International Journal of Electronics Communication and Computer Engineering
Volume 5, Issue 1, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

From XML to RDF: A New Semantic Information Retrieval System

Awny Sayed
Faculty of Science -Minia University-Egypt
College of Applied Sciences IBRI- Sultanate of Oman
Email: awny.ibr@cas.edu.om, Contact No.: 00968-98838296

Sharmi Sankar
College of Applied Sciences IBRI- Sultanate of Oman
Email: sharmi.ibr@cas.edu.om

***Abstract* – The World Wide Web has changed the way of contact and communication between the nations around the world. It lies at the heart of a technology revolution which is transforming the developed world toward a knowledge society. Search engines, such as AltaVista, Yahoo, and Google, are one of the greatest patent which invented by the humans and the main tools for using today's Web. It is clear that search engines contribute on the huge success for using the Web. This development has also changed the way we think of computers and search engines whereas the current web still does not understand the meaning behind the web pages. It returns thousands of relevant and irrelevant data from a single query. Our main purpose is to design and implement advanced information retrieval system based on RDF and semantic web technology and provides information in a specific and accurate manner in the search results. Moreover converting the current Web than just a large repository of information scattered and dispersed to a huge database where information are interrelated and have a specific meaning. In order to enable computers and people to work in better cooperation.**

***Keywords* – RDF, XML, RDFS, XED, SPARQL.**

I. INTRODUCTION

The World Wide Web (WWW) is the largest knowledge database that mankind has ever created. It is simple and open principles enable people to publish or access information very easily. On the other hand, as the web continuously grows, we face problems with the amount of information and its organization. It is becoming more and more difficult to find relevant information and manage heterogeneous knowledge sources [1]. To deal with this problem, the expression *"semantic search"* appeared on the scene. Semantic search is a data searching technique in a which a search query aims to not only find keywords, but to determine the intent and contextual meaning of the words a person is using for search. Semantic search provides more meaningful search results by evaluating and understanding the search phrase and finding the most relevant results in a website, database or any other data repository. Semantic search works on the principles of language semantics. Unlike typical search algorithms, semantic search is based on the context, substance, intent and concept of the searched phrase. Semantic search also incorporates location, synonyms of a term, current trends, word variations and other natural language elements as part of the search. Semantic search concepts are derived from various search algorithms and methodologies, including keyword-to-concept mapping, graph patterns and fuzzy logic.

1.1 Drawbacks of the current web:

Most information is currently available in a weakly structured form, for example, text, audio, and video. From the knowledge management perspective, the current technology suffers from limitations in the following areas: Searching information. Companies usually depend on keyword-based search engines. Despite the Keyword-based search engines is the backbone of the web. However, there are serious problems associated with their use: *High recall, low precision*. Even if the main relevant pages are retrieved, they are of little use if another 28,758 mildly relevant or irrelevant documents were also retrieved. Too much can easily become as bad as too l*ow or no recall*. Often it happens that we don't get any answer for our request, or that important and relevant pages are not retrieved. Although low recall is a less frequent problem with current search engines, it does occur.

Results are highly sensitive to vocabulary. Often our initial keywords do not get the results we want; in these cases the relevant documents use different terminology from the original query. This is unsatisfactory because semantically similar queries should return similar results.Results are single Web pages. If we need information that is spread over various documents, we must initiate several queries to collect the relevant documents, and then we must manually extract the partial information and put it together. Human involvement is necessary to interpret retrieved pages, and to combine information. [2].

Extracting information needs human time and effort which are required to browse the retrieved documents for relevant information. Current intelligent agents are unable to carry out this task in a satisfactory fashion. Maintaining information currently involves issues, such as inconsistencies in terminology and failure to remove outdated information/uncover the information. New knowledge implicitly existing in corporate databases is extracted using data mining. However, this task is still difficult for distributed, weakly structured collections of documents. To view information it is often desirable to restrict access to certain information to certain groups of employees. "Views", which hide certain information, are known from the area of databases but are hard to realize over an intranet (or the Web). [2]

1.2 Semantic Web: Vision and goals

There are several initiatives to improve the situation and delete the drawbacks in the current web. One of them is the semantic web [3], which would give more structure and computer-understandable meaning to the data on the WWW. The semantic web is not a separate web but an

extension of the current one, in which information is given a well-defined meaning, better enabling computers and people to work in cooperation [3].

The Semantic Web proposes to overcome the difficulties listed above by making Web content machine-processable. The key point is that the semantics (meaning) of Web content is explicitly represented and processed. This aim will be achieved by the combination of the following technologies :*Explicit Meta-Data*: Web content will carry its meaning "on its sleeve" through appropriate semantic markup. *Ontologies*: They will describe semantic relationships between terms, and will serve as the foundation for establishing shared understanding between applications *Logical Reasoning*: Automated reasoning-enabled tools will make use of the information provided by meta-data and ontologies. [2]

The aim of the Semantic Web is to allow much more advanced knowledge management systems: Knowledge will be organized in conceptual spaces according to its meaning. Automated tools will support maintenance by checking for inconsistencies and extracting new knowledge. Keyword-based search will be replaced by query answering: requested knowledge will be retrieved, extracted, and presented in a human-friendly way. Query answering over several documents will be supported. Defining who may view certain parts of information (even parts of documents) will be possible. [2]

2. EXTENSIBLE MARKUP LANGUAGE

2.1 Introduction

Extensible Markup Language, abbreviated XML, describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language. By construction, XML documents are conforming SGML documents [4]. Moreover it is designed to describe data and focus on what data is as well as it is used to structure store and to send information. As we say Java is a portable programs as well as XML is portable data.

2.2 XML Tree

XML documents form a tree structure that starts at "the root" and branches to "the leaves. XML documents must contain a root element. This element is "the parent" of all other elements.

XML Example:

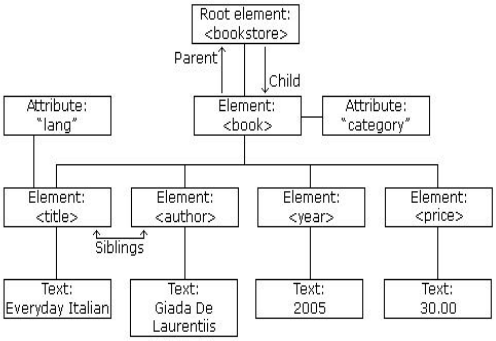

Fig.1. XML Tree

The root element in the example is <bookstore>. All <book> elements in the document are contained within <bookstore>. The <book> element has 4 children: <title>, < author>, <year>, <price>.

<bookstore>

<book category="COOKING">
<title lang = "en">Everyday Italian</title>
<author>Giada De Laurentiis</author>
<year>2005</year>
<price>30.00</price>
</book>
<book category="CHILDREN">
<title lang = "en">Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>29.99</price>
</book>
<book category="WEB">
<title lang = "en">Learning XML</title>
<author>Erik T. Ray</author>
<year>2005</year>
<price>39.95</price>
</book>
</bookstore>

Fig.2. XML Document

2.3 XML storage

The first approach to storing XML documents is to employ traditional databases such as relational database or object-oriented database. The second is to develop a specialized system, which is known as native storage. Recently, there is another approach, namely hybrid storage, which allows data to be stored using mapping to relations and also allows storage of XML sub-trees in its native format [9]. The underlying storage representation has a significant impact on the efficiency of query processing. Basically, a storage strategy can be defined as efficient if the system manages to retrieve data accurately, use storage resources competently and update data and schema correctly. [10]

2.3.1 XML-enabled databases (XED)

XEDs are the databases that store the XML data in other formats such as tabular data, spreadsheet, and objects and so on, other than the XML format itself. Yet, these technologies usually provide a means to shred XML into their underlying format. Using the relational database as the storage, it provides maturity, scalability, portability and stability [11]. On the other hand, object-oriented databases use class inheritance to govern the data integrity and to support complex relationships. There are also some approaches, which use object-relational database to combine both relational and object oriented technologies. Although XEDs are advantageous in terms of providing the database features (such as scalability, concurrency control, recovery service and so on), these approaches are vulnerable especially when evaluating query of a large dataset. [10]

Relational databases: Most enterprise today has long secured the use of relational databases for high-end transaction processing system. They have spent large investments amounting to trillions of dollars. As such, simply replacing relational databases with a pure XML database is not a good choice. Examples of some XML storage models based on relational databases are STORED and XISS/R. Since there are mismatches between the XML-structured data and relational data, mapping plays an important role in providing seamless integration between these database infrastructures [10].

Object-oriented databases: There are only minor discussions and publications based on object-oriented databases. This may be due to several reasons as follows [12]. First, applying object oriented database techniques is less difficult compared to relational technology. Second, any publication on this subject would have to disclose the proprietary technical details. Finally, the absence of a standard object-oriented query language has hindered the development of tests to compare these implementations. [10]

2.3.2 Native XML databases

A native storage basically means building a specialized data manager that contains XML as its fundamental unit of its logical model. These data are stored and retrieved in their original structure, with no mapping process required. Nevertheless, the NXD requires a particular underlying physical storage model, which can be a custom database or any typical database model [13]. Using this approach may work best, especially on scalability, data retrieval and handling of huge amounts of data. Nevertheless, it is not suitable when integration between various heterogeneous XML documents is needed. [10]

Basically, an NXD falls into two main categories: (1) document-based storage and (2) node-based storage. A document based storage (also known as text-based) will store the entire XML document in text form and provide some database transaction support such as indexing, materialized view and so on in accessing the document. A simple strategy for this might store the document as a Binary Large OBject (BLOB) or Character Large Object (CLOB) in a relational table or as a file in a file system and provide XML-aware indexes over the document. In contrast, a node-based (also known as model-based) storage models the XML document as the internal model such as Document Object Model (DOM) or Simple API for XML (SAX) from the document and stores this model. How the model is stored depends on the physical underlying database. For instance, storing the DOM to relational database might result in tables such as Entities, Elements, Values, Attributes, Levels, and so on. TIMBER, XBase and Natix are some examples of native storage [10].

2.3.3 Hybrid Storage

Recently, commercial RDBMS vendors have shown significant interest in providing support for XML data management. These hybrid systems have the added advantage of being capable of storing both relational (structured) and XML (semi-structured) data, thus, allowing applications to access a single data repository. The main objective of such systems is to preserve the benefits associated with commercial RDBMS offerings including high levels of reliability, availability, security, concurrency and customer support while making it easy to manage and integrate existing corporate data with data modeled in hierarchical XML structures. [10].

Figure 3 depicts the general hybrid storage architecture. By using hybrid storage, the user is not limited to either native or relational storage alone. The user may choose to store different parts of the XML based on the required level of granularity [14]. For example, if we want to query the top-level elements of Publication List such as book, title, section, and journal and so on, these elements can be stored in relational tables, while the section details may be stored as XML data type. With this, it gives the flexibility of storing useful and query able information in relational tables while not decomposing the entire XML document. Henceforth, it saves the time and speed needed in reconstructing the document as well as the speed to load data [15].

Being hybrid storage, users can search both XML data and relational data types with structured query language (SQL) or XQuery and also combine SQL with XQuery. In addition, to efficiently process queries of XML data, most of these commercial RDBMSs leverage cost-based query optimization technology to evaluate different data access strategies and select a low-cost option [10].

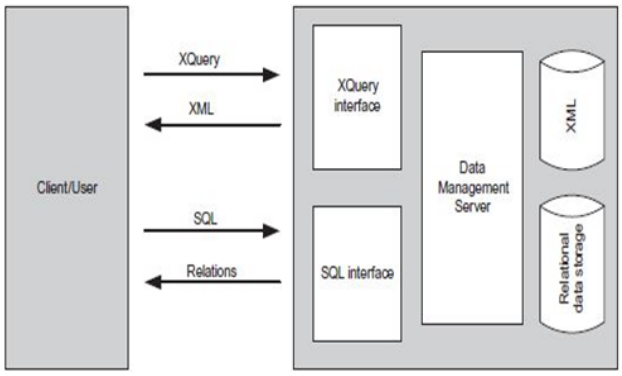

Fig.3. General Architecture of hybrid storage [26].

2.4 XML query:

Much research has been done on XML query languages. Examples of prior works on XML query languages are Quilt, XPath, XQuery and XML query algebras such as XAL, YATL and Lore. On the other hand, XML query languages provide the means to extract and manipulate data from XML documents. Although most of the query languages differ in detailed grammars and representation, they share a common feature, that is, queries usually make use of path expression for query evaluation [16].

Since XML is semi-structured data, there are typically two types of user queries, namely full-text queries (keyword-based search) and structural queries (complex queries specified in tree-like structure) as depicted in Figure5. A keyword search is somehow similar to content retrieval in information retrieval technology. Conversely, a structural search is to retrieve matches on the tree where it has the tags and structure (relationship) specified in the query criteria. Structural queries can be classified further

into path query and twig query. For each type of query, it may consist of only single type parent–child (P–C) relationship, single type ancestor–descendant (A–D) relationship or mixed types of both relationships. [10]

Path query defines query on one single element (consists of only one leaf node) at a time while twig query defines query on two or more elements (consists of two or more leaf nodes). Thus, they are also known as Simple Path Expression and Branching Path Expression respectively. In both cases, query nodes may be elements, attributes or text. However, query edges for path query are either P–C or A–D relationships, whereas query edges for twig query may be P–C, A–D or sibling (preceding and following), which also determine the ordering of the relationships. In XPath notation, P–C relationship is denoted by ‘‘/’’ while A–D relationship is denoted by ‘‘//’’. A complex query, however, is a twig query that consists of at least a branching edge and may contain many edges formed by the basic P–C or A–D relationships. [10]

2.5 XML indexing

Generally, indexing is a well-known technology to improve the efficiency and scalability of query processing by reducing the search space. XML indexes fall into the following categories: Primary XML index and Secondary XML index. The primary XML index is a B+tree and is useful because the optimizer creates a plan for the entire query. It is always better to split the entire XML columns into relational rows and columns. It contains one row for each node in the XML instance. A relational index on the shredded and persisted representation of all tags, values, and paths of XML instances in the xml data type column. The index creates several rows of data for each instance in the column. A primary XML index requires a clustered index on the primary key of the table containing the xml data type being indexed [17].The secondary XML indexes can be created to enhance the performance.There are the types of secondary indexes:

- *PATH secondary XML index*:
  If the use of path expressions on XML columns is prominent, the PATH secondary XML index can speed up the task. It optimizes queries based on path expressions.
- *VALUE secondary XML index*:
  The VALUE index can be used if your task involves querying unknown attribute names. It optimizes value-based queries for paths that include wildcards or are not fully specified

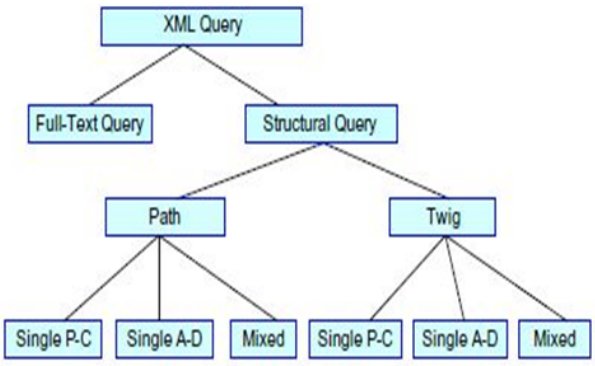

Fig.4. Main classification of query

- *PROPERTY secondary XML index*:
  Clustering paths within each XML instance in the PROPERTY index can be beneficial when multiple values are retrieved from individual XML instances. It optimizes queries based on properties in a specific XML instance stored in a column. [17].

III. RESOURCE DESCRIPTION FRAMEWORK (RDF)

The Resource Description Framework (RDF) is a W3C standard for describing Web resources, such as the title, author, modification date, content, and copyright information of a Web page. It is written in XML. The most important concept of it is metadata which concerned with meaning and integration which linking documents by common vocabularies. It is designed to be read by computers not to be displayed on the web. The RDF language is a part of the W3C's Semantic Web Activity. W3C's "Semantic Web Vision" is a future where: Web information has exact meaning, Web information can be understood and processed by computers and Computers can integrate information from the web. RDF triples are referred to as “triplets”, or “statements”. Triples in RDF terminology are RDF Resource, Property, and Property Value. RDF identifies things using Web identifiers (URIs), and describes resources with properties and property values. The combination of a Resource, a Property, and a Property value forms a Statement (known as the subject, predicate and object of a Statement).

3.1 RDF Basics:

The fundamental concepts of RDF are resources, properties and Property value.

3.1.1 Resources

We can think of a resource as an object, a “thing” we want to talk about. Resources may be authors, books, publishers, places, people, hotels, rooms, search queries, and so on. Every resource has a URI, a Universal Resource Identifier. A URI can be a URL (Unified Resource Locator, or Web address) or some other kind of unique identifier; note that an identifier does not necessarily enable access to a resource. URI schemes have been defined not only for web-locations but also for such diverse objects as telephone numbers, ISBN numbers and geographic locations. There has been a long discussion about the nature of URIs, even touching philosophical questions (for example, what is an appropriate unique identifier for a person?), but we will not go into detail here. In general, we assume that a URI is the identifier of a Web resource. [2]

3.1.2 Properties

Properties are a special kind of resources; they describe relations between resources, for example “written by”, “age”, “title”, and so on. Properties in RDF are also identified by URIs (and in practice by URLs). This idea of using URIs to identify “things” and the relations between is quite important. This choice gives us in one stroke a global, worldwide, unique naming scheme. The use of such a scheme greatly reduces the homonym problem that has plagued distributed data representation until now. [2]

3.2.3 Property value

Statements assert the properties of resources. A statement is an object- attributes value triple, consisting of a resource, a property, and a value. Values can either be resources or literals. Literals are atomic values (strings), the structure of which we do not discuss further. [2]

3.2 How to represent RDF?

RDF describes the interrelationships among resources in terms of named properties and values. These named properties may be thought of as attributes of resources and in this sense correspond to traditional attribute-value pairs. One main difference is that properties defined in RDF only identified by their name (URI) and not like in other object models, where attributes are identified by their name plus the domain class they can describe. So, properties have a URI and therefore are also resources. The value of a property can be another resource or a literal. A literal is simple string or other primitive data type as defined by XML. [19]

RDF provides three representations of the RDF data model namely:

- RDF Graph – a syntax-neutral graphical description of the data
- RDF 3-tuples – the set of statements described in triples
- RDF Syntax – provides some standard ways for describing data using XML.

RDF Example:

```
<xml version =”1.0” encoding=”utf-8”>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:as="http://www.coas.om/~zhp2/2004/0401/univ-bench.owl#">
<owl:Ontology rdf:about="">
<owl:imports rdf:resource="http://www.coas.om/~zhp2/2004/0401/univ-bench.owl"/>
</owl:Ontology>
<rdf:dean rdf:about="http://www.coas.om/~zhp2/2004/0401/univ-bench.owl/dean0">
<dean as:deanname="Ahmed Alriami"/>
<dean as:deandept="dean office"/>
<dean as:deandept="college council"/>
<dean as:deandept="Assistant dean for academic affairs and scientific research"/>
<dean as:deandept="Assistant dean for academic support affairs"/>
<dean as:deandept="Dir. of administration and finance affairs"/>
<dean as:deandept="Post and documentation section"/>
<dean as:deandept="Quality assurance section"/>
<dean as:deandept="Legal researcher"/>
<dean as:deandept="Internal auditor"/>
</dean>
</rdf:RDF>
```

Fig.5. RDF Example

This RDF is describing the organization chart for the Ibri college of Applied science. Its view all the department and section that is order under the dean of college.

3.3 RDF Schema (RDFS):

RDF is a universal language that lets users describe resources using their own vocabularies. RDF does not make assumptions about any particular application domain, nor does it define the semantics of any domain. It is up to the user to do so in RDF Schema (RDFS)[2]. It is not a replacement for XML Schema or the user of DTDs. It used to define specific RDF vocabularies; to specify how the elements of the vocabularies relate to each other [20]. RDF Schemas are Web resources (and haveURIs) and can be described using RDF. Officially called “RDF Vocabulary Description Language” [21].

RDF Schema (see figure 9) defines a data model for the creation of RDF statements

- Abstract data type (class).Class instantiation in RDF via <rdf:type>.
- Hierarchical class model and class inheritance (subclasses and superclasses)
- Syntax for common data exchange. Definition of properties and restrictions.

RDF Schema provides the same functionality as the relational database schema. It also provides the resources (describes objects, attributes and relationships within a specific area of interest) necessary to describe the objects and properties of a domain-specific schema. How to represent this statement (Semantic Web Technologies is managed by Harald Sack)? It has subject (Semantic Web Technologies), property (is managed by) and object (Harald Sack) refer figure 7to know how it’s represented in RDF and RDFS.

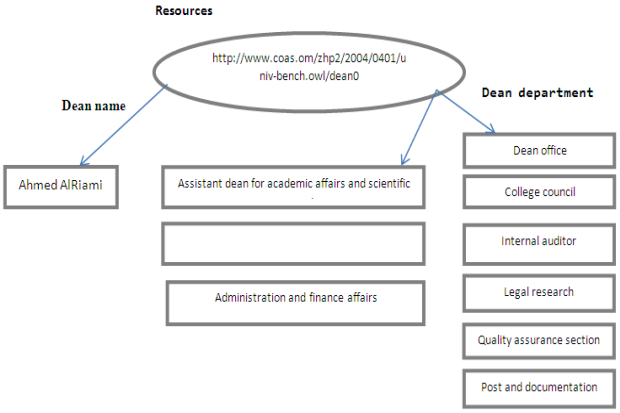

Fig.6. Part of RDF-graph from figure 5

3.4 How to store semantic data in RDF?

An RDF store (or triple store) is a system for storing and managing RDF data, and thus a special kind of Semantic Web tool [22]. Moreover, it is a system, uses to provide a mechanism for persistent storage and access of RDF graphs.A triple store is a purpose-built database for the storage and retrieval of triples, a triple being a data entity composed of subject-predicate-object, like "Bob is 35" or "Bob knows Fred"[23].

Much like a relational database, one stores information in a triple store and retrieves it via a query language.

International Journal of Electronics Communication and Computer Engineering
Volume 5, Issue 1, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

Unlike a relational database, a triple store is optimized for the storage and retrieval of triples. In addition to queries, triples can usually be imported/exported using Resource Description Framework (RDF) and other formats. [23]

3.5 How to query RDF data?

An RDF query language is a computer language, specifically a query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format. SPARQL is emerging as the de facto RDF query language [24]. So, SPARQL (a recursive acronym for SPARQL Protocol and RDF Query Language) is an RDF query language, that is, a query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format [25].Furthermore, SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs. [26]SPARQL allows users to write unambiguous queries. For example, the following query returns names and emails of every person in the dataset.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE {
  ?person a foaf:Person.
  ?person foaf:name ?name.
  ?person foaf:mbox ?email.
}
```

3.5.1 Query forms

In the case of queries that read data from the database, the SPARQL language specifies four different query variations for different purposes

- *SELECT query*
  Used to extract raw values from a SPARQL endpoint, the results are returned in a table format
- *CONSTRUCT query*
  Used to extract information from the SPARQL endpoint and transform the results into valid RDF.
- *ASK query*
  Used to provide a simple True/False result for a query on a SPARQL endpoint
- *DESCRIBE query*
  Used to extract an RDF graph from the SPARQL endpoint, the contents of which is left to the endpoint to decide based on what the maintainer deems as useful information.[25]

IV. THE PROPOSED SYSTEM

Our main objective is to create an Intelligent Information Retrieval System with the following characteristics (figure 7 shows the structure):

1. Easy to use by the normal user, since the proposed system is a keyword-based system.
2. Save time and achieve exact match for the submitted query.
3. Understand the meaning behind information (help to answer queries which cannot be answer by Google),
4. Our system is element-based retrieval not document-based retrieval as traditional.

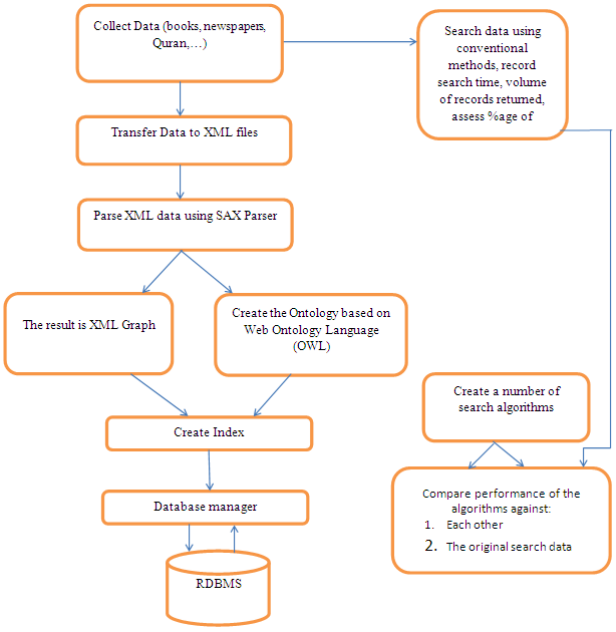

Fig.7. General Structure Of our IR system

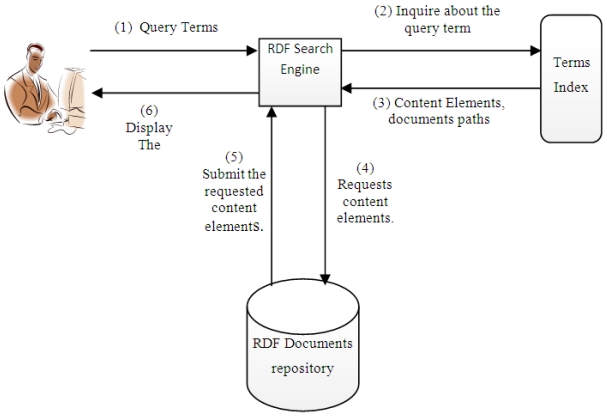

Fig.8. Action flow diagram

IV. SYSTEM DESIGN

Our system has three types of design (user interface design, database design, and security design).

1. *Database design*: In this phase we will draw the entity relationship diagram ERD to represent our data requirements which will describe the way how data will be stored in the database. Our database management system will be oracle 10g. We will store XML data as a graph in the relational data base; each graph has nodes and edges. Each node in RDB will be store (name, OID, parent, child), each Edge will store in RDB as (Source, target). Moreover, the data follow diagram will be used to represent our process requirements. Our design will have leveled and balanced DFD with different levels to represent our main process for the project. Class diagram as an object-oriented approach will be used also.

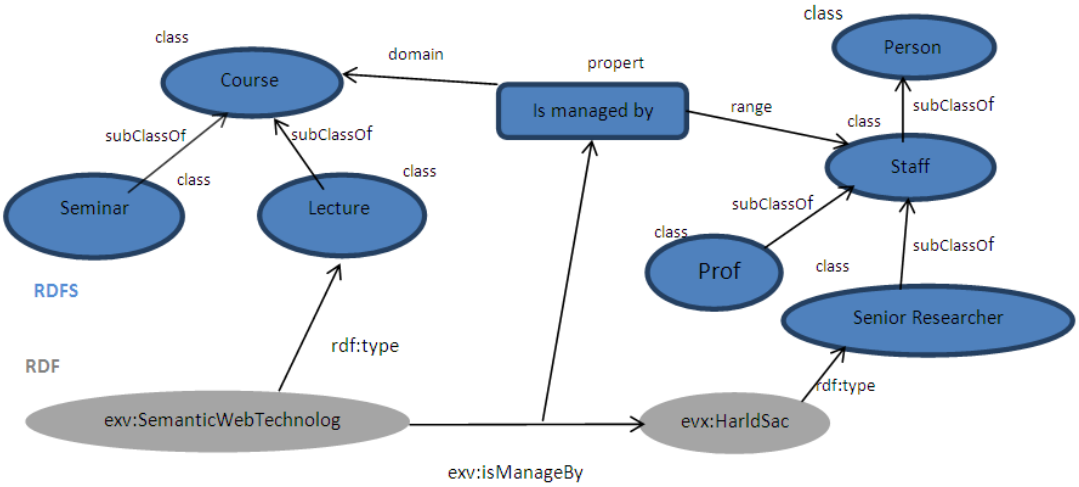

Fig.9. RSF as a graph

2. *User interface design*: This part of design will contain the main menus of our search engine that describe the interaction between the end users and the engine. Our user interface must be more usable and accurate.
3. *Security design*: The last part of our design about security. We plan to apply more advanced algorithms to protect our data. We also will apply the oracle 10g features about security to distribute the authorities between users.

VI. How Our Proposed System Works

1. The user submits his query terms into the XML Search Engine (see figure 8).
2. The XML Search Engine inquires the index about the terms submitted in step one.
3. The Index responds to the Search Engine with documents paths and lines numbers of the content elements contain those query terms.
4. The Search Engine gets the content elements from the database.

VII. Conclusion and Future Work

Semantic search seems to be an elusive and fuzzy target to many researchers. One of the reasons is that the task lies in between several areas of specialization. First, we present how we understand semantic search, the Web and the current challenges. Second, how to convert from XML to RDF to improve the semantic Web search. Third, how the usage of search engines can capture the implicit semantics encoded in the queries and actions of people. Fourth, we explain our prototype of semantic search engine.

References

[1] Svihla,M. Transforming Relational Data into Ontology Based RDF Data(a doctoral thesis). June 2007.
[2] Antoniou, G. and van Harmelen, F. (2004). A Semantic Web Primer. Cambridge: The MIT Press.
[3] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. Scientific American, 284(5), May 2001.
[4] T. Bray, J. Paoli and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0. Available at <http://www.w3.org/TR/1998/REC-xml-19980210>,February-1998.
[5] E.T.Ray. Learning XML, First Edition, January 2001.
[6] Anonymous. xml-schema . Available at <http://lucas.ucs.ed.ac.uk/tutorials/xml-schema/>, Jan 2013.
[7] T.Bray, J.Paoli, C.M.Sperberg, E.Maler, F.Yergeau and J,Cowan. Extensible Markup Language (XML). Available at <http://en.wikipedia.org/wiki/XML#Document_Type_Definition>,1996.
[8] Anonymous. Introduction to XML Schema. Available at <http://www.w3schools.com/schema/schema_intro.asp>.
[9] D. Hall, L. Stromback, Generation of synthetic XML for evaluation of hybrid XML systems, Lecture Notes In Computer Science 6193 (2010) 191–202.
[10] S.C. Haw, C.S.Lee. Data storage practices and query processing in XML databases , Faculty of Information Technology, Multimedia University, 63100 Cyberjaya, Malaysia . 2011.
[11] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D.J. DeWitt, J.F. Naughton, Relational databases for querying XML documents: limitations and opportunities, in: Proceedings of the VLDB, 1999, pp. 302–314.
[12] A.B. Chaudhri, A. Rashid, R. Zicari, XML Data Management: Native XML and XML-Enabled Database Systems, Addison-Wesley, US, 2003.
[13] A. Vakali, B. Catania, A. Maddalena, XML data stores: emerging practices, IEEE Internet Computing (2005) 62–69.
[14] Anonymous. Oracle, Oracle 9i and 10g, <http://www.oracle.com/technology/tech/xml/xmldb/index.html>, 2005.
[15] L. Stromback, M. Asberg, D. Hall, HShreX – a tool for design and evaluation of hybrid XML storage, in: Proceedings of DEXA Workshops, 2009, pp. 417–421.
[16] J.T. Yao, M. Zhang, A fast tree pattern matching algorithm for XML query, in: Proceedings of the IEEE/WIC/ACM, 2004, pp. 235–241.
[17] Anonymous. XML Indexes. Available at <http://www.careerride.com/XML-Indexes.aspx>
[18] S.Sunkle. Introduction to RDF. Available at <http://zentapestry.wordpress.com/2006/10/06/introduction-to-rdf-2/>, Aguast-2006.
[19] Anonymous. Introduction to RDF. Available at <http://www.dbis.informatik.unifrankfurt.de/~tolle/RDF/DBISResources/RDFIntro.html>
[20] Shelley Powers.Practical RDF ,Chapter 1. RDF: An Introduction. Available at < https://www.google.com.om/url?sa=t&rct=j&q=&esrc=s&source=books&cd=1&cad=rja&ved=0CC8QFjAA&url=http%3A%2F%2Fidb.snu.ac.kr%2Fimages%2F7%2F78%2FPractical_RDF_Ch1.pptx&ei=ELZCUrLaGeKJ0AWM9YDQCg&usg=AFQjCNEOM50ye_9pip5nsw84j1CaTvZcqQ&bvm=bv.53077864,d.d2k>

[21] H.Sack. Semantic Web Technologies Lecture. Available at <http://semweb2013.blogspot.com/>, University of Potsdam. 2012.
[22] Anonymous. RDF Schema. Available at <http://www.webopedia.com/TERM/R/RDF_Schema.html>
[23] Anonymous. Triplestore. Available at <http://en.wikipedia.org/wiki/Triplestore>
[24] Anonymous. RDF query language. Available at <http://en.wikipedia.org/wiki/RDF_query_language>
[25] Anonymous. SPARQL. Available at <http://en.wikipedia.org/wiki/SPARQL>
[26] Anonymous. SPARQL Query Language for RDF. Available at <http://www.w3.org/2001/sw/wiki/SPARQL>