8-15-1997

Integrating Information Systems into the World Wide Web

Chao-Min Chiu
Rutgers University, chchiu@pegasus.rutgers.edu

Michael Bieber
New Jersey Institute of Technology, bieber@cis.njit.edu

Integrating Information Systems into the World Wide Web

Chao-Min Chiu
Department of MS/CIS
Rutgers University
Newark, NJ 07102 USA
chchiu@pegasus.rutgers.edu
http://pegasus.rutgers.edu/~chchiu
Michael Bieber
Institute of Integrated Systems Research
New Jersey Institute of Technology
Newark, NJ 07102 USA
bieber@cis.njit.edu
http://megahertz.njit.edu/~bieber

1. Introduction

Hypertext is an approach that allows authors to manage information relationships by creating links within or across small chunks of information (nodes or documents). Links represent an author's perception of inter-relationships among information. Hypertext allows readers to easily and rapidly browse the nonlinear information network in the order most appropriate to the reader's task by activating links.

What benefit do users gain from providing information systems with hypertext support? Users can find it difficult to understand and take advantage of the myriad of inter-relationships in a information system's knowledge base (data, processes, calculated results, reports). Hypertext helps by streamlining access to and providing rich navigational features around related information, thereby increasing user comprehension of information and its context [BK95]. Augmenting an information system with hypertext support results in new ways to view and manage the information system's knowledge, by navigating among items of interest and annotating with comments and relationships (links) [BV97].

The overall goal of this paper is to provide hypertext functionality through the WWW to hypertext-unaware dynamically-mapped information systems (DMIS) with minimal changes to DMISs. DMISs dynamically generate their contents and thus require some *mapping mechanism* to automatically map the generated content to hypertext constructs (nodes, links, and link markers) instead of hypertext links being hard coded over static content. DMISs include spreadsheets, statistical analysis systems, database management systems (DBMS), geographic information systems, expert systems and decision support systems. This paper uses DBMS as a sample domain. Web database development is a hot new field. Most database applications handle queries and generate HTML from query results without inferring links. Those database applications do not meet our definition of integration with DMISs since they do not allow users to access other aspects of database objects. (e.g., metainformation of database tables).

The WWW has radically increased the awareness of the concept of hypertext. The WWW is mainly used to browse predefined HTML files. However, the WWW has the opportunity to integrate hypertext support into information systems. There are many potential approaches for integrating Web servers with information systems: traditional CGI, server API (e.g., Netscape's NSAPI and Microsoft's ISAPI), applets, etc. No systematic approach exists, however, for moving an analytical information system to the WWW and giving users direct access to its interrelationships. This research contributes a systematic dynamic-mapping mechanism and a set of guidelines for DMIS integration on the WWW.

Web search engines do perform mapping using simple mapping rules to map the query result to a dynamically-generated Web page with links underlying the "URL address" and "page title" for each hit. The search engine has a simple dynamic mapping mechanism. In this paper, we propose general guidelines for building more generic mapping rules. First we show an architecture for implementing them on the WWW.

2. System Architecture

In this section, we propose a conceptual architecture with seven logical components. This architecture emphasizes the integration of Web servers with DMISs, providing hypertext functionality to each. Note that actual architectures may physically implement the functionality in different modules than those shown here. Since the Web browser and server are normal Web components, we just describe the functionality of other components.

1. The Master Handler coordinates schema mapping and message passing among different DMIS domains, thus aiding DMIS-to-DMIS integration.

2. A DMIS handler translates and routes messages between its DMIS and the Web server. It also provides its DMIS's mapping rules. A comprehensive DMIS handler will allow us to integrate an existing DMIS with few or no changes.

3. A DMIS is an application system with which users interact to perform some task, which dynamically produces output content for display.

1. A knowledge base stores commands for accessing various relationships on DMIS objects and information that can not be accessed directly from DMISs (e.g., relationships in E-R diagrams).
    1. A linkbase stores user-created annotations (e.g., comments). We have created a prototype of an editable browser extending Sun's 1.0 alpha3 release of the HotJava browser. This browser allows users to create and store comments in local or remote linkbases.

Note that the master handler and DMIS handlers can be implemented using CGI scripts, Java applets, server APIs, etc.

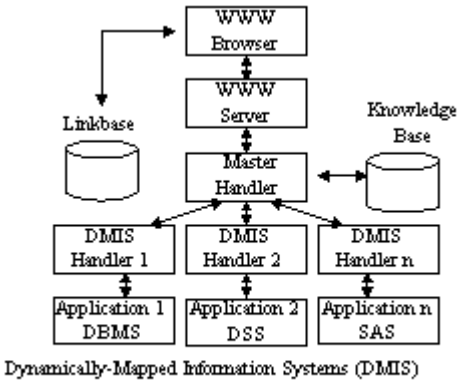

Figure 1. A Conceptual Architecture for Integrating the WWW with DMISs

3. Building Mapping Rules

In this section, we are going to discuss the process of analyzing dynamically-mapped information systems (DMISs) and building mapping rules. We divide the process into four steps. These rules would reside in and become invoked by the DMIS handler. We explain each by using a relational DBMS as the target DMIS.

Step 1. Identify Objects

This step is to identify data objects in which we are interested. In the database application, objects include (1) database: a collection of tables, (2) table: a collection of records (or fields), (3) record: each record

represents a collection of related data values, (4) field: a property that describe some aspect of an entity (5) value: each value represent the fact describing an entity or a relationship instance [W96].

Step 2. Identify Relationships

Bieber and Vitali [BV97] identify several types of relationships for system objects, including those below. Identifying explicit and implicit relationships forces developers to consider which information users are interested in and then build mapping rules to access this. (When identifying a relationship, the developer also should identify the commands used in step 3 to implement the link representing it.)

Each of the following relationships gives the user easy access to some aspect of an object.

- Schema Relationships

Access to the kind of domain-specific relationships one finds in a schema or application design. In a database application, this includes relationships captured in the entity-relationship diagram.

- Operational Relationships

Direct access to information about DMIS objects using operational commands supported by the DMIS. In the database application, this includes SQL commands over specific objects.

- Structural Relationships

Access to related objects based on the application's internal structure. In a DBMS these include "contains" links among databases, tables, records, fields and values.

- Metainformation Relationships

Access to attributes of and descriptive information about DMIS objects. In a DBMS, these include its database size (numbers of tables), table size (number of fields or records), field type, etc.

- Occurrence Relationships

Access to all manifestations of a given object. In a DBMS, for example, we might want to access other tables that have the same key.

- Annotative Relationships

Relationships declared by users instead of being inferred based on system structure. All users should be able to annotate objects even when without direct write access. A linkbase will store user-created comments.

Sometimes meaningful relationships can not be accessed directly from DMISs so developers have to declare those relationships and store them in the knowledge base.

Step 3. Identify Commands for Accessing Relationships on DMIS Objects

Commands underlie the links that give users direct access to various relationships on DMIS objects. After identifying its implementation commands, we can then build a relationship's mapping rules. Table 1 lists some of these commands for database table relationships. The actual system commands will vary among DBMS implementations. The DMIS handler should pass the actual system commands to the DMIS.

Object	Relationships	Commands
table	schema	list related tables
table	operational	insert record
table	structural	list fields
table	metainformation	describe table
table	occurrence	list tables having the same key
table	annotative	create comment

Table 1. Commands underlying database table relationships.

Step 4. Build Mapping Rules

The main purpose of mapping rules is to infer useful links from the output dynamically generated by a DMIS (e.g., DBMS) and commands for operating on DMIS objects. Note that one set of mapping rules can serve all instances of a DMIS. We identify two types of mapping rules: Command_Rule and Object_Rule. Bieber and Wan [W96] implemented mapping rules (or bridge laws) using Prolog. However, for clarity we discuss mapping rules in terms of functional procedural calls. Command rules are invoked when the user selects a link anchor. Object rules are invoked when the user selects a command link and DMISs display documents.

1. Command_Rule(Owning_Sys, Object_Type): This rule infers commands for operating upon the selected object. This rule has two parameters: Owning_System and Object_Type. As figure 1 shows, a Web server can integrate with multiple DMISs so we need the "Owning_System" parameter to discriminate among different DMISs.

This rule should provide the following functions:

- Search the knowledge base for commands accessing various relationships on the selected DMIS object.
- Map commands to links
- Form a HTML document that includes mapped links and sends the document to the Web server.

For example, Command_Rule("DBMS", "Table") will execute the aforementioned functions and create the following HTML document. To conserve space, we just list the first command (list fields) here. In reality the system would present all commands (or a filtered subset). Some available commands are listed in table 1.

```
<HTML><BODY>Select one of the following commands:<A href="http://www.rutgers.edu/cgi-
bin/masterhandler?Owning_System=DBMS&Object_ID=Goods%2Cvendor&Object_Type=Table&Command=list
fields">list fields</A>. . . .</BODY></HTML>
```

In this HTML document, we make the following assumptions: (1) The "Goods" database contains the "Vendor" table. (2) The master handler is a CGI application called "masterhandler."

2. Object_Rule(Command, Owning_System,

Object_ID, Object_Type): This rule should be included in the DMIS handler. It has four parameters. The object identifier (Object_ID) is the key to determine which object of the given system the command should operate on. The following table lists possible identifiers for database objects. The Object_ID, ,Goods,Vendor', means the "Vendor" table of the "Goods" database.

Type	Object_ID	Object_ID Example
Database	Database	"Goods"
Table	Database,Table	"Goods,Vendor"

Field	Database,Table,Field	"Goods,Vendor,Name"

This rule should provide the following functions:

- Map commands to actual DMIS commands.
- Send actual commands and other parameters to the DMIS.
- Receive display output from the DMIS.
- Infer links from the output generated by the DMIS.
- Create the HTML document with inferred links and send the document to the Web server .

Here is an example in which the command accesses a structural relationship (from step 2). Object_Rule("list fields", "DBMS", "Goods,Vendor", "Table") will execute the five aforementioned functions and send the following HTML document to the Web server. To conserve space, we just list the field called "Name".

```
<HTML><BODY>The table, Vendor, has the following fields:<A href="http://www.rutgers.edu/cgi-bin/masterhandler?Owning_System=DBMS&Object_ID=Goods%2CVendor%2Cname&Object_Type=Field">Name</A>. . . .</BODY></HTML>
```

4. Conclusion

The WWW presents a way to integrate hypermedia into information systems and information spaces. We believe that integrating the WWW with information systems and DMISs in the business world should constitute a major thrust for the WWW research. This will go a long way toward making applications more understandable. When reengineering applications for the WWW, dynamic relationship mapping could prove an effective way to add additional hypermedia links. This should help counteract the danger that new Web applications (especially DMISs) will have no hypertext in them [BV97]. We hope this paper will start people thinking about mapping rules for better integrating the WWW and analytical information systems.

Acknowledgments

This research is supported by the New Jersey Center for Multimedia Research, the New Jersey Commission of Science and Technology, the NASA JOVE faculty fellowship program, the AT&T Foundation, and the New Jersey Institute of Technology under Grant 991967.

References

[BK95] Bieber, M. and Kacmar, C. Designing Hypertext Support for Computational Applications, Communication of the ACM, 38(8), 1995, 99-107.

[BV97] Bieber, M. and Vitali, F. Toward Support for Hypermedia on the World Wide Web, IEEE Computer, 30(1), 1997, 62-70.

[W96] Wan, J. Integrating Hypertext into Information Systems through Dynamic linking. Ph.D. Dissertation, New Jersey Institute of Technology, Newark NJ 07102, 1996.