SPECIFICATION OF A MECHANISM FOR DIRECT ACCESS TO DATABASES THROUGH WWW

Vagelis Zorbas
vzorbas@dblab.ntua.gr
Knowledge and Database Systems Laboratory,
National Technical University of Athens,
Zographou 157 73 Greece,
http://www.dblab.ntua.gr/

Yannis Stavrakas
ys@dblab.ntua.gr

Kostas Chatzinas
kchatzin@dblab.ntua.gr

1. SUMMARY

We present a model that allows database access through the Web, and whose specification is based on Web standards. Users of the Web will be able to access distant databases in a way similar to accessing hyper-documents. The model deals with the expression of requests from web browsers to databases, in order to retrieve stored information, and the transfer of results to web clients. In addition, we examine the possibility of presenting results from many sources in the same web page, and of selective incorporation of results into conventional hyper-text documents. A prototype system has been implemented, demonstrating all the features described in this paper. The present paper focuses on specifying in detail the mechanism we propose. A description of the model and the architecture can be found in [1].

2. INTRODUCTION

Until recently, database access methods through the Web restricted all functionality and control for database access to the server. These methods are also characterized by the possible binding of applications to database structure, the limited interaction allowed by applications, and the dependence on a specific web server. In the last years, efforts have been made to implement new techniques for bringing databases closer to Internet users. The Java [15,16] programming language and Microsoft's Dynamic HTML (D-HTML) [9,10] gave the web client the possibility to have more control over the process of database access.

However a global standard that defines web client-driven database access has yet to be defined. The way the Web merges with database information seems to be the result of covering needs as they appear, rather than that of a careful and comprehensive study.

All existing methods for accomplishing database access are based on custom code and especially built applications. Instead of proprietary methods and protocols, a more generic model *based on Web standards* must be considered.

Using a global standard, databases could play a more active role in the Web, increasing the availability of information, and taking advantage of the existence of the Internet. It seems to us that a prerequisite for such a development is that databases become first-class citizens and participate in the Web from a position similar to that of the hyper-documents. Instead of being hidden behind applications, they must be directly visible to Web users.

3. THE PROPOSED MODEL

3.1. Architecture

In promoting databases to first-class citizens in the Web, a basic concept seems to be that the web client must be aware that databases exist and be able to express its demands directly to them [1]. To make that possible, we must examine from this perspective the triplet HTML-URL-HTTP and come up with adjustments that will enable such operations.

- Extensions to HTTP [7,8] led to defining *DBTP (Data Base Transfer Protocol)*, a new communication protocol that supports the requirements and functionality of the proposed mechanism.
- Extending URL [5,6] semantics led us to the definition of *dbtpurl*, a new URL scheme, whose links define operations on databases available over the Web.
- Finally, in order to allow for flexible and controlled manipulation and presentation of database results at client side, we augmented HTML with two new tags.

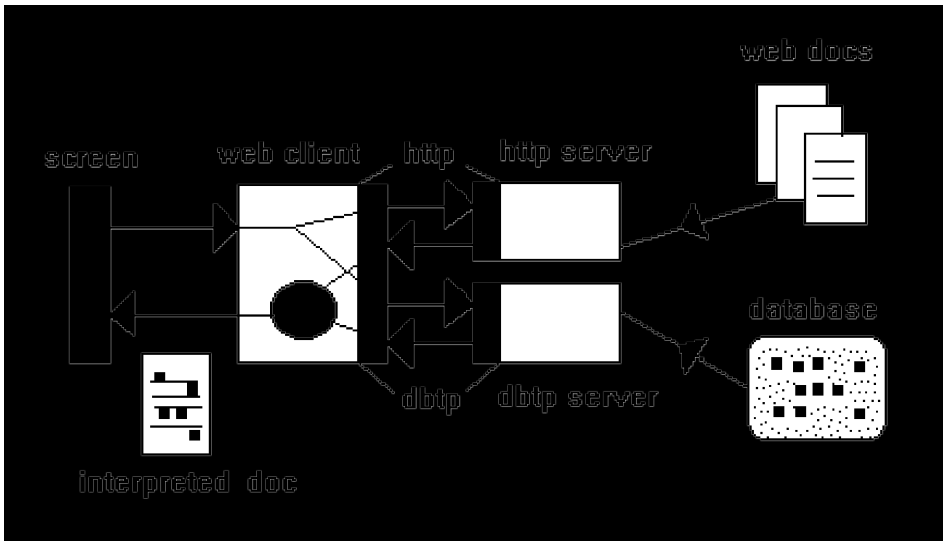

***Figure 1**. Proposed architecture for accessing databases through WWW*

The architecture for the proposed system is illustrated in *Figure 1*. The components in *Figure 1* are the web client, which is an extended browser also called *dbtp client*, and an especially built server named *dbtp server*. These components implement the features presented in the following paragraphs.

3.2. URL Scheme

URL (Uniform Resource Locator) [5] is the standard way for specifying a resource available on the Internet. Following the general URL syntax rules, we defined a new URL scheme, tailored to database resources. The *dbtpurl* allows web users to specify the database to connect to and the operations to perform, in a generic way.

```
dbtpurl = "dbtp://" login ["/"[ ["metadata" | Range] "/"] [ dbName [":" query ]]]
          login = [user[":"password] "@"] hostport
          hostport = host [":"port]
          Range = "Range:" *DIGIT "-" *DIGIT
          DbName = *unreserved
          Query = *xchar
```

Table 1. *Formal definition of the dbtpurl*

The formal definition of the new URL scheme is presented in Table 1 in augmented BNF syntax [4]. Parts of this definition are the server IP address, the database name, the database query, and additional information regarding the request. The tokens "*user*", "*password*", "*host*", "*port*", "*DIGIT*", "*unreserved*" and "*xchar*" are further defined in [5].

Additional request information supported by *dbtpurl* include:

- Username and password for user authorization against the database.
- Indication that the user would like to receive information about which databases are available on a specific dtpt server.
- Indication that the user would like to receive meta-information on the schema of a particular database.
- Indication that the user can receive only a maximum number of results.
- Indication that only a specified part of the database results should be returned to the user.

The validity of the *query* part of the URL is ultimately determined by the database which handles it. It can be any valid expression of the database own query language. For SQL queries we provide the alternative to apply a simple encoding, mainly for the sake of brevity, where SQL keywords are replaced by special characters.

Examples of *dbtp* URLs, accompanied with a short explanation, follow.

1. *dbtp://name:pass@www.dbtpServer.com/range:0-20/BooksDB:# * ? Book*
Apart from the protocol to be used (dbtp), the host, and the port the dbtp server is listening to, this URL indicates the username and password for authenticating the connection to the database and that the user wishes to receive only the first 20 lines of the results. The encoded SQL query is interpreted by the dbtp server as "SELECT * FROM Book".
2. *dbtp://www.dbtpServer.com/*
Alternatively: *dbtp://www.dbtpServer.com/metadata/*
As a result to the above requests, metadata on the services offered by the server are sent by the dbtp server back to the dbtp client. Metadata may include the names and schemas of the databases exposed by the dbtp server, and information on the server capabilities.

3.3. Communication Protocol

The principal question is whether HTTP can accommodate the *dbtpurl* features. The answer is an initial *yes*; HTTP could be used for serving database communication needs, either extended or not:

- Leaving HTTP unmodified means that *dbtpurl*-related information must be added at low level inside the body of the existing HTTP messages, since the higher level HTTP methods refer to documents and are not suitable to databases. This means that if HTTP is used as is, the semantics of a new protocol are "burried" inside HTTP infrastructure. This solution is not satisfactory, as the HTTP methods will be overridden by semantically irrelevant information hidden inside the HTTP messages body.
- Extending HTTP to incorporate new functionality for serving *dbtpurl* requests, is the second solution we considered. It seems that an extended HTTP protocol could support *dbtpurl* needs. However, this approach may present a few drawbacks: (1) HTTP operations could prove inadequate for the required functionality. HTTP security and caching would probably need modifications to stand for databases as well. (2) The new protocol would become heavy and complicated, trying to serve both hyper-documents and databases.

Our approach was to build a new protocol from scratch, because of the advantages this solution has:

- A new protocol can be fully adapted to the *dbtpurl* needs. Operations like connection duration, message encoding, caching and security, can be designed with database communication in mind.
- The new protocol, being independent from HTTP, could be selectively deployed from web browsers and specialized dbtp servers.

```
DBTP-message = DBTP-Request | DBTP-Response

DBTP-request = Request-Line          DBTP-response = Status-Line
               |*(general-header                    |*(general-header
               | request-header                     | response-header
               | entity-header )                    | entity-header )
               CRLF                                 CRLF
               [ message-body ]                     [ message-body ]

Request-Line = Method SP dbName SP DBTP-Version CRLF
Status-Line = DBTP-Version SP Status-Code SP Reason-Phrase CRLF
DBTP-Version = "DBTP" "/" 1 *DIGIT "." 1 *DIGIT
Request-header = Authorization       Entity-header = Content-Length
                 | Maxlines                        | Content-Type
                 | Range
                 | Query-Encoding
Method = "query"                     Status Code = "200" ; OK
         | "metadata"                              "206" ; Partial Content
dbName = "*"                                       "400" ; Bad Request
         | token                                   "401" ; Unauthorized
                                                   "420" ; Bad Query
                                                   "500" ; Server Error
```

"505" ;	Version not supported

Authorization = "Authorization" ":" userid ":" password
*Maxlines = "Maxlines" ":" *DIGIT*
*Range = "Range" ":" *DIGIT "-" *DIGIT*
Query-Encoding = token
*Content-Length = "Content-Length" ":" 1*DIGIT*
Content-Type = "Content-Type" ":" type "/" subtype

(SP = Space, CRLF = Carriage Return, Line Feed)

Table 2. *Formal definition of the DBTP message*

DBTP, a new Internet protocol, has been formally defined in terms of BNF. It is an application level protocol designed to work over a reliable transport protocol (like TCP), with the following features:

- Its operation is based on the request/response paradigm. A dbtp client establishes a connection with a dbtp server, sends a Request message to the server, receives a Response message from the server and finally the server closes the connection.
- It is an object oriented, human readable and stateless protocol.
- The communication unit of the protocol is the message. Request and Response messages carry the client's database request to the server, and the database response back to the client.
- Its operation and the form of messages are similar to those used in HTTP.

The formal definition, in augmented BNF [4], of the dbtp messages is given in Table 2. The tokens "*token*", "*type*", "*subtype*", "*DIGIT*", "*userid*", "*password*", "*SP*", "*CRLF*" are further defined in [8].

A mapping of dbtpurl to DBTP Request messages has been defined in detail. The information is carried into headers alike to HTTP messages.

A sample of dbtp messages follows, based on the 1st example of the dbtp URLs given in a previous section:

- The DBTP Request message issued by the dbtp client:
  " *query BooksDB DBTP/1.0 \r\n*
  Authorization: name : pass\r\n
  Range: 0-20\r\n
  \r\n
  *# * ? Book* "
- A possible answer from the Server could be the following dbtp Response message:
  " *DBTP/1.0 200 Query Executed\r\n*
  \r\n
  <Results From Database> "

3.4. HTML Extensions

The dbtp client can display the database results directly, formatted as HTML table. However, it should be possible for the dbtp client to select data from the results, and insert them at specific positions in a web page that will be the actual response [1]. In

order to enable this selective merging of data and hypertext at the client side, we extended HTML with the following tags:

1. The <fetch> tag specifies a URL, dbtpurl or conventional, whose contents are accessed and stored locally on the client, but not immediately displayed.
2. The <insert> tag, refers by name (through the "src" attribute) to a <fetch> tag and specifies a subset or all of the data corresponding to the <fetch> URL. Those data substitute <insert> at its location, and are subsequently presented as part of the web page. The optional attributes "row" and "col" can be used in the case of referring to database results, to specify pieces of data to insert at given locations of the web page.

```
<!ELEMENT FETCH - O EMPTY>
<!ATTLIST   FETCH
        name    NAME    #REQUIRED
        href    %URL    #REQUIRED
        >

<!ELEMENT INSERT - O EMPTY>
<!ATTLIST   FETCH
        src     NAME        #REQUIRED
        row     NUMBER    #IMPLIED
        col     NUMBER    #IMPLIED
        >
```

Table 3. *Formal definition of the FETCH and INSERT tags*

Multiple <fetch> and <insert> tags can be used in the same HTML page. In case the <fetch> URLs are conventional URLs referring to HTML documents, <insert> tags can be used to dynamically interpose HTML documents in the current page. This is advantageous because documents automatically reflect changes made at their site of origin. In case the <fetch> URLs are dbtpurl however, those tags become even more useful as they offer a way to selectively incorporate data returned from databases in a web page. This mechanism can be thought of as a way to *define views* over hyperdocuments and structured data.

Formally, in terms of SGML [12], the two tags are defined in Table 3.

4. SYSTEM OPERATION

As an example we will consider an extended HTML page residing at a web server with address "latest.resources.com". This page will provide to the user the latest information on XML from other web and dbtp servers. The page will contain <fetch> and <insert> tags and will be automatically updated by the user's extended browser.
Let us assume that there is a dbtp server, the *gorgon.com* server that serves a book database. The XML specification that resides on the W3C organization server, will be also "fetched" by the extended browser.
Analytically the transactions that will take place are the following:

- The URL "http://latest.resources.com/XML/main.html" will be inserted from the user into the extended browser. The following page will then be downloaded by

the extended web client using the HTTP protocol.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
        <title>Latest XML Resources</title>
</head>
<body>
<h3>The latest XML specifications from W3C</h3>
 <p>
 <FETCH NAME="W3C" HREF="http://w3c.com/xml/">
 <INSERT SRC=W3C>
 </p>
<hr>
<h3>Books on XML available from the bookstore gorgon.com</h3>
 <p>
 <FETCH NAME="gorgon" HREF="dbtp://www.gorgon.com/BooksDB:
                                                # * ? Books % Category='XML'">
 <INSERT SRC="gorgon">
 </p>
<hr>
</body>
</html>
```

- The dbtp client will parse the HTML document, and for every FETCH tag, the resource indicated will be downloaded and inserted appropriately into the document.
- The result will be a new HTML page, without extensions (fetch and insert tags), that will be presented to the user using a commercial browser. The result page will be comprised of an HTML page retrieved from the W3C web site and an HTML table containing database results retrieved from the BooksDB database, served by the gorgon.com server.

Both the browser extension and the dbtp server are available for downloading and testing at the Web address *http://www.dbnet.ece.ntua.gr/~ys/dbtp/bin/*. Installation instructions and brief tutorial are available at the same address in the file "manuals.doc".

5. CONSIDERATIONS AND FUTURE ENHANCEMENTS

As possible directions for future work we consider:

- *XML use:*

Using XML [13] for formatting database results would have significant advantages over HTML. With XML the need to define extensions to HTML would be eliminated, and the merging of hypertext and database results could be designed in a more efficient way.

- *Security:*

One of the most important features that have to be added to the system is security, since database information can be very critical.

- *Applet Version:*

The dbtp client is developed as a Java program. An applet version of the program would give the advantage of publishing the program to every web browser supporting Java.

- *Dynamic Insertion:*

It would be desirable that the users be able to dynamically formulate the HTML page after the retrieval of documents and database results indicated by FETCH tag URLs.

- Dynamic metadata manipulation:

Dbtp server metadata are currently only presented to the end-user by the dbtp client. A more interactive model is desirable, where the browser will guide the end user to the information he wants, taking advantage of the server's metadata.

- *Joining results at client side:*

Extending the dbtp client in the direction of performing joins over one ore more database sites is a subject that should be analyzed. Relevant work [14] must be evaluated in conjunction to the designed model

- *Statefull DBTP:*

Database transactions involve consequent queries posed to a database. A persistent connection between the dbtp client and the dbtp server would have many advantages in that case.

- *Caching*

Caching mechanisms similar to the one used in HTTP, could be designed and employed to enhance database access efficiency.

6. REFERENCES

[1] Yannis Stavrakas, Nikos Karayiannidis, Panos Vassiliadis. "Direct Access to Databases through WWW Browsers", Hellenic Conference on New Information Technologies (NIT 98), Athens, Greece 1998 (In Greek).

[2] Kris Jamsa, Suleiman Lalani, Steve Weakly. "Web Programming", Jamsa Press.

[3] Martin Rennhackkamp. "Basic Web Architectures", DBMS, May 1997.

[4] David H. Crocker, "Standard for the format of ARPA Internet text messages", Chapter 2: "Notational Conventions", *http://www.ietf.org/rfc/rfc0822.txt,* August 1982.

[5] T. Berners-Lee, L. Masinter, M. McCahill. "Uniform Resource Locators", *http://www.ietf.org/rfc/rfc1738.txt,* December 1994.

[6] R. Fielding. "Relative Uniform Resource Locators", *http://www.ietf.org /rfc/rfc1808.txt,* June 1995.

[7] R. Fielding, H. Frystyk, T. Berners-Lee. "HTTP Version 1.0", *http://www.ietf.org/rfc/rfc1945.txt,* May 1996.

[8] R. Fielding, J. Gettys, J. Mogul, H. Frystyk Nielsen, T. Berners-Lee. "HTTP Version 1.1", *http://www.ietf.org/rfc/rfc2068.txt,* Jan 1997.

[9] Dave Raggett, Arnaud Le Hors, Ian Jacobs. "*HTML 4.0 Specification*", W3C Proposed Recommendation 7-Nov-1997, PR-HTML40-971107 *http://www.w3.org/TR/PR-html40/cover.html.*

[10] Rick Dobson. "Data Binding in Dynamic HTML", DBMS, March 1998.

[11] "Document Object Model", *http://www.w3.org/DOM/*, September 1997.

[12] "Standard Generalized Markup Language" (SGML). Published specification ISO 8879, *http://www.iso.ch/cate/d16387.html.*
[13] Tim Bray, Jean Paoli, C.M.Sperberg-McQueen. "Extensible Markup Language (XML) 1.0", *http://www.w3.org/TR/1998/REC-xml-19980210.*
[14] S.S. Bhowmick, W.-K Ng , E.P. Lim, S.K. Madria. "Join Processing in Web Databases", DEXA98, August 1998, *http://www.cais.ntu.edu.sg:8000 /~wkn/paper/dexa98.ps.gz.*
[15] M. Campione, K. Walrath. "The Java Tutorial".
[16] Patrick Chan, Rosanna Lee. "The Java Class Libraries", Addison-Wesley.