IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 2, Apr-May, 2014
ISSN: 2320 – 8791 (Impact Factor: 1.479)
www.ijreat.org

An Efficient Data Retrieval Using MashQL in the Data Web

K.Saranya[1], T.Amruthavalli[2]

[1, 2]**Department of Computer Science and Engineering, S.K.R Engineering College, Poonamallee, Tamil Nadu, India**

Abstract

Presenting an efficient query formulation Language called the MashQL in which the MashQL query is seen as tree. Here the initial stage is to provide the authentication to the user for login into the system. The individual public and private key are generated for each and e user which is used to login into the account. MashQL queries are created to retrieve the result. The MashQL is used as a query instead of using as an application. At the back-end the extracted MashQL queries are converted to SPARQL queries and the final results are displayed to the requested user. User can search the results without knowing the schema of the data which will provide more reliability. Online mashup editor and Firefox add-on are the implementation scenarios of MashQL that are created for more efficient retrieval of data and evaluating the implementation on two large datasets. By introducing a Search-box on the top of MashQL to allow keyword search and which is used to filter the retrieved results.

Keywords: *Structured data, Data Web, SPARQL, Mashup, Dashboard, Graph Signature.*

1. Introduction

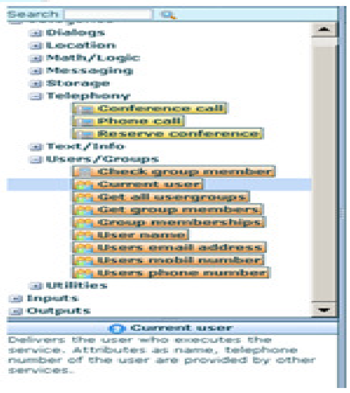
MashQL is a semantic data mashup language. The novelty behind the MashQL is to mashup, query and pipeline the user requested data intuitively. A use Semantic data structure which are usually represented in RDF format [11] results in smaller query efficiency. The companies such as Google Base, Yahoo Local, Freebase, Upcoming, Flicker, eBay, Amazon, and LinkedIn have made their content publicly accessible through APIs. In addition to these companies many of the companies have also started to adopt web metadata standards.

For example, Yahoo started to support websites embedding RDF and micro formats, by better presenting them in the search results; MySpace also started to adoptRDF for profile and data portability; Google, Upcoming, Slideshare, Digg, the Whitehouse, and many others started to publish their content in RDFa, a forthcoming W3C standard for embedding RDF inside web pages so that content can be better understood, searched, and filtered

[1]. A use of RDF and SPARQL as a query language to RDF structured data is often criticized, claiming that efficiency of such a such a procedure is very low and the same can be implemented using the SQL.

2. Motivations and Challenges

Traditional relational and object-oriented database systems force all data to adhere to an explicit schema which illustrated the semi-structured. The major challenges are, before formulating a query, one has to know the structure of the data and the attribute labels which represents a schema of the data. End-users are not expected to examine "what is the schema" each and every time they search or filter information or data from the web. In several cases, a data schema might be even dynamic.



Delivers the user who executes the service. Attributes as name, telephone number of the user are provided by other services.

Fig. 1. The Example represented in MashQL.

Mashup applications mix and merge content from multiple content providers in a user's browser to exhibit a high value web applications. Even other sources might be schema-free, or if it exists, the schema might be inline the data. Allowing end-users to query structured data flexibly is a great challenge and especially when a query involves multiple sources. Example: Figure 1 illustrates the use of keywords search and the necessary fields. Suppose a Web user wants to retrieve the previous conference call data and from the source. These sources do not only disagree completely on property labels (e.g., Phone call and Groups), but also on data semantics.

3. Related Work

In the existing system, Before formulating a query, one has to know the structure of the data and the attribute labels (i.e., the schema). End-users are not expected to investigate what is the schema each time they search or filter information. In many cases, a data schema might be even dynamic, i.e., many kinds of items with different attributes are often being added and dropped. Other sources might be schema-free, or if it exists. Traditional search engines cannot serve such data as the results of a keyword based query will not be precise or clean, because the query itself is still ambiguous although the underlying data is structured. Here the user must have a prior knowledge about the process content which would not display if the keyword is not matching. Allowing end-users to query structured data flexibly is a challenge, especially when a query involves multiple sources. Here the MashQL is used as an application.

3.1 Issues In Existing System

- Everybody must have a knowledge about the process.
- Content would not display if the keyword not matching

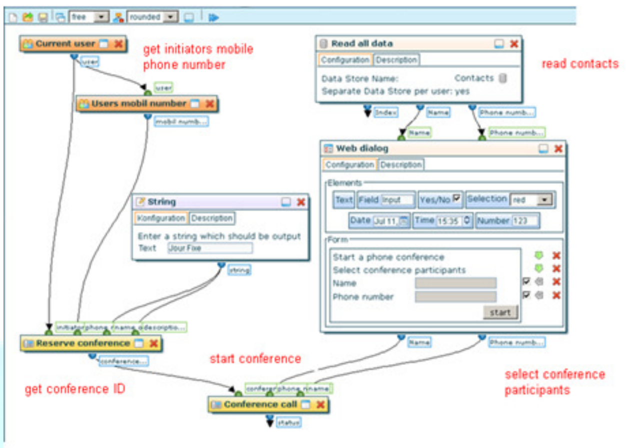


Fig. 2. The MashQL.

The simplest querying method is the Query-By-Form, but it is neither flexible nor expressive. For each MashQL query, a form needs to be developed; and changes to a query involve changing its form. Eventhough, some methods have been proposed to semi-automate form generation and modification but they generally fail with assumptions 2-4.

3.2 Query-By-Example

A known approach in databases, where users formulate queries as filling tables . However, it requires the data be schematized and the users to be aware of the schema (fails with assumptions 1 and 2).

3.3 Conceptual Queries

As many databases are modeled at the conceptual level using EER, ORM or UML diagrams, one can query these databases starting from their diagrams. Users can select part of a given diagram, and their selection is translated into SQL (ECR, RIDL, LISA, ConQuer, Mquery). These approaches assume that data has a schema and users have a good knowledge of the conceptual schema (fail with assumptions 1,2,3, and some with 4).

3.4 Natural Language Queries

It allows the people to write their queries as natural language sentences, and then translate these sentences into a formal language (e.g., SQL , XQuery). Hence, people are not required to know the schema in advance. The main problem is that this approach is fundamentally bounded with the language ambiguity – multiple meanings of terms and the mapping between these terms and the elements of a data schema (fails with assumptions 2, 3, and relatively 4).

3.5 Visualize queries

Several Semantic Web approaches (Isparql, RDFAuthor, GRQL, Nitelight) propose to formulate a SPARQL query by visualizing its triple patterns as ellipses connected with arrows, so that one would need less technical skills to formulate a query. Similarly, some tools had been also proposed to assist formulating XQueries graphically (Altova XMLSpy , Stylus Studio, Bea XQuery Builder , XML-GL , QURSED). Although these approaches vary in their intuitiveness they all intend to assist developers - rather than end-users, as they require technical knowledge about the queried sources and their Schemas/DTDs (fail with assumptions 1 and relatively with 2 and 4). In fact, they are close to the query-by-example approaches as they are studio-based query builders, but for semi-structured data.

Mashup Editors and Visual Scripting. Some mashup editors (e.g., Yahoo Pipes , Popfly , sMash) allow people to write query scripts inside a module, and visualize these modules and their inputs and outputs as boxes connected with lines. However, when a user needs to express a query over structured data, she has to use the formal language of that editor (e.g., YQL for Yahoo). Two approaches in the semantic web community (SparqlMotion and DeriPipes) are inspired by this visual scripting. For example, allows people to write their SPARQL queries (in a textual form) inside a box and link this box to another, in order to form a pipeline of queries. All of these visual scripting approaches are not comparable with MashQL, as they do not provide query formulation guide in any sense. They are included here, because MashQL is also inspired by the way Yahoo Pipes visualizes query modules. However, the main purpose of MashQL is not to visualize such boxes and links, but rather, to help formulating what is inside these boxes . Hence, it is worth noting that the examples of this article cannot be built using Yahoo pipes. Yahoo allows a limited support of XML mashups, using scripts in YQL.

3.6 Interactive Queries

The closest approach to MashQL is Lorel , which was developed for querying schema-free XML, and without assuming a user's knowledge about a schema. The difference between them: (First) Lorel partially handles schema-free queries. Like using the Graph-Signature in MashQL, Lorel uses a summary of the data (called DataGuide). However, unlike the Graph Signature, the DataGuide groups unrelated items as they extrinsically use same property labels, which lead to incorrect query formulation. In authors words, "we have no way of knowing whether O is a publication, book, play, or song. Therefore, a DataGuide may group unrelated objects together". To resolve this issue, the authors proposed the notion of Strong DataGuide; but the problem is that the size of a Strong DataGuide can grow exponentially in case the data is graph-shaped (rather than tree-shaped), thus, can be larger than the original graph: "the worst case running time is exponential in the size of the database, and for a large database even linear running time would be too slow for an interactive session". (Second) Lorel does not support querying multiple sources (assumption 3); and (Third) its expressivity is basic (assumption 4).

4. Proposed Solution

In the proposed system, an interactive query formulation language, called the MashQL is used. Being a language not merely an interface and at the same time, assuming data to be schema-free is one of the key challenges addressed in the context of MashQL design and development. Without loss of generality, this focuses on the Data Web scenario. This regard the Web as a database, where each data source is seen as table[2]. In this view, a data mash up becomes a query involving multiple data sources. To illustrate the power of MashQL querying RDF is mainly focused, which is the most primitive data model[7]. Hence, other models as XML and relational databases can be easily mapped into it. Use of this MashQL will provide the precise information of the data being retrieved. The keyword search is also implemented in the mashup editors and a graph signature algorithm is also implemented[4].

4.1 Advantage

- No need of prior knowledge about the database data.
- Keyword not needed for searching and query will make as per user assumption.

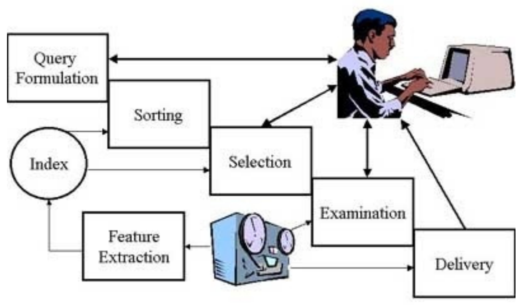


Fig. 3. Syatem Architecture Diagram

4.2 The Definition of MashQL

This section defines the data model, the syntax, and the semantics of MashQL. The discussion on how to formulate a query follows in the next section.

4.2.1 The Data Model

MashQL assumes the queried dataset is structured as (or mapped into) a directed labeled graph, similar to but not necessarily the exact RDF syntax. A dataset G is a set of triples <Subject, Predicate, Object>. A subject and a predicate can only be a unique identifier I (URL or a key). An object can be a unique identifier I or a literal L.

Def.1 (Dataset): A dataset G is a set of triples, each triple t is formed as <S, P, O>, where $S \in I$, $P \in I$, and $O \in I \cup L$.

The only difference with the RDF model is that we allow an identifier to be any form of a key (i.e. weaker than a URI). Allowing this, would simplify the use of MashQL for querying databases. Relational databases (or XML) can be mapped easily to this primitive data model. Figure 3 shows a simple example of mapping (or viewing) a database into a graph. The primary key of a table is seen as a subject, a column label as a predicate, and the data-entry in that column as an object. Foreign keys represent relationships between data elements across tables. Mapping from relational database and XML into RDF is a mature topic and is entering a standardization phase [4].

4.3 User Query Request

User Query Request is the process of mediating requests to data and services maintained by a specific application, determining the requests based on the user needs. Authentication is the first line of defense against compromising confidentiality and integrity. Though traditional login password based schemes are easy to implement, they have been subjected to several attacks. A separate public key and private keys are generated which is more authenticated for file uploading. The user will request a query for efficient retrieval of the result. The input will be in the form of a dataset G is a set of triples, each triples t is formed as <S, P, O>, where S belongs to I, P belongs to I and O belongs to I or L. The RDF model is that it allow an identifier to be any form of a key. Allowing this, would simplify the use of a MashQL for querying databases. Relational databases can be mapped easily to this primitive data model. The assumption is that each object literal to have a data type. If an object value does not have an explicit data type, it can be implicitly assumed, by taking advantage of XML conventions. A types literal is a literal object with a tag specifying its data type D. Every object literal must have a data type D.

5. Creation of Web Page Using Mash Queries

Mashup is seen as a query over one or multiple sources. Instead of developing a mashup as an application that access structured data through APIs. Here mashup is regarded as a query. A simple query language for the Data Web, in a mashup style. MashQL allows querying a data spaces without any prior knowledge about its schema, vocabulary or technical details (a source may not have a schema al all). The assumption of any knowledge about RDF, SPARQL, XML, or any technology to get started is to be known. Users can also use drop-lists to formulate queries which will be implemented in the mashup editors. A database is created for storage of user data and maintained by the administrator. A creative mashup editors are also created for retrieval of data from various sites. A MashQL query Q is seen as a tree. The root tree is called the query subject Q(S), which is the subject matter being inquired. A subject can be a particular instance I or a user variable V. Each branch is a restriction R, on a property of the subject branches can be expanded to allow sub trees, called query paths. This allows one to navigate through the underlying dataset and build complex queries. As a result a web page was created with necessary fields.

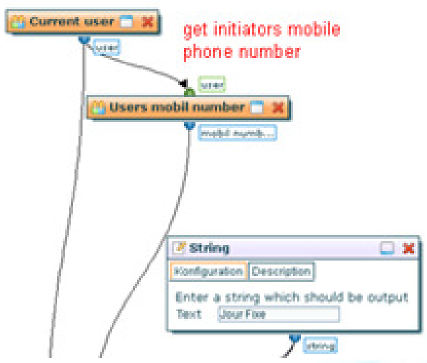


Fig. 4. Result of MashQL Query Execution.

6. Query Formulation Algorithm

This algorithm is used by the MashQL editor. Its novelty is that it, one to navigate through and query a data graphs without assuming the end-user to know the schema or the data to adhere to a schema. In query formulation algorithm, the responsibility of understanding a data source are moved from the user to the query editor. It allows end-users to easily navigate and query a data graph without prior knowledge about it, even if it is schema free. This algorithm is implemented in the mashup editors. Here the MASHQL queries are converted into SPARQL with the help of the data mining tool called the dashboard. The dashboards are installed in computers to monitor information in a database, dashboards reflect data changes and updates onscreen often in the form of a chart or table, enabling the user to see how the business is performing. Historical data also can be referenced, enabling the user to see where things have changed (e.g., increase in sales from the same period last year). This functionality makes dashboards easy to use and particularly appealing to managers who wish to have an overview of the company's performance.

6.1 The Query Subject Selection

That is, after specifying the dataset, users can select S from a dropdown list that contains, either: (i) ST: the set of the subject-types in G, such as Article or (ii) SI: the union of all subject and object identifiers in the dataset or (iii) a user-defined subject label. In the latter case, the subject is seen as a variable (S I V) and displayed the default subject is the variable label anything[7].

6.2 Property Selection

Depending on the chosen subjects in the select query subject, a list of the possible properties for this subject is generated. There are four possibilities:

(i) if (S I ST), such as Article, the list will be the set of all properties that the instances of this subject-type have (e.g., Title, Author, Year).

(ii) if (S I SI), such as A1, the list will be the set of all properties that this particular instances has.

(iii) If the subject is a variable (S I V), the list will be the set of all properties in the dataset.

(iv) Users can also choose the property to be a variable by introducing their own label. Add An Object Filter. There are three types of filters the user can use to restrict P: a filtering function, an object identifier, or a query path. A filtering function can be selected from a list.

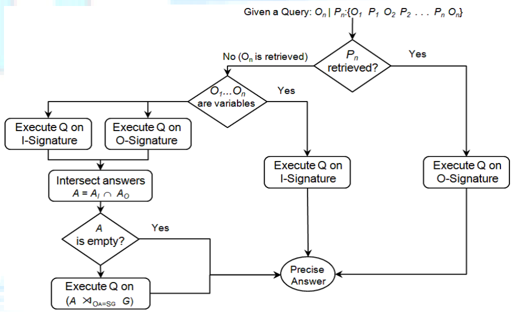

Given a Query: O_n | P_n:{O_1 P_1 O_2 P_2 . . . P_n O_n}
P_n retrieved?
No (O_n is retrieved)
Yes
O_1...O_n are variables
Yes
Execute Q on I-Signature
Execute Q on O-Signature
Execute Q on I-Signature
Execute Q on O-Signature
Intersect answers $A = A_I \cap A_O$
A is empty?
Yes
Execute Q on ($A \rtimes_{OA=SG} G$)
Precise Answer

Fig. 5. Working Model of Query Formulation Algorithm

7. Conclusion and Future Work

A query formulation language, called MashQL has been proposed. Four assumptions that a Data Web query language should have, and shown how MashQL implements all of them. The language-design and the performance complexities of MashQL are fundamentally tackled. Designed and formally specified the syntax and the semantics of MashQL, as a language, not merely a single-purpose interface. Specified the query formulation algorithm, by which the complexity of understanding a data source (even it is schema-free) are moved to the query editor. Addressed the challenge of achieving interactive performance during query formulation by introducing a new approach for indexing RDF data. Two different implementation scenarios of MashQL and evaluated the implementation on two large datasets. Allows people to discover and navigate unknown data spaces without prior knowledge about the schema or technical details. It Can be used as a general purpose data retrieval and filtering of information from various sources.

References

[1] Mustafa Jarrar, Marios D. Dikaiakkos : A Query Formulation Language for the Data Web, 2010.

[2] Abadi D, Marcus A, Madden S, Hollenbach K: Scalable semantic web data management using vertical partitioning. VLDB, 2007.

[3] Athanasis N, Christophides V, Kotzinos D: Generating On the Fly Queries for the Semantic Web. ISWC2004.

[3] Bloesch A, Halpin, T: Conceptual Queries using ConQuer–II. ER 1997.

[4] Chong E, Das S, Eadon G, Srinivasan J: An efficient SQL-based RDF querying scheme. VLDB'05, Springer. 2005

[5] Magesh Jayapandian, H. V. Jagadish: Expressive Query Specification through Form Customization, EDBT 2008.

[6] Goldman R, Widom J: DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. VLDB 1997.

[7] Jarrar M, Dikaiakos M: A Data Mashup Language for the Data Web. Proceedings of LDOW, at WWW'09. ISSN 1613-0073. 2009.

[8] Magesh Jayapandian, H. V. Jagadish: Automated Creation of a Forms based Database Query Interface**,** VLDB 2008**.**

[9] De Keukelaere F, Bhola S, Steiner M, Chari S, Yoshihama S:SMash: secure component model for cross-domain mashups on unmodified browsers. WWW 2008.

[10] Michalis Petropoulos, Yannis Papakonstantinou, Vasilis Vassalos : Graphical Query Interfaces for Semistructured Data: The QURSED System, in ACM SIGMOD International Conference on Management of Data, 2002.

[11] Klyne, G. and Carroll, J. : Resource Description Framework (RDF) : Concepts and Abstract Syntax W3C Recommendation, 2004.