ABSTRACT

Nowadays, data is available everywhere in every field and the abundance of this rapidly expanding data ensures that all of it cannot be fit in one conventional database management system. In general a dataspace system allows the users to analyze the structures and properties offered by various information sources. This further grants access to users to assess the utility and importance of the information source as a whole and to assess the compatibility with the dataspace services.

This concept works well on paper but not in real world practices. To avoid this problem, this project concentrates on dataspace profiling as the key concept. Dataspace profiling works well on any project that involves unfamiliar dataspaces. This project also implements the detailed requirements for dataspace profilers.
# TABLE OF CONTENTS

Abstract.................................................................................................................................................. ii

Table of contents..................................................................................................................................... iii

List of Figures.......................................................................................................................................... v

List of Tables........................................................................................................................................... vii

1. Background and Rationale .................................................................................................................. 1
   1.1 Background ..................................................................................................................................... 1
   1.2 Project Rationale ............................................................................................................................ 3
   1.3 Related Work.................................................................................................................................. 4
      1.3.1 Profiling of Linked Open Data ................................................................................................. 5
      1.3.2 Quarry Metadata Explorer for CORIE .................................................................................... 8
      1.3.3 Problems of existing methods.................................................................................................. 12
   1.4 Advantages of the developed Dataspace Profiler .......................................................................... 12
   1.5 Project Objectives ......................................................................................................................... 13

2. System Design ..................................................................................................................................... 14
   2.1 Components of the developed system .......................................................................................... 14
      2.1.1 Logical Components of Dataspace Profiler .......................................................................... 15
      2.1.2 Dataspace Customization ...................................................................................................... 22
      2.1.3 Dataspace Services ................................................................................................................ 22
   2.2 Dataspace Profiler ......................................................................................................................... 24

3. System Description ............................................................................................................................. 30

4. Testing and Evaluation ....................................................................................................................... 43
   4.1 Test Cases...................................................................................................................................... 43

5. Conclusion .......................................................................................................................................... 49

6. Bibliography and References ........................................................................................................... 50
LIST OF FIGURES

Figure 1.1 An example Dataspace and its objects ................................................................. 2
Figure 1.2 ProLOD tool interface .......................................................................................... 8
Figure 1.3 CORIE Ocean Forecasting System ...................................................................... 9
Figure 1.4 Quarry Metadata Explorer Interface for CORIE .................................................. 10
Figure 2.1 Sample XML file format for non-item based data .................................................. 16
Figure 2.2 Sample XML file format for item based data ......................................................... 16
Figure 2.3 XML file snapshot that is stores in XML repository............................................... 17
Figure 2.4 System Architecture of Personalized Dataspace Profiler ...................................... 23
Figure 2.5 Design of Dataspace Profiler ............................................................................... 28
Figure 3.1 Dataspace Profiling Homepage .............................................................................. 31
Figure 3.2 Keyword is entered to search either from eBay or Commission Junction ............... 32
Figure 3.3 Web Service data from eBay is stored in an xml file.............................................. 33
Figure 3.4 Web service data from eBay stored as list of xml files .......................................... 34
Figure 3.5 Web Service data from Commission Junction is stored in an object file ................ 35
Figure 3.6 Web service data from Commission Junction stored as list of object files ............. 36
Figure 3.7 Schema populated after Do Profiling button is clicked .......................................... 37
Figure 3.8 Properties of item used by searchResult .............................................................. 38
Figure 3.9 Properties of paginationOutput used by findItemsByKeywordResponse ................ 38
Figure 3.10 Values of property shippingType in shippingInfo ............................................... 39
Figure 3.11 Values of property categoryName in primaryCategory ........................................ 40
Figure 3.12 List of xml files for values of categoryName in primaryCategory ......................... 41
Figure 3.13 List of xml files for values of shipToLocations in shippingInfo ............................ 41
Figure 3.14 Database is erased when Empty Database button is clicked ............................... 42
Figure 4.1 Web service data from eBay is stored in xml file ................................................. 43
Figure 4.2 Web service data from Commission Junction is stored in object file ....................... 44
Figure 4.3 Structure of web service data from eBay and Commission Junction ....................... 45
Figure 4.4 Properties of condition used by item used by searchResult .................................. 45
Figure 4.5 Values of property conditionDisplayName ............................................................ 46
Figure 4.6 Xml files of selected values of conditionDisplayName .......................................... 47
Figure 4.7 Cleared page after Empty Database is clicked .................................................... 48
LIST OF TABLES

Table 1.1 Correlation Coefficient of media cluster ................................................................. 7
Table 1.2 Inverse Predicates ordered by occurrences............................................................... 7
Table 2.1 Difference between XML files and OBJ files with an example............................... 16
1. BACKGROUND and RATIONALE

A dataspace system is a multi-model data management system that manages data sourced from a variety of local or external sources. These sources may actually take the form of database management systems for a specific data model. A dataspace is a container for domain specific data. Data in data spaces are linked across spaces and domains to enhance the meaning of internal data. “A data space should be fully supportive of data portability”. [Dong, Halevy 2007]. This means that the object in a data space should be movable and have the ability to be referenced using an identifier such as the Uniform Resource Identifier. The necessity of dataspace systems in recent days comes from the challenges posed by information management schemes. These challenges come from various companies or organizations that depend on the heavy usage of varied data sources.

1.1 Background

The concept of dataspaces was proposed in a 2005 paper by Franklin, Halevy and Maier as a new data management paradigm to help unify the diverse efforts on flexible data models, data cleaning, and data integration being pursued by the database community. [Franklin, Halevy, Maier 2005].

The main idea behind dataspaces is to expand the scope of database technology by moving away from the schema-first nature of traditional data integration techniques and instead, support a spectrum of degrees of structure, schema conformance, entity resolution, quality and so on.
Dataspaces offer an initial, lightweight set of useful services over a group of data sources, allowing users, administrators, or programs to improve the semantic relationships between the data in an incremental and iterative manner. This “pay-as-you-go” philosophy is at the heart of the dataspaces approach. The main difference between dataspaces and other information-integration systems is that dataspace systems provide quality answers even before semantic mappings are provided to the system.

The main advantage of dataspace systems is that they offer various services relating to data without requiring the most important semantic integration that should be done first in other data integration systems. [Franklin, Halevy, and Maier 2008]. Users of the dataspace systems will have the additional leniency of deciding when and where to identify the semantic relationships that are important for data integration. Figure 1.1 depicts the relationships among objects and dataspaces.

![Dataspaces Diagram](image)

Figure 1.1 An Example Dataspaces and its Objects[Franklin, Halevy, and Maier 2008].
1.2  Project Rationale

The necessity of the dataspace comes from the fact that the corporate world is in need of constantly updating obtainable information sources to new uses. When these information sources come from multiple sources, they will be integrated using data integration techniques. But these sources need to be carefully validated before integrating them to ensure uniform capabilities over the integrated data.

One main alternative to the data integration method is the dataspace approach. Dataspace approach ensures that all the information sources within a standard enterprise or company is broadly included in the final dataspace. The advantage of dataspacing is that there is no need to include all the capabilities at the beginning. The capabilities can be gradually increased over the time. But the main problem with dataspacing is that no one will have all the extensive knowledge of all the sources within that dataspace. This problem can be solved using “dataspace profiling”. “Dataspace profiling can be defined as the process of analyzing a dataspace to determine its structure and internal relationships”. [Marshall 2006].

There are main advantages of structural profiling of the dataspace. They include but are not limited to the simplified understanding of the data sources. Another advantage is the connection among the information sources to evaluate the sustainability and suitability of a particular source for a given task or application. The third advantage is having the ability to authenticate the conditions that are base criteria to add a certain capability to more than one information data sources. The fourth advantage is dataspace customization. Dataspace customization enables the option to define one’s own data fields to fit the needs of the system. “Data sources often need to be massaged in various
ways to be suitable for a particular use”. [Howe, Maier, Rayner, Rucker 2008]. In order to complete dataspace profiling, the verification of latent structural characteristics and regularities has to be done with the priority level of one.

The necessity of dataspace is visible in some broad categories like

- Enterprise and government data management
- Digital libraries
- Smart homes
- Personal information management.

As previously stated, the main advantage of dataspace over data integration comes from the fact that dataspace support platforms do not require full semantic integration of the sources in order to provide the required services.

1.3 Related Work

The most acute information management challenges today arise from organizations relying on large number of diverse, inter-related data sources but having no means of managing them in a convenient, integrated and principled fashion. The challenges stem from enterprise and government data management, digital libraries, smart homes and personal information management. Dataspaces are introduced as a new abstraction for data management in such scenarios.
1.3.1 Profiling of Linked Open Data

Linked Open Data (LOD) [Bohm 2010] consists of data that comes from various open data sources such as Wikipedia or from specific topic-based projects. This data is provided as RDF triples covering many domains. Due to this, there is no complete view of the structure of the information and also the data values. To understand this open linked data, we need to understand each single dataset and have an overview of all the data available. As LOD has huge set of open heterogeneous data, classical profiling techniques are not sufficient to deal with.

Profiling of LOD is implemented using iterative and interactive technique of allowing the user to divide the data into groups. Now, review the results at group-level and then make a decision of re-grouping to refine the result of profiling. To implement profiling of LOD, a prototype called ProLOD was developed. [Bohm 2010]. The infobox without ontology mappings and DBpedia, which is a subset of Wikipedia with 34.2 million triples are considered in ProLOD.

Profiling of Linked Open Data (LOD) is done in three steps:

a) Clustering and Labeling

b) Schema Discovery

c) Gather Statistics of Data types and Patterns

a) Clustering and Labeling

Clustering divides the large and heterogeneous data sets into data sub-sets based on the correlation of semantics. The prototype ProLOD does profiling based on a pre-calculated hierarchical clustering of dataset which can be changed dynamically at runtime. The entire dataset is represented by the root cluster and each cluster is further
divided into sub-clusters. Clustering is done based on three strategies. (1) Initial clustering is done in pre-processing phase of profiling. (2) Clustering is done based on the similarity of the schema i.e., only the predicates but not the actual values. (3) Sub-dividing of clusters is decided based on the internal dissimilarity threshold. A mean schema ‘n’ is calculated for each cluster which determines the average schema size of all entities of cluster.

Labeling is done to give more details about the cluster. As the entities are correlated by semantics, most of the entities has description in text form as rdfs:comment or some similar property. If a textual description is not found, the mean schema calculated for the cluster can be used as label.

b) Schema Discovery

To provide basic understanding of the structural information of the data, schema discovery is used beyond the mean schema. Schema discovery includes finding the actual schema, finding equivalent predicates and also finding poor attributes which do not have useful information for data sets and analyze association rules, inverse relations and foreign key relations.

Schema discovery is done by detecting the positive and negative association rules which determines the dependencies among the predicates. The positive dependencies in a cluster is shown using Apriori Algorithm which holds a desired confidence and correlation coefficient $\ell$. For media cluster which has book as entity has occurrences author, isbn, genre whose predicate occurrences are as shown in Table 1.2
The negative dependencies occur when the predicates have equivalent semantic meanings and never appear together. A semantic connection is said to be established when a subject $A$ holds subject $B$ over a predicate $P$ say $A \rightarrow B$ and a stronger interconnection is established when links $A \rightarrow B$ and $B \rightarrow A$ with inverse predicates say $P$ and $Q$.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Confidence</th>
<th>Correlation Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>genre, isbn ⇒ author</td>
<td>0.99</td>
<td>0.67</td>
</tr>
<tr>
<td>isbn ⇒ author</td>
<td>0.92</td>
<td>0.66</td>
</tr>
<tr>
<td>isbn ⇒ author, genre</td>
<td>0.83</td>
<td>0.66</td>
</tr>
<tr>
<td>author, genre ⇒ isbn</td>
<td>0.70</td>
<td>0.66</td>
</tr>
<tr>
<td>author ⇒ isbn</td>
<td>0.64</td>
<td>0.66</td>
</tr>
<tr>
<td>author ⇒ genre, isbn</td>
<td>0.58</td>
<td>0.67</td>
</tr>
</tbody>
</table>

Table 1.1 Correlation Coefficient of media cluster [Bohm 2010]

c) Gather Statistics of Data Types and Patterns

This is done by analyzing the RDF triples. The articles in Wikipedia are modified by many users using different syntax to represent the same fact. Due to this, more number of predicates is used when it is not desired. Profiling helps in detecting such predicates and find valid unique predicates avoiding multiple misuses.

Numeric data types are divided into ranges where as non-numeric data types are determined by normalized patterns by identifying the sequences of same character. Based on pattern and data type, ProLOD determine the actual data values.
The ProLOD tool appears as shown in Figure 1.2. The interface has cluster tree view on left and detailed view on right. The cluster can be further explored and can be used to select desired one. The facts and statistics of the selected cluster can be seen in detailed view and is used to perform schema discovery.

1.3.2 Quarry Metadata Explorer for CORIE

Quarry Metadata Explorer (QME) [Howe 2008] is developed to browse and extract the individual data sets from the metadata available from multiple domains. Quarry does not have schema information of the domains. The properties of the resources are identified using Quarry and form a multi-column table for all the common properties of the resources. Quarry provides iterative environment to search for resources based on property-value conditions. It does not have information of how the properties are arranged in groups at backend. The set of properties are identified for a collection of
resources and all the possible values for each property listed are identified for that collection of resources.

Two functions are assumed to explain the Quarry Metadata Explorer. The $Properties$ function returns all the properties corresponding to a set of resources and the $Values$ function returns all the values corresponding to those properties over same set of resources. The application developed provides the user to select a particular property from the resources that are filtered down and values are obtained for a selected property then choose a value of that property to further display the list of properties corresponding to that value.

The CORIE Environmental Observation and Forecasting System combines the ocean circulation models data obtained from an array of fixed and mobile observations. This raw data is transformed into useful data sets which involve intermediate data sets, log files, configuration files and images which becomes difficult for the users to understand as shown in Figure 1.3. This made to develop the (resource, property, value) triples from the data using Quarry and see the results using API.

![Example: Ocean Circulation Forecasting System](image)

Figure 1.3 CORIE Ocean Forecasting System [Halevy 2008]
The triples are loaded into Quarry backend; they are saved in a table with three string attributes resource, property and value. Signature Manager is used to identify the set of properties $S(r)$ known as Signature of resource $r$. Each resource is viewed as (property, value) pairs and each property is viewed with atleast one tuple for property as (value). The strategy used in quarrying is $|\text{Signatures}| < |\text{Resources}|$.

Quarry Metadata Explorer for CORIE is as shown in Figure 1.4

![Quarry Metadata Explorer Interface for CORIE](image)

**Figure 1.4 Quarry Metadata Explorer Interface for CORIE [Howe 2008]**

In this interface, four functions are used: Describe, Properties, Values, Traverse and Filter. Describe returns the (property, value) pairs for the resource desired. Properties return the list of properties for the resources satisfying a given condition. Values return the values of the property selected for the resources satisfying given condition. Traverse is used to identify the resource. Filter is used to get the subset of resources satisfying given condition.
Consider $R$ as set of resources, $C$ as condition, $r$ as subset of resources, $p$ as property and $v$ as value of property.

Filter: \{Condition\} $\rightarrow$ \{Resource\} $\rightarrow$ \{Resource\}

Properties: Path $\rightarrow$ \{Condition\} $\rightarrow$ \{Property\}

Values: Path $\rightarrow$ \{Condition\} $\rightarrow$ Property $\rightarrow$ \{Value\}

Traverse: Path $\rightarrow$ \{Condition\} $\rightarrow$ \{Resource\}

As in Figure 1.4, when the property variable is selected, the values SAL, sal, Salinity, U, V, Velocity, VRES12 are obtained. When VRES12 is selected, the properties of all the files with variable=VRES12 are listed. When plottype property is selected, the values of plottype for files with variable=VRES12 are listed.

These properties and values are obtained by executing a set of SQL statements on the signature tables. The properties that are processed during SQL statements are only those mentioned in the query to reduce the overhead.

Properties($P$, $C$) = $PropQuery(Traverse(P, C))$

Values($P$, $C$) = $TravQuery((C, p) : P)$

Traverse($\emptyset$, $C$) = $ValuQuery(\{'plottype\'}, C)$

Traverse($P$, $C$) = $TravQuery((C, \{'plottype\'} : P)$

Quarry helps in profiling the data from various domains without apriori information of schema which helps to retrieve the data even without the knowledge of actual schema of the datasets.
1.3.3 Problems of existing methods

In existing methods, relational databases were used as data sources. Also, functional dependencies exist between the attributes of the data sources. Due to these functional dependencies, when an update has been made to any of the data source, these functional dependencies will be violated. In such cases, it can be handled only by human intervention rather than hoping an automated system will find and solve it. Due to dependencies among the attributes in a relational database, the customization of data sources is done based on the derived dependency relation. This involves using join statement a lot.

Based on the attributes shared among the relational databases, customization of data sources into a single data source based on one attribute set to the corresponding values in another attribute set may not be able to connect all tuples due to some fields that are missing.

1.4 Advantages of the developed dataspace profiler

In developed dataspace profiler online tool, the data sources used are XML repositories and Object files which are of two different file formats unlike relational databases. Functional dependency problem does not exist in this system because of different of file formats.

As functional dependencies do not exist, even when an update to any data source is made, profiling is done dynamically without any issues and also human intervention is not needed.
1.5  **Project Objectives**

The objectives of this project are to:

a) Develop a stand-alone tool for dataspaces.

b) Identify services that will be made available via the dataspace tool.

c) Create user defined dataspace customization.

d) Allow users to define custom databases for dataspaceing.
2. SYSTEM DESIGN

The previous sections discussed several existing solutions for dataspaces in general. The various database profiling tools that were developed are ArkZone and Travel XRay tools [Arkenford, 2009]. The difference between dataspace profiling and database profiling is that most of the database profiling tools available out there picks up a starting point of relational data with a domain specific schema where as for the developed dataspace profiling online tool the starting point may not have a deductive schema. This section explains how the system design is done in order to achieve efficient profiling of the dataspaces. The main essence of dataspace profiling is captured in two kinds of data sources for this project in the form of XML files and OBJ files generated from eBay and commission junction data sources respectively. The details description of the data sources is explained in section 2.1.1. The content is organized as follows: section 2.1 describes major components of the developed system.

2.1 Components of the Developed System

The various components of the implemented dataspace profiling online tool are described in this section. The various components include but not limited to logical components of the dataspaces, dataspace services and dataspace systems. The detailed descriptions of these components are given below.
2.1.1 Logical Components of the Dataspace Profiler

The design of the dataspace that was developed in this project contains all of the data that is pertinent to any selected organization regardless of the organization’s format and location. The organizations that were chosen are Commission junction and eBay data sources. The entire sets of logical components are called as participants. The participants in our dataspace design are the individual data sources.

The data sources that are considered are:

- Commission Junction public data source [XML/OBJ data source]
- eBay public data source [XML data source]

For example, here, let us compare Ebay_HP Pavilion1304526236.xml and CJ_HP Pavilion1304526278.obj files.

<table>
<thead>
<tr>
<th>XML files</th>
<th>OBJ files</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML files that are used in the project are generated through the web service called XML Schema Definition (XSD).</td>
<td>OBJ files that are used in this project are generated through the web service called Web Service Definition Language (WSDL).</td>
</tr>
<tr>
<td>XSD defines a schema which is a definition of how an XML document can be structured.</td>
<td>WSDL shows which operations are available and how data should be structured to send to those operations.</td>
</tr>
<tr>
<td>In XML file, the identification number allotted to the returned search string was assigned as an attribute to the tag “itemId”.</td>
<td>Where as in OBJ file, the identification number allotted to the returned search string was assigned as an attribute to the tag “adId”.</td>
</tr>
<tr>
<td><strong>Example</strong> &lt;itemId&gt;150548007826&lt;/itemId&gt;</td>
<td><strong>Example</strong> : adId : 10273676</td>
</tr>
</tbody>
</table>
In XML file, the name of what was being offered was assigned as an attribute to the tag “title”.

**Example**: `<title> HP Pavilion ze4300 speakers </title>`

In XML file, the URL of the product being offered was assigned as an attribute to the tag “viewItemURL”.


In XML file, the currency in which the product is being sold was assigned as an attribute to the tag “currentPrice”.

**Example**: `<currentPrice currencyId='USD'>24.99</currentPrice>`

In OBJ file, the name of what was being offered was assigned as an attribute to the tag “description”.

**Example**: `description : HP PAVILION DV1000/ZE2000/DV40`

In OBJ file, the URL of the product being offered was assigned as an attribute to the tag “buyUrl”.

**Example**: `buyUrl : http://www.pcmall.com/p/product~dpno~8389634~pdp.gebgbjj;s=essionid=AB1D92E22FB1EB1AB2EC22E3F9445439?source=zwb12165`

In OBJ file, currency in which the product is being sold was assigned as an attribute to the tag “price”.

**Example**: `price : 77.9899`

Table 2.1 Difference between XML and OBJ files with example

Commission Junction public data source was selected because it is one of the few public domain data sources that are available copyright free. Commission Junction data source (CJ) falls under Affiliate data feeds category. Affiliate data feeds are a great source of free content for public’s use. CJ offers a public API in the form of Web
Services Description Language (WSDL). WSDL is an XML-based language that offers a standard model for describing web services.

When users try to acquire data from web service and select CJ as the data source, it parses through the available data and saves the data in the form of Object files. The OBJ file format is a text file format.

eBay public data source is also another widely available public domain data sources. When users try to acquire data from web service and select eBay as their choice of data source, the profiler parses through the available data and returns the desired results and saves the result in the form of XML files.

When “Acquire from Web Service” URL link is clicked, it takes to another php page “form.php”, where the option to gather data from eBay or CJ is available. When a keyword was entered in the text box area and suppose “Get from eBay” button was used, it loads a function called getProduct().

getProduct() first checks for the length of the search keyword that was entered. If the length validation is good, it calls one php file called “SOAPhandler.php”. This php file contains a class that will call the required functions to gather data and convert the data into XML files. The initial data will be gathered from eBay public server location http://svcs.ebay.com/services/. This public server can be accessed only through web service calling. In the above mentioned php file, a constructor class function initializes the process of generating XML files.

Constructor function calls initializes a function called buildURLArray() that will generate an indexed URL snippet from the array of item filters. buildURLArray() first iterates through each filter in the array and if any of the filters were found, it further
iterates through each key in the filter and indexes the key for each value. This iteration is
done for each of the filter and each key in those filters until the end of the filters.

Once the buildURLArray () function completes the task, eBay API returns the
data in the form of XML document. This document was handled by storing the data in
“$apicall” variable and calling simplexml_load_file () function. simplexml_load_file ()
function reads the contents of the XML file and interprets the XML file into an object.
The parameters passed to this function is “$apicall”.

In the similar way, when a keyword was entered in the text box area and suppose
“Get from Commission Junction” button was used, it loads a function called getProduct2
().getProduct2 () first checks for the length of the search keyword that was entered. If the
length validation is good, it calls one php file called “SOAPhandler2.php”. Just like the
eBay counterpart, this php file also contains a class “class CJ” that calls all the required
functions to gather raw data and convert that data into OBJ files using wsdl Web service
[SOAP method].

The initial data will be gathered from eBay public server location
https://product.api.cj.com/wsdl/. This public server can only be called by using wsdl web
service that can be invoked by SOAP protocol. Function CJ () triggers the process of
gathering data from the above mentioned CJ public server location. CJ () function first
checks whether the search keyword related data is available in the public server by
calling file_get_contents () function. If this triggers passes through, the web service
returns an error message that “NO WSDL found at” a particular server location.
If the above test fails and the entered search keyword returns some data, an instance of SoapClient () function will be initialized. This instance traces through the public server URL and returns the eligible product data in the form of objects. Once the objects were received, parseObject () parses through the obtained objects and calls simplexml_load_file (). The sample XML file format and its specifications are much better explained in the below showed format. If the XML file contains Non Item-based data, it is saved in the following format as shown in Figure 2.1.

```
<correction>
  <reason>OTHER</reason>
  <ActionID/>
  <oid>t6557</oid>
</correction>
```

Figure 2.1 Sample XML file format for Non-Item based data.

If the XML file contains Item-based data, it is saved in the following format as shown in Figure 2.2.
Figure 2.2 Sample XML file format for Item based data.

Suppose if the search key word is “Dataspace” and the users enter the key word to acquire data from web service, in this case eBay, the result will be saved in a XML file with the search string embedded in the title of the file as shown in Figure 2.3.

```xml
<findItemsByKeywordsResponse>
  <ack>SUCCESS</ack>
  <version>1.10.0</version>
  <searchResult count="0">
    <paginationOutput>
      <pageNumber>0</pageNumber>
      <entriesPerPage>10</entriesPerPage>
      <totalPages>0</totalPages>
      <totalEntries>0</totalEntries>
    </paginationOutput>
    <itemSearchURL>
      http://shop.ebay.com/i.html?LH_BIN=1&LH_IncludeSIF=1&LH_FS=1&nkw=Dataspace&ddo=1&i ingres=.10&mPr accumulation=1&pgn=1&nSiteID=25
    </itemSearchURL>
  </searchResult>
</findItemsByKeywordsResponse>
```

Figure 2.3 XML file snapshot that was stored in XML repository.
In the above image, the searchResult count property shows value as zero indicating the profiler found no entries/data with the name of “dataspace” in eBay data source. As described earlier, the participants are the building blocks of the dataspace system. One of the important participants in the developed dataspace profiler is “Very Structured”.

The examples of Very structured being Commission junction and eBay data sources that are stored as OBJ files and XML files in the repository for the purpose of better profiling. Functional dependency problem does not exist in this system because of different of file formats. As functional dependencies do not exist, even when an update to any data source is made, profiling is done dynamically without any issues and also human intervention is not needed.

As a rule of thumb, dataspace profiler does not intend to make any deductive assumptions about domain specific schema. Also, dataspace profiler design was able to model any kind of relationship between two or more participants. The dataspace profiler recognizes common property patterns for the XML data resources & commission junction data resources automatically and thus provides easier access to the domain schema.

Another important feature of this developed system is that the data sources can overlap on each other. The best example is coming to a common schema from two or more different data sources, in our example commission junction data source, eBay public data source. The goal of dataspace overlapping must stick to provide base functionality over all data sources, regardless of how integrated they are.
2.1.2 Dataspace Customization

In this project, dataspace customization on eBay and Commission Junction is done using a keyword search. When a user wants to look up for a desired item in eBay or Commission Junction, a matching keyword is entered to search and the results are stored as XML files and Object files. The profiling is done on these files saved in the repository and user can look up for the desired item based on the desired property.

For example, when a user want to look up for an ‘Acer Keyboard’, then user is allowed to enter any related keyword say “Acer” and the results obtained from eBay are stored in XML file and saved in the repository. After profiling, by selecting categoryName from property primaryCategory, which displays the list of different categories from which “Keyboards” can be selected to see the desired results from the saved XML file without having to look at the whole XML file.

2.1.3 Dataspace Services

The dataspace services allow many different modes of interaction. This interaction allows the application of different services on various different types of content. The main advantage of dataspace systems is that they offer various services relating to data without requiring the most important semantic integration that should be done first in other data integration systems. Semantic integration is the process of interrelating information from diverse sources, for example calendars and to do lists. The developed dataspace profiler online tool uniquely identifies each property and value associated with the different data sources.

The uses that can be obtained from this dataspace profiler are manifold.
i. Better understanding of a data source.

ii. Understanding the interlinking and connection among the data sources.

iii. Dataspace customization.

This interaction allows the application of different services on various different types of content. The main dataspace services offered by the developed “Dataspace Profiler” online tool are described in Figure 2.4.

Figure 2.4 System Architecture of Personalized Dataspace Profiler

In the above architecture, the data sources used are eBay and Commission Junction. The web service data from eBay is stored in the form of XML files and the web service data from Commission Junction is stored in the form of Object files. These files are saved into a MySQL Database and when profiling is done, the common schema from both the XML and Object files is extracted and displayed. From the properties displayed, when a desired property is selected, a set of SQL statements are executed over MySQL database and the properties and/or values that are associated with the selected property
are displayed. Once the values are displayed, the list of XML files corresponding to data values are displayed which provides easier access to data for the user. The architecture is explained in detail in section 2.2.

2.2 Dataspace Profiler

The dataspace profiler online tool contains two online data sources and one other user added data source. The two online sources, as mentioned earlier, are Commission Junction and eBay. The main purpose of this dataspace profiler online tool is to identify, if property $P_{r1}$ over a set of resources has additional properties $P_{r2}$ & $P_{r3}$ or for every property over the same set of resources satisfying the given condition has values $V_{u1}$ & $V_{u2}$.

When the "Do Profiling" button was clicked, getResource () is used to get the set of selected resources. getProperty () propagates over the set of unique properties that describes the overall schema of the set of selected data sources. When a property is selected, getValue () propagates over the set of unique values that are associated with a particular property that were rendered using getProperty (). This process is done using iterative and interactive methodologies.

Dataspace profiler is used in this way. First, the data is acquired from web services. The raw data from eBay was acquired from web service and rendered into a XML files using Representational State Transfer (REST) method. The raw data was first converted into intermediatery datasets and then finally generated XML files. In similar way, raw data from commission junction was acquired from web service and rendered into OBJ files using SOAP method.
The advantages of REST method are that in REST, component interactions are much more scalable and they can be deployed independently. REST method also allows any HTTP client to talk with any HTTP server with no further configuration needed. Also, REST ensures the use of a uniform interface between components [Ajaxonomy, 2008].

Simple Object Access Protocol (SOAP) is an application protocol with its own set of application protocol semantics. SOAP messages are typically transferred using HTTP protocol. SOAP treats HTTP as a lower-level communication protocol and its specifications define bindings of the SOAP application protocol semantics to other application protocol semantics such as HTTP, SMTP and WebSphere MQ [Ajaxonomy, 2008].

The descriptive triples (Entity, Property, Value) from the data sources like XML and OBJ files were extracted using profiling scripts. The triples that were obtained are stored in a table in the form of three string attributes. When we use getResource(), getProperty() and getValue() on these string attributes, a set of SQL statements are processed to extract the list of properties for a selected set of resources and list of values for a selected property over the same set of resources.

Assume,

\[ S = \{ (R_c, P_r, V_u) \} \]

where \( R_c \) as set of resources, \( P_r \) as property and \( V_u \) as value.

In addition, we define the following symbols:

\( C_n \) as condition which is a conjunctive query of property and value,

\( R_s \) as set of selected resources,
$L_d$ as load that gives the set of properties and set of values

$P_s$ as set of properties for selected resources $R_s$

$V_s$ as set of values for selected property

The dataspace profiler online tool implements the following three functions.

$\text{getResource} () : : \{ C_n \} \rightarrow \{ R_c \} \rightarrow \{ R_s \}$

When a condition $C_n$ is applied over a set of available resources $R_c$, selected set of resources, $R_s$, were obtained as per the condition.

$\text{getProperty} () : : L_d \rightarrow \{ C_n \} \rightarrow \{ P_r \}$

Once, a set of selected resources are generated, condition $C_n$ is applied to get the set of properties for the selected resource.

$\text{getValue} () : : L_d \rightarrow \{ C_n \} \rightarrow \{ P_r \} \rightarrow \{ V_n \}$

Once the set of selected properties are loaded for selected resources, and a condition is applied for a desired property, the value is generated corresponding to the selected property.

Now, the profiler online tool can be described as,

$\text{getResource} (C_n, R_c) = \{ R_s \ | \ (P_s, V_s) \in Cn , (R_c, P_r, V_n) \in S \}$;

getResources() is used to get the set of selected resources when a property-value pair condition is applied where the set of triples (resource, property, value) belongs to set of triples of all the resources.

$\text{getProperty} (L_d, C_n, R_s) = \{ P_s \ | \ (R_c, P_r) \in S \}$;
getProperty() is used to obtain desired set of properties from a set of selected resources when a condition is applied on resource property pairs where resource, property pair belong to set of all triples.

\[
\text{getValue} \ (L_d, C_n, P_s) = \{V_s \mid (R_c, P_r, V_u) \in S\};
\]

getValue() is used to get the desired value for the selected set of properties over the selected set of resources when a condition is applied on set of properties where (resource, property, value) triples belong to entire set of triples of resources.

The profiling ensures schema that contains most common list of properties. When getResource () is used to get the set of selected resources. getProperty () is used to get the set of properties like findItemsByKeywordsResponse, item, primaryCategory, shippingInfo, sellingStatus, listingInfo, paginationOutput et cetera can be observable. If we select primaryCategory property, getValue () is used to return the list of values. we see properties categoryId and categoryName which are related to primaryCategory property. If we select categoryName property, it returns a list of values corresponding to category name.

If a value is selected from the list of values, it displays a list of XML files corresponding to that value as shown in Figure 2.5.
This is how profiling is implemented in this online tool. When profiling is done, the following patterns can be observed.

- For \textit{item} = \textit{country}, all \textit{searchResult} values are in upper case listed in column (CN, US, EC, JP)
- For \textit{item} = \textit{location}, all \textit{searchResult} values have initial capitals (Austin, Seattle)
- For \textit{item} = \textit{globalId}, all \textit{searchResult} values have prefix EBAY (EBAY-US, EBAY-CN)
- The case for \textit{item} = \textit{country} is similar to \textit{item} = \textit{shipToLocations} but with some values listed with initial capitals (US, Worldwide, CN)
- For \textit{item} = \textit{listingInfo}, led to properties without any values.
If we observe both the XML files and Object files, properties of files where () are findItemsByKeywordsResponse, item, primaryCategory, shippingInfo, sellingStatus, listingInfo, condition, paginationOutput, secondaryCategory, out, Product. The properties of files where (ack = success) are itemSearchURL, paginationOutput, searchResult, timestamp, version.
3. SYSTEM DESCRIPTION

The main emphasis of this project is on developing an online tool called “Dataspace Profiler”. Dataspace profiler uses three main data sources to explain dataspace profiling. They are

1. eBay public data source
2. Commission Junction data source

The commission junction data source and eBay public data source were accessed using SOAP method and REST method respectively. Simple Object Access Protocol [SOAP] was used for commission junction data source because it is extensible. SOAP is a minimal set of conventions that uses XML and HTTP to invoke code. While SOAP does not mandate the use of XML schemas, it was designed to allow XML schemas act as its type description language. REST was used for eBay because it is not limited to HTTP protocol and the emphasis here is on simple point-to-point communication over HTTP using plain old XML.

At the startup of the Dataspace profiler online tool, it looks like as shown in Figure 3.1.
Figure 3.1 shows the home page for Dataspace Profiler online tool and all the links that are used in the project. When “Do Profiling” button is used, it will perform the dataspace profiling on the XML files residing in XML repository and Object files that are generated from commission junction data source.

When a desired keyword is to be searched, “Acquire from Web Service” is used. It allows the user to enter the keyword and provides two different data sources either eBay or Commission Junction from which data can be searched as shown in Figure 3.2.
Figure 3.2 Page which shows where the keyword is to entered to search either from eBay or Commission Junction

When the user enters a keyword and selects “eBay” as data source, the data is extracted from the source using REST method and the web service data extracted is saved as XML file as shown in the Figure 3.3.
Figure 3.3 The result from eBay is stored in an XML file

All the XML files that are saved can be seen in the repository using “XML Repository” as shown in Figure 3.4.
When the user enters a keyword and selects “Commission Junction” as data source, the data is extracted from the source using SOAP method and the web service data extracted is saved as object file as shown in the Figure 3.5.
Figure 3.5 The result from commission junction is stored in an object file

All the object files that are saved can be seen in the repository using “Object Files” as shown in Figure 3.6.
The dataspace profiling is done using “Do Profiling” or “Dataspace Profile”. When the button “Do Profiling” is used, it performs dataspace profiling on web service data resulted from the eBay and Commission Junction data sources which are saved as XML files and Object files respectively. The combined schema of the web service data from both data sources is populated as shown in Figure 3.7.

When the button “Dataspace Profile” is used, it populates the previous existing profiling data without dynamically profiling the newly added web service data where as “Do Profiling” dynamically performs dataspace profiling every time it is used including newly added data.

Figure 3.6 Search results from commission junction stored as list of object files
Figure 3.7 Schema populated after Do Profiling button is clicked

The list that is populated has properties of the entire data where in some of the properties have relation with other properties as shown in Figure 3.8 and Figure 3.9.

The relation between the properties is shown using “Used by” at the bottom of the list of properties.
Figure 3.8 Properties of ‘item’ Used by ‘searchResult’

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>autoPay</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>charityId</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>compatibility</td>
<td>OBJECT</td>
</tr>
<tr>
<td>condition</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>country</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>galleryPictureURL</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>galleryURL</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>globalId</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>itemID</td>
<td>OBJECT</td>
</tr>
<tr>
<td>listingInfo</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>location</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>paymentMethod</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>postalCode</td>
<td>OBJECT</td>
</tr>
<tr>
<td>primaryCategory</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>productId</td>
<td>OBJECT</td>
</tr>
<tr>
<td>returnsAccepted</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>secondaryCategory</td>
<td>OBJECT</td>
</tr>
<tr>
<td>sellingStatus</td>
<td>OBJECT</td>
</tr>
<tr>
<td>shippingInfo</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>subtitle</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>title</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>viewItemURL</td>
<td>VARCHAR(0)</td>
</tr>
</tbody>
</table>

Used by:
- searchResult

Figure 3.9 Properties of ‘paginationOutput’ Used by ‘findItemsByKeyword’

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>entriesPerPage</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>pageNumber</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>totalPages</td>
<td>VARCHAR(0)</td>
</tr>
<tr>
<td>totalPages</td>
<td>VARCHAR(0)</td>
</tr>
</tbody>
</table>

Used by:
- findItemsByKeywordsResponse

<table>
<thead>
<tr>
<th>Field</th>
<th>TABLENAME</th>
<th>FIELDNUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>paginationOutput</td>
<td>findItemsByKeywordsResponse</td>
<td>6</td>
</tr>
<tr>
<td>item</td>
<td>item</td>
<td>22</td>
</tr>
<tr>
<td>primaryCategory</td>
<td>primaryCategory</td>
<td>2</td>
</tr>
<tr>
<td>shippingInfo</td>
<td>shippingInfo</td>
<td>6</td>
</tr>
<tr>
<td>sellingStatus</td>
<td>sellingStatus</td>
<td>5</td>
</tr>
<tr>
<td>listingInfo</td>
<td>listingInfo</td>
<td>8</td>
</tr>
<tr>
<td>title</td>
<td>title</td>
<td>2</td>
</tr>
<tr>
<td>condition</td>
<td>paginationOutput</td>
<td>4</td>
</tr>
<tr>
<td>secondaryCategory</td>
<td>secondaryCategory</td>
<td>2</td>
</tr>
<tr>
<td>out</td>
<td>out</td>
<td>4</td>
</tr>
<tr>
<td>Product</td>
<td>Product</td>
<td>18</td>
</tr>
</tbody>
</table>
When a particular property is selected, it gives the list of values associated with that property from the web service data of both data sources as shown in Figure 3.10 and Figure 3.11.

Figure 3.10 Values of property ‘shippingType’ in ‘shippingInfo’
Figure 3.11 Values of property ‘categoryName’ in ‘primary Category’

When a value is selected and clicked, the list of XML files from web service data are displayed below the value. The XML files are those from which the values have been extracted as shown in Figure 3.12 and Figure 3.13.
Figure 3.12 List of XML files for the values of ‘categoryName’ in ‘primaryCategory’

Figure 3.13 List of XML files for the values of ‘shipToLocations’ in ‘shippingInfo’
When “Empty Database” button has been clicked, it erases and empties the database that is used to store the results of the profiling. This button has been added to prove that profiling is done every time dynamically.

Figure 3.14 When ‘Empty Database’ button is clicked, the database is erased.
4. TESTING AND EVALUATION

4.1 Test Cases

Following are the test cases that use detailed procedures to test the working and features of the project.

Test Case 1:

Test whether the keyword entered is searched from the selected source and web service data is stored in the files as per the design.

Output:

When the desired keyword to be searched is entered and eBay is selected to search, the web service data is stored in the form of XML file as shown in Figure 4.1

Figure 4.1 Web service data from eBay is stored in xml file
When the desired keyword to be searched is entered and Commission Junction is selected to search, the web service data is stored in the form of object file and as shown in Figure 4.2.

Test Case 2:

Test whether the “Do Profiling” works and generates the structure of the web service data that is stored and also test whether the properties are displayed as per the design and relation between properties is generated.

Output:

When “Do Profiling” is clicked, the structure of the web service data is displayed as shown in Figure 4.3, properties and relation between those properties are also displayed as Figure 4.4.
Figure 4.3 Structure of the web service data from eBay and Commission Junction

Figure 4.4 Properties of ‘condition’ which is used by property ‘item’ and properties of ‘item’ which is used by property ‘searchResult’
Test Case 3:

Test whether the values are displayed for the properties from the web service data stored and whether the XML files or Object files are listed from which the data has been extracted, when a particular value is selected.

Output:

When a particular property is selected, the values are displayed related to that property extracting from the web service data stored, which is based on search from both eBay and Commission Junction. The results are as shown in Figure 4.5. When a particular value is selected, it displays all the XML files and Object files from which the data is extracted as shown in Figure 4.6.

Figure 4.5 Values of the property ‘conditionDisplayName’
Figure 4.6 XML files of selected values of ‘conditionDisplayName’ are extracted

Test Case 4:

Test whether “Empty Database” works and entire structure of the web service data is emptied.

Output:

When ‘Empty Database’ is clicked, it erases and empties the database that is used to store the results of the profiling as shown in Figure 4.7
Figure 4.7 Cleared page after ‘Empty Database’ is clicked
5. CONCLUSION

This project has been developed to perform the dataspace profiling on the web service data from various data sources. The necessity of the dataspace profiling became as important as managing a database system. This dataspace profiler online tool uniquely identifies each property and value associated with the different data sources. There are multiple uses that can be obtained from this dataspace profiler. It allows the user to easily get the desired value from all the available data sources, based on the property that is listed after profiling. It also provides better understanding of the data sources, understanding the inter-linking and connection among the data sources and also easily identifies the data source customization.

The main advantage of a dataspace profiling system is that it offers various services relating to data without requiring the most important semantic integration that should be done in other data integration systems. This project proves dataspace profiling as a data management abstraction for dataspace support platforms.


