, India**

Abstract

Current Web users usually have their own files, work documents, communications and personal contacts distributed in the storage systems of many widely-used Internet services (e.g. Google Docs, Gmail, Face book, Zoho). Therefore, they face the challenge of being not able to have an integrated view for their related data objects (e.g. mails, pics, docs, contacts). Recently, most of the major Internet services provide standard APIs that allow developing software applications that can read and write data from their underlying data store after providing the credential access information of registered accounts. The World Wide Web is witnessing an increase in the amount of structured content – vast heterogeneous collections of structured data are on the rise due to the Deep Web, annotation schemes like Flickr, and sites like Google Base. While this phenomenon is creating an opportunity for structured data management, dealing with heterogeneity on the web-scale presents many new challenges. In this paper, we highlight these challenges in two scenarios – the Deep Web and Google Base. We contend that traditional data integration techniques are no longer valid in the face of such heterogeneity and scale. We propose new data integration architecture, as means for achieving web-scale data integration.

Keywords— *Cloud Computing, Data Integration, Data Store, Web Services.*

1. Introduction

The great impact of the World Wide Web is rapidly transforming industries, business models and our work culture. The emergence of many popular Internet services such as: Web-based email, multimedia sharing, collaborative tools and social networks plus the increased worldwide availability of high-speed connectivity have accelerated the trend of moving the computing and data storage from PC clients to large Internet services.

In the existing system integration of current, requests and finding the required objects and the relationships between them is not possible by any single system. Neither traditional web search engines nor desktop tools can achieve that goal. The main reason behind this is that the required objects reside in online hidden repositories that such systems are not authorized to access. Web search engines are only able to crawl and index the surface web while desktop tools can only access data and files which are stored in the local file systems. In the Proposed system which is designed to enable the Web users to interact with their Internet services normally while, behind the scene, the information of their objects will be extracted, consolidated, linked and then populated into a single data store where the user can have integrated access to their data objects from anywhere in the world through multiple devices. The system makes use of a significant feature which is recently introduced by almost all of the major Internet services where they provide standard APIs that allow developing software applications that can read and writes data from their services after providing the credential access information usually a username and password of registered accounts.

2. Previous Work

Data, since its inception in 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 (e.g., Flickr, the ESP game, Google Co-op) that enable people to add labels to content (pages and images) on the web, and Google Base, a service that allows users to load structured data from any domain they desire into a central repository.

A common characteristic of these collections of structured data is that they yield heterogeneity at scales unseen before. For example, the deep web contains millions of HTML forms with small and very diverse schemata, Google Base contains millions of data items with a high degree of diversity in their structures, and Google Co-op is producing large collections of heterogeneous annotations. Heterogeneity in this environment is reflected in two aspects: First, the same domain can be described using multiple different schemata (e.g., multiple Google Base schemata describing vehicles); [1]

Second, there may be many ways to describe the same real-world entity (e.g., multiple ways of referring to the same product or person). The presence of vast heterogeneous collections of structured data poses one of the greatest challenges to web search today. To take a concrete example, suppose a user poses the query "Honda Civic" to a web-search engine. We would like the engine to return (and properly rank) results that include, in addition to unstructured documents, links to web forms where the user can find new or used cars for sale, links to sites where car reviews can be found, entries from Google Base that may be relevant, and links to special sites that have been annotated by car enthusiasts as relevant. If the user further specifies a geographic location with the query, the engine should specialise the results appropriately, and perhaps include links to Honda dealers in the area, or a link to an appropriate dealer locater form.

Improving search in the presence of such heterogeneous data on the web leads to a fundamental question: are too many structures (i.e., schemata) akin to no structure at all? In other words, are we doomed to query this content with traditional web-search techniques based on Information Retrieval? Or can we extend techniques from data management, in particular from heterogeneous data integration, to improve search in such contexts? This paper contends that traditional data integration techniques are no longer valid in the face of such heterogeneity and scale.

Thus, we propose new data integration architecture, as a methodology for approaching this challenge. This architecture is inspired by the concept of data spaces that emphasizes pay-as-you-go data management. We begin by describing two data management efforts at Google in some detail, and use them to expose the challenges faced by web-scale heterogeneity. First, we discuss our work on searching the deep web. We describe how the scale of the problem affects two alternative approaches to deep-web querying: run-time query reformulation and deep-web surfacing. The former approach leaves the data at the sources and routes queries to appropriate forms, while the latter attempts to add content from the deep web to the web index. We also describe the first study of the deep web that is based on a commercial index; the study suggests that the deep web contains millions of sources. Second, we consider Google Base and describe how schema when available can be used to enhance a user's search experience. This improvement, however, comes at the expense of large-scale heterogeneity that arises naturally as a result of the large numbers of independent contributions of structured data to Google Base. Additionally, we touch briefly upon annotation schemes to further illustrate the challenges and opportunities of structured data on the web. [1]

The deep (or invisible) 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. This content is considered invisible because search-engine crawlers rely on hyperlinks to discover new content. There are very few links that point to deep web pages and crawlers do not have the ability to fill out arbitrary HTML forms. The deep web represents a major gap in the coverage of search engines: the deep web is believed to be possibly larger than the current WWW, and typically has very high-quality content.

There has been considerable speculation in the database and web communities about the extent of the deep web. We take a short detour to first address this question of extent. In what follows, we provide an estimate of the size of the deep web in terms of the number of forms. This measure will give us an idea of the quantity of structured data on the web and hence the potential and need for structured data techniques on a web-scale.

Extent of the Deep Web: The numbers below are based on a random sample of 25 million web pages from the Google index. Readers should keep in mind that public estimates of index sizes of the main search engines are at over a billion pages, and scale the numbers appropriately. To our surprise, we observed that out of 25 million pages, there were 23.1 million pages that had one or more HTML forms. Of course, not all of these pages are deep web sources. Thus, we refined our estimate by attempting to successively eliminate forms that are not likely to be deep web sources. [1]

First of all, many forms refer to the same action, i.e., the url that identifies the back-end service that constructs result pages. In our sample, there were 5.4 million distinct actions. Many actions, however, are changed on-the-fly using JavaScript and therefore many of them may refer to the same content. As a lower bound for the number of deep web forms, we counted the number of hosts in the different actions urls. We found 1.4 million distinct hosts in our sample.

In order to refine our notion of distinct deep web forms, we computed a signature for each form that consisted of the host in the form action and the names of the visible inputs in the HTML form. We found 3.2 million distinct signatures in our sample. To further refine our estimate, we decided to count only forms that have at least one text field. After all, if a form contains only drop-down menus, check-boxes and radio buttons, it is conceivable that a search engine can try all combinations of the form and get the content in a domain-independent way. We also eliminated common non-deep web uses of forms such as password entry and mailing list registration. In our sample, 1.5 million distinct forms had at least one text box. [1]

Finally, forms that contain a single input are typically (but certainly not always!) of the type "search this site" and do not yield new web content; similarly, forms with a large number of inputs (say, 10) often capture some detailed interaction, e.g., booking an airplane ticket. To further limit our numbers to forms that are likely to offer deep web content, we counted the number of forms that have at least one text input and between two and ten total inputs. In our sample, we found 647,000 such distinct web forms. This number corresponds to roughly 2.5% of our sample of 25 million pages. Scaling this estimate to an index of one billion pages yields 25 million deep web sources. While a simple scaling of the numbers might not be statistically accurate (as would be true of any unique aggregator), the numbers we show will serve to illustrate the number of web forms that estimated 450,000 forms and did not consider any of the filters that led to our final estimate.

To put this estimate in perspective, consider that each deep web source may lead to a large amount of content (e.g., a single form on a used car site leads to hundreds of thousands of pages). Thus, the amount of content on the deep web is potentially huge. In addition to their large number, we observed that the semantic content of deep web sites varies widely. Many of the sources have information that is geographically specific, such as locators for chain stores, businesses, and local services (e.g., doctors, lawyers, architects, schools, tax offices). There are many sources that provide access to reports with statistics and analysis generated by governmental and non-governmental organizations. Of course, many sources offer product search. However, there is a long tail of sources that offer access to a variety of data, such as art collections, public records, photo galleries, bus schedules, etc. In fact, deep web sites can be found under most categories of the ODP directory. [2]

While surfacing does make deep web content searchable, it has some important disadvantages. The most significant one is that we lose the semantics associated with the pages we are surfacing by ultimately putting HTML pages into the web index. By doing so, we overlook the possibility of exploiting the structure during query time. Further, it is not always possible to enumerate the data values that make sense for a particular form, and it is easy to create too many form submissions that are not relevant to a particular source. For example, trying all possible car models and zip codes at a used car site can create about 32 million form submissions a number larger than the number of cars for sale in the United States. Finally, not all deep web sources can be surfaced: sites with robots.txt as well as forms that use the POST method cannot be surfaced. [2]

We would ideally like a solution where given an arbitrary user keyword query, we identify just the right sources that are likely to have relevant results, reformulate the query into a structured query over the relevant sources, retrieve the results and present them to the user. The problem of identifying relevant sources to user keyword queries, which we call *query routing*, will be a key to any web-scale data integration solution as we discuss later. We are currently pursuing the surfacing approach as a first step to exploring the solution space in the context of the deep web.

Google Base

The second source of structured data on the web that we profile, Google Base, displays a high degree of heterogeneity. Here, we detail this degree of heterogeneity and describe some of the challenges it poses to web-scale data integration. Google Base is a recent offering from Google that lets users upload structured data into Google. The intention of Google Base is that data can be about *anything*. In addition to the mundane (but popular) product data, it also contains data concerning matters such as clinical trials, event announcements, exotic car parts, and people profiles. Google indexes this data and supports simple but structured queries over it.

Users describe the data they upload into Google Base using an item type and attribute/value pairs. For example, a classified advertisement for a used Honda Civic has the item type vehicle and attributes such as make = "Honda" and model = "Civic". While Google Base recommends popular item types and attribute names, users are free to invent their own. Users, in fact, do often invent their own item types and labels, leading to a very heterogeneous collection of data. The result is that Google Base is a very large, self-describing, semi-structured, heterogeneous database. It is self describing because each item has a corresponding schema (item type and attribute names). It is semi-structured and heterogeneous because of the lack of restrictions on names and values. [3]

3. Proposed System

3.1 Authorization

The web consists of hidden data which cannot be, currently, achieved by any single system. Neither traditional web search engines (e.g. Google, Yahoo!) nor desktop search tools (e.g. Google Desktop Search) can achieve that goal. The main reason behind this is that the required objects reside in online deep and hidden repositories that such systems are not authorized to access. Web search engines are only able to crawl and index the surface web while desktop search tools can only access data and files which are stored in the local file systems. The module makes use of a significant feature which is recently introduced by almost all of the major Internet services where they provide standard APIs that allow developing software applications that can read and writes data from their services after providing the credential access information usually a username and password of registered accounts. Using this functionality, the main idea of our system is to let the Web users interact with their Internet services normally while behind the scene the metadata and the pointers (URIs) of the user distributed objects will be extracted.

3.2 Object Extraction

The object extraction layer is designed in a very flexible fashion where a tailored crawler for each Internet service is implemented using the service supported API as an independent plug-in. Thus, integrating any additional service simply requires utilizing its available API functionalities to implement an object Extraction plug-in. The extracted objects from the repositories of the different service repositories are naturally heterogeneous; i.e belongs to different object types e.g. person, mail, document, calendar appointment. Therefore, each object has its own schema metadata information. Due to this schema heterogeneity, we rely on Amazon SimpleDB2, a cloud based key-value data store, as an efficient and flexible solution in this context. Additionally, we use the extracted schema information of the different object types in building special indexes that are used to provide enhanced results for the user's search requests.

3.3 Object Matching

Entity resolution is a well-know problem in data integration systems. In practice, information about the same entity may be distributed across different systems. Therefore, different extracted entities may refer to the same real world object and thus they need to be re-linked together. For example, Peter may have John in his contact list of different services e.g. Face book, LinkedIn, Twitter, Gmail. However, these different contacts need to be treated as a single object as they are all refers to the same person. In our implementation, we used the flexible framework for mapping-based object matching, to achieve this goal.

3.4 Object Linking

The extracted object from the repositories of different services can be usually related to each other through different types of relationships. Automatic and complete discovery of these relationships at once is a very challenging task. Therefore, we are going to follow a pay-as-you-go discovery philosophy where these relationships can be gradually identified and discovered in multiple ways such as: Explicit indication by the end-user to link existing objects with specific type of relationships. Defining some heuristic rules that can suggest the potential existence of specific types of relationships between existing objects e.g. Personal contacts with the similar email address are to be identified as potential work colleagues.

4. Results

The concept of this paper is implemented and different results are shown below, The proposed paper is implemented in Advance Java technology on a Pentium-IV PC with 20 GB hard-disk and 256 MB RAM with apache web server. The propose paper's concepts shows efficient results and has been efficiently tested on different Datasets. The Fig 1, Fig 2, Fig 3 and Fig 4 shows the real time results compared.

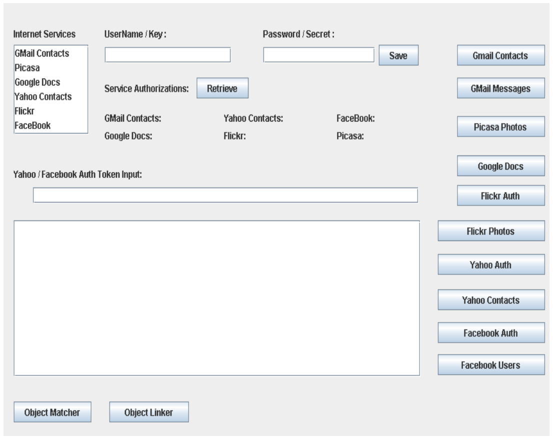
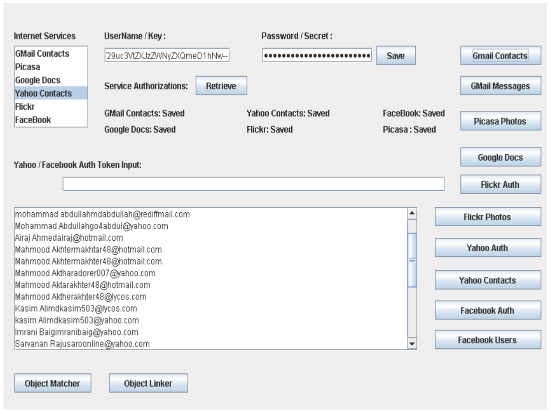


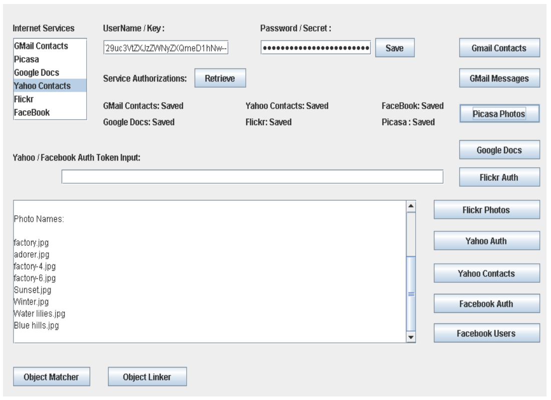
Fig. 1 Proposed system initial interface.



UserName / Key : '29uc3VtZXJzZWNyZXQmeD1hNw--
Password / Secret : ••••••••••••••••••••••••
Service Authorizations:
GMail Contacts: Saved Yahoo Contacts: Saved FaceBook: Saved
Google Docs: Saved Flickr: Saved Picasa : Saved
Yahoo / Facebook Auth Token Input:

```
mohammad abdullahmdabdullah@rediffmail.com
Mohammad Abdullahgo4abdul@yahoo.com
Airaj Ahmedairaj@hotmail.com
Mahmood Akhtermakhtar48@hotmail.com
Mahmood Akhtermakhter48@hotmail.com
Mahmood Aktharadorer007@yahoo.com
Mahmood Aktarakhter48@hotmail.com
Mahmood Aktherakhter48@lycos.com
Kasim Alimdkasim503@lycos.com
kasim Alimdkasim503@yahoo.com
Imrani Baigimranibaig@yahoo.com
Sarvanan Rajusaroonline@yahoo.com
```

Fig. 1 Proposed system performing Object Extraction



UserName / Key : 29uc3VtZXJzZWNyZXQmeD1hNw--

Password / Secret : ••••••••••••••••••••••••

Service Authorizations:

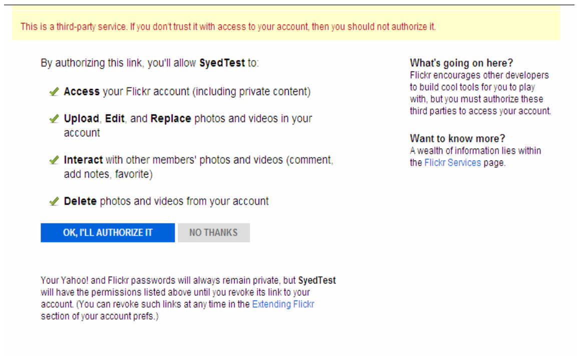
GMail Contacts: Saved
Yahoo Contacts: Saved
FaceBook: Saved
Google Docs: Saved
Flickr: Saved
Picasa : Saved

Yahoo / Facebook Auth Token Input:

Photo Names:

factory.jpg
adorer.jpg
factory-4.jpg
factory-6.jpg
Sunset.jpg
Winter.jpg
Water lilies.jpg
Blue hills.jpg

Fig. 3 Retrieving different types of objects



This is a third-party service. If you don't trust it with access to your account, then you should not authorize it.

By authorizing this link, you'll allow **SyedTest** to:

- ✓ **Access** your Flickr account (including private content)
- ✓ **Upload**, **Edit**, and **Replace** photos and videos in your account
- ✓ **Interact** with other members' photos and videos (comment, add notes, favorite)
- ✓ **Delete** photos and videos from your account

Your Yahoo! and Flickr passwords will always remain private, but **SyedTest** will have the permissions listed above until you revoke its link to your account. (You can revoke such links at any time in the Extending Flickr section of your account prefs.)

What's going on here?
Flickr encourages other developers to build cool tools for you to play with, but you must authorize these third parties to access your account.

Want to know more?
A wealth of information lies within the Flickr Services page.

Fig. 4 Access Authentication from Third Party Objects

5. Conclusion

We described the design of the proposed system which provides the Web users with an integration service for their own Deep Web Data. In practice, the amount of data for each user is growing massively. Therefore, we are currently in the process of designing a more convenient browsing interface for the user objects and the relationships between them using the Mind Maps techniques.

References

[1] H. K¨opcke and E. Rahm. Frameworks for entity matching: A comparison. Data Knowl. Eng., 69(2), 2010.

[2] J. Madhavan, S. Cohen, X. Dong, A. Halevy, S. Jeffery, D. Ko, and C. Yu. Web-Scale Data Integration: You can afford to Pay as You Go. In CIDR, 2007.

[3] J. Novak and A. Ca˜nas. The origins of the concept mapping tool and the continuing evolution of the tool. Information Visualization, 5(3), 2006.

[4] A. Thor and E. Rahm. MOMA - A Mapping-based Object Matching System. In CIDR, 2007.