An Implementation of Semantic Web System for Information retrieval using J2EE Technologies.

1.B.Hemanth kumar , Asst., Prof.,

Dept., of Information Technology,
RVR & JC College of Engineering,
Guntur, A.P. India

2. Prof. M.Surendra Prasad Babu
Dept., of CS & SE
AU college of Engineering,
Andhra University,
Visakhapatnam, A.P, India.

Abstract: Accessing web resources (Information) is an essential facility provided by web applications to every body. Semantic web is one of the systems that provide a facility to access the resources through web service applications. Semantic web and web Services are new emerging web based technologies. An automatic information processing system can be developed by using semantic web and web services, each having its own contribution within the context of developing web-based information systems and applications. This combination, called Semantic Web Services (SWS), provides several potential opportunities and challenges in e-business. Web services provide a verity of dynamic services for accessing web resources, but until now, they have been managed separately from conventional web contents resources. A new system is proposed here for a semantic web information retrieval, which incorporates semantic web, web services and J2EE technologies to enable dynamically locate the web resources that include homogeneous or heterogeneous web contents and web services. In this multi-tier architecture system the middle tier components contains the semantic web services.

Keywords– **RDF, Ontalogy, XML, OWL, JAXR.1.**

I. Introduction

The Semantic web[1] is a concept of having data on the web, defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration, and reuse of data across various applications. The Semantic web is the extension of the current web in which information is given with a well-defined meaning, better enabling computers and people to work in cooperation. Semantic web Applications have been characterized by different authors and by different events[2]. Some of the problems in Semantic web applications are: (a) data has semantics and is represented using formal descriptions, (b) semantic data is reused, manipulated and processed, (c) data sources are heterogeneous and are owned or controlled by different organizations, (d) applications assume an open world (i.e. the information is never complete), (e) multiple natural languages are supported, and (f) RDF(S) and OWL, the open standards recommended by the W3C, are used. In the Semantic web, reuse appears not only at the data level, but also at the application level, as nowadays there is much open software from a wide range of sources that can be reused when building Semantic web applications. In the application level reuse follows three different approaches: 1) a distributed services approach: by integrating web service technology in their architectures; 2) a shared memory approach: by composing components that use a shared space of common memory to communicate as in the case of reusing libraries inside an application; and 3) and a mixed approach: by combining the above two approaches.

II. Semantic web technology stacks

The figure1 shows an entire stack[1] of concepts, languages and standards attributed to the Semantic web.

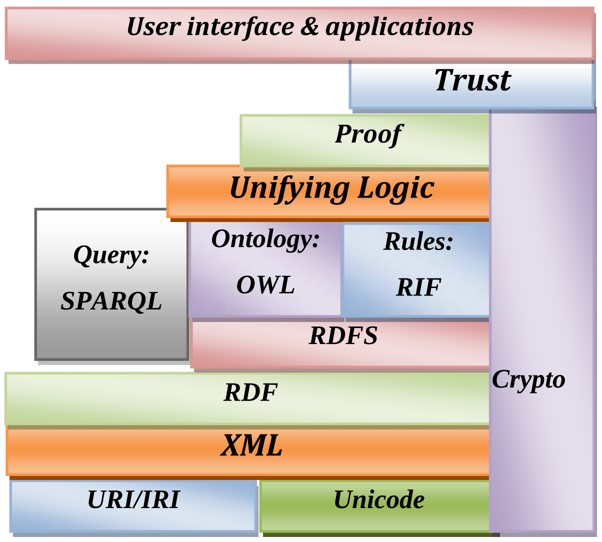


Figure 1. Semantic Web technology stack

A. XML:

XML is taken as the basis for writing semantic tags. This frame work fixes a namespace, a DTD [8], or an XML Schema for a specific domain. XML [9] and its concepts form its underlying foundation; together with the general use of Unicode make everything language-independent. URIs as well as IRIs can be used for unique global identification of dynamic web resources. IRIs (Internationalized Resource Identifiers) are an extension of URIs (Uniform Resource Identifiers) that allow the use of Unicode symbols.

B. RDF – Resource Description Framework

RDF is the underlying unified data model for representing semantics. The data model and XML serialization syntax is used for describing resources both on and off the Web. RDF makes use of unique identifiers (URI, Uniform Resource Identifier) for describing metadata. URIs are used to describe things, also called resources, which could represent people, places, documents, images, databases, etc. All RDF applications adopt a common convention for identifying these things. A subset of URI, the Uniform Resource Locator or URL, is concerned with the location and retrieval of resources, while URI is a unique identifier for things or resources that we describe but that may not necessarily be retrievable. However, RDF provides a consistent, standardized way to describe and query internet resources, from text pages and graphics to audio files and video clips. RDF is based on three core concepts: **resources**, **properties**, and **statements.** A **resource** is the subject or the object of an RDF description. A **property** describes a resource; its value is either a literal or a reference to another resource. A **statement** combines the property of a resource with a value. It can be represented as a subject, a predicate, and an object, where subject and object may be URIs and predicate is the property linking the object to the subject.

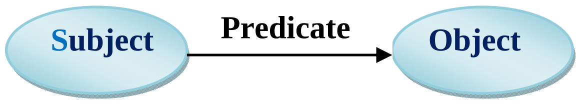


Figure 2 : General form of an RDF statement

C. RDF Schema:

RDF Schema is a standard means to describe the attributes, properties and classes of resources defined using RDF. For example, if a resource is of a particular type, or having a certain relationship to another resource, or having some specified attribute, then it need to uniquely identify these descriptive concepts. RDF uses URIs for identifying the above. Different development communities can invent new descriptive properties (such as person, employee, price, and classification) and assign URIs to these properties. Since the assignment of URIs is decentralized, one can be sure that uniquely named descriptive properties don't get mixed up when we integrate metadata from multiple sources. Both RDF and RDF Schema are based on XML and XML Schema. The existence of standards for describing data or other resource (RDF) and data or other resource attributes (RDF

Schema) enables the development of a set of readily available tools to read and exploit data or other resource from multiple sources. The degree to which different applications can share and exploit data or other resource is called syntactic interoperability. The more standardized and widespread these data manipulation tools are, the higher the degree of syntactic interoperability, and the easier and more attractive it becomes to use the SW approach as opposed to a point-to-point integrated solution. Figure.3 is an RDF document describes a student entry for the college found in a particular department CSE. In the case of CSE department the students Kiran, Ravi, and Syam are the existing students.

```
<? xml  version="1.0" ?>
<rdf: RDF
xmlns:rdf="http://www.w3.org/1992/02/22-rdf-syntax-ns#
xmlns:student="http://www.college.student>
<rdf:Desription
   rdf:about="http://www.college./student/cse">
    <student:cse>
         <rdf:dept>
         < rdf:li>Kiran</rdf:li>
         < rdf:li>Ravi</rdf:li>
         < rdf:li>Syam</rdf:li>
          </rdf:dept>
  </student:cse>
 </rdf:Description></rdf:RDF>
```

Figure: 3 RDF document

D. Ontology:

Ontology is the central component of the Semantic web. The term ontology, also known as a domain model, refers to a hierarchical data structure containing the relevant entities and their relationships and rules within a specific domain. In terms of classes, subclasses, and properties Ontology define data models. In order to understand data – and treat it as information by multiple applications – semantic interoperability is required. Syntactic interoperability is about parsing data correctly, which requires mapping between terms. Semantic interoperability is about content analysis, which requires formal and explicit specifications of domain models, which define the terms used and their relationships. Ontology provides shared semantics to metadata, enabling a degree of semantic interoperability. The challenge for the designer is to identify how to represent, create, manage and use both ontology as shared knowledge representations, but also large volumes of metadata records used to annotate Web resources of a diverse kind. That is, to achieve knowledge integration through ontology, semantic metadata and databases of annotations. The aim of building ontology is to share and reuse knowledge. Since the SW is a distributed network, there are different ontology that describe semantically equivalent things. Therefore, it is necessary to map elements of these ontology if one wants to process information across applications or domains.

E. Web Ontology language (OWL):

W3C's OWL (Ontology Working Language) Web Ontology Language is a poplar language used to represent ontology on the Web. OWL provides greater machine interpretability for Web content than supported by XML, RDF, and RDF Schema by providing additional vocabulary along with formal semantics. Classes, properties, and individuals are the basic components of OWL. Classes are the basic building blocks of OWL ontology. A class is a concept in a domain. Classes usually constitute a taxonomic hierarchy (for example, a subclass super class hierarchy). Properties have two main categories: (a) Object properties, which relate individuals to other individuals; and (b) Data type properties, which relate individuals to data type values, such as integers, float, and strings. OWL makes use of XML Schema for defining data types. Instances of classes are known as individuals, and properties can relate one individual to another. Examples of ontology are catalogs for online shopping sites.

F. SPARQL:

(Protocol And RDF Query Language) is query language for accessing collections of RDF data currently under development. RIF stands for the Rules Interchange Format (under development). For specifying the meaning of the components and terms included ontology, a logical language is often used, as it is for defining the formal semantics of an ontology. A logical language comes with a proof theory or model theory in order to be able to make statements about consistency, completeness, or correctness. A logical language can also provide formal semantics for a query language for ontology. If all of these items are indeed used for automated integration of information, for eliminating inconsistencies or incompleteness, and for consolidating distributed information, from the web, and if accompanied with proper security, verification, and encryption mechanisms, the result will be a Web that goes for beyond capabilities of what we are used to today. Trust can be seen as a top goal for the development of the semantic web, where users can trust the information that is accessible via Web or the information processes are that executed on the Web. Finally a user interface and applications layer appears at the top of the stack.

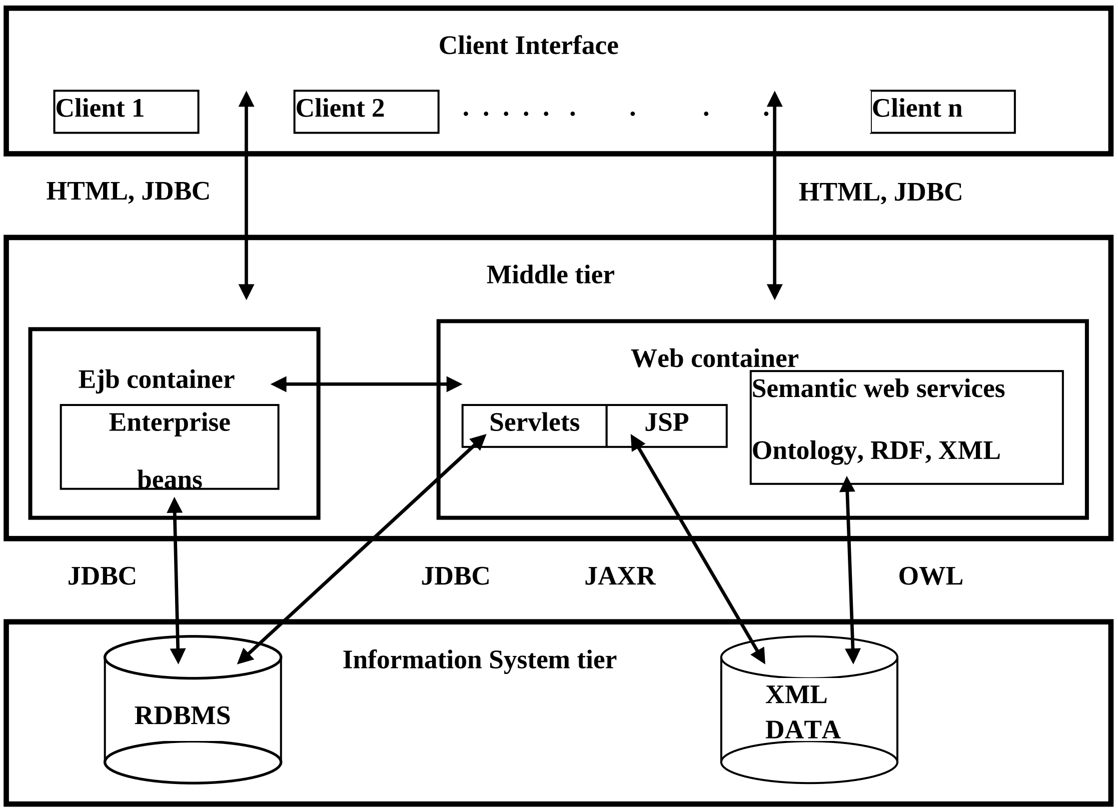


Figure :4 SYTEM ARCHITECTURE

III. System architecture:

Figure.4 shows the over all architecture of the proposed system. The technology and economy of today have intensified the need for more efficient, faster, and larger-scale information management solutions. The J2EE system satisfies these challenges by providing a programming model that improves development productivity, standardizes the platform for hosting enterprise applications, and ensures portability of developed applications with an extensive test suite. J2EE supports multi-tier architecture of type component-based development enterprise applications. A J2EE multi-tier architecture application system includes the following three tiers:

III.A Three tier architecture

- **Client tier**: Web components like Servlets and JavaServer Pages (JSPs) [6], or java applets and stand alone applications provide a dynamic interface for the client to communicate with middle tier.
- **Middle tier**: In middle tier the system maintains server side applications, enterprise beans and Web Services that encapsulate reusable, distributable business logic for the application. J2EE Application Server maintains these server-tier components, which provides the platform for these components to perform actions updations of data.
- **Enterprise data tier**: Enterprise's data is stored and persisted in data tier in the form of a relational database and XML registries.

Development of J2EE applications includes development of different types of components, containers, and services. Web components at application level like servlets and JSP provide dynamic responses to requests from a Web page. EJB components contain Server-side business logic for enterprise applications and these can be implemented in EJB components. EJB and WEB modules are supported by EJB host services.

III.B JAXR:

XML registry is the repository for information about the availability of web services and information necessary to access web services. Java API for XML Registries(JAXR)[17] is used to access an XML registry. Application Server provides the ability for the clients to publish, discover, and manage content within XML registries using the implementation of Java^{TM} API for XML Registry (JAXR). The JAXR architecture is Organized into 2 groups. One is the JAXR client and the other is JAXR provider. JAXR client is a java program that access information contained in an XML registry by interacting with a JAXR provider. JAXR provider implements the registry service interface and allows JAXR client to access registry.

To develop JAXR client [16] 1.Get access to a registry, 2.Establish a connection, 3.Query a registry

1.Get access to a registry: Through any public XML registry…

You must obtain permission from the registry to access the registry. A JAXR client can then perform queries, adds data to registry, or update registry data. To register with one of the public UDDI (Universal Description, Discovery and Integration) go to one of the following Web sites and follow the instructions:
www-3.ibm.com/services/uddi/testregistry/inquiryapi, http://uddi.microsoft.com/inquire,

http://test.uddi.microsoft.com/inquire, http://test.uddi.microsoft.com/publish

2. Establish a connection: Create a ConnectionFactory object, using ConnectionFactory.newInstance(); Create a set of properties and set the properties to the ConnectionFactory object, Call the createConnection() method to connect to the XML registry., Create an instance of RegistryService object which is used to interact with the xml registry.

3.Making a query: Create an instance of BusinessQueryManager which is used to query the registry., three common queries can be performed by clients: locating businesses that have published services on the registry., locating services of a particular business, retrieving information on how to bind to a particular service.

The BusinessQueryManager object contains interfaces for each type of query., they are findOrganization(), findServices(), findServiceBindings()

Queries can be made based on one of the following **criteria**: finding organizations by name., finding organizations by classification., finding organizations by WSDL descriptions.,finding services and service bindings

III.C. Publishing services to registry:

A business can make web services available to other organizations by inserting new information into an xml registry. This information consists of : Name of the organization, Name of the primary contact, Primary contacts telephone number and email, Industry classification and code of the business, Name and description of web service supported by the business, Binding description and binding URI.

III. D Inserting and removing a service:

Insert: Connect to an xml registry; Insert the new information into the xml registry.

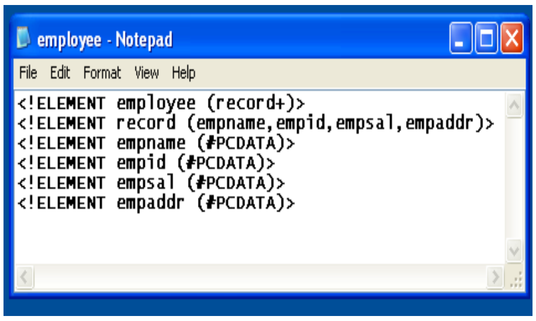
Removing a service from registry: Inquire the registry for service, Delete from the registry.

IV. Results & Discussion:

A.Sample data sets: Data set 1.

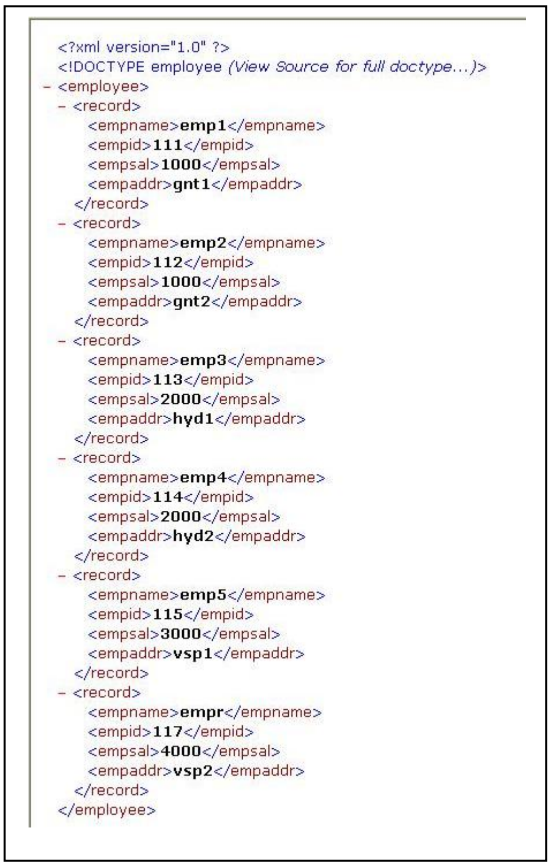
employee			
empname	**empid**	**empsal**	**empaddr**
emp1	111	1000	gnt1
emp2	112	1000	gnt2
emp3	113	2000	hyd1
emp4	114	2000	hyd2
emp5	115	3000	vsp1
empr	117	4000	vsp2

Table 1. RDB for employee



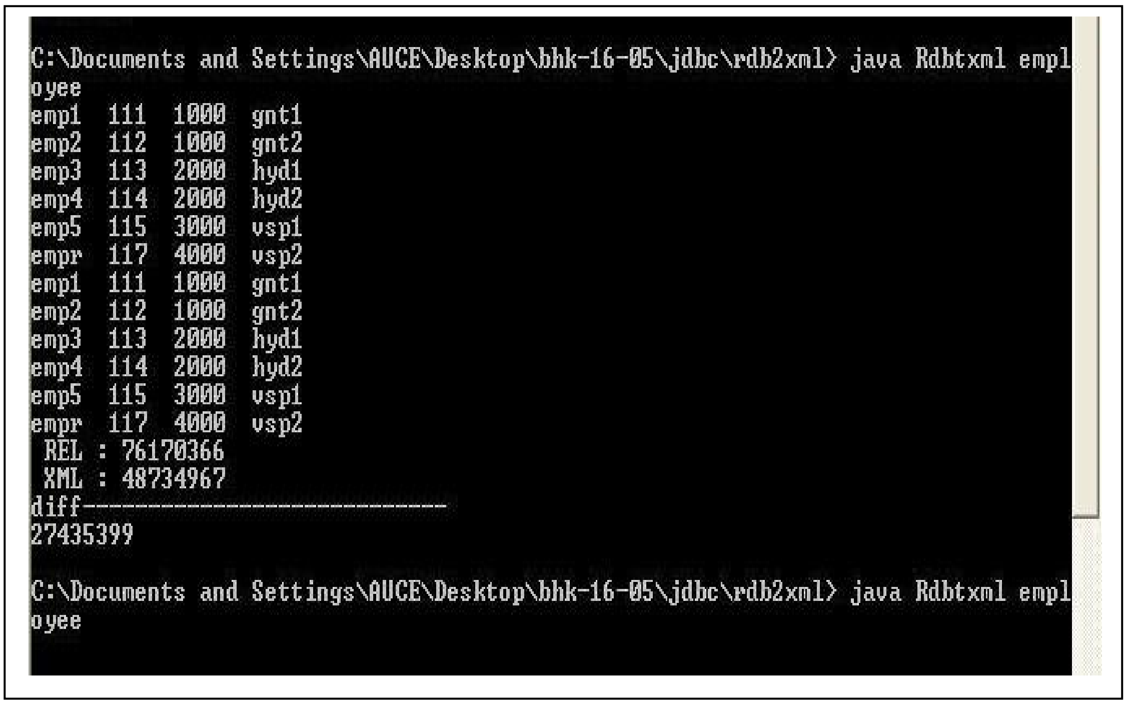
```
<!ELEMENT employee (record+)>
<!ELEMENT record (empname,empid,empsal,empaddr)>
<!ELEMENT empname (#PCDATA)>
<!ELEMENT empid (#PCDATA)>
<!ELEMENT empsal (#PCDATA)>
<!ELEMENT empaddr (#PCDATA)>
```

DTD for the xml document:



```
<?xml version="1.0" ?>
<!DOCTYPE employee (View Source for full doctype...)>
- <employee>
  - <record>
      <empname>emp1</empname>
      <empid>111</empid>
      <empsal>1000</empsal>
      <empaddr>gnt1</empaddr>
    </record>
  - <record>
      <empname>emp2</empname>
      <empid>112</empid>
      <empsal>1000</empsal>
      <empaddr>gnt2</empaddr>
    </record>
  - <record>
      <empname>emp3</empname>
      <empid>113</empid>
      <empsal>2000</empsal>
      <empaddr>hyd1</empaddr>
    </record>
  - <record>
      <empname>emp4</empname>
      <empid>114</empid>
      <empsal>2000</empsal>
      <empaddr>hyd2</empaddr>
    </record>
  - <record>
      <empname>emp5</empname>
      <empid>115</empid>
      <empsal>3000</empsal>
      <empaddr>vsp1</empaddr>
    </record>
  - <record>
      <empname>empr</empname>
      <empid>117</empid>
      <empsal>4000</empsal>
      <empaddr>vsp2</empaddr>
    </record>
  </employee>
```

XML data set for the employee



```
C:\Documents and Settings\AUCE\Desktop\bhk-16-05\jdbc\rdb2xml> java Rdbtxml empl
oyee
emp1  111  1000  gnt1
emp2  112  1000  gnt2
emp3  113  2000  hyd1
emp4  114  2000  hyd2
emp5  115  3000  vsp1
empr  117  4000  vsp2
emp1  111  1000  gnt1
emp2  112  1000  gnt2
emp3  113  2000  hyd1
emp4  114  2000  hyd2
emp5  115  3000  vsp1
empr  117  4000  vsp2
 REL : 76170366
 XML : 48734967
diff--------------------------------
27435399

C:\Documents and Settings\AUCE\Desktop\bhk-16-05\jdbc\rdb2xml> java Rdbtxml empl
oyee
```

Extracting data from RDB set and XML data set

Time comparison to extract data from RDB and XML for the data set 1:

From RDB it takes: 76170366 Nanoseconds

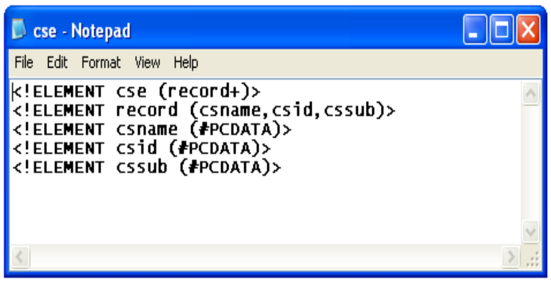
From XML it takes: 48734967 Nanoseconds.

Time difference is: 27435399 nanoseconds

B. Data set2:

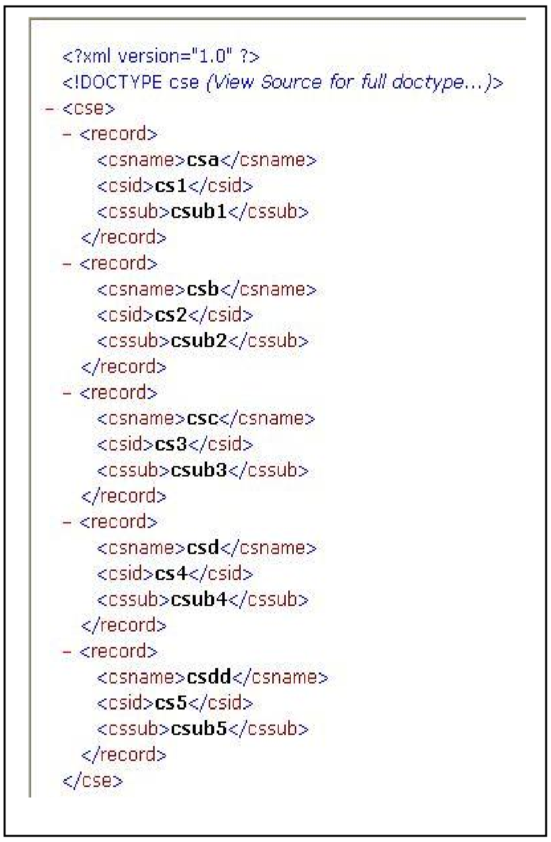
cse		
csname	csid	cssub
csa	cs1	csub1
csb	cs2	csub2
csc	cs3	csub3
csd	cs4	csub4
csdd	cs5	csub5

Table 2. RDB for CSE



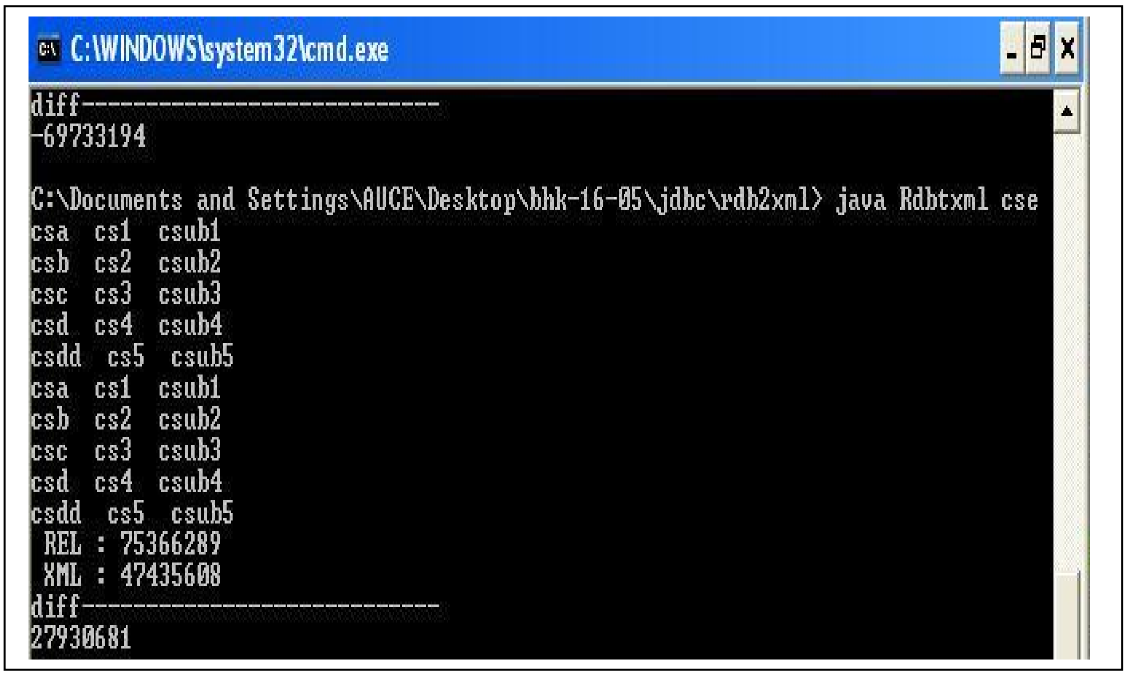
```
<!ELEMENT cse (record+)>
<!ELEMENT record (csname,csid,cssub)>
<!ELEMENT csname (#PCDATA)>
<!ELEMENT csid (#PCDATA)>
<!ELEMENT cssub (#PCDATA)>
```

DTD for the xml document:



```
<?xml version="1.0" ?>
<!DOCTYPE cse (View Source for full doctype...)>
- <cse>
  - <record>
      <csname>csa</csname>
      <csid>cs1</csid>
      <cssub>csub1</cssub>
    </record>
  - <record>
      <csname>csb</csname>
      <csid>cs2</csid>
      <cssub>csub2</cssub>
    </record>
  - <record>
      <csname>csc</csname>
      <csid>cs3</csid>
      <cssub>csub3</cssub>
    </record>
  - <record>
      <csname>csd</csname>
      <csid>cs4</csid>
      <cssub>csub4</cssub>
    </record>
  - <record>
      <csname>csdd</csname>
      <csid>cs5</csid>
      <cssub>csub5</cssub>
    </record>
  </cse>
```

XML data set for the CSE



```
diff----------------------------
-69733194

C:\Documents and Settings\AUCE\Desktop\bhk-16-05\jdbc\rdb2xml> java Rdbtxml cse
csa  cs1  csub1
csb  cs2  csub2
csc  cs3  csub3
csd  cs4  csub4
csdd  cs5  csub5
csa  cs1  csub1
csb  cs2  csub2
csc  cs3  csub3
csd  cs4  csub4
csdd  cs5  csub5
 REL : 75366289
 XML : 47435608
diff----------------------------
27930681
```

Extracting data from RDB set and XML data set

From RDB it takes: 75366289 Nanoseconds

From XML it takes: 47435608 Nanoseconds

Time difference is: 27930681 nanoseconds.

Form the above results it is more efficient to extract data from xml than relational data base. It takes less time to extract data from XML than RDB. The time factor entirely depends on processor speed.

V. Conclusions

Multi tier architecture plays a dynamic role in web service applications for information retrieval system. Data represented in XML format is well defined structure. JAXR is one of the application to access XML data in java. In semantic web XML is the base to represent RDF and Ontology. RDF describes the data in XML format and Ontology define the data models by using XML syntax. In this paper we have studied new system for information retrieval system for semantic web using J2EE architecture. It is shown that it takes less time to extract data from XML than RDB. By using JAXR applications the client can access the XML registries published by the provider.

References

[1] Gottfried Vossen, Stephen Hagemann, “Unleashing web 2.0 from concepts to creativity” Morgan Kaufmann publishers,.
[2] Amar Nayak, Jitendra Agarval, “Enterprise Architecture for Semantic web mining in education”, @2009 IEE.
[3] Haibo Yu,Tsunenori Mine and Makoto Amamiya “ An Architecture for Personal Semantic Web information Retrieval System Integrating Web services and Web contents ” IEEE International conference on web services(ICWS’05).
[4] M. Burstein and C. Bussler. “A Semantic Web Services Architecture” , Version 1.0, January, 2005. http://www.daml.org/services/swsa/note/swsa-note_v3.html
[5] Eyal Oren, Armin Haller, Manfred Hauswirth, “A Flexible Integration Framework for Semantic web2.0 Applications”. @2007 IEEE.
[6] “http://www.sun.com/products/jsp/overview.html” .
[7] “http://www.hpl.hp.com/semweb”
[8] “http://www.w3schools.com”
[9] Jussi Myllymaki, “Effective Web Data Extraction with Standard XML Technologies ” *WWW10*, May 1-5, 2001, Hong Kong.ACM 1-58113-348-0/01/0005.
[10] XML Path Language (XPath), W3C Recommendation,” http://www.w3.org/TR/xpath.html”
[11] XML Schema Part 0: Primer, W3C Working Draft, “http://www.w3.org /TR/xmlschema-0/”.
[12] OASIS:Organization for the Advancement of Structured Information Standards. “http://www. oasisopen.org / home /index.php”
[13] “http://www.w3schools.com/wsdl/wsdl_uddi.asp ”
[14] “http://schemas.xmlsoap.org/wsdl/
[15] “http://www.w3.org/TR/rdf-sparql-protocol/”
[16] http://www.java-tips.org/java-ee-tips/java-api-for-xml-registries/java-api-for-xml-registries.html
[17] http://download.oracle.com/javaee/1.3/tutorial/doc /JAXR2.html
[18] http://www.uddi.org/pubs/uddi_v3.htm.