29 November 2010 (v3, 14 December 2010)

jSPyDB, an open source database-independent tool for data management

Antonio Pierro for the CMS Collaboration

Abstract

Nowadays, the number of commercial tools available for accessing Databases, built on Java or .Net, is increasing. However, many of these applications have several drawbacks: usually they are not open-source, they provide interfaces only with a specific kind of database, they are platform-dependent and very CPU and memory consuming.

jSPyDB is a free web based tool written using Python and Javascript. It relies on jQuery and python libraries, and is intended to provide a simple handler to different Database technologies inside a local web browser. Such a tool, exploiting fast access libraries such as SQLAlchemy, is easy to install, and to configure. The design of this tool envisages three layers. The front-end client side in the local web browser communicates with a backend server. Only the server is able to connect to the different databases for the purposes of performing data definition and manipulation. The server makes the data available to the client, so that the user can display and handle them safely. Thanks to jQuery libraries, moreover, this tool allows data exportation to different formats, such as XML and JSON. Finally, by using a set of pre-defined functions, users are allowed to create their customized views for a better data visualization.

In this way, we optimize the performance of database servers by avoiding short connections and concurrent sessions. In addition, security is enforced since we do not provide users the possibility to directly execute any SQL statement.

Presented at *CHEP2010: International Conference on Computing in High Energy and Nuclear Physics 2010*

jSPyDB, an open source database-independent tool for data management

Giuseppe Antonio Pierro[1], **Francesca Cavallari**[2], **Salvatore Di Guida**[2], **Vincenzo Innocente**[2]

[1] INFN-Bari - Bari University, Via Orabona 4, Bari 70126, Italy
[2] CERN Geneva 23, CH-1211, Switzerland

E-mail: antonio.pierro@gmail.com

Abstract. Nowadays, the number of commercial tools available for accessing Databases, built on Java or .Net, is increasing. However, many of these applications have several drawbacks: usually they are not open-source, they provide interfaces only with a specific kind of database, they are platform-dependent and very CPU and memory consuming.

jSPyDB is a free web-based tool written using Python and Javascript. It relies on jQuery and python libraries, and is intended to provide a simple handler to different database technologies inside a local web browser. Such a tool, exploiting fast access libraries such as SQLAlchemy, is easy to install, and to configure. The design of this tool envisages three layers. The front-end client side in the local web browser communicates with a backend server. Only the server is able to connect to the different databases for the purposes of performing data definition and manipulation. The server makes the data available to the client, so that the user can display and handle them safely. Moreover, thanks to jQuery libraries, this tool supports export of data in different formats, such as XML and JSON. Finally, by using a set of pre-defined functions, users are allowed to create their customized views for a better data visualization.

In this way, we optimize the performance of database servers by avoiding short connections and concurrent sessions. In addition, security is enforced since we do not provide users the possibility to directly execute any SQL statement.

1. Introduction

Database Management Services are a crucial element for the success of the CMS experiment due to the huge amount of detector data that needs to be handled. In principle, access to the CMS database should be granted to each user, to permit the checking of data by means of some Oracle database client. However, the CMS database comprises millions of records and even one bad SQL query could impact on the performance of the database. For this reason it is not safe to allow each and every user to run unchecked queries on the database. However by means of web interfaces it is possible to control features available when accessing the database. Indeed, in order to help non-expert users, the client provides a simple graphical interface to relational database.

Each event/action is mapped to a specific method: the user can list the databases or tables simply by clicking on a drop-down menu; he can view the contents of a data table in a simple grid format on a web page that also supports querying, sorting, filtering, and joining data across web services without having to know Structured Query Language.

Because it is built using javascript (front-end) and SQLAlchemy (back-end), it can access any database engines simultaneously running on any machine, allowing remote access to multiple databases. In detail, using *jSpyDB*, the user can:

- easily view and edit data in any database engines,
- view the databases meta-data such as: time and date of creation, creator or author of data, column names and types for a given table such as String, Integer, default values etc,
- work with multiple databases on both local and remote machines,
- use a single, consistent interface to work with different database engines.

This paper is organized as follows. Firstly we discuss strategies that motivate the use of *jSpyDB*. Secondly, we list the advantages and disadvantages of existing open source client tools to manage the database server. Following this, we go through the details of *jSpyDB* three-tier architecture, the *Presentation-tier*, the *Logic-tier* and the *Data-tier*. Finally we discuss open issues affecting the future of jSPyDB, most of which relate to new web technology that is currently under development e.g. *web sockets*, and describe practical experience using the tool.

2. Motivation

Many people encounter difficulties in understanding concepts of relational database, table design, SQL statements, optimization, normalization and security.

Oracle SQL Developer[1] is one of the most widely used DB clients for inspecting objects in an Oracle DB. Oracle SQL Developer is a free-ware and fully supported graphical tool for database development. It provides functionality for browsing database objects, running SQL statements and scripts, as well as editing and debugging PL/SQL statements. Moreover, it supports a large number of reports, as well as creating and saving user-defined reports. Despite this, the main interest of the average user is browsing the database in order to check for physics quantities, rather than PL/SQL editing. Thus the goal should be to provide the end user with a very simple tool that allows to easily and more quickly accomplish tasks whilst eliminating the risk of compromising performance, security or, even worse, getting the back-end database stuck.

Often these situations can arise because of users that forget to quit their client database application. Another disadvantage of DB clients such as *Oracle SQL Developer* is that, since they provide for any database operation, they can be difficult to use and they can have significant side effects if not used correctly causing unnecessary use of computational resources. In fact, most of the administrative functions implemented on *Oracle SQL Developer* would never be used by average users.

To significantly decrease the risk of problems we provide a very simple tool to accomplish tasks easily and more quickly without the risk of downgrading the performance of the backend. Details will be provided in Section 4.

3. Comparison with other Products in the Market

First to deploy a new Database Browser application we conducted a market investigation of existing tools. This is a list of some of the most used database browsers that help users explore objects in a relational database:

- phpPgAdmin[2]: phpPgAdmin is a web-based administration tool for PostgreSQL. It is perfect for PostgreSQL DBAs, beginners and hosting services.
- phpMyAdmin[4]: phpMyAdmin can manage a whole MySQL server (needs a super-user) as well as a single database. To accomplish the latter you'll need a properly set up MySQL user who can read/write only the desired database.

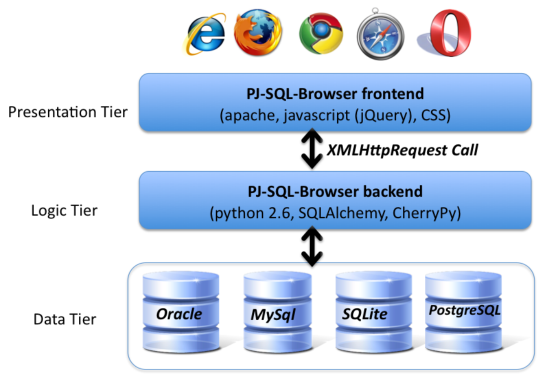
- phpOracleAdmin: it uses PHP4 to offer simple Oracle object browsing, multiple database connections and easy viewing of Oracle 8i database structures. It needs the *with-oci8 php* option which might not be installed by default. Unfortunately, the development cycle is very slow, no improvement has been seen since January 2004.
- Oracle SQL Developer[1]: is a free and fully supported graphical tool for database development. With SQL Developer, you can browse database objects, run SQL statements and SQL scripts, and edit and debug PL/SQL statements. You can also produce a large number of reports, as well as create and save your own.

The coverage of operations offered by these tools also implies some drawbacks:

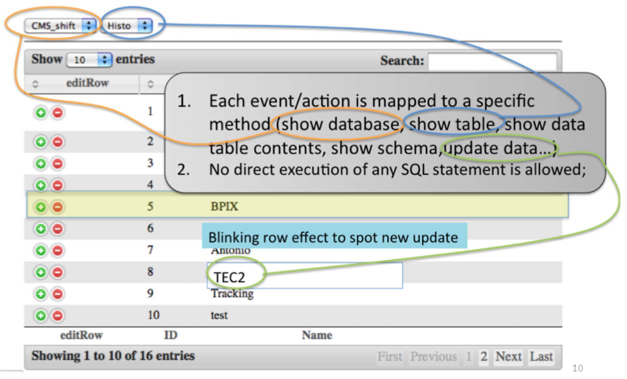
- They permit execution of arbitrary SQL commands. Bad SQL queries can downgrade the performance of the whole system.
- They are designed to work only with a particular DBMS. If you want to change the DBMS you need to change the application.
- They maintain open database connections for a long time: The problem with long-lived connections is that they can be left in unpredictable states by clients, and moreover it might not be entirely evident that the connection is still there. A network failure, server restart or *stateful firewall*[1] forgetting some of its state could all result in a *stale connection* which looks open, but then gives an error when you try to use it.

4. Technical Details

The *jSpyDB* is a three-tier system i.e. it is a client-server architecture in which the user interface (*Presentation Tier*), functional process logic (*Logic Tier*), computer data storage and data access (*Data Tier*) are developed and maintained as independent modules, most often on separate platforms (see Figure 1a). The policy used to deploy each tier is that the changes of a specific Tier don't affect the other Tiers. This results in high functionality, high reliability, easy debugging, low maintenance and ease of management.



(a) *Architecture of* jSpyDB.



1. Each event/action is mapped to a specific method (show database, show table, show data table contents, show schema, update data...)
2. No direct execution of any SQL statement is allowed;

Blinking row effect to spot new update

editRow	ID	Name
	1	
	2	
	3	
	4	
	5	BPIX
	6	
	7	Antonio
	8	TEC2
	9	Tracking
	10	test

Showing 1 to 10 of 16 entries

(b) *Frontend of* jSpyDB.

Figure 1: The *jSpyDB* system is designed as a three-tier architecture1a that draws a clear distinction amongst its information, logic and presentation. The *jSpyDB* GUI1b supports a wide range of operations on the data-tier without the need to write any SQL statements.

[1] In computing, a stateful firewall is a firewall that keeps track of the state of network connections traveling across it.

4.1. jSpyDB - Presentation Tier

The top-most level is the *jSpyDB* front-end (the *jSpyDB Presentation-tier*). The main function of this module is to translate tasks and results to something the user can understand. We use *jQuery*, *JavaScript frameworks*, in order to:

- create alerts if a server or network device is offline or to prevent and correct possible mistakes made by the users (for example inserting of wrong expressions in filter search field).
- manipulate or get information about objects in the HTML DOM. Objects in an HTML page have methods (actions, such as opening a new window or submitting a form) and properties (attributes or qualities, such as color and size).
- make a request to the server, interpret the results and update the current screen. In details, when a user selects a particular table/method the AJAX engine is called, which sends the request to the server using XmlHttpRequest object, parses the response, and updates the relevant portion of the page; therefore the whole page is not reloaded.
- finally, to create the interface table where data can be sorted, searched, and filtered[11] (see Figure 1b).

4.2. jSpyDB - Logic Tier

The *Logic Tier* is the jSpyDB back-end. This layer coordinates the application, processes commands, makes logical decisions and evaluations and performs calculations. It also moves and processes data between the two adjoining levels, the *Presentation Tier* and the *Data Tier.*

In the bottom-most level the information is stored and retrieved from a database. The information is then passed back to the *Logic Tier* for processing, and then eventually back to the user. We use SQLAlchemy[8] to access the back-end database. Thanks to SQLAlchemy we can make database-independent queries accessing relational databases such as Oracle, DB2, MySQL, PostgreSQL, and SQLite.

In detail The *Logical Tier* is responsible for:

- filtering all requests from outside in order to prevent unauthorized access and inappropriate use of data,
- mapping user requests to the handler that connects to the database and performing database calls,
- making cache checks to allow to use cache files if it is less than sixty seconds old in order to get a faster response. and to avoid overloading the Backend Database Systems
- converting data retrieved from database into JSON strings that are sent back to the *presentation Tier* (see Figure 2),

For the *Logic Tier*, we provide an *RPM package*, so that the installation could be easily automated. If you want to install it on other Operating System, we provide you the source files of the whole *Logic Tier*; in this case, before the installation, you need to compile them.

The *jSpyDB logic-tier* is currently used by the following web application regarding the CMS conditions data: *popcon monitoring*[5], *global tag browser*[6] and finally *Payload Inspector*[7], a web application for displaying the condition data. These three web applications are used in order to have a centralized control of all the workflows for the population of the production DBs, and to reduce any web-related security issues.

4.3. jSpyDB - Data Tier

The instances of the *jSpyDB* support most of the commercial and open source database management system to handle their administration over the world wide web.

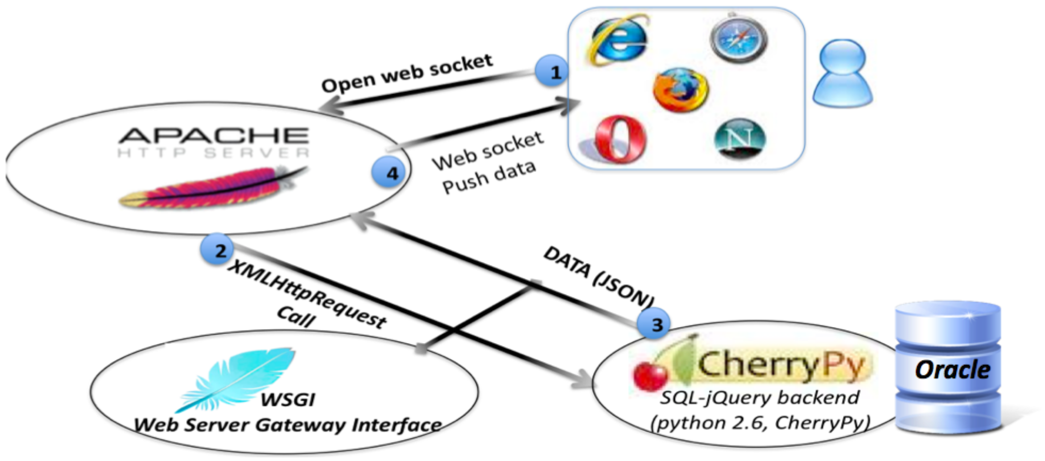


Figure 2: Web-request workflow.

In the *jSpyDB* context, the data-tier is an entity that contains all the database objects and instance objects whose connections are used by the logical tier application.

The *jSpyDB* data-tier can settle *Oracle*, *MySQL*, *PostgreSQL* and also the *Frontier*[9] system to access the caching servers for the condition data of the CMS experiment.

5. Open Issues

Here we list some of the problems encountered whilst still we were developing this tool.

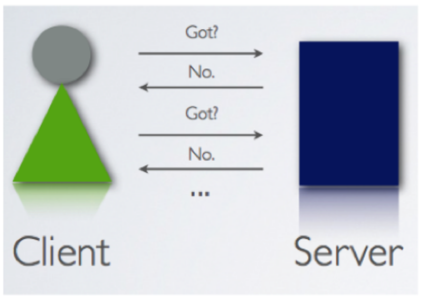
5.1. Display of Data Updates in Real-Time

At the moment, we are using AJAX to make the data display in real-time. With the current web technology, polling the server with Ajax is the easiest way to retrieve the latest data and events from the server for a Web application. The Web application essentially polls the server on a regular basis, based on a timer. The timeliness of the data is directly proportional to the polling frequency. The cost of polling frequently is very high, both in terms of network bandwidth as well as server infrastructure needed to support a huge number of polling requests. The server must have the capability to buffer the data that it needs to send to the client until the next request comes from the client (see Figure 3).

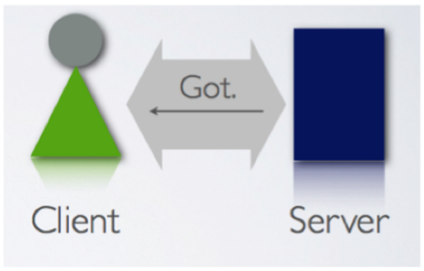
The Communication section in the HTML version 5[10] introduces *Server Sent Events* and *WebSockets*: these new features enable a new paradigm of web application programming that will revolutionize the way to develop and deploy web applications.

Thanks to WebSockets we can implement an *event-driven* communication: when an event occurs in the database backend (insert, edit, delete record in database), it implies that something changes in the database backend. In this case the *Logical Tier* triggers the client to upload the GUI with new data.

Since these specifications are not yet supported in all browsers, only *Google Chrome* and *Safari* implement these APIs, we have not yet put it into production. Whilst waiting for the implementation of these standards in *Firefox 3.7* and *Internet Explorer 9*, we successfully started the test cycle with these APIs.



(a) *Ajax call.*



(b) *HTML5 Web Sockets.*

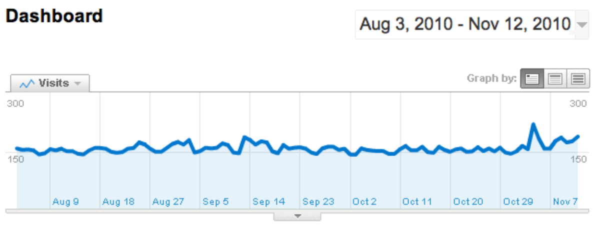
Figure 3: One major limitation of AJAX is that the browser must initiate the request for data from the server. The Communication section in the HTML 5 specification solves this problem by allowing the server to push data to the browser at any time.

5.2. Session Management

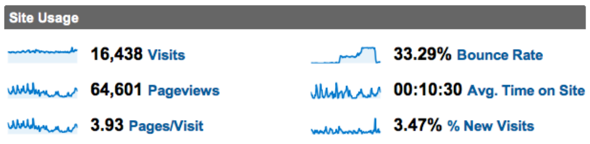
Session management is the process of keeping track of a user's action. At the moment, each user has a different session object so that session information is not overwritten or shared. This solution could be not scalable, since it introduces a lot of connections for a single user. Therefore, in order to save memory usage and bandwidth, in the next release of *jSpyDB* we shall share a session across multiple users on the same database connection.

6. Practical Experience

The results of using the tool in practice are displayed in the form of two simple graphs reporting the site usage statistics for the last four months. During this period the *jSpyDB logic-tier* has had an average number of contacts of 180 different IPs per day without any service interruption (Figure 4).



(a) *Graph showing number of visits per day.*



Site Usage

16,438 Visits	33.29% Bounce Rate
64,601 Pageviews	00:10:30 Avg. Time on Site
3.93 Pages/Visit	3.47% % New Visits

(b) *jSpyDB usage statistics.*

Figure 4: Google Analytics snapshot covering the period from 3 August to 13 November 2010. The image 4a shows the number of visits per day per *jSpyDB logic-tier*. The image 4b shows different statistics usage. Among others you can see the the total number of visitors and the total pages viewed[12].

Hence the system has proven to be reliable and very robust with high performance and availability both for the GUI and the database back-end.

7. Conclusions

Part of this paper has been dedicated to illustrating advantages and disadvantages of existing database browser tools. A trade-off between usability and performance has been possible by choosing to deploy this server on the Web Browser granting in such a way the possibility of access from any location and at any time and from various types of client machine, namely desktop, laptop or even handheld palm-sized devices. Nevertheless this strategy implies some drawbacks in achieving compliance with CERN standards, whilst at the same time it is required to keep up with evolution of web technologies.

Web technology is evolving at a very fast pace and the next release of web browsers may bring some improvements; indeed some APIs like *Server Sent Events* and *Web Sockets* are going to be enabled in all web browsers in order to achieve compliancy with the highest web programming and security standards.

References

[1] Oracle SQL Developer: a free and fully supported graphical tool for database development `http://www.oracle.com/technology/products/database/sql\_developer/index.html`

[2] phpPgAdmin: a web-based administration tool for PostgreSQL `http://phppgadmin.sourceforge.net/`

[3] `http://phporacleadmin.org/`

[4] PhpMyAdmin: a free software tool written in PHP intended to handle the administration of MySQL over the World Wide Web. `www.phpmyadmin.net`

[5] Persistent storage of non-event data in the CMS databases. M. de Gruttola et al. Jan 2010. 20pp. Published in JINST 5:P04003,2010.

[6] Global tag Browser for CMS Condition Database - `http://cms-conddb.cern.ch/gtlist/`

[7] CMS Payload Inspector - a web application for displaying the condition data `http://cms-conddb.cern.ch/payload_inspector/`

[8] Sqlalchemy: The Python SQL Toolkit and Object Relational Mapper `http://www.sqlalchemy.org/`

[9] CMS conditions data access using FroNTier - B Blumenfeld et al 2008 J. Phys.: Conf. Ser. 119 072007 doi: 10.1088/1742-6596/119/7/072007

[10] Network section of the whatwg HTML5 draft page `http://www.whatwg.org/specs/web-apps/current-work/\#network`

[11] DataTables `http://www.datatables.net/`

[12] Google Analytics - `http://www.google.com/analytics/`