Volume 3, Issue 9, September 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Self-Regulating Query Generation in Information Web Search

S.Kavitha[1] , G.Hemanth Kumar Yadav[2]
Assistant Professor
Department Of Computer Science and Engineering
Madanapalle Institute of Technology and Science
Chittoor, Andhra Pradesh, India- 517 325

Abstract:-Mobile applications are used for retrieve and to manage the data from remote servers is an important service today. For this reason the developers are required to develop applications continuously that will perform this task. An automated mobile applications query generator called MashQL is presented. The MashQL is used to ease the developers' work. This will generate mobile query processing which is used for accessing a remote database. It will provide a user friendly interface for both, the client and the server sides and it generates the Java ME applications at client side. Remote databases are been managed by using JSP and My SQL. Hence once the information has been retrieved, the data will be converted into an appropriate format for storing it persistently on the mobile device by using the Record Management System. Here the display area and the resource restrictions of small devices like cell phones, currently, most of the reports specify that the database query systems developed for these devices that can be posed by users are only offering very small set of pre-determined queries.

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

I. INTRODUCTION

Several languages are been used for querying and for specifying views over the RDF/S description bases have been proposed in the literature such as the RVL, and RQL. These languages mostly targeting the experienced users, who want to comprehend not only the RDF/S data model, but also for learning the syntax and semantics of each language to formulate a query or view and translate them in some textual format. Making user friendly Graphical User Interfaces for browsing and then separating the RDF/S report bases while using them in a transparent way the expressiveness of the declarative languages is still an open and important issue for emerging Software technology. The main idea of using the suitable visual constructs for accessing the data collections that originates from the database systems of early days. The purpose of graphical user interface presented here is to combine the simplicity of software browsers with the feeling of database query generators to navigate the Semantic Web which generates the queries quickly. The interface specified in the proposed system, called as GRQL, which completely relies on the RDF/S data model to construct and in a transparent way for the end-user, the queries are been expressed in a declarative language. After choosing an entry point more precisely, the users can discover the individual [11] RDF/S class and their property definitions will continue browsing by generating path expressions at each step of the RQL that is required to access the interested resources.

So, we believe to have a generic database information query system for the small mobile devices, the system must be constructed for supporting of various queries as well as for the unplanned queries. In order to do so, the system must use the nominal resources possible. Thus, here in this paper, a new database information query language that is appropriate to be implemented for the query formulation method on the display area and also on the resource restricted mobile devices such as cell phones. Here the information query language is implemented in a database query system [18] prototype for the cell phones. If this language works on the mobile phones, then it is capable to be adopted by the other "thicker" small devices. For the unplanned and imprecise queries it will plays an important role today, with the improvement in the technology for both the data communication networks as well as for the accessing devices, the activity specified will be carried out by using many small mobile devices like Personal Digital Assistants (PDAs), this information query language can provide a much better interface for the users in formulating the queries through mobile phones and the system will only provide minimal querying capabilities. Hence, these possible queries that are been formulated on these systems are mostly determined previously by the developers as a set of options that are provided in menu. The Cell phones has less number of resources i.e. it contains very amount of less memory and also the display area is very small, so to implement such method on these will form a challenging task and these are even applicable to other device also. So Interface is provided to the user to formulate the query [10]. This has the ability to support the unplanned queries in order to make them as generic. It also reduces the number of queries as input in the place of relations. Most of these queries are of these types, which will benefit the devices with poor resource.

The remaining of this paper is organized as follows. Section II will highlights some related works, Section III introduces the important concepts of the query language, Section IV presents the database information query system prototype for the cell phones, Section V presents the usability tests conducted on this prototype, and Section VI presents the conclusion.

II. RELATED WORK

A. Visual scripting and Mashup editors:

Some of the Mashup editors will allow people to write the query scripts inside a device, and then visualize these devices with their inputs and outputs as the Boxes connected with these lines. However, when the user wants to present a query over the structured data, they need to use the normal formal language of that editor. There are two approaches in this Semantic Web community that are inspired by this visual scripting. Consider an example, Tummarello et al will allow people to write their own SPARQL [11] [15] queries inside a box and then link this box to another box to form a pipeline of these queries. The visual scripting methods all of them are not comparable with the MashQL, as they do not provide a query formulation guide. They have been included here as the MashQL is inspired by Yahoo Pipes visualizes query devices. But, the main use of the MashQL is not visualizing such boxes and their links, but rather for helping to formulate to know what is inside these boxes. So, the examples of this paper cannot be built using Yahoo pipes. Yahoo will allow limited support for the XML Mashups, which uses the scripts in YQL.

B. RDF Querying Scheme:

The results of the RDF_MATCH table function are further processed by the SQL's rich querying capabilities and combined seamlessly with the queries on traditional relational data. Further, an SQL query has been rewritten with the RDF_MATCH table function invocation, that is used to avoid the run-time table function procedural overheads. It also allows optimization [18] of the rewritten query in the conjunction with rest of the query. The resulting query is executed well by using the B-tree indexes as well as the specialized subject-property materialized views. The functionality of the RDF_MATCH table function is described for querying RDF data, which can will optionally include the user defined releases, [20] and it discusses the implementation in Oracle DBMS. It will also presents an new study characterizing the overhead which are eliminated by avoiding the procedural code at runtime, performance is characterized under different input conditions, and demonstrating the scalability using 80 million RDF triples from the UniProt protein and annotation data

C. Generating Fast Queries for the Semantic Web

Construction of user-friendly Graphical User Interfaces for the browsing and filtering RDF/S description bases by exploiting in a transparent way the expressiveness of the declarative query/view languages is very important for various Semantic Web applications. Here we present the novel interface, which are called as GRQL, which will rely completely on the full power of the RDF/S data model for building queries on the fly expressed in the RQL. More exactly, a user can do navigation graphically through individual RDF/S class and the property definitions and can generate transparently the RQL path expressions that are required to access the resources for information retrieval.

D. Graphical Rql Interface (Grql):

These GRQL expressions will capture precisely the meaning of its navigation steps by means of the class sub assumption and/or their associations. The users can develop the generated queries with the filtering conditions on attributes of the class currently visited. While these can easily specify the resource's classes that in which they are appearing in the query result. The first application is GRQL application, which is independent Graphical User Interface that is able to generate unique RQL queries which internments the increasing effect of an entire user navigation session.

Further, the "unplanned" word in database querying is rather independent. This is because, for computer-related operations there will be always a limit to the operations which can be executed. This range of operations will [5] depends on the level of expressiveness that an information query language exhibits for database. The mentioned work specifies that a language can support the relational level of expressiveness at the bottom level to computable expressiveness at the top end and the languages specified can be of different forms. Consider SQL is a textual language which demonstrations at least the relational level of expressiveness and on other hand QBE [13] is a graphical language which is capable of supporting relational expressiveness. For the mobile devices, only one work is specified which has discussed the issue of language expressiveness can be found. The Evripidou, Samaras and Polyviou discussed the expressive queries in their work, in which they implemented the directory-like interface for query formulation. But however, their method is suitable for only with devices that have the pen input mechanism.

III. THE QUERY FORMULATION LANGUAGE

The query formulation language is used to form a query on the structured data in web easily. The main purpose of the MashQL is it allows people to use it with limited IT skills to explore and perform queries one [3] [9] or more data sources without any knowledge about the vocabulary, structure, schema or any of the other technical details of these sources. The most important thing is about MashQL is to be robust and it cover most of the cases in practice, we do not assume that the data sources should contain an offline or online schema. This MashQL have several language design and various performance complexities that are been tackled fundamentally. To show the query formulation power of the MashQL, and without loss of any generality [19] we have chosen the Data web scenario. We have also chosen the querying RDF, which is the primitive data model. Hence, the MashQL can be used for querying the relational databases and also the XML. We have presented two implementations of the MashQL, one is online Mashup editor, and another a Firefox with add on.

A. The MASHQL: The MashQL assumes that the queried data set is structured as a directed labeled graph, which is similar one, but not the exact RDF syntax. Consider a data set G is a set of triples say <Subject, Predicate, and Object >.

A predicate and Subject can only be a unique identifier I (key). An Object can be a unique identifier I or can be a literal L. The difference between MashQL with the RDF model is MashQL allows an identifier as any form of a key i.e., it may be weaker than a URI. Thus allowing this will simplify the use of the MashQL for querying the databases. The Relational databases or XML can be mapped to this primitive data model easily.

B. Query Formulation Algorithm: A novel query formulation algorithm is presented in which complexity and responsibility of learning a data source [17] are been moved from the user to query editor. This will allow end users to navigate more easily and query an unknown data graph. The people are able to learn the content and structure of a data set while performing navigation on it. This algorithm does not require the data to have specific information or any tags, except are being semantically correct RDF, as they are discussed in the query model.

FROM Id triples t1, t2, t3
WHERE t1.PropertyID = 14 AND t2.PropertyID = 11
AND t2. Object ID = 4 AND t3.PropertyID = 29
Then a self-join query will be generated based on the matching variables across triples in the pattern:
WHERE t1.SubjectID = t2.SubjectID AND
t2.SubjectID = t3.SubjectID
Next, internal IDs are been joined with the UriMap table for generating the join result in the URI 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 the 'r' is a URI, so there is no type associated with it, whereas the 'c' and 'a' have a date type (c$type, a$type) associated with them.

C. Query language for mobile phones: Before to the construction of query language, a very small survey involving 45 IVth year students from the Department of Computer and Information Sciences at the University Technology PETRONAS, Malaysia, who had been just returned from their eight-month industrial internship program, that has conducted to gather the types of queries users have been normally issuing to a database. In this survey, a test database schema was presented to the respondents in a narrative form. According to this scenario, each of the respondents was asked to provide at least five possible queries that they may want to query to the database. Based on this a result of 262 queries were returned. On this major query operation each of the queries required five different query groups has been identified. These five groups were selection, join, projection, set difference and the union.

To provide for the above requirement, a free-form query language has been proposed. The Free-form is a concept which is based on universal relation [14]. This can be used to reduce the number of terms that are needed in a query. Mostly for the queries of the join type, the most number of query terms that can be greatly reduced by not specifying foreign key relationships between them. In keyword-based querying [7] [8] universal relation concept has been applied. But, free-form query language varies with the keyword-based querying which normally uses Database instances as the query term. The approach that has been opted for the language here uses schema terms instead. This is due to various reasons which are normally related to accessing devices that are been using, i.e., the cell phones.

IV. ANDROID

Figure .1 Android Phones

A. About J2ME:

To support our language to its intended capabilities, one prototype has been developed. This prototype consists of J2ME midlet as an interface on Java phone emulator, and there are several Java servlets for the execution of the queries. This interface that is developed follows the guidelines that are given in most of the references on J2ME such as those of Mahmoud [15] which has been suggested that an interface for the small devices should be simple and has to be used with as many as possible high-level APIs.

For cell phone devices the Android is a software platform and an operating system, this software is based on the Linux kernel, and has been developed by Google after the Open Handset Alliance. This has allowed the developers to write managed code in Java and then controlling the device with Google-developed Java libraries (Fig.1). The applications are written in C and we can run other languages that can be compiled to ARM native code and then run, but the Google has not officially supported development path.

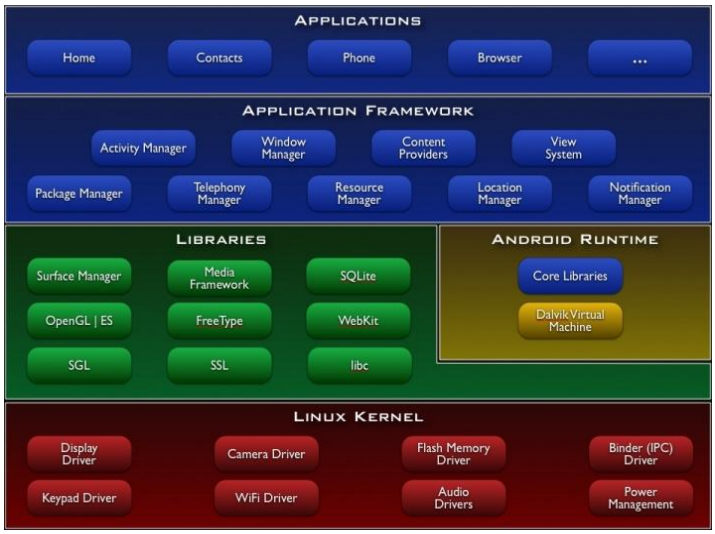

Figure.2 Application Framework

B. Open Handset Alliance:

Open Handset Alliance, is association of several companies which includes Google, Motorola, Intel, T-Mobile, HTC, Qualcomm Sprint Nextel and NVIDIA. These companies aim to develop the technologies which significantly reduce the cost of developing and distributing the mobile devices and their services. The Android platform is the first step to this direction which is a fully integrated mobile "software stacks" which includes a middleware and an operating system, user-friendly interface and some applications.

C. Operating System:

Linux is used in Android for its device drivers, networking, memory management and process management. But you are not programming to this layer directly. The Android runtime includes the Dalvik Virtual Machine, it runs the dex files, which are been converted at compile time to their standard class and jar files. These files are more efficient and compatible than the class files, one important thing to consider is the limited memory and battery powered devices are the main Android targets. Android runtime is also a part of the core java libraries. These are written in Java for everything above this layer. Here, it will provide a substantial subset of the Java 5 Standard Edition which includes Collections, I/O, and packages so on.

D. Architecture:

The central design point of this Android security architecture does not have any application. By default it has the permission to perform any operations that would impact other applications, the OS in Fig.2. This includes reading or writing the user's private data such as contacts or e-mails, writing or reading any another application files also, performing the network access, keeping all the devices awake, etc.

E. Performance:

Devices hosting the Android applications have limited functionalities. That is the reason the code should be efficient and avoid all unnecessary memory allocations, method calls and so on.

F. IDE and Tools:

As it is possible to develop Android applications with every modern IDE Google recommends, can be done by using the Eclipse IDE with a special purpose plug-in called Android Development Tools(ADT). The Android Development Tools uses of all the Dev Tools that comes along with the SDK and which supports and simplifies all the steps starting from assembling the classes over the packaging and signing them to run the final application on the emulator. The Android Development Tools is not only just speeding up the testing process but it also relieves the

developers work in terms of User Interface creation and the application description. For this reason the Android Development Tools offers the developer the graphical representations or else have to be written in XML. We can hope that the next versions of the Android will overcome the limitations and that the future possibilities will became a reality.

G. Programming language:

The formally supported programming language on Android platform is Java. It also recommends having some knowledge of the XML as the descriptor file as well as the UI of an application is based on that. The Linux kernel of Android platform is based upon ARM processor architecture, so it would also be possible to write the code in C or other languages and then compile them to ARM native code.

V. SYSTEM PROTOTYPE

Figure.3 Free Form Query Language in Phone

The mobile phones accept various queries as well as unplanned queries by taking imprecise inputs. As the mobile phones having fewer resources as compared to the other mobile devices, implementing such a method on them successfully would mean that it is applicable to any other devices also. The Free form language can provide simpler interface for the users to formulate the queries. This language will help in reducing the total number of query inputs especially in cases like where joins of relations are to be needed. Since majority of the queries which might be issued are mostly of this type, as long as such a method would benefit the users of resource-poor devices. By using the structure above in Fig.3 the queries such as q1, q2, q3 and q4 below are all the valid queries. The queries, q1 combines with two relations, while, q2 combines with two attributes which are not necessarily from the same table. A query is also acceptable which will combines a relation and an attribute (see q3) and order of the terms in query can be reversible as well. A query, q4, which will combines two conditions, is also allowable.

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

Further, the language will also allows a union or a set difference query can be implemented by including respective operator once, at anywhere in the query. But however, the position of set operator will determine the components of query which needs to be changed. For example, a query, q5, requests for the students to be combined with the staff who teaches third year students.

q5: STUDENT.studname U STUDENT.studyear=3
STAFF.staffname
q6: STUDENT.studname STUDENT.studyear=3 U
STAFF.staffname
q7: STUDENT.studname STUDENT.studyear=3 U
STAFF.staffname TEMP1
q8: TEMP1 SUBJECT

A. Server

Here, we are using MySQL as data server and Apache Tomcat 5.0 as web server. These two are the necessary as serves as core of the server side programming. Here this application has been deployed in Apache Tomcat so that all Http Requests and responses can be handled easily. The parameters passed from request can be retrieved using the method get Parameter () of Http Request Object.

B. Connection:

The Connection between the J2ME applications in mobile with the Web Server is maintained by object Http Connection in javax.Microedition.io. Using this connection the module can retrieve the database information by passing Http Request. The request attributes are been sent as parameters of the URL built for Http Connection.

C. Design of the Application:

Based on the request sent by the client, the server responses by processing the data and is responded to client, which is received by open Data InputStream () in the Http Connection Object. The client application is designed according to Field information of tables that are retrieved from the server. The Activity Manager will manage lifecycle of these applications. This cycle passes through various states starting from Started, Running, Background, and Killed. It also maintains a stack allowing users to navigate from one application to another. The Content Provider will allow one application make its data available to another. Consider an example like contacts application makes its data available for other applications that may be necessary. The phone or the email application may need to consult the contacts.

D. Query Generation:

The design of this application is done by using data types which are used in the database. The choice that is about to use, according to that the query will be generated at background. After the completion of operation by selecting the choices we execute the query by passing it the parameter to the server we can get our information.

VI. CONCLUSIONS

It is possible to develop a database query formulation system to retrieve information for smart phones which will accepts some unplanned queries, by allowing the free-form inputs. As the cell phones having fewer resources compared with other mobile devices, the most important thing in implementing a method on those devices would be very much necessary. Indefinite query method for information extraction in the form of free-form language can provide a simpler interface for the users to formulate the queries. This method will also helps in reducing the total number of query inputs, especially where joins of relations are necessary. Most of the queries which are issued of this type benefit users with 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.

[11] A. Bloesch and T. Halpin, "Conceptual Queries Using ConQuer- II," Proc. Int'l Conf. Conceptual Modeling (ER), 1997.

[12] S. Comai and E. Damiani, "Computing Graphical Queries over XML Data," ACM Trans. Information Systems, vol. 19, no. 4, pp. 371-430, 2001.

[13] E. Chong, S. Das, G. Eadon, and J. Srinivasan, "An Efficient SQLBased RDF Querying Scheme," Proc. Int'l Conf. Very Large Data Bases (VLDB '05), 2005.

[14] B. Czejdo, R. Elmasri, M. Rusinkiewicz, and D. Embley, "An Algebraic Language for Graphical Query Formulation Using an EER Model," Proc. Computer Science Conf., 1987.

[15] F. De Keukelaere, S. Bhola, M. Steiner, S. Chari, and S. Yoshihama, "SMash: Secure Component Model for Cross-Domain Mashups on Unmodified Browsers," Proc. Int'l Conf. World Wide Web (WWW), 2008.

[16] O. De Troyer, R. Meersman, and P. Verlinden, "RIDL on the CRIS Case: A Workbench for NIAM," Proc. IFIP WG 8.1 Working Conf. Computerized Assistance during the Information Systems Life Cycle, 1988.

[17] J. Dionisiof and A. Cardenasf, "MQuery: A Visual Query Language for Multimedia, Timeline and Simulation Data," J. Visual Languages and Computing, vol. 7, no. 4, pp. 377-401, 1996.

[18] R. Goldman and J. Widom, "DataGuides: Enabling Query Formulation and Optimization in Semi structured Databases," Proc. Int'l Conf. Very Large Data Bases (VLDB), 1997.

[19] E. Griffin, Foundations of Popfly. Springer, 2008.

[20] R. Henzinger, A. Henzinger, and W. Kopke, "Computing Simulations on Finite and Infinite Graphs," Proc. Ann. Symp. Foundations of Computer Science (FOCS), 1995.

