SQL+PaWS: *SQL and People as Web Services*

Steve Goschnick
University of Melbourne
Agent Lab & IDG Group,
Department of Information Systems
111 Barry St, Australia, 3010
+61 3 8344 1527
stevenbg@unimelb.edu.au

ABSTRACT

This paper introduces SQL+PaWS - a simple technology that allows SQL databases and people to become minimalist Web Services. The data delivery mechanism is a simple two dimensional uniform table in HTML, used to represent sets or a vector of data. SQL+PaWS has two levels of use both of them are easy and flexible: SQL+PaWS is *readable* by people and is also machine-readable by client-side programs that consume web services. SQL+PaWS is *writable* either by people with web-page editors following a simple protocol, or via a small piece of code that retrieves data from SQL-oriented database management systems. The users who wish to use SQL sourced data feeds can prototype and hone their SELECT statements via a single text-area field in a HTML form - almost all web-servers have an SQL DBMS on them of some sort, giving broad availability. Individuals who author web-pages with editors such as DreamWeaver, can become providers of web-services, in turn read by people or by software. The client-side software may be simple consumers of just the data such as a spreadsheet program, or it may be used to put together mashup applications, or even something as sophisticated as a DSS (Decision Support System) or other data warehousing products such as a part of a data warehouse extraction phase.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *Interaction Styles*. C.2.4 [**Computer Systems Organization**]: Computer-Communication Networks – *Distributed Applications*.

General Terms

Management, Human Factors, Standardization.

Keywords

Internet interfaces, web services, end-user computing, SQL, DBMS, HCI, SOA, SaaS, client-server, DSS, software agents.

1. INTRODUCTION

This paper describes the general adoption and usage of a small piece of technology written in a few days but heavily used everyday since. It uses two mundane, broadly adopted technologies - SQL and the humble HTML <table> tag - to deliver *web services* which are easy to prototype easy to host (all technology needed is already in place) easy to test and easy to use. Initially it was developed to make management data available to users at home, and to Information System (IS) students in the lab and at home. The invention of it came out of frustration with the formal definitions of web services: their continually growing complexity, the different vendor interpretations and shifting standards, and the transient nature (on air, off air) of some of the freely available SOAP [4] and WSDL [5] web services. The management need was to access certain workplace-hosted data from home through the web browser, and the student need was for multiple data streams from different web services, brought together in mashup applications - which they model and develop in lab exercises. In both cases, the underlying need was to use existing technology, that didn't require the installation of a web service server.

2. People Can be Web Services Too

The use of SQL+PaWS is potentially very generic and includes a convention that allows people to post their own data in a particular HTML format. I.e. Individuals who routinely collect data, can simply post it on the Internet as a Web Service to other people and/or organizations, without the need to use an SQL database at all. Figure 1 shows that either people *and/or* SQL-oriented databases can be the *source* of an SQL+PaWS web service, and either people *and/or* software clients can be the *consumers* of SQL+PaWS web services.

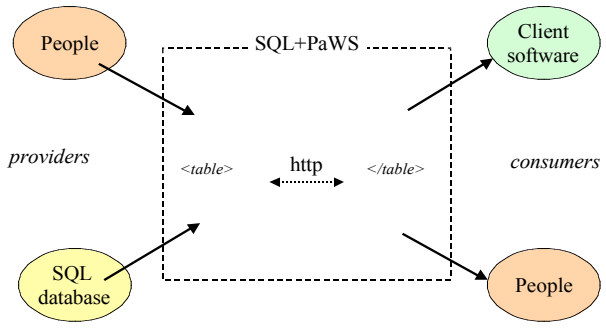


Figure 1. Overview of SQL and People as Web Services

Often, what a manager or an office worker needs on the road or at home, is not data from some remote 3rd party server but the data from ones own work-related web servers. And whether its data from work's web servers or from other 3rd parties out there on the Internet, what is often needed format-wise, is a simple set of data in a tabular 2D format. However, almost all of the web service technologies and stacks (SOAP, WSDL, XML-RPC, and others) that return data over the Internet protocols (HTTP, SMTP) in a machine-readable format, return it in a format that is hierarchically structured.

Figure 2 shows an example country code/currency code SQL+PaWS web service, as it is displayed in the browser, while figure 3 shows a manually constructed SQL Select command within the simple HTML form interface to SQL+PaWS, which produced the result seen in figure 1 – same URL. While figure 2 is what a human *consumer* of the web service will see in the web browser, a client-side application program consumes the same data programmatically. Similarly, figure 3 shows the HTML form interface that an end-user would use to prototype and test an SQL command, but once the SELECT statement works satisfactorily for an often-used data retrieval, it can then be cut-and-pasted into a client application program which will automate the web service call, as in the dialog-window based interface shown in figure 4.

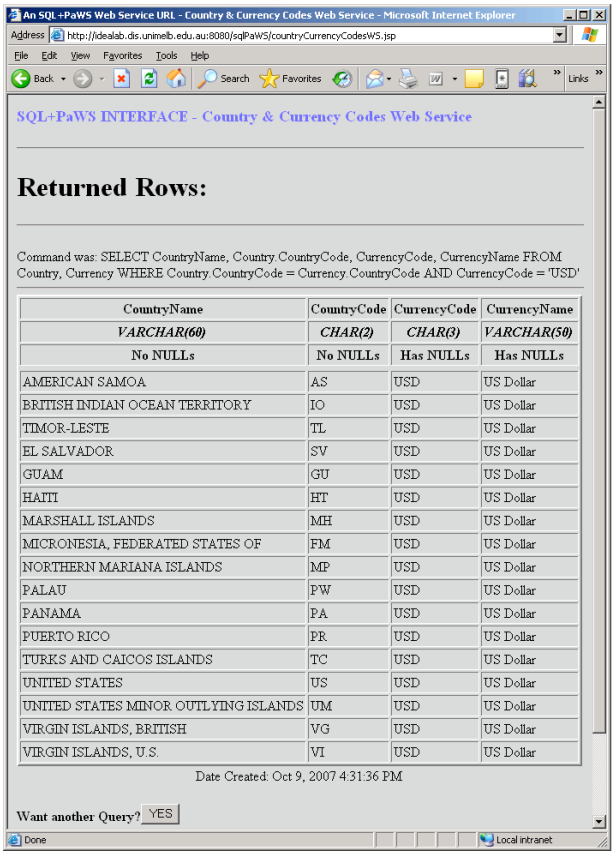
Clearly, a HTML table like that in figure 2 can easily be *authored* by a person, rather than produced by an SQL Select statement querying against a database. A human *consumer* of the table is none-the-wiser. A client application program using it merely needs to be told that it does not require an SQL statement, and can thereafter simply retrieve the table from the URL it is given, as with a SQL database produced table.

3. The Mechanics and Protocol

The HTML table carrying the data payload for SQL+PaWS is always a 2 dimensional uniform grid i.e. set-based data, including a vector (a single row). It may be imbedded any where within any sort of HTML file with any number of other tables. The start of the SQL+PaWS table is marked with an *anchor* tag – see line 1 in Listing1 below. (Nb. Standard HTML usually uses such anchor tags - a variant on the link tag - to allow browsers to start the display of a page, part way down the file). The opening table tag <table> immediately follows the relative anchor. See the SQL+PaWS Specification [6] for more details.

Listing 1. HTML code fragment with essential insertions

```
<a name="START-SQL-PaWS"></a>
<table>
<caption align="bottom">Date Created: Sep 28,
2007 1:27:12 PM</caption>
<thead valign="top">
<tr>
    <th>CountryName</th>
    <th>CountryCode</th>
    <th>CurrencyCode</th>
    <th>CurrencyName</th>
</tr>
<tr>
    <th>VARCHAR(60)</th>
    <th>CHAR(2)</th>
    <th>CHAR(3)</th>
    <th>VARCHAR(50)</th>
</tr> ...
</thead>
<tbody>
<tr>
    <td>AMERICAN SAMOA</td>
    <td>AS</td> ...
</tr> ...
</tbody>
</table>
<a name="END-SQL-PaWS"></a>
```



CountryName	CountryCode	CurrencyCode	CurrencyName
VARCHAR(60)	*CHAR(2)*	*CHAR(3)*	*VARCHAR(50)*
No NULLs	No NULLs	Has NULLs	Has NULLs
AMERICAN SAMOA	AS	USD	US Dollar
BRITISH INDIAN OCEAN TERRITORY	IO	USD	US Dollar
TIMOR-LESTE	TL	USD	US Dollar
EL SALVADOR	SV	USD	US Dollar
GUAM	GU	USD	US Dollar
HAITI	HT	USD	US Dollar
MARSHALL ISLANDS	MH	USD	US Dollar
MICRONESIA, FEDERATED STATES OF	FM	USD	US Dollar
NORTHERN MARIANA ISLANDS	MP	USD	US Dollar
PALAU	PW	USD	US Dollar
PANAMA	PA	USD	US Dollar
PUERTO RICO	PR	USD	US Dollar
TURKS AND CAICOS ISLANDS	TC	USD	US Dollar
UNITED STATES	US	USD	US Dollar
UNITED STATES MINOR OUTLYING ISLANDS	UM	USD	US Dollar
VIRGIN ISLANDS, BRITISH	VG	USD	US Dollar
VIRGIN ISLANDS, U.S.	VI	USD	US Dollar

Figure 2. HTML Table produced, with data names and types

After locating the correct table via the relative anchor, the standard HTML table <caption> tag is used to hold a time-stamp of the publication of the data. Next, the metadata (attribute name and type) necessary to identify the data attributes held in the table proper, are put within the standard HTML <thead> *metadata* </thead> tags. Then the actual data is held in rows (the <tr> tags) all within the standard HTML <tbody> *data* </tbody> tags.

So, for those people-authored pages, what is the incentive for the authors to write their pages in this SQL+PaWS format over simply publishing them in any old HTML formatted page? Primarily motivation is that the pages of the same format can be written by the SQL database generated web services too, and that there is a vast number of such databases out there on the Internet, holding much of the world’s data. Any client software applications that can read the SQL generated web services, can also read the people-authored pages. People are on an equal footing as sources of data.

Note: There is a second version of the form presented in figure 3, which includes both a user-name and a password field, for secure access to a particular SQL+PaWS web service.

4. Subtle Change in Thinking over Time

It was a very subtle change in thinking needed to bring SQL+PaWS about - the sort of change that is often more difficult to make than when a large change in thinking is necessary. While it took less than a day to write the code behind the web-interface in figure 3 (even though it is totally general purpose against any SQL-oriented database on a web-server), it took several months beforehand to gradually see the

subtle changes needed to web services to make them this simple to build and use, via technology already in place. So, a question that comes to mind, to which I try to uncover possible answers in the next section, is: *why hadn't it been done before?*

When a web-publisher or a casual web-page author thinks of the <Table> construct in HTML it is often associated with thoughts and experiences of some of the less-predictable behaviour of HTML editing. The non-2D-grid format variant of HTML tables have predominantly been used within web page content, to carve up well-designed graphics and artwork, making some parts of those graphics interactive. Tables used as such, have given graphic designers the ability to control the display of their carefully constructed graphic elements in a WYSIWYG (what-you-see-is-what-you-get) manner, in an otherwise non-WYSIWYG technology (i.e. HTML in the Web Browser). However, the humble HTML table can also clearly be used to describe a simple uniform 2D table, and that in turn can be used to define attribute names and attribute types in a uniform and machine-readable manner too - and that constitutes a *data schema*, metadata that can be read accurately and with purpose by both people and application software.

The other well-established convention in web services is to encode all machine-readable information into a tagged language predominantly based on the XML language. Given that XML is for the definition of hierarchical structures, it means that even set data will end up in an often deeply structured hierarchy. It is useful to reflect on the history of the tagged languages to see how this sits with SQL-oriented relational DBMSs which are also termed *structured databases*.

4.1 Structured and Textural Databases

Prior to tag languages being in the opposite corner to structured database management systems (relational DBMS), the so-called free-text searching (or textural) DBMSs filled that position - represented by products on mainframes a generation ago with names like STAIRS and BASIS, upon which vast collections of textual information was stored, indexed and searched. These were the precursors to the Internet and to Google and the other search engines. Technically oriented libraries (the sort that hold books and have readers visit them) used the likes of STAIRS and BASIS to index and retrieve free-text documents - files with large text fields such as the abstract of a paper, and another field holding the whole paper itself. These technical librarians and their information analyst associates and their needs eventually led to the first great tag language SGML, the mother-language of both HTML and XML, but more akin to XML. Users of both SGML and XML can create their own tags via a separate schema file, much as one does in the internal schema of a relational DBMS.

So, prior to XML there were two disparate but highly professional languages in the management of information realm, serving two distinct markets: SQL for structured DBMSs, and SGML for free-text search DBMSs. Then along came XML and in particular XML Schema the variant of XML schema language you use to define your own tags, (rather than the earlier alternative schema language: XML DTD). XML together with XML Schema effectively represents a merging of the two previously divergent main streams of DBMS - both *structured* data and *free-text/textural*. But do we always want or need a language that merges both structured and free-text data? Will it be the most efficient, or the most user-friendly method to use a language that merges structure data and free-text data?

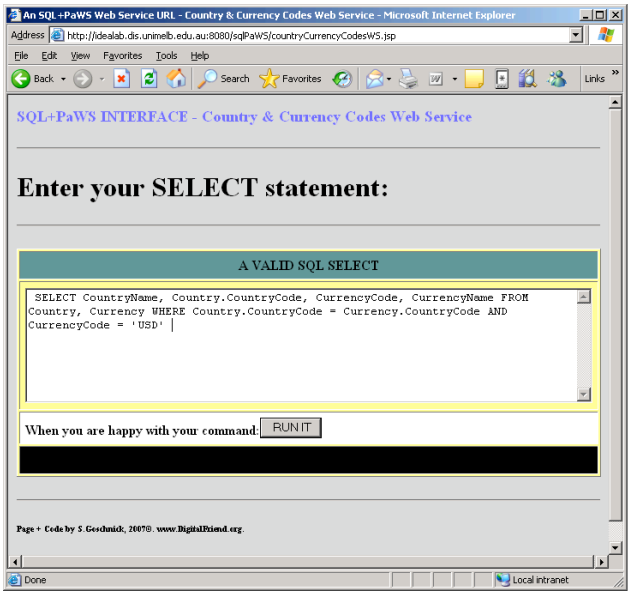


Figure 3. The simple HTML input form with an SQL select command that produces the table in figure2

When you discard all of the technicalities, it all comes down to *sets* (flat files) and *hierarchies* (trees structure files). In a relational DBMS new sets can be projected from joining relationships between existing sets using the SELECT command, which features in figure 3. Some things naturally require or use hierarchies - the sort of textual data in a book that is described by the index of that book, is the sort of information hierarchy that XML can easily accommodate. Some applications thrive on sets - regular, rectangular arrangements of data, with a fixed number of columns (attributes) and a variable number of rows (records) - for example the sheets in a spreadsheet program.

There are things that XML shines at and things that SQL shines at but often they are different things. Hierarchies are good for representing (programmed) objects and their relationship with other objects. Sets are good for representing uniform groupings of like data. SQL+PaWS is designed for set data and data vectors (a single row of attributes – similar to the attributes that a program procedural call will return, as in RPC – remote procedural call).

Because web services (e.g. SOAP, WSDL) have been delivered over Internet transport protocols (mainly HTTP), the use of XML-derived languages is the default choice for such purposes. Web Services often carry data from server-side SQL-oriented DBMS, and that data gets passed through a number of hierarchically-structured XML defined tag languages before getting to a user, even though it is very often placed back into a form or a spreadsheet, which is back in a set format! In the case of set data (the only sort held in a Relational DBMS), SQL+PaWS bypasses those many levels of tag-based language, except for HTML itself.

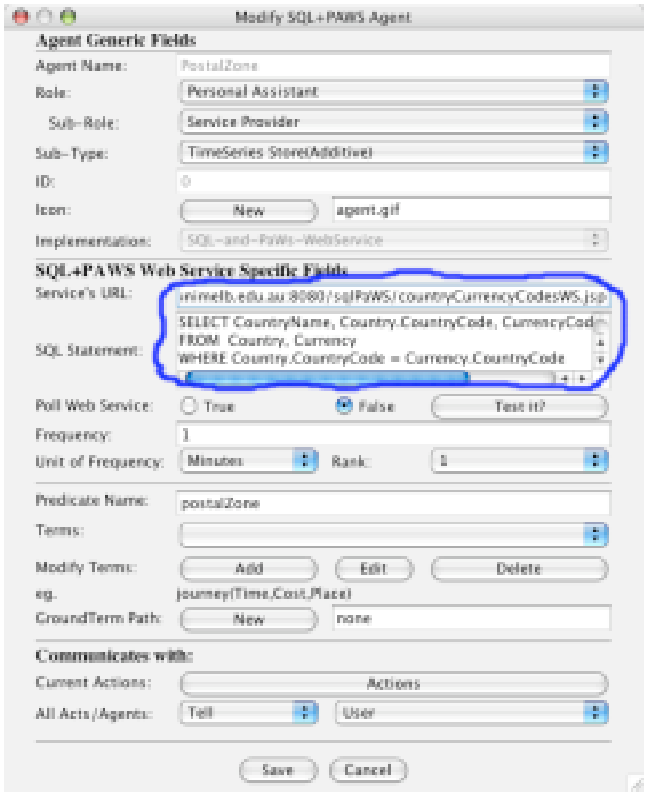


Figure 4. Simple cut'n'paste of tested URL and SELECT into a client-side tool – the DigitalFriend in this case

5. CONCLUSION

The concept of Web Services is simply about providing an interface to web/distributed systems that *our software* can use, in an analogous way that we humans are happily using the Web browser interface. I.e. I would like my software to *find, monitor, filter* and *condense* information from the Internet, while I am happily doing something else like social networking or something more leisure-oriented. Then it should present what its found, when I need it, not before and not after. Web Services are the Internet-side of making that a reality. Anything more technical than that description of Web services is just that - *technicalities*. By considering the use of mundane technologies alone for the delivery of web services (in this case: SQL-oriented DBMS on the server – a very common situation), came the realization that all of the extra technology to make conventional web services work, can be bypassed and is probably hindering the large scale building and usage of them.

The use of the HTML table to present subsets and projections of data and metadata from database systems has previously been overlooked perhaps due to two factors: a). The HTML table has been appropriated [1] by the authors of web pages to do WYSIWYG graphic design. b). There has been a headlong rush to present all data that needs to be machine readable, in an XML derived <tag> language designed for the purpose (e.g. WSDL, XML-RPC), when a perfectly serviceable, appropriated and domesticated tag language is broadly available and used in the form of HTML.

The SQL relational DBMS is one of the main technologies behind the success of Web 2.0 companies [3] - it is a state-machine in marriage with the stateless protocol of the Internet HTTP protocol. Most large organisations hold their operational data in a relational DBMS. Whether their DBMS is Oracle, Sybase, MS SQL Server, IBM's DB2, MySQL or PostgreSQL, they are all capable of running SQL Standard queries via the flexible and powerful SELECT command [2]. Most website hosting companies offer an option with a Relational DBMS backend and a server-side scripting language (either: ASP, JSP or PHP – used by SQL+PaWS server-side) to build dynamic web pages. SQL+PaWS offers a simple way to harness those resources as low-tech, low technical threshold web services, without any extra infrastructure or installation.

SOAP and WSDL web services have often failed to allow us to cross boundaries between work, school and home, partly because of the complexity of setting them up, and partly because of evolving and burgeoning standards involved (e.g. SOAP 1.1, WSDL 1.1, WSDL 2.0, WS-Adressing, WS-Security, WS-ReliableMessaging, WS-Policy, etc.) – i.e. they *still* represent a moving and technically expanding target. SQL+PaWS is a simple, open source, low-tech solution that crosses these boundaries by using only mundane technologies - going from sets at the source end, and presented as sets at the consumer end, with little transformation through hierarchical file structures enroute. While it uses two of the most widely used computer languages available - SQL and HTML - it also allows people to be either the *source* or the *consumer* of the data in such a web service. The client-side software may simply be a consumer of just the data such as a Web browser, a spreadsheet program, or it may be used to put together mashup applications, or even something as sophisticated as an agent system, a DSS (Decision Support System) or other data warehousing products such as a part of a data warehouse extraction phase.

6. ACKNOWLEDGMENTS

My thanks to the many students of my subject titled Database Systems and Information Modelling at the University of Melbourne for being the reason and the need, and hence the inspiration for SQL+PaWS web services, and the first users of it.

7. REFERENCES

[1] Carroll, J., Howard, S., Peck., J. and Murphy, J. (2003). From adoption to use: the process of appropriating a mobile phone, Australian Journal of Information Systems, 10:2, 38-38.

[2] Date, C.J. 2005. Database in Depth: Relational Theory for Practitioners, O'Reilly Press, 230 pages.

[3] O'Reilly, T. 2005. What Is Web 2.0. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

[4] W3C. 2000. Simple Object Access Protocol (SOAP) Version 1.1. http://www.w3.org/TR/soap/

[5] W3C. 2007. Web Service Description Language (WSDL) Version 2.0. http://www.w3.org/TR/wsdl20/

[6] SQL+PaWS. 2007. SQL+PaWS Specification. http://www.DigitalFriend.org/technology/SQL+PaWS/