Publication of Little Lion Scientific R&D, Islamabad PAKISTAN
Journal of Theoretical and Applied Information Technology
15th April 2011. Vol. 26 No.1
© 2005 - 2011 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN IMPLEMENTATION OF ODBC WEB SERVICE

SEIFEDINE KADRY
American university of the Middle East, Kuwait

ABSTRACT

Web applications built using .NET technologies usually access relational databases via ODBC API. This requires a database system specific ODBC driver to be installed on the application side. On the other hand, a paradigm shift is taking place in web application architectures. Future web applications will be built around Service-Oriented Architectures (SOA) where applications will be assembled using remote “web services” components. These newly assembled applications will provide functionalities using remote web services components over Internet via XML messages. Same paradigm shift will eventually apply to the data communication between applications and databases. Future applications will utilize standard “database web services” for data storage and querying requirements. Here we propose a new architectural model and present a prototype “database web service” for relational database systems. In our model, web applications do not need to deal with database drivers but leave the driver-oriented communication to a database web service. Our database web service eliminates the need for installation, maintenance, and other issues involved in maintaining ODBC drivers in distributed web application development.

Index Terms—*Web Services, SOAP, ODBC, Distributed Databases.*

I. INTRODUCTION

Extensible Markup Language (XML) is emerging as the new standard for data and message exchange over Internet [4]. New distributed application integration frameworks such as “Web Services” are based on XML standards [5]. Web Services framework allows applications to exchange data and provide remote services using XML formatted data and messages.
Web services framework consists of the following standards: (1) Extensible Markup Language (XML) for formatting data and messages, (2) Simple Object Access Protocol (SOAP) for exchanging messages, (3) Web Services Description Language (WDSL) for describing services, and (4) Universal Description, Discovery, Integration (UDDI) for providing service directories. All of these standards are based on XML [5].

Current web services framework only provides an infrastructure for building distributed applications utilizing web services infrastructure. Many real-world business requirements such as
integrity, automation, transactional support, and security issues are not completely dealt with yet in the current framework. Ongoing work is trying to address these issues [6].

Most of data in the web is currently held in relational database systems today. Web applications simply provide an interface to query and present that data in HTML format to the end user; web browsers interpret and display the HTML tagged data in an intended format. Therefore, the data presented in HTML format is not usable by other applications; it is more intended for directing the browser to present data for human reading. On the other hand, providing direct access to databases is not feasible due to security, performance, scalability, and reliability issues. There is a need for a standard mechanism to allow applications to access databases just like users access data on databases via web browsers.

Earlier, ODBC provided a standard application programming interface (API) for applications to access relational database systems in a uniform way [1]. An application can access a remote or local relational database using these APIs as long as there is a vendor-specific driver provided for the intended database system. Currently, almost all database systems have ODBC compliant drivers developed. Problem with this approach is that a client application that wants to access a ODBC-accessible database needs to get a hold of the relevant driver for the database system and install on the client side. It is clear that this is not feasible for large-scale distribution of applications. If drivers get updated, this even complicates the issue with many problems: driver updates, maintenance of multiple versions, backward compatibility to name a few. Problem is exactly the same as in the case of early client-server based applications. Web provided a solution to this by a multi-tier application architecture where client is scaled down to a thin-client model with only presentation layer of the application on the client side, and the database access is only done in an application server on the server side.

Today’s computing devices are getting smaller continuously. Wireless-connected personal digital assistants (PDAs) and cell phones are ubiquitous. These devices with their limited resources are not capable of handling complex database drivers. Applications developed for small devices like these need other mechanisms to access data seamlessly. Database web service provides an option; an application developed for these devices can access remote databases via web services, and this approach requires only a small XML messaging client.

In this paper we address the need for universal access to data resources without complicated installation and maintenance issues. Specifically, we utilize the newly arrived Web Services approach for application integration and interoperability for data access. To this end, we developed a standard database web service. This web service is deployed on a server where the database is installed. And, querying and updates on the database are enabled via the database web service. Client on the other hand only needs a web services client application that will only send and receive messages in XML format (SOAP messages).

Section 2 discusses the architecture of this new model where access to relational database is provided via database (ODBC) web service. Section 3 presents the current implementation, an ODBC Web Service, and its advantages are discussed. We discuss implementation and evaluation of ODBC-WS in section 4. Section 5 presents the related work and we conclude in section 5.

II. ACCESSING RELATIONAL DATABASE VIA WEB SERVICES

Relational databases can be accessed by general-purpose using standard ODBC API [1]. This requires applications to use a ODBC driver specifically developed for ODBC access to the database system that needs to be accessed. Therefore, wide-distribution of such an application requires the driver to be also shipped to the client. Future changes to the driver require driver updates for every client.

Problem is even more complicated if the application needs to access many different relational database systems. Basically, this requires distribution and maintenance of many drivers in client-side. Here we propose a new approach for applications to access database systems. It is based on Web Services framework. In this approach, client is reduced to a thin client, just like web applications took the client-side computations to server side. In this case, database drivers are the problem, so they are placed on a server and moved away from the client. And, applications are provided a new interface to access database systems in a uniform way, similar to ODBC API provided a uniform access mechanism
for all relational database systems. Figure 1 depicts the proposed architecture. In this architecture, client accesses a “database web
service” using the standard web services access mechanism, which is sending and receiving XML SOAP messages over standard Internet protocols to request database (web) services.

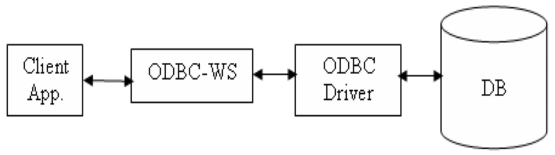

Fig. 1: web service-based access relational database

Publication of Little Lion Scientific R&D, Islamabad PAKISTAN
Journal of Theoretical and Applied Information Technology
15th April 2011. Vol. 26 No.1
© 2005 - 2011 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

III. ODBC WEB SERVICE

We implemented a database web service called ODBC Web Service (ODBC-WS). ODBC-WS will provide a standard ODBC-like interface that can be used by any Web Services client. ODBC-WS is accessed via a URL that is the connection string needed in ODBC. ODBC-WS clients do not need a database driver unlike usual ODBC clients. In the case of ODBC-WS, specific database and driver are completely transparent to the client. As long as the same web service interface (methods, parameters) are provided, database system and driver can be changed completely.

In the standard ODBC-based clients, a username and password needs to be provided for the database connection. Here, ODBC-WS can either authenticate the user once (or keep a session for consecutive operations) or every operation can be authorized via username/password passed along SOAP messages. ODBC-WS accepts standard SQL queries embedded within SOAP messages. Following are some examples:

Scenario 1: To query the names of authors from the author table in a database, XML message to be sent to ODBC-WS is:

```
<query>
SELECT name FROM author
</query>
```

And, response expected will be:

```
<result>
<row>
<name>J. Gosling</name>
</row>
<row>
<name>J. Ullman</name>
</row>
<row>
<name>D. Knuth</name>
</row>
…
<reccount>…</reccount>
</result>
```

Scenario 2: To insert a new author name to the author table in a database, XML message is:

```
<query>
INSERT INTO author (name)
VALUES (“J. Gray”)
</query>
```

And, if the record is inserted to the table successfully, the response expected will be:

```
<result>true</result>
```

Using a standard XML-interface to access relational and tabular databases, client applications access a number of different database systems using similar ODBC-WS interfaces and do not assume anything about specifics of the data source. Data sources are interchangeable and replaceable as long as the same interface is provided. This approach provides a complete distributed database framework for the client applications. Figure 2 provides a general overview of the distributed databases framework ODBC-WS provides over multiple kinds of data sources. ODBC-WS provides a number of advantages over standard ODBC access solutions:

1. Multiple client programs can use a single instance of the web service to access data. There is no need to write database related code in each individual application.
2. No driver installation or distribution in client-side.
3. Users need to install ODBC drivers and configure ODBC data sources only on the server machine that hosts the web service. Client programs can be running on other machines over the network.
4. Client programs are not limited to .NET or Windows platform. All they have to do to access database is call a web service.
5. Client is resilient to driver changes, updates.
6. Database connections can be shared among different client programs.
7. Client is resilient to database system changes. ODBC-WS hides all database system and corresponding driver installations from client.
8. Access to databases can be controlled and monitored from a central location (the web service).
9. If an “ODBC connection pooling” solution is provided in the server side, this is again completely hidden from client side and server side changes can be done anytime to increase the system performance as needed.

This web services approach can in effect be made transparent to the end user who can develop their applications using ODBC. They only need to make a few minor changes like the packages they include and the connection string. This can be achieved by

Publication of Little Lion Scientific R&D, Islamabad PAKISTAN
Journal of Theoretical and Applied Information Technology
15th April 2011. Vol. 26 No.1
© 2005 - 2011 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

providing client classes that override the existing ODBC classes. For example executing a query in the ODBC application would actually create an XML SOAP request message with the SQL query embedded in the message and that would be sent via HTTP to the remote server where it will be executed. The main advantage of this is that application programmers can use legacy ODBC-based applications with the new ODBC-WS implementation without making many changes.
The disadvantage of this approach could be that client and server side needs to process XML messages via standard XML parsing techniques. On the other hand, client side driver initiation, connection establishment is eliminated altogether. Therefore, performance evaluation of this approach needs to done against standard ODBC-based access.

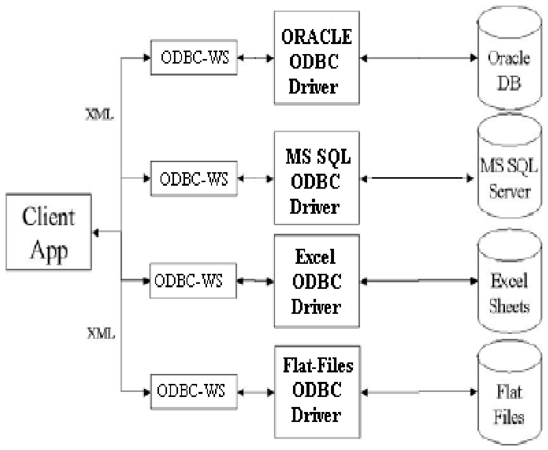

Fig. 2: distributed databases via ODBC-WS

IV. IMPLEMENTATION

The web service keeps a pool of database connections in its memory. When the client program calls GetXMLStringEx, ExecuteEx, and CreateObject, the server will use an existing connection (with the same connection string) if it is available and lock it for the client, otherwise it will create a new one.
At the end of the GetXMLStringEx and ExecuteEx methods or when ReleaseObject is called, the locked database connection will be returned to the connection pool for future use.
Whenever a database failure occurs, the connection will be closed by the server automatically, the server will also try to reconnect to the database whenever needed. It is not necessary to close a database connection explicitly. The web service will write errors and diagnostic information to log files in a system *Log* folder. A log file will be generated each day. By modifying the *web.config* file, you can control how logging
is done. The following is part of the *web.config* file. TraceLevel determines how much information will be logged, the possible values are 0, 10, 20, 30, and 40, where 0 means no
logging and 40 means the most detailed logging. TraceCleanup is the number of days to keep the log files, old files will be deleted automatically.

```
<appSettings>
<add key="TraceLevel" value="40" />
<add key="TraceCleanup" value="7" />
</appSettings>
```

A prototype ODBC-WS package has been developed [9] using Java and Java Web Services Developer Pack from Sun Microsystems [3]. Package utilizes the external configuration files to set up the system for any data resource such as relational databases, spreadsheets, or flat files, without recompilation. In the configuration file, system admin specifies the data source driver, its location, connection parameters, connection pooling configuration, and other ODBC-WS setup parameters. Current implementation's architecture is depicted in Figure 3. Current implementation follows the ODBC API specification [1]. It implements the DriverManager, Connection, ResultSet, and Statement interfaces. These interfaces basically provide a gateway for accessing Current implementation is using Oracle 10g database as the data source and the Oracle provided ODBC drivers are used in the testbed. ODBC-WS API we developed is used by the testbed to access the ODBC data sources to query the database.

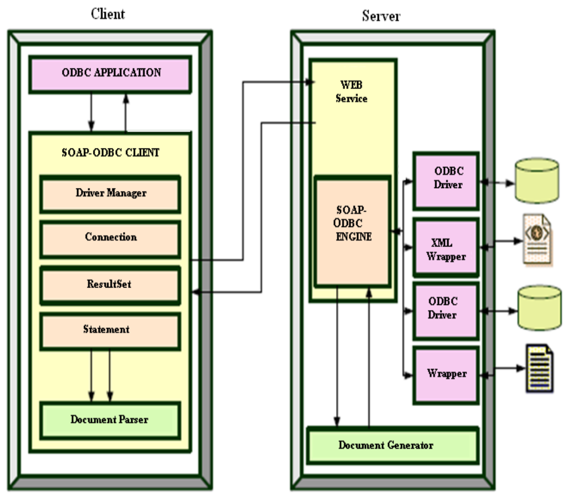

Fig. 3: SOAP-ODBC design

System is currently being tested and evaluated against a standard ODBC implementation. A wide-variety of scenarios are going to be evaluated for both cases, such as queries with short results vs. long results, select queries vs. update queries, standalone vs. concurrent access, etc. System will be also be evaluated to report performance over different data resources.

V. CONCLUSION

Web services framework has a promising future in distributed computing area due to the ubiquity of Internet and overwhelming industry support for web services. Web services will bring the true interoperability and application integration in a language and platform agnostic manner. Databases have provided excellent data storage and querying mechanisms. A natural approach is to provide database access over web services.
We have developed a new database access framework in that database servers are hidden behind a new database web service. Database web service provides a uniform SQL based querying interface and query results are returned to the client in XML format. We have developed a prototype version of database web service called ODBC-WC. ODBCWC, when installed for a database server, provides a database and driver transparent interface to the database for any client application that can implement SOAP messaging.
Currently, the system is being evaluated and tested. The work presented in this paper is part of a larger framework we are working on, where ODBC-WC will be used as a standard interface for any data source with SQL querying capabilities. In the extended framework we will address the following issues among others: transaction processing (web services composition), concurrency control, distributed query evaluation and optimization, database replication, distributed integrity constraint and constraint enforcement.

APPENDIX

Appendixes, if needed, appear before the acknowledgment.

REFERENCES

[1] G. O. Young, “Synthetic structure of industrial plastics (Book style with paper title and editor),” in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.
[2] W.-K. Chen, *Linear Networks and Systems* (Book style).Belmont, CA: Wadsworth, 1993, pp. 123–135.
[3] H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.
[4] B. Smith, “An approach to graphs of linear forms (Unpublished work style),” unpublished.
[5] E. H. Miller, “A note on reflector arrays (Periodical style—Accepted for publication),” *IEEE Trans. Antennas Propagat.*, to be published.
[6] J. Wang, “Fundamentals of erbium-doped fiber amplifiers arrays (Periodical style—Submitted for publication),” *IEEE J. Quantum Electron.*, submitted for publication.
[7] C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995.
[8] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interfaces(Translation Journals style),” *IEEE Transl. J. Magn.Jpn.*, vol. 2, Aug. 1987, pp. 740–741 [*Dig. 9th Annu. Conf. Magnetics* Japan, 1982, p. 301].
[9] M. Young, *The Techincal Writers Handbook.* Mill Valley, CA: University Science, 1989.
[10] J. U. Duncombe, “Infrared navigation—Part I: An assessment of feasibility (Periodical style),” *IEEE Trans. Electron Devices*, vol. ED-11, pp. 34–39, Jan. 1959.
[11] S. Chen, B. Mulgrew, and P. M. Grant, “A clustering technique for digital communications channel equalization using radial basis function networks,” *IEEE Trans. Neural Networks*, vol. 4, pp. 570–578, July 1993.
[12] R. W. Lucky, “Automatic equalization for digital communication,” *Bell Syst. Tech. J.*, vol. 44, no. 4, pp. 547–588, Apr. 1965.
[13] S. P. Bingulac, “On the compatibility of adaptive controllers (Published Conference Proceedings style),” in *Proc. 4th Annu. Allerton Conf. Circuits and Systems Theory*, New York, 1994, pp. 8–16.
[14] G. R. Faulhaber, “Design of service systems with priority reservation,” in *Conf. Rec. 1995 IEEE Int. Conf. Communications*, pp. 3–8.

[15] W. D. Doyle, "Magnetization reversal in films with biaxial anisotropy," in *1987 Proc. INTERMAG Conf.*, pp. 2.2-1–2.2-6.

[16] G. W. Juette and L. E. Zeffanella, "Radio noise currents n short sections on bundle conductors (Presented Conference Paper style)," presented at the IEEE Summer power Meeting, Dallas, TX, June 22–27, 1990, Paper 90 SM 690-0 PWRS.

[17] J. G. Kreifeldt, "An analysis of surface-detected EMG as an amplitude-modulated noise," presented at the 1989 Int. Conf. Medicine and Biological Engineering, Chicago, IL.

[18] J. Williams, "Narrow-band analyzer (Thesis or Dissertation style)," Ph.D. dissertation, Dept. Elect. Eng., Harvard Univ., Cambridge, MA, 1993.

[19] N. Kawasaki, "Parametric study of thermal and chemical nonequilibrium nozzle flow," M.S. thesis, Dept. Electron. Eng., Osaka Univ., Osaka, Japan, 1993.

[20] J. P. Wilkinson, "Nonlinear resonant circuit devices (Patent style)," U.S. Patent 3 624 12, July 16, 1990.

[21] *IEEE Criteria for Class IE Electric Systems* (Standards style), IEEE Standard 308, 1969.

[22] *Letter Symbols for Quantities*, ANSI Standard Y10.5-1968.

[23] R. E. Haskell and C. T. Case, "Transient signal propagation in lossless isotropic plasmas (Report style)," USAF Cambridge Res. Lab., Cambridge, MA Rep. ARCRL-66-234 (II), 1994, vol. 2.

[24] E. E. Reber, R. L. Michell, and C. J. Carter, "Oxygen absorption in the Earth's atmosphere," Aerospace Corp., Los Angeles, CA, Tech. Rep. TR-0200 (420-46)-3, Nov. 1988.

[25] (Handbook style) *Transmission Systems for Communications*, 3rd ed., Western Electric Co., Winston-Salem, NC, 1985, pp. 44–60.

[26] *Motorola Semiconductor Data Manual*, Motorola Semiconductor Products Inc., Phoenix, AZ, 1989.

[27] (Basic Book/Monograph Online Sources) J. K. Author. (year, month, day). *Title* (edition) [Type of medium]. Volume(issue). Available: http://www.(URL)

[28] J. Jones. (1991, May 10). Networks (2nd ed.) [Online]. Available: http://www.atm.com

[29] (Journal Online Sources style) K. Author. (year, month). Title. *Journal* [Type of medium]. Volume(issue), paging if given. Available: http://www.(URL)

[30] R. J. Vidmar. (1992, August). On the use of atmospheric plasmas as electromagnetic reflectors. *IEEE Trans. Plasma Sci.* [Online]. *21(3).* pp. 876—880. Available: http://www.halcyon.com/pub/journals/21ps03-vidmar