Zelig: Schema−Based Generation of Soft WWW Database Applications

Carlos A. Varela, Caroline C. Hayes
Department of Computer Science
University of Illinois at Urbana−Champaign
cvarela@cs.uiuc.edu, hayes@cs.uiuc.edu

URL: http://fiaker.ncsa.uiuc.edu:8080/WWW94.html

Abstract

The World−Wide Web brings a global information universe into existence using available technology [Berners−Lee et al. 1992]. In order to fully realize the benefits of this information system, methodologies need to be developed for the creation of scripts that query existing databases and produce effective user interfaces. Present practice falls short of this goal in two areas; first, interface changes require direct modification of the scripts, and second, user interfaces are *hard*, in the sense that they don't adapt to database usage.

We present Zelig, a schema−based approach to HTML document generation that addresses both these problems. First, Zelig uses ZHTML schemata, which are HTML documents commented with directives for document generation. And second, Zelig contains an expert module which gives advice regarding the underlying data structures and interface design issues. This approach allows *soft* or *evolving* database applications that keep track of usage and self−adapt to increase database efficiency and to improve human−computer interaction. As an example, we have used this approach to automatically generate four different WWW interfaces to the CCSO phone nameserver database software.

1. Introduction

In order to fully realize the benefits of the World−Wide Web, we need to be able to effectively interchange information with existing databases. Common Gateway Interface scripts [McCool 1993] are programs developed to perform this communication between database servers and WWW clients.

These scripts are written in any programming language (like C, C++ or PERL) and their main functions are:

- To receive the information from the WWW client under the hypertext transfer protocol (http) [Berners−Lee 1993].
- To perform a query for the database server, allowing the WWW server to act as a database client.
- To parse the database server results.
- To generate an HTML document [Berners−Lee & Connolly 1993] and send it to the WWW client.

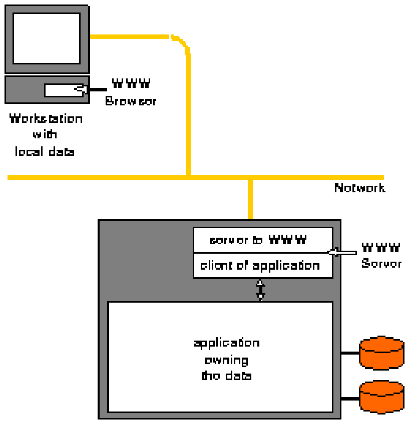

Fig 1. Purpose of a script in a WWW server [Berners–Lee & Cailliau 1992].

We are addressing two difficulties that CGI script developers find very often: how to modify their generated user interfaces without changing the source code of the programs, and how to generate flexible enough interfaces to meet the needs of very different types of users.

In order to modify the user interfaces created by CGI scripts, currently developers have to go through a difficult **two–level** process. First, to change the script that generates the hypertext documents. Then, to make sure that the generated documents satisfy the original interface specifications. Our schema–based HTML generation reduces the complexity of this task by using schemata, where changes to the interface can be performed more directly.

Another problem lies in the generation of *soft* user interfaces to existing databases. Current WWW hypertext interfaces to existing databases [Ng 1993] are *hard*, in the sense that they do not evolve according to usage statistics or particular user data inputs. This issue becomes crucial if we consider the difficulty in predicting the usage of a world–wide accessed database both in terms of the way in which data is used and the change in demand for items over time. We have followed a schema–based approach, inspired by other program synthesis researchers [Dershowitz 1983, Bhansali 1991], to overcome this problem.

The presented approach allows the interface and the database to evolve based on usage statistics and expert rules for database design and human–computer interaction. Thus, we have identified two main components in the problem of generating design evolution decisions for database applications at run–time: changes in the underlying data structures and in the user interface.

Following is a guide to the paper: In section 2, we present a methodology for schema selection and instantiation based on current database transaction results and ZHTML schemata. Section 3 is a description of the main heuristics used for modification to underlying data structures and user interfaces. Section 4 presents several running examples based on the database software for phone directory nameservers developed at CCSO [Dorner 1992]. Finally, in section 5, we present some conclusions.

2. Schema–Based HTML Generation

In traditional HTML generation, user interfaces are created by scripts directly. This implies that changes to interfaces have to be performed at the level of the source code of the script. We present a methodology based on schemata, to allow designers to debug and maintain the user interfaces without directly changing the scripts.

In this section, we will explain the information that is provided by the schemata to the scripts for document generation. Then, we will give a description of ZHTML, the language to write these schemata, which is an enhancement to HTML incorporating directives for database interface generation.

The scripts base their hypertext generation not only on the current parsed database query results, but also on existing ZHTML schemata. We can further categorize this information as:

- **Current database transaction results.** These are the results generated by a query to a database. We assume that the script knows how to interact with the database server, parse its results, and standardize the query information to the following taxonomy:
  - On an *application* level.
    The most general database information. For example: name, number of tables, default query table.
  - On an *object* level.
    The information for a specific object of our application. For example the table PEOPLE may have: name, number of records, number of fields, default field for queries.
  - On a *field* level.
    The most specific information about a query for a given record. For example: field names, default field values, current field values, length, type, and access policy.
- **Existing ZHTML schemata.** These are provided by the interface designer, but can be easily created by modifying an HTML document with the desired output from the current database query. We have created several schemata for the phone database example:
  - *WAIS*–based user interface schema.
    This schema is based on the Wide Area Information Servers system [Kahle & Medlar 1991]] and uses an HTML <ISINDEX> tag. It is important to note that there are many indexed databases in the Web that use this information system as the basis for database search.
  - *Forms*–based user interface schema.
    This schema makes use of the forms–based WWW browsers to provide a more friendly user interface with menus and widgets to perform the most common database operations.
  - *Application*–specific user interface schema.
    This schema is usually an advanced interface tailored to the specific needs of a database user interface. It still allows evolution, in the sense that certain constructs don't imply any order or level of access in the generated HTML documents.

A ZHTML schema is an HTML document which has been annotated with comments, which are used as directives for the script. These comments are parsed and executed by the script, and the resulting text is placed instead of the original comment. This is performed at run–time, using the current database query results. Future work includes writing script code generators departing from these schemata.

The ZHTML comments are similar to HTML constructs. They are generally of the form:

```
<!ZTAG> ZHTML body <!/ZTAG>
```

There are several Zelig directives with different functionality, including: print a variable value, run an external function printing its output, conditionally include the ZHTML body, traverse all the current database records invoking Zelig recursively on the ZHTML body and traverse all the fields in a specific object.

Following are the main constructs of this Document Type, even though a formal Document Type Definition (like the DTD shown for HTML in [Berners–Lee & Connolly 1993]) is still in progress:

- `<!ZPRINT variable>`
  returns the current value of an HTML form variable, or an *application*–level variable. If the variable is *object*–level or *field–level*, then it needs to be in the scope of a `ZFOR` tag.
- `<!ZRUN external_fn>`

runs the script function *external_fn* and returns its output.

- <!ZIF *cond-expr*> *ZHTML body* <!/ZIF>
  returns the output from *ZHTML body* if *cond−expr* is true. It returns the null string otherwise.
- <!ZFOR TYPE=*traversal_type*> *ZHTML body* <!/ZFOR>
  traverses all the tables, records or fields of the current query, depending on the *traversal_type* (which can vae the value TABLES, RECORDS, or FIELDS) and returns the output from *ZHTML body* instantiated to each of the loop elements in the query.

3. Soft WWW Database Applications

Current WWW hypertext interfaces to existing databases are *hard*, in the sense that they do not evolve according to usage statistics or particular user data inputs. This issue becomes crucial if we consider the difficulty in predicting the usage of a world−wide accessed database both in terms of the way in which data is used and the change in demand for items over time. We have identified two main components in the problem of generating design evolution decisions for database applications at run−time: changes in the underlying data structures and in the user interface.

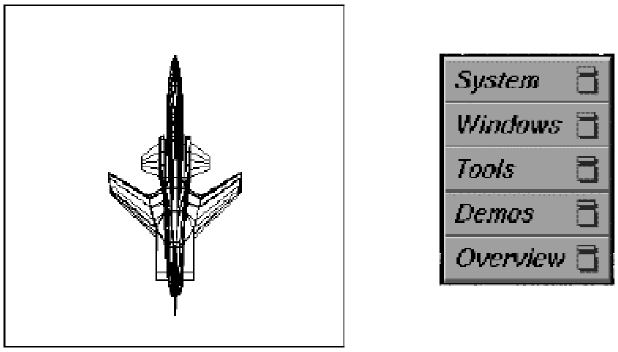

Fig 2. Difference between hard and soft designs [Varela & Hayes 1993].

3.1. Underlying Data Structures

Modifications to the *underlying data structures* of the database are intended to optimize access time and storage space in a global manner (they are directed to the database server, and are invisible to the WWW clients.) These modifications include database physical reorganization issues such as sorting tables and creating and deleting indexes. Our proposed methodology allows two specific data structure changes [Varela & Hayes 1993]:

- simplification of many−to−many relationships between objects because of large branching factors, and
- creation of new objects versus field replication to improve access times.

The rules for deriving the changes in data structures, were written in OPS5, a production system developed at CMU [Forgy 1981].

3.2. User Interfaces

Modifications to the *user interface* are intended to provide a more effective human−computer interaction for database operations. Different users may have very different views of the data, individual interfaces need not be the same, instead they should evolve as a result of collected usage statistics. This interface evolution includes:

- changing default values to reflect most commonly accessed data,
- reorganizing the order of user interface actions, to make variables with high variance first. These are the variables in the forms whose value is more often

changed from the default value.

- changing the level of access to certain database operations (like sorting, filtering and reporting options) according to their usage. Frequently used operations would be directly accessible, while rare operations would go to a second level of access (following a link.) This would allow the generation of smaller documents and provide a more friendly interface.

4. An Example: The CCSO Phone Nameserver Database

We have automatically generated four different interfaces following our schema–based approach to writing scripts. If you are reading this abstract in a forms–based WWW browser, such as NCSA Mosaic [NCSA 1993], you can follow the links to execute these interfaces.

- **phfa:**

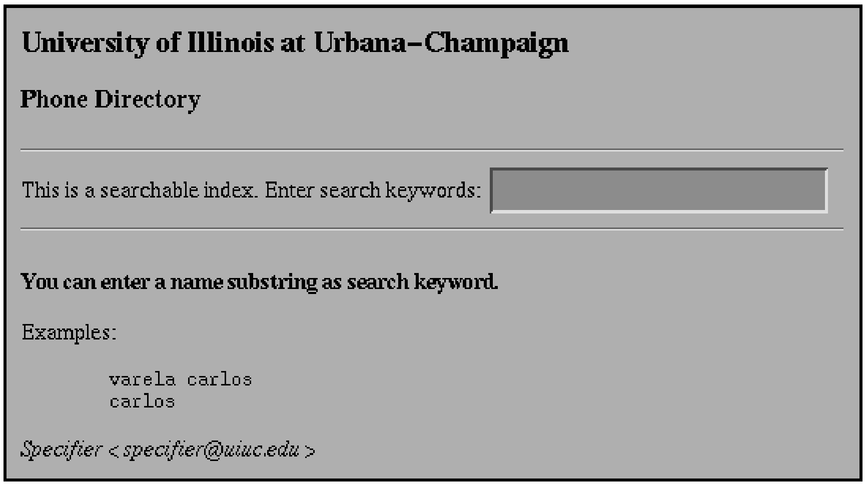

Fig 3a. A WAIS–based user interface using the HTML `ISINDEX` tag.

- **phfb:**

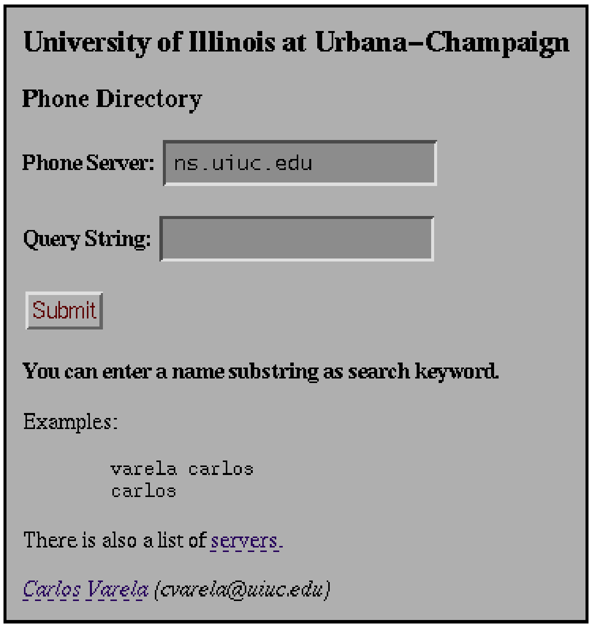

Fig 3b. A simple user interface for this database application. It uses a form and is

not sophisticated in the output handling, yet it is a useful and simple gateway to phone directories.

- **phfc:**

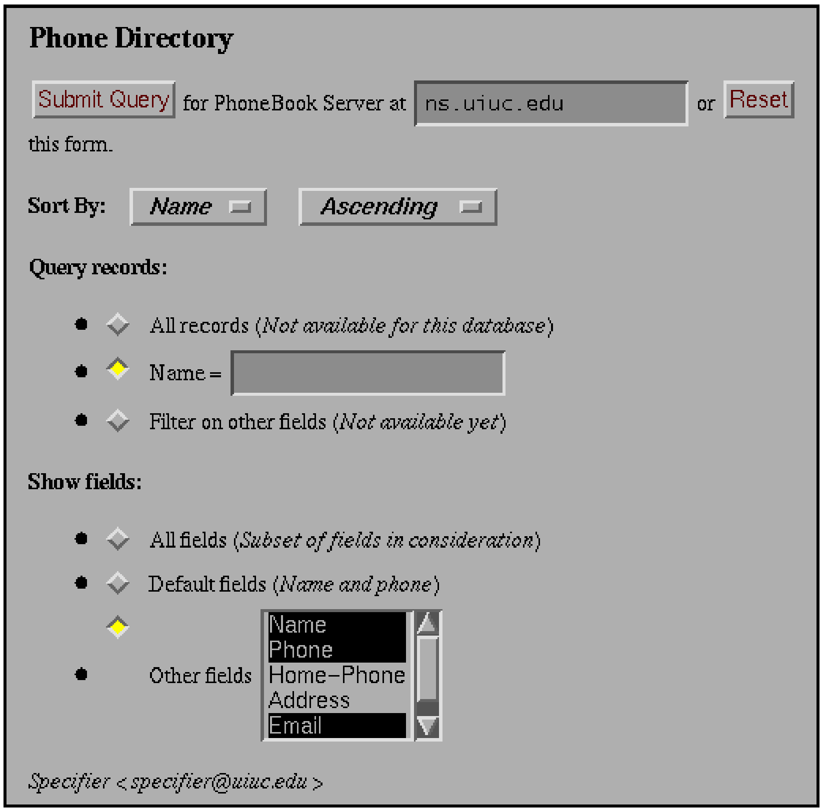

Fig 3c. A forms–based user interface using forms, and parsing the database output to allow different sorting, filtering and showing options. The usage given to these options is fundamental towards the generation of advice regarding both the user interface and the underlying data structures. This schema has been written to be used by different database applications, in other words it is domain independent.

- **phfd:**

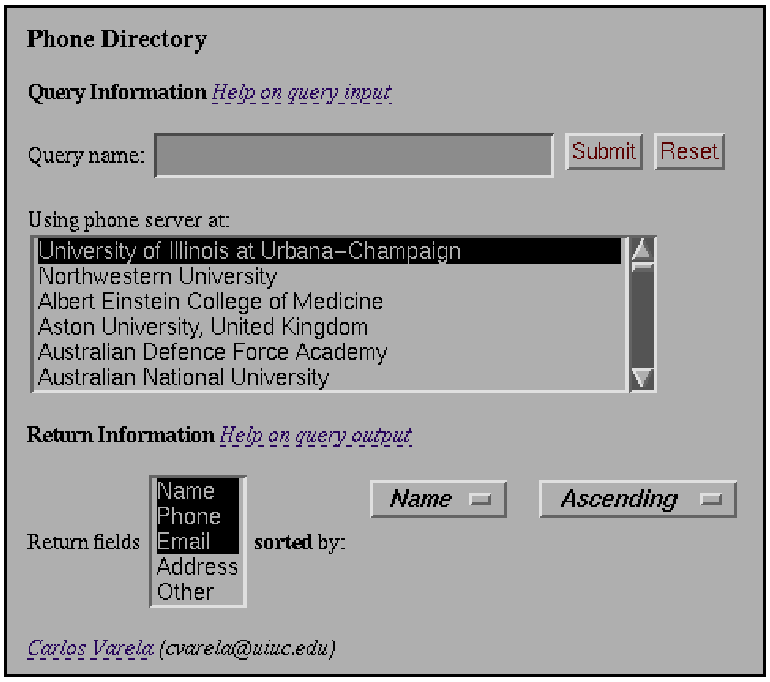

Fig 3d. And finally an advanced user interface for this database application. This interface is very domain specific, in the sense that it has been adapted to work *mainly* for the phone database. The idea is to let the programmer move from the previous application to this one by making simple changes to the generated code.

There are some additional CSO nameserver database interfaces found in the WWW, for reference and comparison:

- CCSO Phonebook by Ed Kubaitis at CCSO, UIUC.
- Form for CSO PH query by Jim Browne at NCSA, UIUC.
- ETHZ Staff Phone Book by Andi Karrer at ETH Zuerich, Switzerland.

5. Conclusions

The presented methodology offers an evolving solution to database interface design using an expert system. This schema–based approach to HTML run–time document generation, allows us to make modifications to WWW database applications easily by changing the schemata instead of directly changing the scripts which generate the hypertext documents. This approach also gives the interface designers some freedom in the schemata selection method, which proves useful for generating *soft* user interfaces.

As a result of having *soft* database applications, we have improved previous methodologies in two ways. On the one hand, there is increased efficiency in the underlying data structures resulting in faster access to the data. On the other hand, there is a more effective communication between the database and the users, who may have different views of the same information. This is of great significance because of the greatly varying needs of the different users in the WWW.

While the initial cost of programming parsers for the ZHTML schemata may be high, the long term benefits of the automatic generation of interface design far outweight the initial effort. Ultimately, this procedure offers the user a more efficient, fully customized interface, further closing the gap between information provision and use.

Acknowledgements

Thanks to the NCSA Software Development Group for their great work with Mosaic and helpful comments on this research. Additional thanks to the NCSA Alpha Shapes Development Team for their support to the first author and their excellent research and working environment.

References

[Berners−Lee 1992]
T. Berners−Lee. *Hypertext Transfer Protocol Requirements.* Internet Working Draft. CERN. Work in progress.
http://info.cern.ch/hypertext/WWW/Protocols/HTTP.html

[Berners−Lee et al. 1992]
T. Berners−Lee, R. Cailliau, J. Groff, B. Pollermann. World−Wide Web: The Information Universe. *Electronic Networking: Research, Applications and Policy*, 2(1), pp. 52−58, Meckler Publications, Westport CT, Spring 1992.
ftp://info.cern.ch/pub/www/doc/ENRAP_9202.ps

[Berners−Lee & Cailliau 1992]
T. Berners−Lee, R. Cailliau. World−Wide Web. Submitted to *Computing in High Energy Physics 1992.*
ftp://info.cern.ch/pub/www/doc/chep92www.ps

[Berners−Lee & Connolly 1993]
T. Berners−Lee, D. Connolly. *Hypertext Markup Language: A Representation of Textual Information and Metainformation for Retreival and Interchange.* Internet Working Draft. CERN, Atrium Technology Inc. Work in progress.
http://info.cern.ch/hypertext/WWW/MarkUp/HTML.html

[Bhansali 1991]
S. Bhansali. *Domain−Based Program Synthesis Using Planning and Derivational Analogy.* Ph.D. thesis. University of Illinois at Urbana−Champaign. May 1991.

[Dershowitz 1983]
N. Dershowitz. *The Evolution of Programs.* Birkhause, Boston, 1983.

[Dorner 1992]
S. Dorner. *The CCSO Nameserver, Server−Client Protocol.* Computing and Communications Services Office. University of Illinois at Urbana−Champaign. July 1992.

[Forgy 1981]
Forgy, Charles L. *OPS5 User's Manual.* Department of Computer Science, Carnegie Mellon University.

[Kahle & Medlar 1991]
B. Kahle and A. Medlar. An Information System for Corporate Users: Wide Area Information Servers. *ConneXions − The Interoperability Report,* 5(11), pp 2−9, Interop, Inc., Nov. 1991.

[McCool 1993]
Rob McCool. National Center for Supercomputing Applications, University of Illinois at Urbana−Champaign. *Common Gateway Interface Overview.* Work in progress.
http://hoohoo.ncsa.uiuc.edu/cgi/overview.html

[NCSA 1993]
National Center for Supercomputing Applications, University of Illinois at Urbana−Champaign. *NCSA Mosaic. A WWW Browser.* Work in progress.
http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/help−about.html

[Ng 1993]
J. Ng. *GSQL: A Mosaic−SQL gateway.* National Center for Supercomputing Applications. University of Illinois at Urbana−Champaign. Work in progress.
http://www.ncsa.uiuc.edu/SDG/People/jason/pub/gsql/back/starthere.html

[Varela & Hayes 1993]
C. Varela, C. Hayes. *Automating Design of Database Management Systems.* Department of Computer Science. University of Illinois at Urbana−Champaign. Working paper.