Detection and Elimination of Duplicate Data from Semantic Web Queries

Zakia S.
Faisalabad Institute of Cardiology, Faisalabad-Pakistan

Abstract— **Semantic Web adds semantics to World Wide Web by exploiting machine understandable metadata. The metadata can be described by resource descriptor framework (RDF). When these resources are queried by a Web browser, the duplicate records may cause different problems: slow down indexing time, reduced search time efficiency, and extra storage space. We tackle this issue by implementing an algorithm which uses hash values to eliminate duplicate data records. The goal is to optimize the storage space and to increase the speed of returned results. We validate our approach on structured query language bases schema.**

Keywords— **RDF. Semantic and Web**

I. INTRODUCTION

Semantic Web (SW) is an extension of the current World Wide Web. In order to be easily understandable by the search engines, SW serves as a means for the information contents: documents, data, applications, e-services, images, audio/video files, personal Web pages, etc. Consequently, web users, designers, and warehouses developers can easily integrate, share, and query data information. Thus, SW does not only create links between web pages but also describes relationships like A is the writer of document B, and properties like size, location, and document type. Metadata is represented using Resource Description Framework (RDF) [1], an important part of the SW layer architecture [2].

Metadata describes characteristics of multiple resources, that are not limited to web pages but can also include items, knowledge, or concepts that can be identified on the Web, that have great utility, for instance, properties for shopping like price and availability. Thus, generated information and that of the consumers can be linked by shared metadata. The idea is not only to improve resource discovery, but also involves: administrative control (e.g. Admin Core), security, for instance the need for securing the process of transforming one digital format or platform to another one, personal information (Vcard), management information (Instructional Management Systems), content rating (Platform for Internet Content Selection, PICS), rights management (Uniform Resource Characteristics, URC), and preservation (object's provenance and context) [3].

The RDF graph model can be well mapped for expressing relational data. Relational database model consists of tables which further contain rows and columns constituting data records. A record is nothing but contents of its fields similarly. RDF nodes are nothing but the connections: property values. The mapping is direct as a record is an RDF node. The field (column) name is RDF property type. Finally, the record field (table cell) is a value [4]. To simplify queries that involve self joins for RDF data, simple arcs are used in the graph which represents the foreign key relationship [5].

The main SW task, organized as a huge relational database, is the metadata recovering. Thus, a large number of languages have been used to query data available on the SW. Due to the mapping process, some of them are inefficient: taking time to convert data from Structured Query Language, SQL to the targeted query language that normally has different format. Consequently, integration of these query languages is a difficult task.

While SQL based scheme [6] avoids these problems, specifically, it introduces a SQL table function RDF_MATCH query RDF data. This function allows different parameters for carrying user query: pattern for triples and RDF model, rule bases for inference, and aliases for name spaces. As a result of this function, a table of holding data is generated [6]. Furthermore, these results can be processed by using traditional SQL query construct along with other relational data.

Due to the lack of ability for handling duplicate query results SQL based approach has not been used yet to handle duplicate data on SW. A major problem in retrieving information from SW using search engines is that there may be multiple records referring to the same entity, increasing amount of data, and also leads inconsistent information. In addition, the growing data increases the duplicate data detection from the user perspective point of view and from the retrieval systems side.

The redundant data in SW queries is the main issue discussed in this paper. Firstly, we describe a brief review of RDF data query (Section 2). We explain how to detect and eliminate redundant information by hash values, using the Knuth Algorithm (Section 3). We validate our approach with a case study (Section 4). Finally, we conclude and outline some future perspectives.

II. RELATED WORK

The Structured Query Language (SQL) based scheme is used to query RDF data [6]. RDF represents a collection of triples of <subject, property, object> that can be easily stored in a relational database. Most of current research is based on efficient and scalable querying of RDF. However, approaches like RDQL [7], RQL [8], SPARQL [9], SquinshQL [10] define new languages for querying RDF data, which in turn, issue SQL to process user requests. But, due to the required transformation from the SQL data to the corresponding language, some data processing overheads involve. To overcome this problem, a scheme was presented which include a SQL table function RDF_MATCH to the query RDF data, which can search and infer from RDF data available on the SW.

RDF_MATCH function is implemented by generating SQL queries against tables that contain RDF data. To improve efficiency, subject-property matrix materialized join views and indexes are used. This join view is a group of subjects and single valued property occurring together in the same row. This procedure eliminates the additional joins occurring. Also indexes on data and frequently rule bases are used. Pre-computed triples, stored in a separate table, can be used during the query processing. Additionally, some kernel enhancements have been provided. It is an extension of RDBMS table function in the form of an interface. It allows the rewriting of table function as a SQL query and avoids copying results into table function variables, reducing the run time overhead. The proposed approach is an enhancement of this SQL-based scheme, which extends its functionality to remove redundant metadata from query results.

Redundant data are analyzed in [11], research that was primarily motivated by the setting of Internet advertising commissioners, who represented the middle persons between Internet publishers and advertisers. A bloom filters algorithm was developed; a comprehensive set of experiments were run, using both real and synthetic click streams, to evaluate the performance of the proposed solution. In the contexts of sliding and landmark stream windows, the space and time requirement for running was tested. Various applications including fraud detection, utilizes data streams for finding similarity among data. The technique is basically used for data streams and there is a difference between theoretical and practical error rates when applied on synthetic and real data. i.e., it depends on the nature of data used.

Duplicate web documents are studied in [12], and the strategies to eliminate them at storage level during the crawl. Architecture for storage system was designed, implemented, and evaluated, addressing the requirements of web documents. Duplicated web documents are detected before storing them on disk. Three modes checking fake duplicates need a comprehensive knowledge that how much chance is there for occurring duplicates in the collection.

Most of the research work done is based on duplication removal from the web documents either by using some specification [12] or other algorithms [11]. The problem of elimination of duplicate data from SW has not received sufficient attention. In order to overcome this problem, we utilize the metadata retrieved from the Web resources using SQL and then applying Knuth algorithm [13] to find out duplicate query results, as described as follows.

III. PRPOSED ALGORITHM

As a result of multiple resources retrieved from different heterogeneous information consulted, duplicate data can be obtained, for instance SQL-based approach [6]. Thus, we propose a methodology using SQL-based system (see Figure 1).

- The first query result is sent to the user, because no comparison takes place (see Figure 2). When it is the first run, index value from the query result is stored.
- For each data source, after getting the first result from the first data source, the query results is checked depending on their number, follow them processing.
- Each result undergoes the calculating hash process value [13] on the key value which in our case is the Web resource (URL page).
- Each result is compared with all stored data in order to check duplication.
- Each result is compared with all stored data in order to check duplication.

```
METHOD:  Detecting and eliminating duplicate metadata.
INPUT: Query results from semantic data source RDF.
OUTPUT:   Query results free of duplicate metadata.
1 DO   Read user query statement
2 WRITE Query to Data Source
3 READ Query Results
4 COMPUTE Hash Index
5 SAVE the Hash Index and Pointer In the Hash Table.
6 DISPLAY results to the user
7 WHILE not end of query results DO {WHILE LOOP}
8     READ Query Results
9     COMPUTE Hash Index
10      IF Hash Index values in STEP 9 IS EQUAL to
             Hash Index in STEP 4
        THEN
              IF Query Results in STEP 8 IS EQUAL to
                  Query Results in STEP 3
             THEN
                  DISCARD result
              ELSE
                SAVE Hash Index and Pointer in HashTable
                DISPLAY results to user.
              {END inner IF}
    ELSE SAVE Hash Index and Pointer in HashTable
            DISPLAY results to the user
  {END outer IF}
END {WHILE LOOP}
```

Fig. 1: Duplicate data removal algorithm

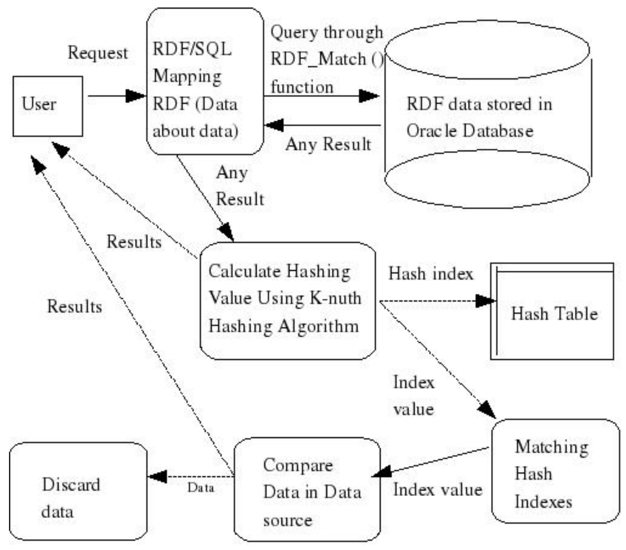
Each Web page metadata is collected and any attribute (e.g. URL) is taken as a key value. After getting the hashing value as an output of the hash function, it is stored with the pointer in a hash table and the result is displayed. Similarly, only more results are stored in the hash table if obtained results are not included. When the key values match, the data against the keys is checked to find collisions, if any. No result is displayed when data are already stored, decreasing the memory requirement and increasing the performance of the system.

A. Detection of Duplicate Query Results

Data are collected from heterogeneous sources during warehouse implementation or data integration process (see Figure 3). As many collected data can be similar, rather than comparing the whole information stored in the Web sites, and whether duplicate information exist or not, this information is recovered in the form of metadata. It is easier to determine when some metadata are similar: current obtained result with those already stored.

1). Hashing Function

The multiplicative scheme hash function [14] accepts an integer hash key and returns an integer hash result, producing a relatively normal distribution of values. The obtained value is used either as index to store or to find similar data. This key is the input for computing directly an address and then mapping it to some index value which is stored in the hash table.



Request
User
RDF/SQL Mapping RDF (Data about data)
Query through RDF_Match () function
RDF data stored in Oracle Database
Any Result
Any Result
Results
Results
Calculate Hashing Value Using K-nuth Hashing Algorithm
Hash index
Hash Table
Index value
Matching Hash Indexes
Index value
Compare Data in Data source
Data
Discard data

Fig. 2: Data flow diagram for removing duplicate data from Semantic Web

Thanks to this key unnecessary search is produced. A constant value is multiplied by the key value, then some necessary bits are extracted from it to index it into the table depending on the size of the table. For example, five most significant bits out of 25 table size are extracted. Equation 1 shows how it can be calculated [14].

$$h(K) = \lfloor M\,(((A/w)*K) \bmod 1)\rfloor \qquad \ldots 1$$

The steps are as follows:

- K key value is multiplied by the constant **A**, which lies in $0<A<1$, where w is the word size of the computer.
- *KA* is the fractional part extracted.
- This fraction is further multiplied by *M.*
- The floor is taken

This scheme is easy to implement and assure no information is lost. The ***M*** value usually takes a power of 2, which is not critical and may works efficiently on most of different computer architectures.

The complexity of the proposed algorithm is calculated in terms of the hashing function and in the number of the table entries to be compared. The complexity of hashing function is O(1) rather than O(n) as the data can be simply retrieved/looked up within a linear-time array [16]. If each query entry from SW has to be checked with all entries in the hash table, say *n,* then the complexity of the proposed algorithm is determined as follows: n*O(1).

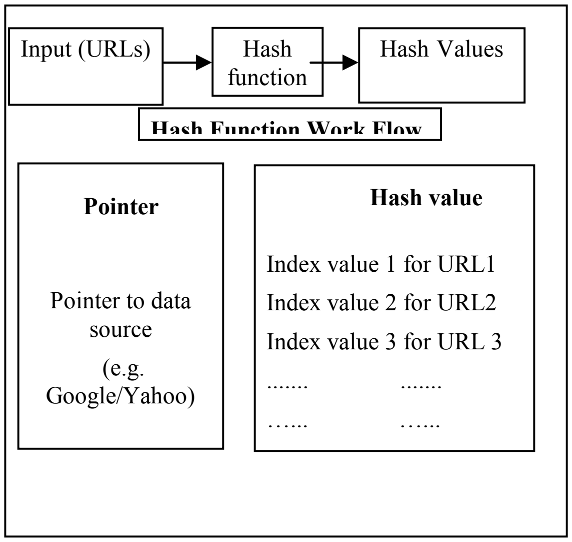


Fig. 3: Hash table working

The main advantages of the Knuth hashing algorithm along with SQL lie in:

1. It is an efficient solution, unless if SQL is the only language used for querying RDF data from SW, because the mapping requirement from other language to SQL is removed (e.g. SPRQL [9]).
2. An implicit pass-through architecture [15] can be used, which results in sending non-similar query results to a web user

So, overall overhead is reduced firstly, by using SQL and avoiding complexities caused by formats specified for representing RDF and second, by reducing memory requirement , which incorporates the storage of only unique query results. The main objective of our scheme is to enhance the efficiency and performance of the process.

In this process, queries are sent to the metadata sources once at a time, allowing the results from each source to be processed. When a user receives the results from SW source, a hash index is computed which is then stored in the hash table with a pointer to the first SW source.

The same procedure is performed for the following SW sources, but before storing the calculated hash index, it is compared with the index of the data source stored, avoiding index collision. When a pair of index values matches, another check is made to see whether query results gathered from both the sources are same or not. We decided to use as hash values for the URL because this can be the only attribute which vary from site to site.

Indeed, converting URLs directly into a hash value and comparing it, it is an efficient way rather than converting URLs into ASCII values. Also, hashing makes fast searching process as elimination of looking into the whole table. The following main steps are performed when user queries the data:

- The URLs are given as input to the system.
- This algorithm is passed to the system and its length is calculated.
- The hash valued is calculated using the Knuth hashing multiplicative scheme, and shift operators for finding the most significant bits.
- After calculating, the hash value is returned.
- All process is repeated for all URLs queried.

B. Removal of Duplicate Query Results

After finding similar results already queried and stored, its index is not stored and results are not returned/displayed to users. A Web spider is used for finding and removing duplicates, extracting the metadata information about Web pages for any specific query.

1). Search Engine Specification

There exists a large list of search engines, however, we chose two data sources: Google and Yahoo. Both support Semantic Web concepts like setting different attributes, search engine name, word/phrase to be searched, number of results needed, language, preferences, new information, etc.

2). Metadata

In order to specify what kind of metadata will be extracted, a list of check boxes is provided from which they can be selected. Also it is possible to add and delete some metadata, as required

As a result of using a search engine (e.g. Google) a list of metadata information along with URLs of web pages requested is obtained. After getting the collection of metadata, the steps described in Section 3 are applied.

IV. CASE STUDY

In order to evaluate the performance of our approach, a case study was used, taking the pass through architecture [15]. The metadata about the web pages containing queried information from different search engines and a web-spider was used. Some duplicate results were present within these obtained results. Then, we applied our approach on these data to get and eliminate duplicates data.

The indexing is done based on the URL page and given along with the related URL column. The index for the given results is computed using the Knuth hashing scheme [13]. The duplicate results received from both sources are highlighted as shown in Figure 4 and 5.

All user queries are sent to Google and Yahoo web data sources to find out RDF metadata information like ID, URL, title page, Kbs page size, last modified date. and time (see Figure 4 for Google and Figure 5 for Yahoo). The hashing index is calculated as explained in Section 3.

The highlighted results in Figure 4 and 5 (at serial number 1, 2, and 3) are the same. While, the un-highlighted sections (at serial number 4 and 5) are dissimilar data. As the results returned from Google only have no chance for results to be similar, the hash value for each one is stored in the hash table.

Any obtained results are stored until they are compared and verified that they are not duplications.

Figure 6 is the results after applying the proposed steps in Section 3. It shows the query results (at serial 1- 4) obtained from Google and Yahoo after removing duplicates. The un-highlighted results of Figure 4 and 5 are shown here as an effect of duplication removal process.

The above methodology discussed provides a way to build large warehouses, using RDF data for data integration purposes from multiple resources at a time. This does not only helps in getting related data (using RDF) but also eliminates similar data during integration process.

V. CONCLUSION AND FUTURE WORK

Search engine services enable users to recover metadata from Semantic Web sources that usually are heterogeneous, as well as distributed. Usually, many duplicate results are published under the Internet. Thus, when a compendium of precise information is required, it is suitable not to store redundant information. Although one particular data source may not return duplicate results, it is often possible to have duplicate results in overall set of returned results.

For removing duplicate information, some methods require to store the total obtained results, entailing storage space, memory for processing, and least performance. In our approach a hash function is used [14], to detect duplicate query.

As a future work, we will improve our technique by preserving records that may be ignored due to collision and may work on optimizing the self joins queries that usually occur during querying RDF data. The proper selection of join method and join order plays an important role in the efficient query processing. Also the problems with UNION OPERATOR for comparing RDF data which relies on column ordering based on matching values rather than matching data types will also be considered. Besides this an alternate storage representation based on partial or name space based normalization, for storing RDF triples may also be considered.

ID	Page URL	Hash Value	Page Title	Page Size (kb)	Last Modified Time
1	http://infolab.stanford.edu/~melnik/rdf/api-doc/org/w3c/rdf/model/package-summary.html	-8275171272461040767	RDF API Draft Revision 2001-01-19: Package org.w3c.rdf.model	7	01/19/01 18:31:35
2	http://infolab.stanford.edu/~melnik/rdf/api-doc/org/w3c/rdf/syntax/package-summary.html	2751386994955643173	RDF API Draft Revision 2001-01-19: Package org.w3c.rdf.syntax	5	01/19/01 18:31:35
3	http://infolab.stanford.edu/~melnik/rdf/api-faq.html	-4783385828439381862	RDF API FAQ	1	10/28/00 02:15:20
4	http://jodi.tamu.edu/Articles/v02/i02/Anutariya/	-5016994694055057729	RDF Declarative Description: Anutariya et al.: JoDI	10	Information not available
5	http://www.cs.rpi.edu/~puninj/rdfeditor/	6584039355575166544	RDF Editor	10	Information not available

Fig. 4: Query Results from Google

ID	Page URL	Hash Value	Page Title	Page Size (kb)	Last Modified Time
1	http://infolab.stanford.edu/~melnik/rdf/api-doc/org/w3c/rdf/model/package-summary.html	-8275171272461040767	RDF API Draft Revision 2001-01-19:Package org.w3c.rdf.model	7	01/19/01 18:31:35
2	http://infolab.stanford.edu/~melnik/rdf/api-doc/org/w3c/rdf/syntax/package-summary.html	2751386994955643173	RDF API Draft Revision 2001-01-19:Package org.w3c.rdf.syntax	5	01/19/01 18:31:35
3	http://infolab.stanford.edu/~melnik/rdf/api-faq.html	-4783385828439381862	RDF API FAQ	1	10/28/00 02:15:20
4	http://protege.stanford.edu/doc/users_guide/rdf_support.html	-8482678325992358661	RDF(S) Support	5	Information not available
5	http://www.w3.org/TR/2004/REC-rdf-mt-20040210/	-815960418018445888	RDF Semantics	228	02/10/04 15:29:29

Fig. 5: Query Results from Yahoo

ID	Page URL	Hash Value	Page Title	Page Size	Last Modified Time
1	http://jodi.tamu.edu/Articles/v02/i02/Anutariya/	-5016994694055057729	RDF Declarative Description: Anutariya et al.: JoDI	10	Information not available
2	http://www.cs.rpi.edu/~puninj/rdfeditor/	6584039355575166544	RDF Editor	10	Information not available
3	http://protege.stanford.edu/doc/users_guide/rdf_support.html	-8482678325992358661	RDF(S) Support	5	Information not available
4	http://www.w3.org/TR/2004/REC-rdf-mt-20040210/	-815960418018445888	RDF Semantics	228	02/10/04 15:29:29

Fig. 6: Results after applying the proposed method

REFERENCES

[1] Manola, F., Miller, E., RDF Primer. In Publication of W3C Semantic Web Activity, February 10, 2004.

[2] Dumbill, E., Building the Semantic Web, available on http://www.xml.com/pub/a/2001/03/07/buildingsw.html, March 2007.

[3] Taylor, C., An Introduction to Metadata, published by University of Queensland Library, Library Web site, available on http://www.library.uq.edu.au/iad/ctmeta4.html, July, 2003.

[4] Berners-Lee, T., Relational Databases on the Semantic Web, available on http://www.w3.org/DesignIssues/RDB-RDF.html, 1998.

[5] http://www.w3.org/2007/03/VLDB/

[6] Chong E. I, Eadon Das, G., Srinivasan J., An Efficient SQL-based Querying Scheme, Proceedings of the 31st International Conference on Very Large Data Bases, pp. 1216-1227, Trondheim, Norway, 2005.

[7] http://www.**w3.org**/Submission/2004/SUBM-**RDQL**-20040109

[8] Karvounarakis G., Alexaki, S., Christophides V., Plexousakis D., Scholl M., RQL: A Declarative Query Language for RDF, May in Proc. of International World Wide Web Conference, WWW2002, 7-11, 2002, Honolulu, Hawaii, USA.

[9] http://www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/.

[10] Miller L., Seaborne A., Reggiori A., Three Implementations of SquishQL, a Simple RDF Query Language, First International Semantic Web Conference (ISWC2002), Sardinia, Italy, published by Springer LNCS2342, pp 423-435, June 2002.

[11] Ahmed M., Divyakant A., Abbadi A., Duplicate Detection in Click Streams, In the publication of International World Wide Web Conference Committee (IW3C2), ACM 1-59593-046-9/05/00, May 10-14, 2005 pp. 12-21, 2005, Chiba, Japan.

[12] Daniel G., Andre L. Santos, Mario J. S., Managing duplicates in a web archive, *P*roceedings of the 2006 ACM Symposium on Applied Computing (SAC), Dijon, France, ACM 1-59593-108-2/06/000, pp. 818-825, April 23-27, 2006

[13] http://www.cs.utk.edu/~eijkhout/594LaTeX/handouts/hashing-tutorial.pdf

[14] Donald E. Knuth, Art of Computer Programming, vol. 3: Sorting and Searching 2nd edition, Published by Addison-Wesley Professional, Apr 24, 1998.

[15] Ramanathan V. Guha, Pass-through Architecture via Hash Techniques to Remove Duplicate Query Result, patent number 6081805, Netscape Communications, Corporation, Mountain View, California, USA, June, 2000.

[16] http://www.netlib.org/bibnet/tools/emacs/hash-1.00/ hash.pdf