EPCC-SS98-07

WebDB:Bringing Structure to the WWW

John Lin

Abstract

The World Wide Web owes much of its success to its flexibility, and ease of use. Many tools do exist that aid the process of creating web pages, but hand coding is still just as easy. However, the free form nature of web authoring also creates inconsistencies, missing links, and hard to maintain web sites. In effect a web site that is not scalable. This report defines a higher level interface to solve some of the shortcomings of the web. In particular, we shall explain how through the use of a database these problems associated with the web can be solved.

Contents

1 Introduction

The World Wide Web (WWW) owes its enormous success to its simplicity and attractive user interface. Despite the establishment of a standard for HyperText Mark-up Langauge (HTML), the language that forms basis of the World Wide Web, the dynamic nature of the Web coupled with proprietary additions to the mark-up language by the browser vendors make writing and maintaining a web site an arduous undertaking. To use an analogy the current web is at the level of assembly code - all the links are hard coded and absolute, and this is at the heart of why the web is so difficult to maintain.

Several tools exist to help write and maintain pages, but we suggest that the problem, fundamentally, is that the wrong model is being used. What is needed is the Web equivalent of compilers or interpreters to allow the construction of Web pages to take pace at a higher level. To this end, we propose that the Web should be seen not as a collection of files but as a structured storage and retrieval mechanism - a database. There are already many initiatives underway to interface Web servers with databases. This is not the same thing. We are not trying to define an interface, we are trying to redefine the Web. In a sense the intention is to bring the WWW closer to its better-designed, but less popular, cousin Hyper-G. Hyper-G is a much more sophisticated distributed hypertext system. It allows the same set of documents to have different routes through them depending on the context they are presented in, without having to modify the document source. In a chain of documents with forward and backward links, insertions and deletions can be made in a single operation. The database underlying the system understands these structural aspects of the pages, so once set up the author can let these happen automatically.

In order get a better understanding of what is required in such a database standard, we shall review some of the problems with the current WWW. The requirements for a good solution will then become more apparent.

2 Problems and Solution Requirements

Modern Web pages are essentially pieces of information loosely bounded together via HTML. HTML allows authors to control both the physical and logical structure of a Web page. Though flexible, the approach promotes many bad design habits. Often, authors will design the look and feel before the actual content of the page. Often, considerations for non-GUI based browsers are ignored. Many people use the bold tag <B> instead of the em tag <EM> for the purpose of drawing attention to a particular piece of text.

Another problem is that anchors and links serve to establish the relationship between two pages. Links are hard coded and tied to a specific file. Any change to the name or the location of storage for the file will require changes to all the related links. A process that is both tedious and error prone; subjecting the user to the often annoying "HTTP Error 404 (Not Found)" message. Even when the link is not broken, it might not be the correct document that is referenced due to the absoluteness of link.

What is needed to solve these problems is a fundamental shift of our point of view. We suggest the use of data centric model to in place of the old file centric model. Essentially, we want to separate the content and logical structure of a page from the physical structure. Thus we ask the question of what to present before considering the question of how to present it. By designing the web site around the content, we develop a more structured, adaptable Web page suitable for browsers with varying capabilities. Links are no longer just relationship between files. They represent explicit relationship among pieces of information.

To better manage the data in this new model, we propose the use of a database. Web pages can now be thought of as different views of the data in a database. This is far different from an attempt to create a Web interface to a database. What we aim to achieve is to exploit the strong relationship associated with the data stored and bind the presentation specification to the data through a standardised interface. The database just happens to be a good medium of storage for both data and the associated relationship. With these design requirements in mind, we shall now give an overview WebDB.

3 Overview of WebDb

WebDB is composed of two distinct components a database and a compiler. Figure 1 illustrates the general architecture of the WebDB.

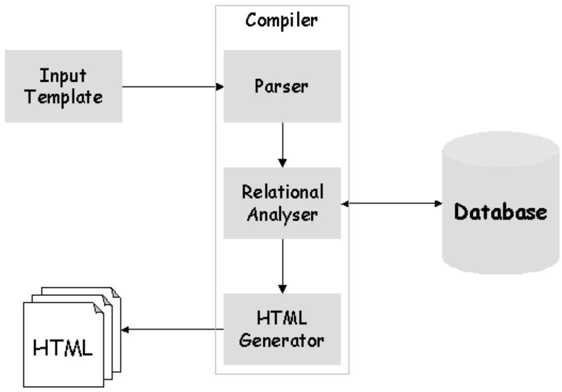


Figure 1: Architecture of WebDB

The database is where the contents of the pages are stored. The compiler forms the bridge between the database and the final Web pages. A set of templates, submitted by the user, form the input to the compiler. The output to the compiler is a set of web pages.

3.1 The Database

The database is the storage unit of the data that is to be presented on the web. The main advantage for using database is that the context associated with the data is also stored. For example, the string "Bobby" can represent any name. However, when stored in a table named "Dog" rather than "Employee", we know that Bobby is a name of a dog, rather than a person. It should be noted that not all text that is in the final HTML is stored in the database. Only the relevant information of the page is stored. WebDB allows high degree of flexibility in terms of the types of database that can be used with the system. Any relational database can be used in WebDB. The only requirements is that database must be normalised to at least the third normal form. The tables in the database must satisfy the following criteria:

- All column must contain atomic value
- Non-key attributes fully dependent on primary key
- All functional dependency are eliminated

The motivation behind normalisation is to eliminate duplicate data in the database. It will become more obvious later that the process guarantees the uniqueness of the anchor or locations referenced by hyperlinks. A primary key that allows the unique identification of the rows of a table is created during normalisation. Since fields uniquely identify the columns of a table, the addition of primary key allows us to address any piece of data within in a table through its primary key and the field name.

3.2 The Compiler

The compiler is the glue that binds the data stored to the layout information. Compilation is divided into three stages - parsing, relational analysis, and output. The input is a set of files that contains layout information specified by the user. Relational analysis is performed after the input file is completely parsed. The compiler then generates the appropriate web pages.

3.2.1 Input templates

The input template is used by the user to control both the logical and physical structure of the output. Each template eventually be translated into a web page or a part of a web page. The user must envision a web page as a collection of related information. Each resource that resides on the page physically partitions out a section. A template specifies the appearance and the resource for that section. All the sectional templates are included in a main template. This defines the final web page that is partitioned by the sectional templates.

The syntax for the template is essentially HTML with a few extra compiler directives. To represent entities in the database, we decided to borrow the syntax for entities from XML and SGML. Each field of a table in the datable is specified by enclosing the table and field name within a ampersand and an semicolon. Similar to SQL, the period "." serves as a separator between the table name and field name. Since entities in SGML behave like macros, you can construct more complex entities from fields of the database. For example, in most databases the attribute "name" is usually stored in two separate fields - "first_name" and "last_name". To refer to the name of something on the template, one can make use of the following entity declaration:

```
<ENTITY name "&first_name; &last_name;" >
```

There is an explicit space character inserted between "first_name" and "last_name". For clarity of syntax, we have omitted the prepending of the table name to the field name.

Non-textual entities can also be represented by the entity declarations. For example, instead of displaying "name" as a string, we would like to use the actual picture of the person. This is achieved with the following declaration:

```
<ENTITY employee.picture "/photos/&name;.gif">
```

To actually display a picture in the final HTML:

<IMAGE src=&employee.picture;>

The main template file must specify a table as the primary table. This table will be used as the starting point during the relational analysis phase of compiling. It is the relationship between the primary table and other tables used in the template that will determine the physical organisation of the web page.

A two-tier indirection is used as the architecture for links and anchors. The syntax for specifying links is still the same as that of HTML. The only difference is in the logical specification. Instead of referring to an actual web page, the user refers to the table name of the entity of interest. After all the web pages are generated from the templates, the compiler will begin the process of binding the pages to their respective links.

3.2.2 Output files

For this project, only HTML is generated. Ideally, the compiler can be modified to generate other mark-up language such as XML. The output files consist of a set of HTML files placed in appropriate directories. These correspond to the document hierarchy for the web site.

There is no restriction as to how many entities in the database are presented on a single output file. Based on each entity's relationship with other entities on the same page, appropriate number of frames, if any are generated. The details of how the compiler determine the actual HTML structuring elements used such as frames and lists will be discussed in the next section.

4 Exploiting Relationship

After the completion of the normalisation process, only one-to-one (1-1) or one-to-many (1-n) relationships exist among the tables in the database. The types of relationship between entities specified in the input template determine the final structure of the Web pages produced. We are now ready to discuss the analysis performed by the compiler on the input to generate the proper final output. There are basically two general types of possible scenario, one for 1-1 relationship, and another for 1-n relationship. The 1-n relationship further depends on whether the numbers of tables are directly related. An indirectly related table will have intermediate tables between the two. The opposite condition applies to directly related. In addition, two different views can be taken for the 1-n relationship. We can either view the relationship from the perspective of the one side of the 1-n relationship or from the many side. Each of the conditions will now be examined.

4.1 Directly related tables

4.1.1 One to one relationship

For a 1-to-1 relationship, the presentations of entities are straightforward. Each entity from one table will map to exactly another entity that resides in another table. Since no entity is repeated in this relationship, we can just substitute the appropriate data from the database whenever it is referenced in the template file. This general scheme is still applicable even if the tables are not directly related. For the special case that there is only one table referenced in the input template, it is obvious that this rule applies as well.

4.1.2 One to many relationship

The table on the one side of the relationship is the primary table. The table that references the primary table via the foreign key is the foreign table or related table. Recall that each entry in the foreign key field must have only one corresponding value in the primary key of the primary table. The foreign key must always reference a primary key of another table.

Recall that in the input template, the user must specify a base view table. Two different scenarios are possible depending on whether the base view table is the primary table or the foreign table. If the base view table is the primary table, then a new web page is created for each row of the base view table. Thus each of these web pages will be the anchor for an entity in the entire web site. Each of the newly created web pages will have a name corresponding to the values of their primary keys. The compiler will create a directory named after the base view table. Since values in the primary key field are unique, there would never be a problem of conflicting file names. A new frame is created to contain all the entity that has the same foreign key value as the primary key value of that particular web page.

If the base view table is a foreign table, i.e. one or more of its fields references the primary key of another table, then only one HTML page is required. All the data in the primary and foreign table are listed in the same web page. The user has the option of making these database references a link. If this is the case, then the compiler would link the appropriate entity to their corresponding anchors.

4.2 Indirectly related tabels

4.2.1 One to one relationship

By the same logic of the one-to-one relationship of the directly related table we can apply the same rules here.

4.2.2 One to many relationship

We can decompose this type of relationship by analysing both the primary and related table as well as all the intermediate tables. To accomplish this we can make use of a relationship tree. Each node in the tree is a table. The relationships are branches on the tree. The leaf is the actual table referenced in the input template. The root of of the tree is the base view table. We begin by traversing down the tree toward the leaf. For each pair of tables, we apply the rules shown in the directly related tables section.

5 Case Study

We shall now apply our theory to an example web site. We want web pages that show the relationship between employees and project. The web pages must express the "work" relationship.

For each employee, we want to know all the project(s) that he is working on. For each project, we want to know all the employee(s) that are participants of the project.

Since this is a fairly simple problem, we will do away with the modelling, decomposition, and normalisation of the data. The final tables are, stored using MS-Access, are shown below.

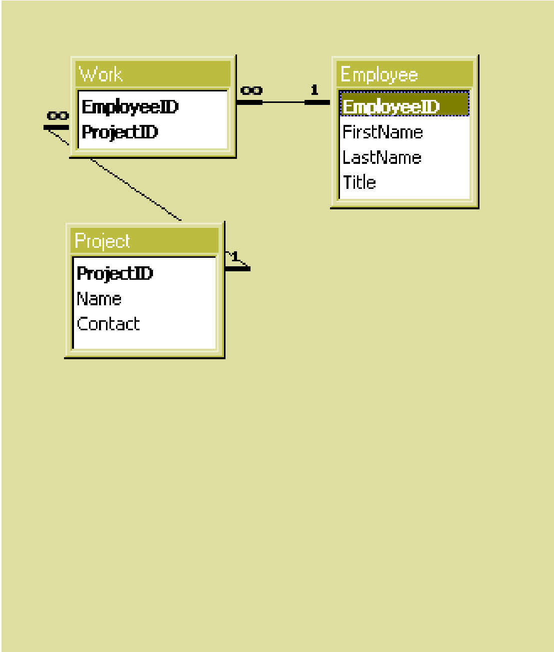


Figure 2: Relationship of Project, Employee, and Work tables

An intermediate table "Work" is created. Both employee and project has a one to many relationship with the "Work" table. An employee can work on multiple projects; each project has multiple workers selected from the employee table. EmployeeID, ProjectID are the primary keys. The "work" table has a primary key that is composite, i.e. more than one field are used to uniquely identify each record in the table. The composite key of the "Work" table is also a foreign key.

The next diagram shows how the compiler translates the entities in project and employee tables to their corresponding web pages.

Because employee is on the one side of one to many relationship, the compiler will generate the following:

- A new HTML for each entity in the table
- A directory named "employee" that contains all the web pages generated from the "Employee" table
- a new frame is created to contain all the project that matches the corresponding employee entity
- each of the project is presented as a link to their respective project pages
- a single index page listing all the information on the employee table

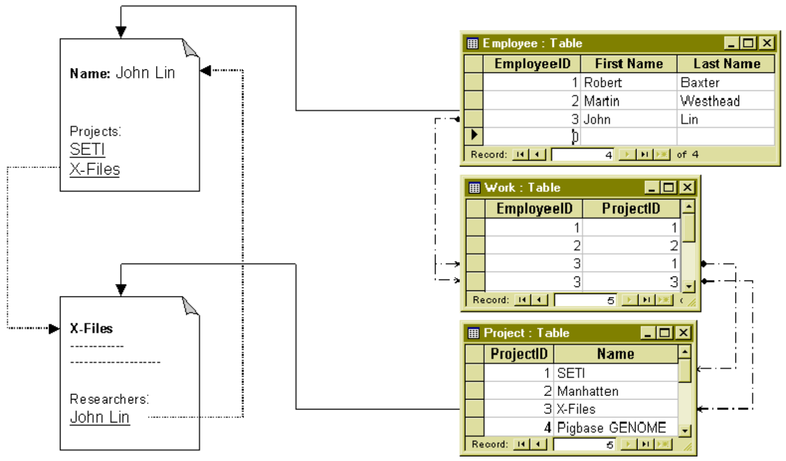


Figure 3: Example translation of tables into web pages

The same process is repeated for the project table. The binding of the links is the last stage performed on all pages generated. In the end three directories - project, employee, work are created by the compiler. Although not shown, one web page exists in the work directory. It contains a list of links to employee and project web pages The project directory contians four web pages. Each has an frame attached to it listing all the employees. The employee web page is similar in structure.

6 Future Direction

The following are some problems not yet addressed by this report

6.1 External link management

WebDB guarantees the consistency and availability of information within the local site. However, external links still have potential to be broken. One simple solution is to store all external references in a master table. Periodically, a server process can check if the links are broken and notify the site manager.

6.2 Dynamic notification of change

Every time the data in the database changes, the user has to recompile the entire site. The problem is aggravated when an entire row in a table is deleted or the structure of a table changes. A nice possible feature is to use CGI to dynamically generate the web pages so that the site is always consistent.

6.3 Pre-design relational analysis

Currently the input template file is still coded by hand, a subject for potential typing errors. A graphical user interface that can aid the generation of the input file would be a great feature to add to WebDB. By analysing the relationship of the underlying database beforehand, the user becomes more aware of the relationship of their contents. This enables the correct usage of data at design time rather than at compile time. A good example of a class of application that satisfies web-authoring requirement are page layout applications. Most page layout programs require the designer to import all the contents into the program. This clear division of content and layout complements our design requirement.

7 Conclusion

WebDB began as a proof of concept project. What we set to prove is that a higher level of specification for web authoring is possible using a database. Using the relationship of data in a relational database, we show that a translation from database to web page is possible. By sacrificing some of the freedom in design, we show that a well-structured web site can be easily created and maintained. Most importantly this new approach does not require numerous modifications to existing systems.

References

[1] "Inside Hyper-G" available on the WWW at URL http://www.fed.de/hgcourse-/inside

[2] SoftQuad Inc.:"The SGLM Primer", (1995)

[3] Khare, R., Rifkin, A.: "X Marks the Spot" (1997)

[4] Garshol, L. M.:"Introduction to XML" (1998)

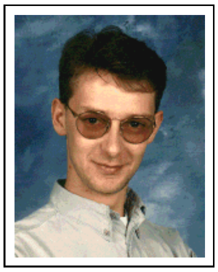
[5] Marchal, B.:"Basic SGML" available on the WWW at URL http://www.pineapplesoft.com/reports/sgml/basic.html (1997)

[6] Flohr, U.:"Hyper-G Organises the Web". BYTE magazine, November (1995), 59-64

[7] Maler, E.: "XML and Broken Links" available on the WWW at URL http://www.sil.org/sgml/maler980331.htm (1998)

John Lin
Computer Engineering
University of California, San Diego, USA
E-mail `johnlin@ucsd.edu`

Martin Westhead
Supervisor
Edinburgh Parallel Computing Center



Rob Baxtor
Supervisor
Edinburgh Parallel Computing Center

Special thanks to Hon Yau for his assistance with JDBC.