Adapting Databases and WebDAV Protocol

Bita Shadgar
Chamran University of Ahvaz, Iran
Department of Computer Science
University Road
0098 611 3331040
shadgar@cs.bris.ac.uk

Ian Holyer
University of Bristol, U.K.
Department of Computer Science
Merchant Venture's Building
0044 117 954 5148
Ian.Holyer@bristol.ac.uk

ABSTRACT
The ability of the Web to share data regardless of geographical location raises a new issue called remote authoring. With the Internet and Web browsers being independent of hardware, it becomes possible to build Web-enabled database applications. Many approaches are provided to integrate databases into the Web environment, which use the Web's protocol, i.e., HTTP to transfer the data between clients and servers. However, those methods are affected by the HTTP shortfalls with regard to remote authoring.

This paper introduces and discusses a new methodology for remote authoring of databases, which is based on the WebDAV protocol. It is a seamless and effective methodology for accessing and authoring databases, particularly in that it naturally benefits from the WebDAV advantages such as metadata and access control. These features establish a standard way of accessing database metadata, and increase the database security, while speeding up the database connection.

Categories and Subject Descriptors
H.3.4 [**World Wide Web**], H.2.4 [**Database Manager**].

General Terms
Design, Performance.

Keywords
Software Engineering, Web Authoring, Web Protocols, Databases.

1. INTRODUCTION
There are many benefits to integrating databases with the Web with regard to remote accessing of databases such as platform independence, cross-platform support, graphical user interface, scalable deployment and so on [1]. Integrating databases and the Web involves the use of dynamic Web pages along with a methodology or an approach to transport data between the browsers and servers, and to process this data. The applied methodology defines where the interaction is handled.

These approaches basically can be divided into three categories with regard to the efficiency and the necessary modification to the Web software that have to be made. Server-side, middle-layer and client-side approaches constitute those approaches [2, 3].

WWW 2004, May 17-22, 2004, New York, New York USA.
ACM 1-58113-844-X/04/0005.

Server-side approaches focus on extending the functionality of Web servers to access databases via the Web. Those methodologies normally use URL munging or RPC[1] via POST method to deliver data from clients to servers. Common Gateway Interface (CGI) is the oldest and probably most widely used approach to access databases, which falls in the server-side approaches category. Furthermore, Netscape Server and Microsoft Internet Information Server introduced a methodology to extending the Web server using the dynamic link library files called NSAPI and ISAPI, respectively. We can also name the Active Server Pages (ASP), Java Servlets, and Java Server Pages (JSP) as some other improved server-side methodologies [4-7].

Client-side approaches are another methodology introduced to handle the interaction between servers and clients. The idea is to distribute the application and send it to the client. The client then executes the code locally on the user`s machine. Parts of the user interface can be rebuilt on the Web and then run on the client computer. One simple example is to send a compiled program to the user and execute it there. This approach can lead to a better performance and high scalability. JDBC (Java DataBase Connectivity) and scripting languages such as JavaScript, JScript are examples of client-side methodologies [7, 8].

While the previous two approaches are concerned with accessing a single database from the Web, the middle-layer approaches allow for integration of data from distributed and heterogeneous data sources over the Web. One of the most popular middle-layers used today is based on CORBA (Common Object Request Broker Architecture) [9].

Although all of these approaches try to increase the functionality of servers and clients by different methodologies, they still use the same road map to transport the data. In other words, they apply the HTTP protocol as their transfer protocol to exchange data between servers and clients.

Moreover, none of those approaches provide database metadata. JDBC is the only method that provides some flexibility to enable the user to extract the database metadata. However, the programming is not too easy, and also it is still vendor dependent.

This paper is organized to explain how HTTP is an inappropriate protocol with reference to remote authoring in Section 2, followed by the solution, which involves the WebDAV protocol. Section 3 defines some of the WebDAV specifications in brief. This is followed by introducing a new methodology for integrating databases and the Web to capture databases as a resource of the WebDAV protocol in Section 4. This methodology, which is based on the WebDAV protocol, represents a major difference in

[1] RFC stands for Remote Procedure Call.

the protocol level comparing to other methodologies that are currently used. The section discusses advantages and disadvantages of this method whilst comparing it with others. Finally, Section 5 provides the conclusions and future work.

2. HTTP SHORTFALLS

Distributed authoring on the Web requires scaling content access across the number of resources, number of users, and transaction rates. Moreover, it must enable support for a broad user community that is made up of workgroups consisting of multiple users who are playing different roles. Supporting these requirements demands a decentralized repository with a simple, standard, multi-user, multi-version interface. Access is required based on open, nonproprietary document formats where possible. Collection operations are required in order to organize authored content into logical groupings for complexity management and namespace collision avoidance. Also extensibility is required across document content and views, resource metadata, and links between resources.

Many repository managers and source code control systems address these requirements, which are typical of any distributed, multi-user, and multi-version repository manager. However, current systems have proprietary interfaces and content formats which limit their appeal and applicability for web authoring. In other words, for Web authoring we need to provide:

- Support for efficient, scalable and secure remote editing.
- Improved efficiency of common editing operations.
- Locking mechanisms to prevent overwrite conflicts.
- Improved link management for non-HTML content types.
- An extensible attribute/value metadata facility for capturing information about a resource.
- Support for container data types (collections).
- Integrated versioning, variants, and configuration management into the Web.
- Supported efficient searching of resource properties and contents.

HTTP fulfills some of these goals and requirements. It is the remote procedure call protocol used to retrieve content by all current web browsers. It is widely implemented and deploys results in a stable, reliable communication. HTTP is a stateless, relatively secure, and authenticated protocol supporting persistent and pipelined connections, potentially through proxy architectures. Finally, HTTP is easily extended through a variety of mechanisms including CGI programs, Servlets, Java Server Pages, Active Server Pages and so on. HTTP/1.1 does all this with seven methods known as GET, HEAD, PUT, POST, DELETE, OPTIONS, and TRACE [10].

Despite all these capabilities, HTTP is not enough to support remote authoring on the Web [11]. One of the most important reasons is resulted by the HTTP POST method. This method has a sufficiently open definition so that almost any operation can be invoked using it. The server performs the stated operation and returns a message body in the response, which gives the results of the operation. Most of the methodologies discussed in Section 3.4 use the POST method to implement the transfer of different structures of data, such as database queries and result sets, from clients to servers and vise versa. The POST method also allows domain specific marshalling of parameters. In other words, parameters do not need to be mapped into HTTP methods and headers, which has the benefit of reducing unanticipated interactions with the rest of HTTP's operations.

However, the POST method ends up being a security hole through which almost any operation can be executed. Trying to look into the POST message body in order to determine what operation is being performed is a very difficult task when programming. Because each individual extension is free to marshal its parameters in separate ways, the POST method does not allow intermediaries to reveal the nature of a given request, whether it is safe or not. For similar reasons, performing access control on POST-based operations is extremely difficult, which causes insecurity of the HTTP protocol.

Altogether, the HTTP protocol provides no means of organizing the complex content that is typical of a Web server supporting many Web applications. HTTP does not support any metadata facility, nor provide any way to link documents other than those whose content models directly support links (e.g., HTML).

Versioning must be done with other repository manager or source code control systems, and then made available to the Web through a separate publish step. Older versions are often no longer accessible. Furthermore, the HTTP PUT method does not provide any means to prevent multiple authors from simultaneously updating the same resource, overwriting each other`s work. This is surely inadequate, especially in geographically diverse locations.

The solution is to improve the HTTP protocol to satisfy remote authoring needs. WebDAV, Web Distributed Authoring and Versioning, consists of extensions to HTTP/1.1 to support remote authoring of Web resources. It addresses the problem of distributed authoring on the Web and resolves these issues by providing a standard protocol for distributed authoring support based on the highly successful and widely adopted HTTP protocol.

WebDAV uses the *add new methods* approach to extend the HTTP functionality. This approach takes advantage of existing features such as operation precondition headers like If-[None-] Match. Operation-based security and access control are also easy, since the operation always occurs at a predictable location in the protocol stream (normally at the beginning of the first line of the request called as Request-Line). However, by using this approach it turns out that HTTP headers are not sufficient for marshalling many kinds of parameters. Furthermore, since existing HTTP headers can be applied to any new method, the interactions between existing headers and new methods need to be explicitly defined. Finally, the existing mechanisms for extending HTTP servers do not easily accommodate adding new methods.

However, in the end, the security and access control advantages of adding new methods outweighed the (relatively minor) drawbacks of the approach. In those cases in which parameters could not easily be marshalled into HTTP headers, WebDAV (Web-based Distributed Authoring and Versioning) as an extension to HTTP, has used XML in the message body to encode the parameters. Thus it is gaining some of the RPC via POST advantages without its security and access control disadvantages. The next section describes the WebDAV protocol in more detail.

3. WebDAV DEFINITIONS

WebDAV is an extension of the HTTP protocol to provide a coherent set of methods in order to provide authoring mechanisms on the Web. WebDAV is an official Internet Engineering Task

Force (IETF) standard protocol that was introduced in 1999 to support some operations on properties, collections and namespaces, and locks [12]. The nature of these operations is discussed as below.

- **Property operations:** Provide the ability to create, remove, and query meta-information about documents. They also provide the ability to specify links, which connect media types that are otherwise unable to contain embedded links. These are provided via the PROPFIND and PROPPATCH methods of the WebDAV protocol.

- **Namespace operations:** Provide the ability to create sets of related documents, and to retrieve a hierarchical listing of their members. They also present the ability to copy and move web resources and collections of resources. MKCOL, MOVE and COPY methods are defined in the WebDAV protocol in order to establish the namespace operations.

- **Locking operations:** Control access to resources in order to avoid lost updates in a distributed, multi-user authoring environment using the LOCK and UNLOCK methods.

WebDAV addresses the HTTP deficiencies by adding new methods to HTTP in order to support metadata, access control through locking, and namespace and collection management. These methods include new request and response headers as well as entity request and entity response body definitions.

The related parameters for each method are specified as request headers, or in the entity request body, or both. Complex methods use entity request bodies as well as passing the parameters with simple structure through the headers. Responses are again returned through either headers or entity response bodies. The response body generally contains the result of the method while the response headers contain information about the response.

WebDAV working groups also defined other complementary protocols and Internet drafts to WebDAV in order to provide versioning through DeltaV, and access control using the WebDAV Access Control Protocol. It also provides a mechanism for searching the resources and properties by introducing DASL (DAV Searching and Locating) via the SEARCH method.

As is evident, WebDAV provides some content management functionality. However, that functionality is accessible by HTTP as well, such as Microsoft FrontPage [13, 14], Zope1.10.3 [15, 16]. This raises the question of what advantages make WebDAV-based applications better than non-standard HTTP-based client

play a significant role in the structure of the Web, also need to be considered in the WebDAV protocol.

4. WEBDAD METHODOLOGY

The WebDAV protocol has basically been designed to complete the original vision of the Web as a medium for collaboration. The WebDAV protocol is particularly concerned with distributed Web authoring. We have invented WebDAD in order to provide an integrated and easy way for authoring databases by using the WebDAV protocol [26]. It describes an architecture to carry database-oriented requests from the client to the server, and send the response back from the server to the client.

WebDAD uses WebDAV methods to convey requests and responses between servers and clients. In other words, WebDAD describes a way to express a given SQL query in the form of a set of WebDAV methods. However, before being able to map SQL statements into the WebDAV methods, it is necessary to provide a database data model compatible with the WebDAV data model.

WebDAV methods are defined to operate on resources and collections (Section 3). WebDAD maps each record into a resource. It also considers each table as a collection of records in a Relational Database (RDB), which is the essential part of the mapping. But how to define URL addresses for records and tables is the issue discussed in Section 4.4.

WebDAD selects a relatively complete subset of SQL queries to map into the WebDAV methods. SQL includes statements that are used by database designers to define conceptual and internal schema for the database. Those statements are known as Data Definition Language (DDL) statements such as Create, Alter, Drop, Grant and Revoke. Furthermore, SQL provides statements to manipulate the database, known as the Data Manipulation Language (DML). Typical manipulations include retrieval, insertion, deletion, and modification of the data [27].

Our subset includes both DDL statements and DML statements. The WebDAD DDL statements are Create Table, Alter Table, Drop Table, Grant and Revoke. Also, WebDAD chooses Update, Insert, Select, and Delete statements as DML statements. Section 4.5 discusses the details of mapping each query into the relevant WebDAV methods. However, each SQL statement is normally mapped to more than one WebDAV method. Also since each SQL query is considered as one single transaction in a database, we need some kind of mechanism, which maps each sequence of relevant WebDAV methods corresponding to a given SQL query as one single transaction. For these we use the ATOMIZE method. This method is used to provide atomicity of each SQL statement, and is described in more detail in Section 5.1.

As illustrated in Figure 1, WebDAD consists of five components: SQL Parsing, ATOMIZE Method Generator (AMG), ATOMIZE Method Handler (AMH), Response Generator, and Result Producer. When a user asks for an SQL query, the SQL Parsing parses that query. During the parsing, the different elements of the query are recognized and saved. When the SQL Parser parses the SQL queries, if there is any syntax error, the WebDAV client reports that error to the user. The AMG parameterizes WebDAV methods relevant to the given SQL query, and encapsulates those methods into a ATOMIZE method. It then passes the ATOMIZE method onto the server. The AMH opens the ATOMIZE method to find out which SQL query has been requested.

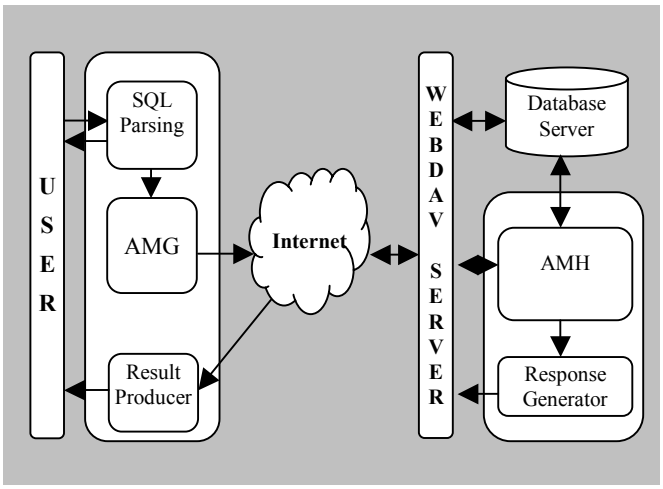

Figure 1. The WebDAD Architecture.

At this stage, the WebDAV server checks the permission of the user to access the data involved in the requested SQL query. If the user is authorized, the SQL query is sent to the database server. The database server runs the query and provides the result set for the user. Meanwhile the related methods included in the ATOMIZE method are run on the WebDAV server. The WebDAV server mirrors the metadata about objects in the database. The Response Generator generates the proper response for the ATOMIZE method based on the interactions in AMH. Also the Response generator reflects any exception or error in the WebDAV server or the database server to the client by using a proper message status code. Finally, in the client part, the Result Producer does the presentation work for the response and produces a proper result for the user.

4.1 Why WebDAD

This section explains some of the main reasons for introducing WebDAD. Basically the WebDAD specification and development arise from the fact that it is based on the WebDAV protocol, and so it naturally benefits from WebDAV's advantages with regard to remote authoring as discussed in Section 3.

The WebDAV protocol claims not only to support file systems, but to support all Web resources identified by URIs. To prove this claim it is necessary to consider other resources that satisfy the condition. WebDAD illustrates a way to express the database resources by URIs and investigate whether the WebDAV statement is true or whether it is just an exaggeration.

WebDAD references a new methodology of accessing databases based on the WebDAV protocol, which has essentially been defined to support the authoring of Web resources. The WebDAV functionality for file systems, which has been implemented by many different application vendors, shows that it is far better than HTTP-based client applications for authoring. However, there is no WebDAV-based application for authoring databases, which can practically be compared to the HTTP-based one. WebDAD make this comparison possible.

This new architecture for authoring databases moves the control on the users` access from inside the database to the outside. Therefore it provides more protection for the database in case of any unauthorized access, which increases the database security.

As well as the security issue, there is another crucial aspect, which is very time consuming in busy databases, called connection time. The WebDAD structure allows users to access the database by using a single connection and connection pool, which speeds up the connection time considerably.

Finally the last reason, which emphasizes WebDAD, regards the provision of a standardized mechanism to extract metadata. Considering the importance of metadata in our world today, the retrieval and storage of metadata are important aspects of the metadata phenomenon. Providing a standardized and easy way to access database metadata is another motivation for the provision of WebDAD.

4.2 WebDAD vs. Other Methodologies

The protocol level is the main difference between WebDAD, compared to other methodologies for authoring databases. All the methodologies that we discussed in Section 1 for the remote authoring of a database use the HTTP protocol to carry requests from a client to a server and return responses from the server to the client. However the WebDAD methodology uses the WebDAV protocol to transfer data between clients and servers.

As we already mentioned in Section 2, the HTTP POST method is an inappropriate method because it tunnels any kind of data, such as SQL queries, inside its body. However, WebDAD maps SQL to the WebDAV methods, so that each SQL query is expressed by a set of relevant WebDAV methods. As a result, WebDAV does not apply the POST method in order to pass queries into the server and thereby the security hole problem, which is raised by the POST method, is solved by the WebDAD architecture.

Moreover, WebDAD uses the access control policy that is provided by the WebDAV protocol using the Access Control protocol. In fact, it moves the access control check for different users from database servers to WebDAV servers. So, only one database connection exists for all the different users. Now using a pooled connection can significantly increase the speed of making a connection to a database. Normally when different users are accessing a database, it is necessary to make different connections to the database. The database server checks those connections and user authentication to access the database. Under this circumstance, if the network is busy, it is likely to hang up the database server, and consequently network efficiency decreases dramatically.

Another WebDAV advantage concerns metadata. It is important for database users to know about database metadata. Furthermore, since there is no standard language to extract this metadata, each database provides its own metadata-oriented commands and instructions.

The *java.sql.DatabaseMetadata* and *java.sql.ResultSetMetaData* interfaces are provided by JDBC to partially solve this problem. However, for some purposes users still need to know the name of tables, which are used to store metadata. Those table names are different in each database [7].

Moreover, working with JDBC needs Java and Web programming skills. Hence these issues do not make JDBC programming an easy way to access database metadata. In the absence of a standard methodology to extract database metadata, WebDAD uses the WebDAV metadata mechanism, which prepares database metadata on the WebDAV server. Therefore users can extract database metadata without even connecting to the database by using the simple method of WebDAV, i.e., PROPFIND.

Apart from extracting the database metadata in a standard way, this mechanism has more advantages. First, it is vendor independent. Second it is easy, because it does not need any programming. Users can simply extract the database metadata using the PROPFIND method. Finally, it provides the database metadata without needing to access the database. Therefore it makes the access more secure for users who are not allowed to access the data in the database, but who need to know about specific metadata as database application designers and programmers.

Finally, WebDAD provides an SQL Parser that parses SQL statements before they are passed to the database. Therefore, any malicious SQL statement is rejected before being passed to the database. This means WebDAD prevents SQL poisoning, which increases security. Also the SQL Parser reflects any syntax error in the SQL statement to the user, without sending the request on to the server. Regarding the fact that usually syntax errors happen a lot, the SQL Parser causes remarkable savings of server time and load.

4.3 WebDAD Drawbacks

Like any other methodology, WebDAD also raises some problems. Tailoring the database data model and operations onto the WebDAV data model and methods is the root of all the problems with which WebDAD is faced. Firstly, WebDAV methods are mostly intended to support file systems. This causes us not to be able to represent the very detailed aspects of SQL queries by WebDAV methods, so that at the moment it is necessary to pass the SQL query as a part of the request to the server.

Secondly, almost all of the SQL queries are represented by more than one WebDAV method. To deal with the sequence of WebDAV methods relevant to a given SQL query as one single transaction, WebDAD needs a mechanism for atomicity. We provide this mechanism by inventing a new method called the ATOMIZE method, discussed in detail in Section 5. However, the ATOMIZE method is not part of the WebDAV protocol and has not been standardized yet, which raises an interoperability problem.

Furthermore, the current specification of the WebDAV properties is not very efficient [28], especially when they are dealing with resources that present a high rate of repetitive properties such as databases. The types and representations of WebDAV properties affect their space and their search time.

In the WebDAD framework, logically, each attribute is considered as a property of the resource. However it is not an efficient design with the current specification of WebDAV properties, since this design is space and time consuming, especially for databases with big tables. In the current framework of WebDAD, each resource carries a pointer to the relevant record in the database [29]. The values of the records are stored in the tables belonging to the database repository. However, we propose Inheritance as a new specification for the WebDAV properties that solves the problem in Section 5.

Finally, WebDAV as an extension to the HTTP protocol is a stateless protocol. Therefore it faces the same problem as the HTTP protocol when handling transactions. This problem is

solved in current approaches by using cookies, which can be used in WebDAD as well.

4.4 WebDAD Data Model

Our architecture for the distributed authoring of databases based on the WebDAV protocol applies a data model that maps the database data model onto WebDAV resources. The WebDAD architecture is flexible enough to be applied to relational databases or object-oriented databases. We present a data model for both types of databases.

In a RDB, we propose to consider each record as a separate resource, and each table as a collection within WebDAV. Since URIs identify WebDAV resources, we represent each table by a URI using the schema and table name. Figure 2 illustrates a URL for a given database table.

```
http://host[:port]/rdb/schema/table
```

Figure 2: Representing a table in a RDB by a URL.

Furthermore we use the schema, table name and record in the structure of the URL to identify each record in a table, as shown in Figure 3.

```
http://host[:port]/rdb/schema/table/record
```

Figure 3: Representing a record in a RDB by a URL.

Since the primary key uniquely identifies each record in a table, the value of the primary key can be applied to distinguish between records in a table. The row identification is another alternative for uniquely identifying each record in a table; since the table may not provide a primary key [30]. To accomplish this, the URL in Figure 3 will change to one of the URLs in Figure 4.

```
http://host[:port]/rdb/schema/table/primaryKey
http://host[:port]/rdb/schema/table/rowIdentifier
```

Figure 4: URLs representing a record in a RDB uniquely.

In an OODB, objects and classes are analogous to RDB records and tables. Therefore we use URL patterns similar to Figure 2 and Figure 3 to represent each class and object respectively. Indeed, since the Object Identifier (OID) uniquely identifies each object in an OODB [30], we express each class and object in the database by URLs such as those given in Figure 5.

```
http://host[:port]/oodb/schema/class
http://host[:port]/oodb/schema/class/oid
```

Figure 5: URLs representing a class and an object in an OODB.

4.5 WebDAD Operations

The next issue is to map WebDAD operations into the WebDAV methods. In fact, the WebDAD operations are database operations such as SQL statements. For our application, we select a primitive subset of SQL statements. The semantics of this subset is supported by SQL99 and two of the most popular commercial database packages on the market (Microsoft SQL server and Oracle8i), and two of the best-known open source database products (PostgreSQL and MySQL) [31]. The syntax chosen for the subset is compatible with Oracle8i. The subset covers the most important SQL statements such as the Create Table, Alter Table, Delete, Drop Table, Insert, Select, Update, Grant and Revoke statements. WebDAD sends an SQL statement to the database and sends the corresponding WebDAV methods to the repository. Table 1 illustrates each SQL query and its relevant methods.

Table 1. SQL statements and relative WebDAV methods.

SQL Statement	Relative WebDAV Methods
Create Table	MKCOL, PROPPATCH
Alter Table	MOVE | PROPPATCH
Drop Table	DELETE (collection)
Select	SEARCH
Insert	[SEARCH ,] PUT
Update	SEARCH (where clause) [, SEARCH (set clause)]
Delete	[SEARCH,] DELETE (resource)
Grant	ACL (to set properties)
Revoke	ACL (to remove properties)

5. PROPOSAL FOR WEBDAV

The WebDAD framework could improve if the WebDAV protocol provided a better foundation for supporting methods and properties. In this section we propose two extensions to the WebDAV protocol in order to establish an improved protocol for authoring.

5.1 WebDAV and Atomicity

The WebDAV methods execute basic and simple operations. A complex operation such as a database operation is normally implemented through a sequence of WebDAV operations. Hence, it is necessary for a complex operation to be atomized as a single operation, so that the WebDAV methods relevant to the complex operation are all done or none are. Therefore, we propose to make use of an ATOMIZE method in order to provide atomicity in the WebDAV protocol.

Although the ATOMIZE method is not a standard method in WebDAV, there are, however, many discussions about it among the WebDAV group under the title of the BATCH method. This method has been suggested in order to increase performance and also implement transactions. However, for some reasons such as low priority and lack of consensus, it has been postponed [32-35].

In these discussions, the BATCH method is suggested as a mechanism for passing a group of methods in one request in order to improve the performance of the network. However, it is refused because it raises the same problems as the POST method does. The BATCH method is also proposed for implementing transactions so that a group of methods that form a transaction are passed to the server in one request using the BATCH method. Under this circumstance, intermediaries are not able to find out what methods and how many of them are tunneled by the BATCH method. This is not RESTful and so is rejected [36-37].

Although the ATOMIZE method is very similar to what is discussed for transactions, there is a major difference that leaves

the ATOMIZE method RESTful. This method is used to provide the atomicity of database operations, intermediaries can recognize the content of the method simply by retrieving the Pragma header as discussed in the next section.

5.1.1 ATOMIZE Method Request

This section presents the syntax and semantics of our proposed ATOMIZE method. Like the other WebDAV methods, the ATOMIZE method request consists of two parts, the header and the body. The header of the ATOMIZE method has the same syntax as other HTTP methods. We also set the value of the Pragma header[2] to the type of SQL query, such as create-table, select, insert, and so on.

However, we propose the syntax shown in Figure 6, for the ATOMIZE method`s entity body. The body is an XML document with an atomize element to include attributes and elements. The header and body of each method relevant to the given SQL query are passed to the server through a request element inside the atomize element.

```
<!ELEMENT atomize (request+)>
<!ELEMENT request (header, body, expected-status)>
<!ELEMENT header (#PCDATA)>
<!ELEMENT body (#PCDATA)>
<!ELEMENT expected-status (#PCDATA)>
<!ATTLIST atomize sql-query ENTITIES #REQUIRED>
<!ATTLIST atomize involved-objects CDATA #REQUIRED>
```

Figure 6: Syntax of a ATOMIZE method request body.

The expected status element representing the status code results from running the related method correctly. It is used to make sure that the result of running this method is the same as what is expected.

To present an instance of an ATOMIZE method we use an example concerning the creation of the Registrations table. Figure 7 illustrates an ATOMIZE method request related to the SQL query in the `sql-query` attribute, with relevant WebDAV methods as shown in Table 1, i.e., the MKCOL method followed by the PROPPATCH method.

5.1.2 ATOMIZE Method Response

The ATOMIZE method response is an XML document similar to other WebDAV methods. It presents the response of the last method in the ATOMIZE method that has been executed.

However, if the ATOMIZE method fails during the running of sub-requests in the ATOMIZE method, it returns the error status for the current request that has failed.

[2] The Pragma header is an HTTP request header, which contains any additional information that the client wishes to specify to the server [38].

```
ATOMIZE /rdb/schema HTTP/1.1
Host: databases.example
Content: text/xml; charset="utf-8"
Content-Length: xxxx
Pragma: create-table

<?xml version="1.0" encoding="utf-8" ?>
<atomize xmlns="http://authoring.db/webdad"
  involved-objects ="/rdb/public"
  sql-query = "Create Table Registrations(
     student varchar(6), unit varchar(9),
     year varchar(6), mark int, result varchar(1),
     primary key (student, unit, year),
     foreign key (student) references Students,
     foreign key (unit, year) references Units);">
   <request>
     <header>
         Header of MKCOL(Registrations)
     </header>
     <body>Body of MKCOL(Registrations)</body>
     <expected-status>201</expected-status>
   </request>
   <request>
     <header>
     Header of PROPPATCH(Registrations, properties)
     </header>
     <body>
     Body of PROPPATCH(Registrations, properties)
     </body>
     <expected-status>207</expected-status>
   </request>
</atomize>
```

Figure 7: ATOMIZE method request for creating a table.

5.2 WebDAV and Inheritance

Inheritance in the WebDAV properties is an issue that can save time and space in handling and searching the WebDAV properties. Since each single resource on the Web can be linked to an unrestricted number of properties, the rate of repetitive properties associated with resources grows remarkably when the size of Web sites increase.

For example, a university`s publication Web site includes hundreds of papers and books. Each book can be associated with properties such as the name of the author or authors, date, title, publisher, number of pages, and category. Also each paper can be linked to properties such as name of author or authors, date, title, name of conference or journal, and keywords.

As is evident in the above example, some resources have similar but not identical properties. If there were a large degree of similarity, it would be useful to be able to share the common properties. Inheritance allows one resource to be defined as a special case of a more general resource. These special cases are known as sub-resources and the more general cases are known as super-resources. By default a sub-resource inherits all the properties of its super-resource. Furthermore it defines its own unique properties, plus a live property named `subResourceOf`, so that its value refers to the URI of its super-resource. Each collection in WebDAV can be a super-resource or a sub-resource. However a non-collection resource can only be a sub-resource. This inheritance is a single inheritance[3].

In the above given example, Books and Papers collections are sub-resources of the Publication resource. The Publication resource is a

[3] Single inheritance declares that the sub-resources inherit from no more than one super-resource [26].

collection associated with the properties such as authors, date, and title, which are common properties between the Books and Papers collections. They inherit all the properties from their super-resource, i.e., the Publications resource. They also have their own unique properties, which are added to the inherited properties. The Books and Papers collections include all the published books and papers of the university as their internal resources. Each resource inherits all the properties from its super-resource.

Figure 8 illustrates the inheritance hierarchy for this example. As you see, the inheritance avoids repetition of the property names, which are repetitive in the non-collection resources.

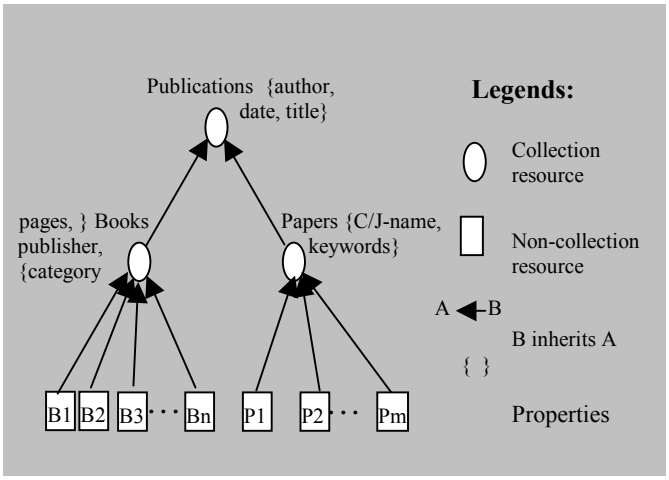

Figure 8: Inheritance hierarchy for university publications.

If the WebDAV protocol supported inheritance for properties, the WebDAD framework would change remarkably. Under this circumstance, WebDAD could define each table as a collection with the table attributes defined as properties for the collection. The value of each property carries the data type of the given attribute. The resources in this collection, which are in fact the records of the table, inherit all properties of the collection. However, the value of each property for the given record carries the corresponding field of that record. This collection is considered as the super-resource and records are its sub-resources.

In this scenario, the value of each record is represented via a sequence of property values, so that the WebDAV repository stores the records of the database. Furthermore, the table constraints are considered as a new collection in the table, which includes sub-resources such as Primary key, Foreign keys and other constraints. Each sub-resource defines its own properties.

Figure 9 illustrates how the Registrations table is represented via the inheritance hierarchy.

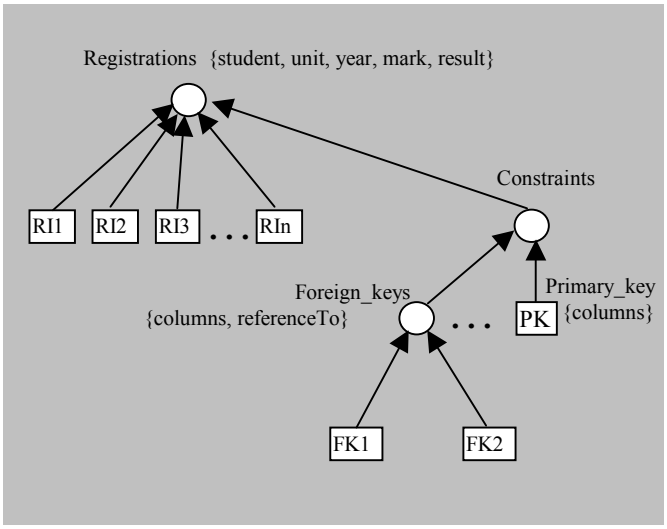

Figure 9: Inheritance hierarchy for the Registrations table.

6. CONCLUSIONS

The development of the WebDAV protocol in order to encompass remote authoring of every kind of Web resource has been investigated in this dissertation which promotes the WebDAV protocol from an authoring protocol for file systems to an authoring protocol for every kind of Web resource.

The paper referenced the need for authoring and accessing databases via the Web. It also presented a collection of possible mechanisms which are currently used for accessing databases through the Web.

Furthermore we introduced a new methodology for authoring databases, which almost preserves the advantages of other methods, but does not reflect their drawbacks (see Section 4). We invented this methodology, called WebDAD, based on the WebDAV protocol. This methodology is a seamless and easy way of authoring databases over the Web.

Considering the benefits and advantages of WebDAD compared to other methodologies for authoring databases, this methodology emphasizes the fact that the WebDAV protocol and its complementary Internet drafts such as DASL and WebDAV Access Control provide a better foundation for remote authoring of databases as well as file systems. This became more apparent when we mapped SQL statements into WebDAV methods.

Furthermore, the WebDAV specifications such as metadata and properties easily provide a repository of database metadata so that a user can extract the metadata in a standard way. Regarding the fact that there is no standard way for extracting database metadata, the role of WebDAV as a foundation for remote authoring databases becomes more comprehensible.

Also, we observed how the WebDAD framework using the WebDAV access control resolves the connection problem that occurs when JDBC is used in order to remotely author databases.

On the other hand, considering the differences between the structure of file systems and databases, it states that WebDAV needs more extensions to be considered as a package protocol for authoring every kind of Web resources. We suggest some of those

in Section 5, which make the protocol more flexible and practical to apply to every kind of Web resource.

7. ACKNOWLEDGMENTS

The authors would like to thank the WebDAV and Jakarta Slide mailing lists for their supports and cooperation.

8. REFERENCES

[1] Greg Riccardi. Principles of Database Systems with Internet and Java Applications. Addison Wesley. 2001. ISBN: 020161247X.

[2] Ralf Kramer. Databases on the Web: Technologies for Federation Architectures and Case Studies. *Proceedings of the International Conference on Management of Data and Symposium on Principles of Database System,* pages 503-506, USA, 1997.

[3] Daniela Florescue, Alon Levy and Alberto Mendelzon. Database Techniques for the World Wide Web: A Survey. *SIGMOD Record*, 27(3): 59-74, 1998.

[4] Tony Beveridge and Paul McGlashan. High Performance ISAPI/NSAPI Web Programming. The Coriolis Group. 1997. ASIN: 1576101517.

[5] Marco Bellinaso and Kevin Hoffman. ASP.NET Website Programming: Problem, Design, Solution. Wrox. 2003. ISBN: 0764543865.

[6] Mark Felton. CGI: Internet Programming in C++ and C. Prentice Hall. 1997. ISBN: 0137123582.

[7] M. Campione, K. Walrath, A. Huml, and the Tutorial Team. The Java Tutorial Continued. Addison Wesley. 1999. ISBN: 0201485583.

[8] H. M. Deitel, P. J. Deitel, T. R. Nieto, T. M. Lin and P. Sadhn. XML How to Program. Prentice Hall. 2001. ISBN: 0130284173.

[9] Bob Blakely. CORBA Security: an Introduction to Safe Computing with Objects. Addison Wesley. 2000. ISBN: 0201325659.

[10] R. Fielding, J. Gettys, J. C. Mogul, L. Masinter, P. Leach, H. Frystyk and T. Berners-Lee. Hypertext Transfer Protocol - HTTP/1.1. RFC 2616. 1999. http://www.rfc-editor.org/rfc/rfc2616.txt.

[11] E. James Whitehead, Yaron Y. Goland, WebDAV: A network protocol for remote collaborative authoring on the Web. *Proceedings of the Sixth European Conference on Computer Supported Cooperative Work (ECSCW`99)*, pages 291-310, Denmark, September 1999.

[12] Y. Goland, E. Whitehead, A. Faizi, S. R. Carter and D. Jensen. HTTP Extensions for Distributed Authoring – WEBDAV. RFC 2518. 1999. http://www.rfc-editor.org/rfc/rfc2518.txt.

[13] Mark Reed. The Fast Track to the Web: FrontPage Express. 1998. http://www.microsoft.com/windows98/usingwindows/internet/Articles/003Mar/FPExpress.asp.

[14] Microsoft Office. FrontPage 2003 Version Comparison. 2003. http://www.microsoft.com/office/frontpage/prodinfo/compare.mspx.

[15] Brian Lloyd. An Introduction to Zope. 2003. http://www.zope.org/Resources/Zope Intro.

[16] Zope Org. Zope Changes. http://www.zope.org/Products/Zope/Products/Zope/2.0.0a4/CHANGES.txt

[17] Lisa Dusseault. WebDAV efficiency. 2003. http://lists.w3.org/Archives/Public/w3c-dist-auth/2003AprJun/0087.html.

[18] KCura. 2002. http://www.kcura.com.

[19] Joe Orton. Cadaver. http://www.webdav.org/cadaver.

[20] Xythos WFS. www.xythos.com/home/xythos/wfs.html.

[21] Lisa Dusseault. WebDAV: Next Generation Collaborative Web Authoring. Prentice Hall. 2003. ISBN: 0130652083.

[22] WebDAV Projects. http://www.webdav.org/proj ects.

[23] Oracle. Oracle HTTP Server: WebDAV-enabled Collaboration. http://otn.oracle.com/products/ias/daily/dec04.html.

[24] Oracle Internet File System. http://products.datamation.com/dms/im/958570191.ht ml.

[25] Sung Kim, Kai Pan and Elias Sinderson. Mod_dav_dbms: A database backed DASL module for Apache. 2002. http://www.webdav.org/catacomb.

[26] B. Shadgar, I. Holyer, An Application for WebDAV-based Authoring of Databases - WebDAD, *Proceedings of the Twelfth International World Wide Web Conference WWW2003,* Hungary, May 2003.

[27] Thomas Connolly and Carolyn Begg. Database Systems, A Practical Approach to Design, Implementation, and Management. Addison Wesley. 2002. ISBN: 0201708574.

[28] E. James Whitehead and Yaron Y. Goland, The WebDAV Property Design. 2003. http://www.webdav.org.

[29] B. Shadgar, I. Holyer, WebDAD: A WebDAV implementation for authoring databases, *Proceedings of the IADIS International Conference WWW/Internet 2002,* pages 827-828, Portugal, November 2002.

[30] Ramez Elmasri and Shamkant B. Navathe. Fundamentals of Database Systems. Addison Wesley. 2000. ISBN: 0201542633.

[31] Kevin Kline and Daniel Kline. SQL in a Nutshell. O`Reilly & Associates. 2001. ISBN: 1565927443.

[32] WebDAV Mailing List. Interest in standardizing Batch methods. 2002. http://lists. w3.org/Archives/Public/w3c-dist-auth/2002JanMar/0001.html.

[33] WebDAV Mailing List. Proposal: WebDAV and transactions. 2002. http://lists.w3. org/Archives/Public/w3c-dist-auth/2002JulSep/0172.html

[34] WebDAV Mailing List. Proposal: WebDAV and Transactions. 2002. http://lists.w3.org/Archives/Public/ w3c-dist-auth/2002JulSep/0172.html.

[35] WebDAV Mailing List. RE: Interest in standardizing Batch methods. 2003. http://lists.w3.org/Archives/Public/w3c-dist-auth/2003JulSep/0023.html.

[36] Roy T. Fielding and Richard N. Taylor. Principled Design of the Modern Web Architecture, *ACM Transactions on Internet Technology*, 2(2): 115-150, May 2002.

[37] Robert McMillan. A RESTful approach to Web services. Network World. 2003. http://www.nwfusion.com/ee/2003/eerest.html#chart.