Developing Web Entry Forms Based on METADATA

Abdualrraouf A. Elbibas and M J. Ridley

Department of Computing, University of Bradford, Richmond Road,
Bradford BD7 1DP, UK
{a.a.m.elbibas, m.j.ridley}@bradford.ac.uk

Abstract. Reformatting information currently held in databases into Hypertext Mark-up language (HTML) forms suitable for the World-Wide Web (WWW) requires significant effort both in creating the forms initially and their subsequent maintenance. To avoid these costs, we make use of metadata that can be extracted from the catalogue tables in relational database system, using java database connectivity (JDBC). With this development framework the implementation and maintenance of various database backed web applications will be easier, faster and less error prone.

1 Introduction

Database technology has played a critical role in the information management field during the past years. It is believed that the integration of the Web and database technology will bring many opportunities for creating advanced information management applications [1]. With the increasing popularity and advancement of Web technology, many organizations want to Web-enable their existing applications and databases without having to modify existing host-based applications. This not only gives all of the existing applications a common, modern look and feel but also can deploy them on corporate Intranets, the public Internet, and newer Extranets [2]. Taking simple data from a database and placing it on the Web is a relatively simple task. However, in most cases, corporate data is maintained in a variety of sources, including legacy, relational, and object databases. It is much more complicated when these diverse data sources must be queried or updated [3]. A variety of new technologies and tools are being developed and brought to the market. Like Microsoft's Active Server Pages, Allaire's Cold fusion, and many others are used to simplify the implementation of integrated Web-DBMS sites. However, these technological advances are not matched by parallel efforts in data abstraction and modeling. Although it has been widely accepted in the community of Database Research that "the Web changes everything", [4], little effort has been devoted to adapt data design methods to the use of the web as the fundamental data interface. This paper propose an approach that can be applied to develop "safe" and easy to maintain HTML forms based on metadata extracted from system catalogue tables, using Java and particularly Java Database Connectivity

(JDBC), which provides a very general method for addressing databases and also includes Metadata features.

2 Metadata and JDBC

Metadata is information about data [5]. Databases store user data, and they also store information about the database itself. Most Database Management Systems (DBMS) have a set of system catalogue tables, which list tables in each database, column names in each table, primary keys, foreign keys, stored procedures, and so forth. Each DBMS has its own functions for getting information about table layouts and database features [6]. This metadata contains the database logic such as integrity constraints and referential constraints that specify restrictions on the data values accepted into our relational database tables. Java programs can interact with any RDBMS by using a JDBC API, as given in the java.sql package [7], which consists of a collection of interfaces and classes, such as: A Connection represents the interconnection between the Java client program and the database. The DatabaseMetaData provides information about the data contained in the database, such as table's name and the names and types of attributes. The Statement interface provides methods for executing queries, and is created in the context of a Connection. The result of a Statement or metadata retrieval is returned as a ResultSet, with methods for iterating through the set of results. The ResultSetMetaData provides information about the ResultSet such as attribute names and types [6, 7].

Tables 1, 2 shows the level of Metadata that exists and can be extracted by using for example DatabaseMetadata interface (getColumns, getIporetdKeys) for a given database schema described in Example 1.

Example 1: Part of suppliers parts Database Schema and primary and foreign key relation

```
table citylist  (
City varchar(15) primary key);
table suppliers(
     Supplier_No int not null,
     Supplier_Name varchar(30) not null,
     Status int,
     Street varchar(15) not null,
     City varchar(15) not null,
     Post_Code varchar(8) not null,
     Phone varchar(12),
     Fax varchar(12), )
primary key(Supplier_No),
foreign key(City) references citylist);
```

```
table spj(Supplier_No int,
      Part_No int,
      Project_No int,
      Qty int,
      Date date,
primary key (Supplier_N
        Part_No,Project_No),
foreign key(Supplier_No)
       references suppliers,
foreign key(Part_No)
       references part,
foreign key(Project_No)
       references project);
```

Table 1. Shows the metadata returned by using getColumns method (DatabaseMetadata interface)

TABLE_NAME	COLUMN_NAME	DATA_TYPE	TYPE_NAME	COLUMN_SIZE	ORDINAL_POSITION	IS_NULLABLE
supplier	supplier_no	4	Int4	4	1	NO
supplier	supplier_name	1	bpchar	20	2	NO
supplier	status	4	Int4	4	3	YES
supplier	street	1	bpchar	15	4	NO
supplier	city	1	bpchar	15	5	NO
supplier	post_code	1	bpchar	8	6	NO
supplier	phone	1	bpchar	12	7	YES
supplier	fax	1	bpchar	12	8	YES
….						
….						

Table 2. Shows the metadata returned by using getImportedKeys method (DatabaseMetadata interface)

PKTABLE_NAME	PKCOLUMN_NAME	FKTABLE_NAME	FKCOLUMN_NAME	UPDATE_RULE	DELETE_RULE
citylist	city	suppliers	city	3	3
part	Part_no	spj	Part_no	3	3
project	Project_no	spj	Project_no	3	3
suppliers	Supplier_no	spj	Supplier_no	3	3
….					
…					

3. Previous related work

One example of building a web interface design utilising information (Metadata) about a database is the Surveillance, Epidemiology and End-Results (SEER) database see Figure.1 [8]. Its purpose was to develop a web-based database interface engine whose content and structure is generated through interaction with a metadata table see Table.3 .

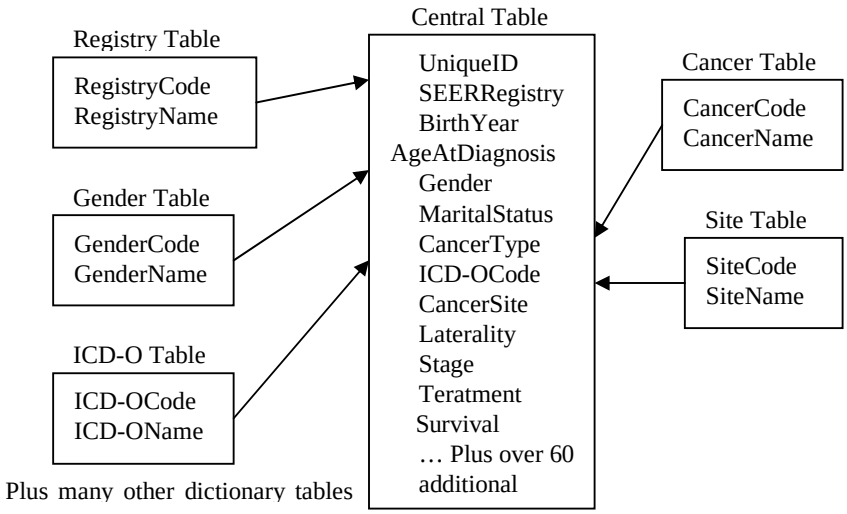

Fig.1 shows a subset of the SEER data model showing a large central table containing literal data (AgeAtDiagnosis, BirthYear) and codes whose definitions are provided in numerous linked dictionary tables [8].

Table 3. shows the handmade metadata table that represents the SEER data model diagrammed in Fig.3 Each field in the main central table that contains a code requiring a dictionary lookup is listed along with its associated linked table. Entries without linked table information are interpreted as actual values [8].

Table Name	Field Name	Field Data Type	Linked Table	Linked Table Field
Central Table	UniqueID	Number		
Central Table	SEERRegistry	Foreign	SEERRegistry	Registr Code
Central Table	BirthYear	Number		
Central Table	CancerType	Foreign	CancerType	CancerCode
Central Table	ICD-OCode	Foreign	ICD-OCode	ICD-OCode
Central Table	SiteCode	Foreign	SiteCode	SiteCode
. . .	Continues for each field in the Central Table Table			
Registry Table	RegistryCode	Number		
Registry Table	RegistryName	String		
Gender Table	GenderCode	Number		
Gender Table	GenderName	String		
Cancer Table	CancerCode	Number		
Cancer Table	CancerName	String		
. . .	Continues for each field in each lookup Table			

Compared to approaches like those shown above we have been working on extracting similar information about fields automatically at run time from a database using the JDBC Metadata classes as mentioned in section 2. These enable us to do things such as:

- If a field (based on a database column) is of type character with length (n) then automatically create the correct sized text field using n for the size attribute and display the field name as a text field label, see Fig. 2.
- If a field (based on a database column) is a foreign key, dynamically representing this field as a drop-down list, with data listed from the primary key table, this is to maintain referential integrity e.g. city field at supplier form, see Fig. 2.
- If a field (based on a database column) is of a Boolean type, represent it as a radio button with TRUE and FALSE values.

These are based on dynamically creating HTML alone; using Java and the metadata we can extend this technique to display help messages to the user and validate the input data, for example extending the previous examples:

- If a field (based on a database column) is of type character or integer, dynamically display a hint, saying this text field accepts character or integer values, see Fig 2.
- If a field (based on a database column) is required (not null) , dynamically display a hint saying this field is required, see Fig. 2.
- When the form is submitted, the data entered to the form will be validated and a proper feedback message will appear to the user, if submission was successful or not, see Fig. 3.

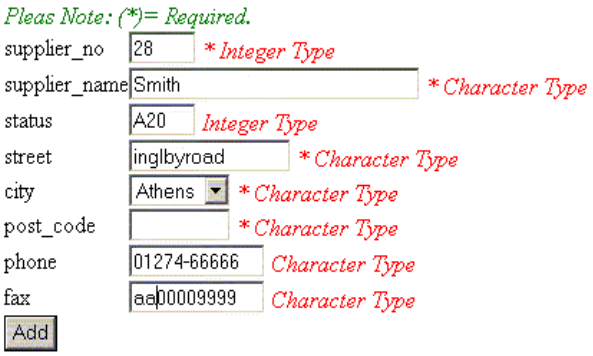

Fig. **2** Shows suppliers html entry form generated using metadata

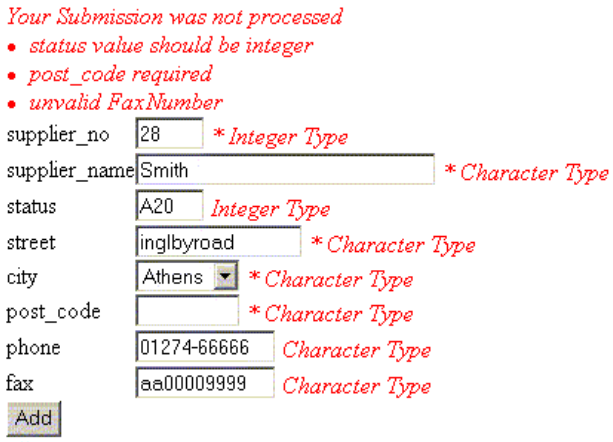

Fig.**3** Shows the feedback message after pressing ADD bottom

4. Three-Tier Solution

Fundamentally the database presentation aspect of our work is made up of three tiers [9]: web browser, servlets middleware, and database server. The three tiers are illustrated below in Fig.4.

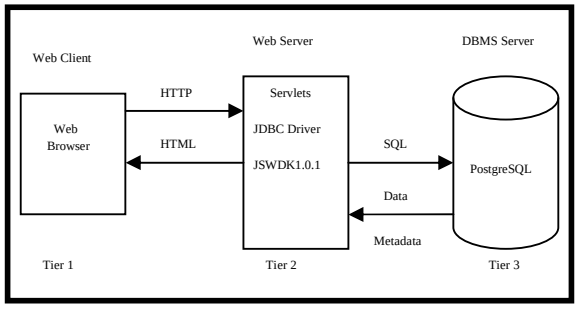

Fig.4 Three-tier solution

-Tier one: the Web browser

The first tier uses a Web browser to take advantage of the installed user base of this universal client. An HTML form is used for user-input and the results of the database query are returned as an HTML page. These pages are not necessarily static .html files and may well be generated by the servlets [9, 10]. Using HTML for user-input and displaying the data lowers the requirement of the client's browser version. This reduces the level of complexity required on the client's browser, this has numerous advantages, it reduces the complexity of the Web Applications wide deployment and it also makes the home user more confident as they won't have to down load the latest Java run time environment or the latest browser version to run the web application [10].

-Tier two: the Servlets/Web Server

The second tier is implemented with a Web server running Java servlets. The Java servlet is able to access the database and return an HTML page. Servlets are runnable on most commercial web servers. For this work we choose to use the (JSWDK) 1.0.1 web server [11].

-Tier three: the Database Server

The third tier is the back-end database server. The Java servlet can access information in the database provided that a JDBC driver exists. As we mentioned JDBC implmetations are available for the wide majority of databases. We choose to use the PosgreSQL DBMS an open source fully SQL92 compliant database [12] making the transfer of data easy should we decide to move to any other enterprise server.

5. Application Interaction

As can be seen, the application is partitioned into three different tiers. The basic steps of the interaction are described below:

1. The user enters information into servlet generated HTML form. The form data is passed to the Java servlet running on the Web server.

2. The Java servlet parses the form data and constructs an SQL statement. The SQL statement is passed to the database server using the Java Database Connection (JDBC).

3. The database server executes the SQL statement and returns a result set to the Java servlet.

4. The Java servlet processes the result set and constructs an HTML page with the data. The HTML page is then returned to the user's Web browser.

6. Features of our approach

- HTML forms that provide the query interface to the database will be generated and validated automatically.

- Because HTML is not stored, no maintenance of the HTML forms or database source is needed. Changes made to the database are reflected the next time the data is accessed.

- WWW access to a database can be provided to existing databases. No changes to the database are required. Rather, our method uses the structure and information contained in the database to generate HTML pages and links when the data is accessed.

- Display and database independence. If new HTML, or other display, standards emerge the database does not need to be changed.

- Links between Columns in Tables will be created and checked automatically. For database browsing this can block impossible queries or provide help on possible values. For database updates this can ensure integrity by automatically requiring all necessary data.

7. Conclusions and Future Work

The basic proposed model has made automated generation of HTML forms which include database logic in the form of constraints possible. The servlets/JDBC/PostgreSQL proved to be a powerful and fast technology. Most of the problems encountered were configuration problems that required carefully reading the associated documentation.

At present we have been investigating the possibilities and limitations of accessing metadata. We have established that only Java offers us a generalised metadata based approach. Other programming languages would require access to DBMS specific system catalog. There are however, issues of metadata support across JDBC implementations, and questions about performance implications of accessing metadata. Further work is also required in these areas when the approach is extended from querying metadata automatically to querying the data itself. This occurs if, following the ESSR example, we need to select all the possible languages from a database table to dynamically build a form element such as a pulldown list.

8. References

[1] L. Fang and H. Lu, Integrating Database and Web Technologies, International Journal of World Wide Web, vol. 1 No. 2 pp, 73-86,1998.

[2] J. Lu, W. Zhao and B. Glasson, Formal specifications of Web-to-database interfacing models, In Proceedings of Asia Pacific Web Conference (APWeb98), pp. 133-140, 1998.

[3] J. Carriere and R. Kazman, WebQuery: searching and visualizing the Web through connectivity, In Proceedings of the 6th International WWW Conference, pp. 701-711, 1997.

[4] P. Bernstein, M.Brodie, S. Ceri, D. DeWitt, M. Franklin, H. Garcia-Molina, J. Gray, J. Held, J. Hellerstein, H. V. Jagadish, M. Lesk, D. Maier, J. Naughton, H. Pir hesh, M. Stone-

braker, and J. Ullman, The Asilomar report on database research,ACM Sigmod Record, vol. 27 No. 4 pp. 74-80, Dec. 1998.
[5] K. G. Jeffery, Metadata the Future of Information Systems, 12th Conference on advanced information systems engineering, vol 1789, Jun 2000.
[6] G. Hamilton, R. Cattell and M. Fisher, JDBC Database Access with Java: A Tutorial and Annotated Reference}, Addison Wesley ISBN:
0201309955, 1997.
[7] Sun Microsystems, Inc., JDBCTM Data Access API, http://java.sun.com/products/jdbc/}.
[8] M. Weiner, M. Sherr and A. Cohen, Metadata tables to enable dynamic data modeling and web interface design: the SEER example, International Journal of Medical Informatics, Volume 65, Issue 1, Pages 51-58, ISSN: 1386-5056, April 2002
[9] S. Rajagopalan, R. Rajamani, R. Krishnaswamy and S. Vijendran, Java Servlet Programming Bible , Hungry Minds Inc, USA. ISBN: 0-7645-4839-5, 2002.
[10] J. Hunter and W. Crawford, Java Servlet Programming, O'Reilly and Associates, First Edition, USA. ISBN: 1-56592-391-x, 1998.
[11] JavaServerTM Web Development Kit, Version 1.0.1, Sun Microsystems, Inc, http://java.sun.com/products/servlet/README.html.
[12] PostgreSQL Inc. The world's most advanced Open Source Database, http://www.postgresql.org .