An Automatic Query Generation For Data Web Search

Swagat Venkata Nadella[*], H Devaraju[#], Y Ramesh Kumar[*],

Final MTech student, Avanthi Institute of Engineering and Technology (JNTUK),Cherukupally, VizianagaramDist,

Andhra Pradesh, India

Assistant professor Department of CSE, Avanthi Institute of Engineering and Technology (JNTUK),Cherukupally,

VizianagaramDist , Andhra Pradesh, India

Associate Professor Department of CSE, Avanthi Institute of Engineering and Technology (JNTUK),Cherukupally,

Vizianagaram,Dist ,Andhra Pradesh, India

Abstract:-Using mobile applications to retrieve and manage data from remote servers is an essential service today. Due to this reason developers are continuously required to implement applications that perform this task. In this proposed system an automated mobile applications generator called MashQL is presented. MashQL is intended to ease developers' work by generating mobile query processing that requires accessing a remote database. It provides a user friendly interface and builds both, the client and the server sides. It generates Java ME applications at the client side. The remote databases are managed using JSP and My SQL. Once information has been retrieved, data has converted to an appropriate format in order to store it persistently on the mobile device using the Record Management System (RMS). In this paper display area and resource restrictions of small cell phone devices, currently, most reported database query systems developed for them are only offering a small set of pre-determined queries that can be posed by users.

Keywords: **Query formulation, RDF, MashQL, J2ME and Android.**

I. INTRODUCTION

In this paper, several declarative languages for querying and specifying views over RDF/S description bases have been proposed in the literature such as RQL, and RVL. However, these languages are mainly targeting experienced users, who need to understand not only the RDF/S data model but also the syntax and the semantics of each language in order to formulate a query or a view in some textual form. Building user friendly GUIs for browsing and seperating RDF/S description bases while exploiting in a transparent way the

Expressiveness of declarative languages like RQL/RVL is still an open issue for emerging SW technology. The idea of using suitable visual abstractions in order to access data collections originates from the early days of the database systems. The graphical interface presented in this paper aims to combine the simplicity of SW browsers with the expressiveness of DB query generators in order to navigate the Semantic Web by generating queries on the fly. The proposed interface, called GRQL, relies on the full power of the RDF/S data model for constructing, in a transparent to the end-user way, queries expressed in a declarative language such as RQL. More precisely, after choosing an entry point, users can discover the individual[11] RDF/S class and property definitions and continue browsing by generating at each step the RQL path expressions required to access the resources of interest.

Therefore, we believe that in order to have a generic database query system for small mobile devices, the system must be made supportive of different types of queries as well as unplanned queries. In doing so, the system should use minimal resources possible. Thus, in this paper, we shall introduce a new database query language that is suitable to be implemented as the query formulation method on display area and resource restricted mobile devices. In our study, we have implemented the language in a database query system[18] prototype for mobile phones. We advocate that if the language works on mobile phones, then it should be able to be adopted by other "thicker" small devices. Unplanned and imprecise queries performs very important role Today, with the advancement of technology in both data communication networks and accessing devices, the above activity can be carried out using many small mobile devices such as the Personal Digital Assistants (PDAs),Query language can provide a much simpler interface for users to formulate queries through mobile phones the systems only provide minimal querying capabilities. Hence, possible queries that can be formulated on these systems are mostly pre-determined by the developers as sets of options provided on a menu. Cell phones are poor in terms of its resources i.e. it contains less memory and also very small display area, so implementing such

method on them will make challenging and even applicable to other device also. It provides Interface to the user to formulate query. [10]Thus it has ability that it supports unplanned queries in order to make it as generic. It reduces number of queries as input, especially in the place of relations. Most of the queries are of these types, thus benefits resource in poor devices.

Hence, the remainder of this paper is organized as follows. Section 2 highlights some related works, Third Section introduces the main concepts of the query language, Section 4 presents the database query system prototype for cell phones, Section 5 discusses the usability tests conducted on the prototype, and Section 6 provides conclusions.

II.RELATED WORK

Mashup editors and visual scripting:

Some mashup editors allow people to write query scripts inside a device, and visualize these devices and their inputs and outputs as Boxes connected with lines. However, when the user wants to express a query over structured data, they have to use the formal language of that editor. Two approaches in the Semantic Web community are inspired by this visual scripting. For example, Tummarello et al allow people to write their SPARQL [11][15] queries inside a box and link this box to another box, in order to form a pipeline of the queries. All these visual scripting methods a are not comparable with MashQL, as they did not provide query formulation guide in any sense. They have included here, because MashQL is also inspired by the way Yahoo Pipes visualizes query devices. However, the main purpose of MashQL is not to visualizing such boxes and links, but rather, to help for formulating what is inside these boxes Hence, it is worth noting that the examples of this paper cannot be built using Yahoo pipes. Yahoo allows limited support for XML mashups, using scripts in YQL.

RDF QUERYING SCHEME

The results of RDF_MATCH table function can be further processed by SQL's rich querying capabilities and seamlessly combined with queries on traditional relational data. Furthermore, the RDF_MATCH table function invocation has rewritten as a SQL query, thereby avoiding the run-time table function procedural overheads. It also enables optimization[18] of rewritten query in conjunction with the rest of the query. The resulting query is executed efficiently by making use of B-tree indexes as well as specialized subject-property materialized views. This paper describes the functionality of the RDF_MATCH table function for querying RDF data, which can optionally include user defined releases, [20] and discusses its implementation in Oracle DBMS. It also presents an experimental study characterizing the overhead eliminated by avoiding procedural code at runtime, characterizing performance under various input conditions, and demonstrating scalability using 80 million RDF triples from UniProt protein and annotation data

Generating On the Fly Queries for the Semantic Web

Building user-friendly GUIs for browsing and filtering RDF/S description bases while exploiting in a transparent way the expressiveness of declarative query/view languages is vital for various Semantic Web applications. In this paper we present the novel interface, called as GRQL, which relies on the full power of the RDF/S data model for constructing on the fly queries expressed in RQL. More precisely, a user can do navigate graphically through the individual RDF/S class and property definitions and generate transparently the RQL path expressions required to access the resource of interest

Graphical Rql Interface (Grql):

These expressions capture accurately the meaning of its navigation steps through the class (or property) sub assumption and/or associations. Additionally the users can enrich the generated queries with filtering conditions on the attributes of the currently visited class while they can easily specify the resource's class (es) appearing in the query result. It is the best of our knowledge. The first application is GRQL application which is independent GUI able to generate a unique RQL queries which captures the cumulative effect of an entire user navigation session.

Furthermore, the word "unplanned" in database querying is rather subjective. This is because, like any other computer-related operations, there is always a limit to the kind of operations which can be executed. For database, this range of operations would [5] depend on the level of expressiveness that a query language exhibits. Chandra mentioned in his work that a language can support relational level of expressiveness at the lowest level to computable expressiveness at the highest end; and these languages can be of different forms. For example SQL is a textual language which exhibits at least relational level of expressiveness and QBE is a graphical language which is also capable of[13] supporting relational expressiveness. For mobile devices, only one work which discussed the issue of language expressiveness can be found. Polyviou, Samaras and Evripidou discussed expressive queries in their work which implement directory-like interface for query formulation. However, their method is suitable only to be used with devices that have pen input mechanism.

III.THE QUERY FORMULATION LANGUAGE

We present the query formulation language (called MashQL) in order to form a query on structured data on the web easily. The main novelty of the MashQL is that it allows people with limited IT skills to explore and query one [3][9] (or multiple) data sources without prior knowledge about the structure, vocabulary, schema or any other technical details of these sources. Most important thing is to be robust and cover most cases in practice, we do not assume that data source should contain an offline or in line schema. This posses several language-design and performance complexities that we tackle fundamentally. To illustrate the query formulation power of MashQL, and without loss of generality, [19]we chose the Data web scenario. We can also chose querying RDF, it is the primitive data model; hence, MashQL can be used for querying relational databases and XML. We present two implementations of the MashQL, an online mashup editor, and a Firefox can add on.

The MASHQL:

MashQL assumes the queried data set is structured as (or mapped into) a directed labeled graph, similar one but not necessarily the exact RDF syntax. A data set 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. 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.

Query Formulation Algorithm:

We present the novel query formulation algorithm, in which the complexity and the responsibility of understanding a data source [17] (even if it is schema free) are moved from the user to the query editor. This allows end users to navigate easily and query an unknown data graph(s). i.e,, people can learn the content and the structure of a data set while navigating it. The algorithm is not required the data to contain specific information or tags, except being symantically correct RDF, as discussed in the query model.

```
FROM Id Triples t1, Id Triples t2. Id Triples t3
 WHERE t1.PropertyID = 14 AND t2.PropertyID = 11
  AND t2. Object ID = 4    AND t3.PropertyID = 29
```

Then a self-join query is generated based on matching variables across triples (e.g. '?r') in the pattern:

```
WHERE t1.SubjectID = t2.SubjectID AND
      t2.SubjectID = t3.SubjectID
```

Next, the internal IDs are joined with the UriMap table to generate the join result in the URI (and literal) format:

```
SELECT u1.UriValue r, u2.UriValue c,
u2.Type c$type, u3.UriValue a,
u3.Type a$type
FROM  UriMap u1, UriMap u2, UriMap u3
WHERE  t1.SubjectID = u1.UriID AND
            t1.ObjectID =  u2.UriID AND
            t3.ObjectID =  u3.UriID
```

Note that 'r' is a URI, so there is no type associated, whereas 'c' and 'a' have a date type (c$type, a$type) associated.

Query language for mobile phones:

Prior to the construction of the query language, a small survey involving 45 fourth year students from the Department of Computer and Information Sciences (CIS) at University Technology PETRONAS, Malaysia (UTP) who had just returned from an eight-month industrial internship program, was conducted for gathering the types of queries users would normally issue to a database. In the survey, a test database schema (of a university scenario) was presented to the respondents in a narrative form. Based on the scenario, each respondent was asked to provide at least five possible queries that they may want to issue to the database. As a result, 262 queries were returned. Based on the major query operation each query required, five different query groups were identified. The five groups were selection, projection, join, set difference and union.

To cater for the above requirement, a free-form query language is proposed. Free-form is a concept which is based on the universal relation [14]. It can be used to reduce the number of terms needed in a query. Especially for queries of the join type, the number of query terms can be greatly reduced by not specifying the foreign key relationships between them. The universal relation concept has been known to be applied in keyword-based querying [7][8]. However, it Varies to keyword-based querying which uses Database instances as query terms, the approach that we opted for our language uses schema terms instead. It is due to several reasons which are related to the accessing devices that we are using, i.e., the cell phones.

IV.ANDROID

Figure .1 Android Phones

About J2ME:

In order to show that our language can support its intended capabilities, one prototype was developed. This prototype consists of a J2ME midlet for the interface on the Java phone emulator, and several Java servlets for the execution of queries. The interface developed follows the guidelines given in most references on J2ME such as those of Mahmoud [15] which suggested that an interface for small devices should be simple and use as many as possible high-level APIs.

Android is a software platform and operating system for cell phone devices, based on the Linux kernel, and developed by Google after that the Open Handset Alliance. It is allowing the developers to write managed code in the Java and, controlling the device via Google-developed Java libraries(Fig.1). Applications written in C and other languages can be compiled to ARM native code and run, but the development path has not officially supported by Google.

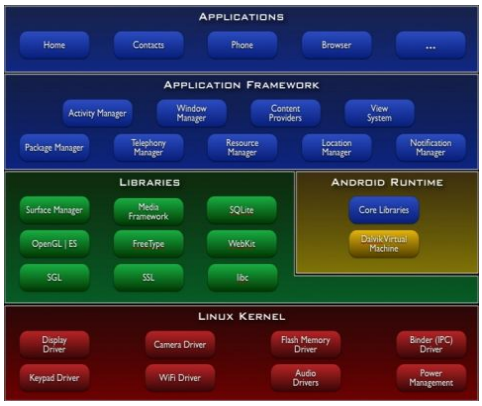

Figure.2 Application Framework

Open Handset Alliance:

Open Handset Alliance, is a consortium of several companies which including the Google, , Intel, Motorola, HTC, T-Mobile, Qualcomm Sprint Nextel and NVIDIA, ... These are the companies which aim to develop technologies that will significantly lower the cost of developing and distributing mobile devices and services. Android platform is the first step in this direction -- a fully integrated mobile "software stack" that includes of middleware and an operating system, user-friendly interface and applications.

Operating System:

Android uses Linux for its device drivers, memory management ,networking and process management. However you will never be programming to this layer directly. Next is the Android runtime, it includes the Dalvik Virtual Machine. It runs the dex files, which are converted at compile time from standard class and jar files. This files are more efficient and compact than class files, one important consideration for limited memory and battery powered devices are Android targets. The Android runtime is also a part of core java libraries.. They are written in Java, as is everything above this layer. Here, it provides a substantial subset of the Java 5 Standard Edition includes Collections, I/O, packages and so forth.

Architecture:

A central design point of the Android security architecture has no application, by default, it has permission to perform any operations that would adversely impact other applications, the operating system(Fig.2), or the user. This includes reading or writing the user's private data (such as contacts or e-mails), writing or reading another application files also, performing network access, keeping the devices awake, etc.

Performance:

Devices hosting Android applications have limited capabilities. That's why code should be efficient, avoid all unnecessary memory allocations, method calls (it takes a lot of time) and so on.

IDE and Tools:

Although it is possible to develop Android apps with every modern IDE Google recommends doing so is by using the Eclipse IDE with a special plug-in called ADT (Android Development Tools). The ADT

makes use of all the Dev Tools that come with the SDK and therefore supports and simplifies all the steps from assembling the classes over packaging and signing to running the final application on the emulator. The ADT is not just speeding up the testing process but also relieves the developers work in terms of UI creation and application description. For that reason the ADT offers the developer graphical representations of what has otherwise have to be written in XML. We can only hope that the next versions of Android have overcome the actual limitations and that the future possibilities became a reality.

Programming language:

The officially supported programming language on the Android platform is Java. It is also recommended to have some knowledge of XML as the descriptor file as well as the user interface of an application is based on that. As the Linux kernel of the Android platform is based upon an ARM processor architecture it would also be possible to write code in C or other languages and compile it to ARM native code.

V.SYSTEM PROTOTYPE

Figure.3 Free Form Query Language in Phone

The mobile phones which accepts different types of queries as well as unplanned queries, by allowing imprecise inputs. Since mobile phones are poor in terms of resources as compared to other mobile devices, the success of implementing such a method on them would mean it is applicable to the other devices. Free form language can provide a much simpler interface for users to formulate queries. The language helps in reducing the number of query inputs especially in cases where joins of relations are needed. Since the majority of queries which might be issued are of this type, providing such a method would benefit users of resource-poor devices. Using the structure above (Fig.3), queries such as q1, q2, q3 and q4 below are all valid queries. Queries, q1 combines two relations, while, q2 combines two attributes (not necessarily from the same table). A query which combines a relation and an attribute is also acceptable (see q3) and the order of the terms in the query can be reversible as well. A query, q4, which combines two conditions, is also allowable.

q1: STUDENT SUBJECT
q2: SUBJECT. Sub name STAFF. Stafname
q3: SUBJECT STAFF.stafname
q4:STAFF.stafID=’e0001’AND SUBJECT.subjcrhr>2

Furthermore, the language also allows a union or a set difference query to be implemented by including the respective operator once, anywhere in the query. However, the position of the set operator determines the components of

The query which needs to be manipulated. For example, a query, q5, requests for students to be combined with staff who teaches third year students.
q5: STUDENT.studname U STUDENT.studyear=3 STAFF.stafname
q6: STUDENT.studname STUDENT.studyear=3 U STAFF.stafname
q7: STUDENT.studname STUDENT.studyear=3 U STAFF.stafname TEMP1
q8: TEMP1 SUBJECT

Server

In our project, we are using MySQL as the data server and Apache Tomcat 5.0 as the web server. These two are the main core of the server side programming. Our application has been deployed in the Apache Tomcat so that all kind of Http Requests and response can be handled easily. All the parameters passed from the request can be retrieved using the get Parameter () method of Http Request Object.

Connection:

The Connection between the Client (i.e. J2ME application in mobile) and the Web Server is maintained by the object Http Connectionin javax. Microedition.io. Http Connection. Using this connection module we could retrieve the database information by passing the Http Request. The requesting attributes is sent as the parameters of the url built for Http Connection.

Design of the Application:

According to the request sent by the client, the server processes the data and it is responded to the client, which is further received by the help of open Data InputStream() in Http Connection Object. So the client application is designed according to the Field information of the tables retrieved from the server. *Activity Manager* Manages the lifecycle of applications. This cycle through various states: Started, Running, Background, Killed. Maintains a common stack allowing the user to navigate from one application to another. *Content Provider* Allows one application to make its data available to another. For example, the contacts application makes its data available to other applications that may need it. The phone or email application may need to consult contacts.

Query Generation:

The design of the application is done by the data types which we used in the database. The choice which we are about to use, according the query will be generated behind the screen. After the complete operation of selecting the choices we are supposed to execute the query by passing it as the parameter to the server through Http Request.

VI.CONCLUSIONS

We conclude that it is possible to develop a database query formulation system for smart phones which accepts unplanned queries, by allowing free-form inputs. As cell phones are poor in terms of resources as compared to other mobile devices, the successfulness of implementing such a method on them would mean it is applicable to the other devices. Indefinite query method in the form of free-form language can provide a much simpler interface for users to formulate queries. The method can also helpful in reducing the number of query inputs, particularly in cases where joins of relations are needed. Since most of queries which might be issued are of this type (as seen by the queries given by respondents), providing such a method would benefit users of resource smart phone devices.

REFERENCES

[1]. R. Alonso, and H. F. Korth, "Database system issues in nomadic computing", in Proc. 1993 SIGMOD Conference, Washington D.C., 1993, pp. 388-392.

[2]. K.Hung, and Y-T. Zhang, "Implementation of a WAP- Based telemedicine system for patient monitoring," *IEEE Transactions on Information Technology in Biomedicine*, Vol. 7, No. 2, June 2003, pp. 101-107.

[3]. A. Koyama, N. Takayama, L. Barolli, Z. Cheng, and N. Kamibayashi, "An agent based campus information providing system for cellular phone," in *Proc. 1st International Symposium on Cyber Worlds*, Tokyo,2002, pp. 339-345.

[4]. P. Boonsrimuang, H. Kobayashi, and T. Paungma, "Mobile Internet navigation system," in *Proc. 5th IEEE International Conference on High Speed Networks and Multimedia Communications*, Jeju Island, 2002, pp.325-328.

[5]. A. Bergstrom, P. Jaksetic, and P. Nordin, "Enhancing information retrieval by automatic acquisition of textual relations using Genetic programming," in Proc IUI 2000, New Orlean, 2000, pp. 29-32.

[6]. H-M. Lee, S-K. Lin, and C-W. Huang, "Interactive query expansion based on fuzzy association thesaurus for web information retrieval, " in Proc. IEEE International Fuzzy Systems Conference, Melbourne, 2001,pp. 724-727.

[7]. S. Agrawal, S. Chaudhuri, and G. Das, "DBXplorer: A system for keyword-based search over relational databases," in Proc. IEEE 18th International Conference on Data Engineering (ICDE'02), San Jose, 2002, pp. 5-16.

[8]. P. Calado, A.S. da Silva, A.H.F. Laender, B.A. Ribeiro-Neto, and R.C.Viera, "A Bayesian network approach to searching web databases through keyword-based queries, " Information Processing and Management, Vol. 40, No. 5, September 2004, pp. 773-790.

[9]. N. Athanasis, V. Christophides, and D. Kotzinos, "Generating On the Fly Queries for the Semantic Web," Proc. Int'l Semantic Web Conf. (ISWC '04), 2004.

[10]. BEA Systems, Inc., "BEA AquaLogic Data Services Platform - XQuery Developer's Guide. Version 2.5," 2005.