Integration of Databases and World Wide Web Based on Open-Source Technologies

Dobrica Pavlinušić
University of Zagreb,
Faculty of Organization and Informatics, Varaždin
dpavlin@foi.hr

Alen Lovren ić, M.S.
University of Zagreb,
Faculty of Organization and Informatics, Varaždin
alovrenc@foi.hr

Abstract: This paper deals with the problem of developing applications which use databases with World Wide Web front ends, based on open-source technologies. This involves some limitations and problems that are discussed in more details.

Keywords: open-source, database, World Wide Web, WWW, integration

---

1. What had to be done?

First, let us examine the goal of this project, which was to build application for article submission for conference “International Conference on Information and Intelligent Systems” in Varaždin. We had several requests that had to be taken into account:

- we need repository for submitted articles which can be accessed over Internet and intranet (because users of this will come from all over the world using Internet and we want to update database from our local area network)
- database should be assessable from web browser over Internet and desktop database applications (like Microsoft Access) from intranet for easy updates
- database should provide authors all over the Internet with ability to change data about theirs articles in database after authorization, and contain all our information needed for conference
- we had limited budget which prevented us from evaluating commercial solutions
- our project ought to be used for at least five years, so we shouldn’t choose some property solution which will cease to exist in that time-frame

With all those points in mind, we decided to build application that is based on relational database with World Wide Web front end based on open-source technologies.

2. What is World Wide Web?

The World Wide Web (known as "WWW', "Web" or "W3") is the universe of network-accessible information, the embodiment of human knowledge [11].
It is basically composed of two main protocols. One of them is called HyperText Transfer Protocol (HTTP) and the other is HyperText Markup Language (HTML). HTML is language that describes appearance of text on screen (which is in fact displayed and positioned on

screen by your web browser) and links or references in form of hypertext. HTTP is protocol based on TCP/IP, used to transfer HTML pages over network, from HTTP server to client's web browser, which is in this architecture client for accessing HTTP server. More information about HTML and HTTP can be found in [7] and [3].

3. Why should we use relational databases?

The database is a data structure, usually rather big and stored in secondary memory, which is specialized for easy processing of large amount of different queries, and other operations among large scale of different data. There are many different database management systems (DBMS), that are used as interface between database user and computer, so user can look at his database from logical point of view, without any need to know physical way DBMS use for data storage. From that point of view DBMS can be seen as back-end CASE tool for static part of information system.

From early 70's when F.E. Codd developed it, relational model is the most used model for database design, so almost all DBMS used today are relational (RDBMS). Advantages that made figured relational model almost the only one used for database management today are formal foundations, complete independence of logical and physical level of database, easy way of connecting database objects at logical level etc.

Relational model is consists of 2 classes of objects – relations, attributes. We can say that the attribute is atom of relational model. *Attributes* consist of *attribute name* and *domain*. Domains are usually standard data types known from procedural programming.

Next step is to define relational scheme. *Relational scheme* is finite sets of attributes. Relational scheme a pattern for building relations.

Let us say that (A_1,D_1), ..., (A_n,D_n) are attributes that form relational scheme R. Let D_1, ... D_n are domains of that attributes. We define $D=D_1\cup...\cup D_n$. Then we can define *row* or *record* as function $t:R \rightarrow D$, with property that $t((A_i,D_i))\in D_i$. Intuitively said, records are n-tuples of values, where i^{th} value is chosen from domain of i^{th} attribute. In cases when attribute domains are known form semantic context, we usually define attributes only with their names. In that case we write relation as $R(A_1,...,A_n)$
Now we can say that *relation* is finite set of records. Relations can be presented as two-dimensional tables. For example, if we have the relation called BOOK with relational scheme (NO#, NAME, ISBN), we can represent relation among this scheme as table:

BOOK	NO#	NAME	ISBN
	1	Hitchhiker's Guide to the Galaxy	0345391802
	2	The Hobbit	0395282659
	3	Do Androids Dream of Electric Sheep?	0345404475

Table 1: relation representation

In this table heading row represents relational scheme, and body rows represent records. In head of each column is attribute name.
To have access to every record in database, in each relation we highlight one or more attributes as a *primary key*. Attributes in primary key have to be unique for each record in the relation.
Now we can say that *database* is set of relations.
We build databases as models of real world. But, in reality objects are not independent. There are connections between them. The database, as the model of real world, has to represent

those connections too. Because of that we introduce a concept of relationships. *Relationship* is a connection between two or more records of relations. It doesn't mean that there are no relationships between record of the same relation.
Relationships in relational model are not represented. Relationships are created as they are used for queries RDBMS receive. Because of that relational model is flexible for very large scale of different queries that connect more than one relation from database. To achieve relationships between records, we use a concept of *foreign keys*. Foreign key is set of attributes in some relation that is primary key of some other record of some relation in database. To illustrate that concept, we propose next example:

AUTHOR	NO#	NAME
	1	R.R. Tolkin
	2	Douglas Adams
	3	Philip K. Dick

AUTHOR-BOOK	AUTHOR-NO#	BOOK-NO#
	1	2
	2	1
	3	3

Table 2: example of foreign keys

Relations AUTHOR and BOOK are connected via relation AUTHOR-BOOK. Primary key in relation BOOK is NO#, as well as primary key in relation AUTHOR is NO#. In relation AUTHOR-BOOK primary key is set {AUTHOR-NO#, BOOK-NO#}. But in the same time BOOK-NO# is a foreign key that corresponds to attribute NO# in relation BOOK. In the same manner, AUTHOR-NO# is a foreign key that corresponds to primary key of relation AUTHOR.
As we can see, this connection between books and their authors is not implemented directly in the database, but with proper query it can be obtained from data stored in the database.
To resume, relational database is an easy way to create and manage large amount of data in organized and easy-to-retrieve way.

4. Integration of RDBMS and WWW

Now that we introduced RDBMS and World Wide Web, we should answer to main question: why should we integrate databases and WWW? There are many different interfaces for database-human interaction, so why should we use WWW? There are many answers, but lets just outline few most important ones for us:

- databases are best used for storing all kind of information,
- HTML enables various forms of data representation (which are suitable for displaying data from relational databases), which is platform and location independent,
- modern trends in database design include client/server architecture that can be implemented using proposed model quite easily.

If we review those points relating to our initial goals, we can see that they are almost perfect fit. So, choosing RDBMS and WWW was logical choice.

4.1. Connecting WWW and Databases

There are two ways for connecting World Wide Web and databases. One is to use CGI that starts external programs (that process is computing intensive) and the other is to use some sort of extension of HTML language.

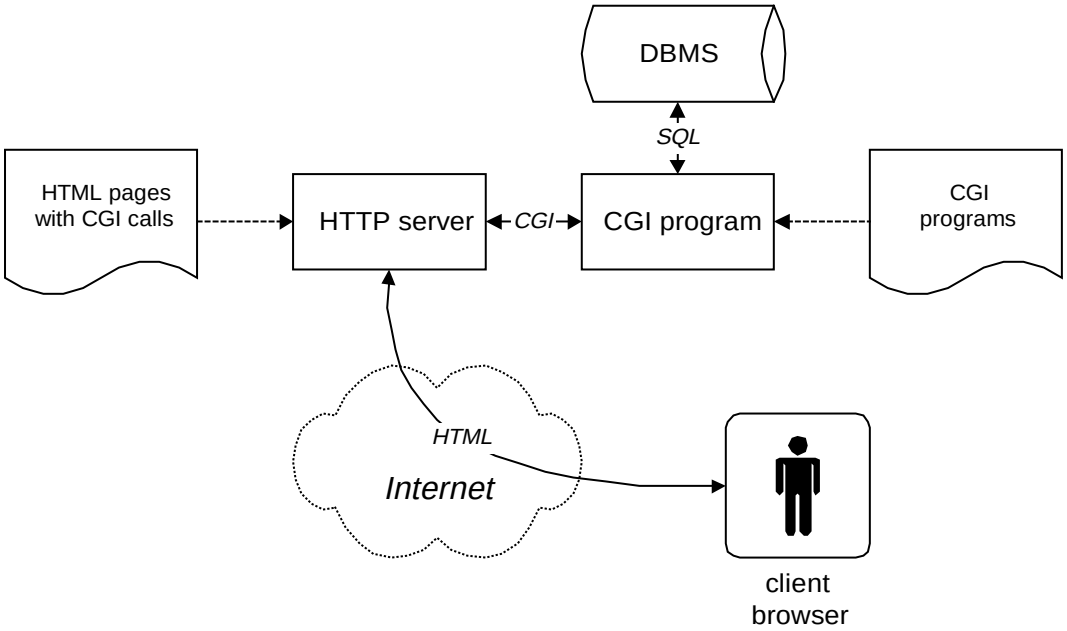

Figure 1: example of CGI-based application

CGI (Common Gateway Interface) is an API that defines how applications "talk to" web serves (and exchange data with them). HTTP defines way in which web clients send data to web servers, which in turn, transfers that data to CGI programs which process them. CGI programs are, in our case, programs that are used to access databases. They can create output, which is in most cases HTML pages, which are then transferred to client machines using HTTP protocol by HTTP servers.

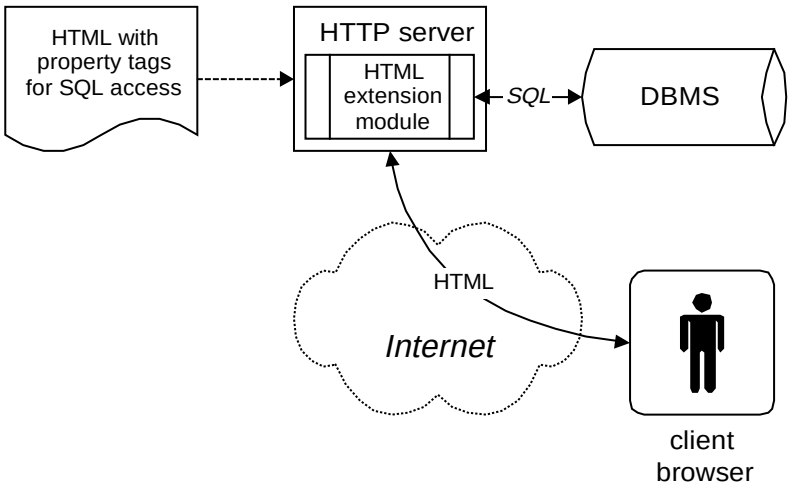

Figure 2: example of HTML language extension

The other way to access databases from WWW is to use some kind of HTML language extension. These extensions define special tags, which are not transferred to client web

browsers, but executed and replaced by the result of that execution (which is in most cases again HTML code). However, second approach has advantage in easier maintenance (code and HTML aren't divided) and speed of execution (as no external programs are forked. All processing is done inside web server).

4.2. Double client/server architecture

To add to general confusion, client/server architecture in this case is twofold. First, we have client/server architecture in WWW. In this case, clients are programs (browsers) which display data received from servers (HTTP servers) in HTML format on user screens. On the other hand, from database view, clients are CGI scripts which access database directly, but also JavaScript programs which are run "inside" clients browsers whose main purpose is to check validity of input data. Server, from database view, is, of course, relational database management system. Example of such architecture that is used in our project is shown in Figure 3.

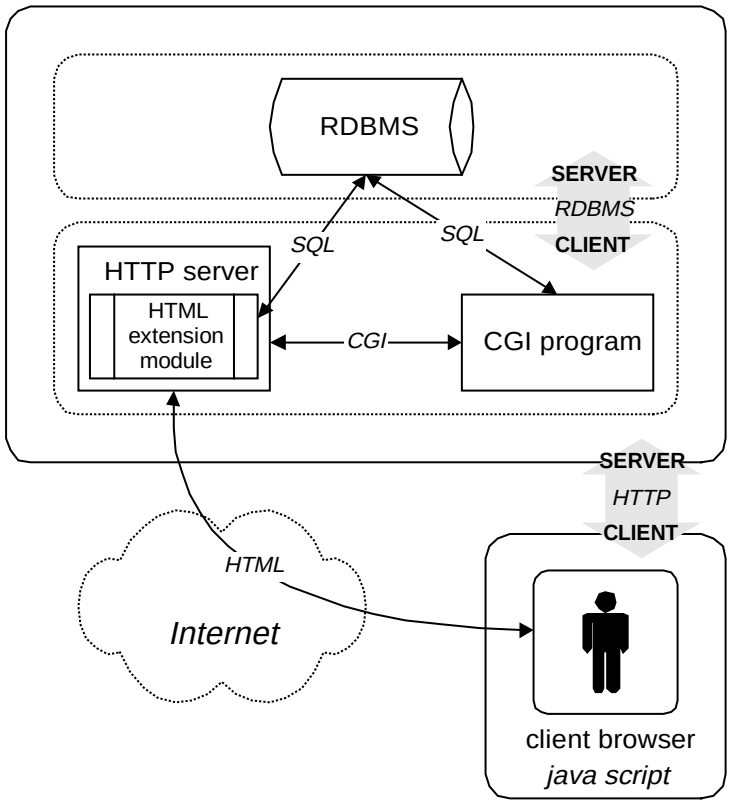

Figure 3: example of double client/server architecture

All of components mentioned before (client browser, HTTP server, CGI program and RDBMS) can be distributed throughout the network to provide load balancing or just to accommodate flood of requests that is possible.
So, if we look at client/server architecture described above as a distributed system, there is one more important thing to note: system architecture must also be open. Here are reasons why: [1]

- benefits of interoperability and portability extend to all components in the architecture,
- the architecture can be specialized or can evolve by changing the implementation of individual components, and
- the architecture can be extended by introducing new components at a later date.

By choosing HTTP as main transport protocol, HTML as a language for data representation to user and SQL as a query language for database, we have fulfilled requirement for open systems. As we will see in chapter 6, that gave us the ability to change tool for displaying of information in middle of project.

5. Open-source

5.1. Introduction

Open-source is definitely buzzword of today. It is first introduced in Eric Raymond's article "The Cathedral and the Bazaar" [3], while making a case for open-source development within an extended developer community as a way to create better software. This article motivated Netscape to open-source Netscape Navigator product [4] that became Mozilla.

Open-source software and open-source development projects have existed for many years under the general term "free software." The word "free" has traditionally two meanings: as in *free speech* and as in *free beer*. Unfortunately, IT community in general had negative connotation to everything free (in *beer* sense, although that was important factor in choosing open-source for this project). And, they don't really care for free in *speech* sense also.

As contrary to that, Richard Stallman, a most vocal free-software advocate did not argue in his "GNU Manifesto" [10] that software development should always be an unpaid or nonprofit activity. Rather he proposed that for-profit business models should treat software as a professional service rather than as intellectual property.

I will not get into details about distinction between free and open-source (interested readers are referred to [9]), but I would like to point out that freedom involved with open-source is often essential to many free-software developers.

Officially, open-source means more than that source code is available. The source must be available for redistribution without restriction and without charge, and the license must permit the creation of modifications and derivative works, and must allow those derivatives to be redistributed under the same terms as the original work. Licenses that conform with the Open Source Definition include the GNU Public License (GPL), the BSD license used with Berkeley Unix derivatives, the X Consortium license for the X Window System, and the Mozilla Public License.
But open-source is more than just a matter of licenses. Some of the most significant advances in computing, advances that are significantly shaping our economy and our future, are the product of a little-understood "hacker culture." It is essential to understand this culture and how it produces such innovative, high-quality software. What's more, companies large and small are struggling to understand how the ethic of free source code distribution affects the economic models underlying their present businesses. [3]

5.2. Support, reliability and design practice

Open-source has support in form of e-mail or Usenet group. Although there is a 24-hour support for some of open-source products, that is not always the case. Database selected for this project, MySQL, has commercial support available.
Reliability is achieved through peer user community that can have a look at source code and examine it. That usually happens when some user (who, in most cases, is actually a member of peer community) finds a bug. As the user isn't limited with restrictive licenses and since he has source code, he can start fixing bugs right away. There are no long waits for next release to fix well-known bugs. The same process is used to add new features to product.

Everything with open-source is not roses. As we will find out during our implementation phase with www-sql, common problem is that open-source product are usually done in hands-first manner, without good design before implementation [12]. That can lead to tools which just can't expand as much as needed.

5.3. Developer conflicts

Krishna Kolluri, Healtheon's VP for applications [12] has interesting thought about this: "The beauty of open-source is getting so many brilliant minds working on something". However, open-source developers are busy people working on different projects, so there could be conflicts. As one interesting post on slashdot.org noticed [14], sometimes developers have different visions of project. However, since open-source allows fork-off, it is not really the problem for developers. That leaves users in a difficult position of choosing "one" or "other" version of same package. Luckily for us, we hadn't had that problem.
Nevertheless, it is still better to have choice of two different projects that try to achieve same goal than to have one, which is closed and may not be "the right thing".

5.4. Advantages and drawbacks of using open-source

Open-source, as stated before, has it's own advantages and drawbacks. However, in this chapter I will focus only on pros and cons of using open-source products for implementing of database and World Wide Web integration. We shell try to sum up what had we gained and lost using open-source technologies, as opposed to some other, commercially available solutions.
In development stage, we really didn't have any RAD or CASE tool that we could use. That was not big problem, as our project is small enough, so we could do it using just pen and paper (in reality we used Microsoft Access to design database and later exported its structure via SQL). However, there are no RAD or CASE tools that are open-source. That field is strictly commercial for now.
The biggest problem we encountered so far is lack of some features in our open-source database. But, we will say few words about that later.

6. Implementation

6.1. Web server

For our Web server we have chosen Apache Web server [12], which is used by over 50% of all Web servers on Internet [17]. This choice was logical because of our positive prior experience with Apache and the fact that it is often cited as one of major open-source projects available today.

6.2. Database

Our choice of relational database was rather simple one: we had to use one which provided SQL and was open-source. We didn't have so much alternatives, so we ended up using MySQL [16] that is also well supported by our development tools.
The biggest problem with this database is that it supports just subset of SQL92 standard. However, we where able to write work-around solutions for all problems that could, otherwise, be solved using some of more advanced features of SQL92.

6.3. Development tools

While initial evaluation of development tools we have considered following alternatives:

(a) Perl with DBI interface for database - this was possible solution but we didn't want to maintain separate perl scripts and HTML pages. Also, complexity of code needed to access database from perl via DBI was too large.
(b) Embedded perl - special version of perl that is based on idea that perl code can be written inside HTML file solved maintenance problems. However, complexity of DBI was still here.
(c) Various additional modules for integration of perl, databases and WWW were considered, but rejected because none of them used SQL language.
(d) www-sql, simple language embedded in HTML based on one additional HTML tag, <!sql>. Commands inside www-sql tag can do simple nesting, conditional branching and database access using SQL language.

After doing initial tool evaluation we have selected tool for implementation of our project: www-sql [5]. During the implementation of our project, we also had to use some other tools which included perl [18] for some additional CGI scripts and pre-processing and PHP: Hypertext Preprocessor [19].

A first problem we encountered using www-sql was creating vast amount of HTML forms needed for input of various data. So we took www-sql source code and added couple of custom tags to solve this problem. That hands-on approach can be used only on product which has source code available (which is one of characteristics in open-source), and www-sql is one of them. Advantage of www-sql was also that it was simple to learn and expand. It is written in C programming language, so expanding it to our needs was easy.
Patches made against www-sql distribution for our project can be found at *http://www.foi.hr/~dpavlin/projects/www-sql/* and used for other projects. GPL license [15] of www-sql mandates that all patches which are not strictly for internal use must be made available for download. That is also a momentum of open-source community. Somebody else can improve our work and give it back to community again.

However, when our project needed to send mail, we turned to perl that is more general-purpose scripting language. In fist step only sending of mail was implemented using perl CGI program.

After first part of project was completed, we had to create several screens with just results from database. In that time, motivated by need for more flexible solution than www-sql, we started to evaluate PHP that proved to be good solution for almost all our problems. If we would do this project again, it would be probably completely written in PHP.

Change of development tool didn't affect our already implemented part, because server communicates with www-sql pages, perl CGI scripts and PHP pages in same way. So we can seamlessly integrate those three languages into one compound project.

7. Conclusion

Results of our work can be found at *http://www.foi.hr/iis/*. We tried several tools and almost all of them could do the job. Although we did initial evaluation of more than ten tools, we still ended up using three of them for this, relatively simple project. However, we don't see that as a drawback.

We also encountered some problems that are typical for open-source projects: tools that doesn't fit user needs perfectly, but can be easily extended and small problems which would call to support solve. However, we had to solve them ourselves.

Altogether, we have created a working project which meets all our goals from beginning of this article.

8. References:

[1] Gordon Blair, Jean-Bernard Stefani (1998): *Open Distributed Processing and Multimedia*, Addison Wesley Longman Ltd, London

[2] Chris Dibona (Editor), et al (1999): *Open Sources: Voices from the Open Source Revolution*, O'Reilly, *http://www.oreilly.com/catalog/opensources/book/toc.html*

[3] J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee (1999): *Hypertext Transfer Protocol -- HTTP/1.1, http://www.w3.org/Protocols/rfc2616/rfc2616.html*

[4] Frank Hecker (1998): *Setting Up Shop: The Business of Open-Source Software*, IEEE Software, Jan/Feb 1999, p.45-51, *http://people.netscape.com/hecker/setting-up-shop.html*

[5] James Henstridge (1998): *The WWW-SQL Home Page, http://www.daa.com.au/~james/www-sql/*

[6] Tim O'Reilly (1999): *Lessons from Open Source Software development*, Communications of the ACM, April 1999/Vol. 42, No. 4, p.33-37

[7] Dave Raggett, Arnaud Le Hors, Ian Jacobs (1998): *HTML 4.0 Specification, http://www.w3.org/TR/REC-html40/*

[8] Eric Raymond (1996): *The Cathedral and the Bazaar, http://www.tuxedo.org/~esr/writings/cathedral-bazaar*

[9] Aaron M. Renn (1998): *"Free", "Open Source", and Philosophies of Software Ownership, http://www.urbanophile.com/arenn/hacking/fsvos.html*

[10] Richard Stallman (1985): *The GNU Manifesto, http://www.gnu.org/gnu/manifesto.html*

[11] W3C (1997), *About The World Wide Web, http://www.w3.org/WWW/*

[12] Greg Wilson (1999): *Is the open-source community setting a bad example?*, IEEE Software, Jan/Feb 1999, p.23-25

[13] …(1999) *Apache Server Project, http://www.apache.org/httpd.html*

[14] …(1999) *Conflicting Open Source Developers, http://slashdot.org/article.pl?sid=99/07/12/1639202&mode=nested*

[15] …(1991) *GNU General Public License, http://www.gnu.org/copyleft/gpl.html*

[16] …(1999) *MySQL home page, http://www.mysql.com/*

[17] …(1999) *The Netcraft Web Server Survey, http://www.netcraft.com/survey/*

[18] …(1999) *Perl Mongers home page, http://www.perl.org/*

[19] …(1999) *PHP: Hypertext Preprocessor home page, http://www.php.net/*