Integrating Hypermedia Functionality into Database Applications

Anirban Bhaumik*+ anirbanbhaumik@usa.net
Deepti Dixit* deeptidixit@hotmail.com
Roberto Galnares* galnares@homer.njit.edu
Aparna Krishna* aparna7@yahoo.com
Manolis Tzagarakis** tzagara@cti.gr

Michalis Vaitis** vaitis@cti.gr
Michael Bieber* bieber@njit.edu
Vincent Oria* oria@homer.njit.edu
Qiang Lu*/*** qiang@suda.edu.cn
Firas Alljalad* firas@homer.njit.edu

Li Zhang* lxz9848@oak.njit.edu

* *Collaborative Hypermedia Laboratory, CIS Department, New Jersey Institute of Technology, USA*
** *Computer Technology Institute, University of Patras, Greece*
*** *Suzhou University, Peoples Republic of China* +Primary Author

Abstract

The general goal of our research is to automatically generate links and other hypermedia related services to analytical applications, such as geographic information systems and decision support systems. Using a dynamic hypermedia engine (DHE), we propose to automate the following features for database systems, both on and off the Web. First, we automatically generate links based on the database's relational (conceptual) schema and its original (non-normalized) entity-relationship specification. Second, the application developer can specify which kinds of database elements are related to diverse elements in the same or different database application, or even another software system. Our current DHE prototype illustrates these for a relational database management system. We propose integrating data warehousing applications into the DHE. We also propose incorporating data mining as a new kind of automated link generation. Passing the application element selected by a user, a data mining system would discover interesting relationships for that element. DHE would then map each relationship to a link. DHE's linking is based on the structure of the application, not keyword search or lexical analysis based on the display values within its screens and documents. DHE aims to provide hypermedia functionality without altering applications by building "application wrappers" as an intermediary between the applications and the engine.

Keywords

Hypertext, hypermedia, database, automated linking, metadata, Dynamic Hypermedia Engine, data mining, data warehousing, E-R Diagram, database schema, wrapper

1. Introduction and Motivation

Database queries typically return results in a plain text format. Some applications on the World Wide Web generate link anchors for database elements, but these anchors normally hold a single link to the most obvious destination for the dominant type of user.

We could consider each element within a database application as a potential starting point for information exploration. Each element could have multiple links, each representing a different relationship (schema-based or otherwise). The ability to explore a piece of information in more detail could help users resolve doubts about or simply better understand that item, as well as the analysis or display of which it is a part. Users may wish to dig deeper around data values and symbols they see, labels on graphs or user input forms, options in pop-up lists, or even on the menu commands they can invoke.

To complicate the developer's job, users often have different mental models of an application and its underlying domain than the developer. Even when developers work closely with users, the end result might not be intuitive for all users or serve each user's individual tasks equally well. Many people visit a given application's screen aside from the most dominant type of user(s) for which it was developed. These include other users of the application, customer service representatives, company analysts, managers, trainees, people inside the company designing new databases or applications based on the current one, external analysts, and stockholders, among others. Each may be interested in different aspects of application elements, according to their current task-at-hand. Customization is one solution, but even so users often might wish to explore several different relationships from a given anchor, and therefore should have several links available.

The purpose of this chapter is to explore all aspects of hypermedia support for database applications. We base much of our discussion on our experience designing the Dynamic Hypermedia Engine (DHE). DHE automatically generates anchors, sets of links and metadata within database applications, as well as supporting users with other types of hypermedia structuring, navigation and annotation functionality, including guided tours and annotation. As we explain later, the links provide direct access to a broad range of relationships among application elements, including those outside a database schema. Metadata, defined as data about data, shows users parameters and other characteristics about any given element.

This work makes many contributions to both the database and hypermedia fields. Many database applications do not take as much advantage of hypermedia as they could. This chapter puts forth a series of opportunities for integrating hypermedia and database systems. As we shall describe in the next section, DHE is the only tool that provides automated linking and hypermedia services based on the application structure (as opposed to search or lexical analysis), without altering applications. Thus it is uniquely suited to support databases and other analytical applications on the Web that generate the contents of their displays dynamically in response to user queries.

This chapter proceeds as follows. After introducing the Dynamic Hypermedia Engine, we describe hypermedia support for databases. First, we show how DHE can support existing relational database management systems. Then we describe how DHE uses databases internally to support hypermedia. Next we look at integrating data warehousing and applications that use database support. Then we consider generating links through data mining. After this we present a literature review of hypermedia and database integration, followed by a short conclusion. Table 1 gives the set of acronyms we use in this chapter.

Acronym	Full Name	First Discussed
DHE	Dynamic Hypermedia Engine	§2
DSM	Database Schema Mapper	§3.3
JDBC	Java Database Connectivity	§3.1
OLAP	On-line Analytical Processing	§5
RDBMS	Relational Database Management System	§3.1
RDF	Resource Description Framework	§3.2
RDWM	Relational Database Wrapper Module	§3.1
RMI	Remote Method Invocation	§2
SQL	Structured Query Language	§1
UI	User Interface	§2
URI	Uniform Resource Identifier	§3.1
XML	Extended Markup Language	§2

Table 1: Acronyms in this Chapter

2. The Dynamic Hypermedia Engine

We have just developed the very first cut of a Web-based prototype of the Dynamic Hypermedia Engine (DHE), which redesigns an older PC-based prototype (Bieber 1999). Figure 1 shows a screenshot of a database query result in the top frame. DHE has added anchors to all parts of the query result, including the field names at the column heads. The user has clicked on "Counseling Center - department," resulting in metadata for the element in the bottom center frame and a list of links in the bottom right-hand frame. Selecting any link will generate a standard Structured Query Language statement ("SQL query") to create the appropriate result. Currently the list of links includes only database structural links, such as finding the primary keys for this element. As we later describe, however, we are developing a module to add links automatically based on a database's original (non-normalized) Entity-Relationship schema. The bottom left-hand frame contains menus for any integrated application or DHE internal module. Links represent relationships and relationships have "meta-information" as well. Selecting an asterisk next to any link will provide metadata and a list of links for it. DHE's next release will provide these for menu items as well. The metadata frame currently displays the full Resource Description Framework record—see §3.2. DHE's next release will format the metadata nicely. Future versions will filter and rank order the links and metadata based on the user task and preferences.

<Place Figure 1 about here>

As this demonstrates, DHE link generation does not result from any type of lexical analysis. Our focus is not on the display content of the link anchor, but rather on the application elements underlying each anchor. A "mapping rule" encodes each relationship found between two elements of interest at the "class level". For example, suppose an application display shows the name of a university department. Departments generally have professors and courses taught (based on the standard entity-relationship diagram within a database system), as well as a Web page, an annual budget (within the accounting system), hires-in-progress (within the personnel system), a location on a map (within a geographic information system), etc. Individual mapping rules contain an algorithm or computation (set of commands) leading to the appropriate component in these respective systems. When the user selects a particular department, DHE constructs these commands with the actual department instance selected and sends them to the appropriate destination system, which then retrieves—or more often generates—the resulting page. For example, one mapping rule could state that an element of type "department" would be related to an element of type "annual budget" through a relationship with the semantic type "annual budget for" and with a parameterized command to retrieve annual budgets from the accounting system. Developers may take advantage of this to integrate database applications with other applications without altering their contents; they only have to add new mapping rules for the relevant element types.

DHE executes concurrently with database management systems, database applications, and other applications such as the accounting system, providing automated link generation and other hypermedia functionality without altering them. Developers write an independent application "wrapper" and a set of mapping rules. Note that once a wrapper is written and the mapping rules are specified for each type of application (geographic information system, relational database management system, accounting package, etc.), DHE will support all instances of that application in the future (new maps, database contents, budget sheets, etc.).

DHE executes as follows. Applications or their wrappers connect to DHE through World Wide Web components, such as servlets and JavaServer pages. DHE intercepts all messages passing between the application and its user interface, and uses the mapping rules to map each appropriate element of the message to a hypermedia anchor. Our Web browser wrapper merges these anchors into the document being displayed and passes the resulting HTML document through the Web component servlet to the

user's Web browser. When the user selects an anchor, the browser wrapper passes it to DHE, which returns a list of possible links (one for each appropriate relationship as determined by the mapping rules) and metadata. If the user selects a DHE link (e.g., to add an annotation or stage in a guided tour), DHE processes it entirely. If the user selects a relationship with a destination in a known application, DHE infers and instantiates the appropriate SQL queries or other application commands from the relationship's mapping rule and passes them to the target application for processing. If the user selects a user-created annotation or tour, etc., DHE retrieves it. Thus DHE automatically provides all hypermedia linking (as well as navigation) to applications, which remain hypermedia-unaware and in fact often entirely unchanged.

Figure 2 shows DHE's logical engine architecture. We shall describe some of the major components here. The others are described on our project Web site (http://dhe.njit.edu). For our current Web prototype, we have programmed all modules in Java. We use XML as our message format. While the browser wrapper currently produces HTML documents for display, we intend to migrate to XML documents, which take advantage of the Web's new XLink, and XPointer standards to handle anchors and links. We use RMI for inter-module communications. We intend to keep up with Web standards as they become available, whenever practical for our environment.

<Place Figure 2 about here>

User Interface Wrappers serve three important functions: First, they translate DHE's internal messages from DHE's standard format to a format the browser (or other User Interface or UI) can process, and vice versa. Second, they handle communication between the engine and the UI. Third, they implement any functionality DHE requires from the UI (e.g., maintaining parameters), which the UI cannot provide itself.

The *Message Manager Module* enables the communication between all DHE modules, routing all DHE internal messages.

The *Mapping Rules Module* maps the application data and relationships to hypermedia objects at run-time. The Mapping Rules Module maps the element instances in the virtual document to global element types (classes), and infers all relevant relationships (links) and metadata for the given element classes. These links and metadata are passed in messages to the UI Wrapper for display.

Application Wrappers, like user interface wrappers, manage the communication between DHE and their application systems, such as database applications and DBMS. They translate user requests from DHE's internal format to the application's programming interface (if any). They receive output from the application, convert it to the DHE format, mark the elements for the mapping rules module, and send it to DHE for eventual display on the UI.

Other Hypermedia Services: We are planning to implement a series of other service modules over the next few years. Most will implement various kinds of hypermedia structuring, navigation and annotation functionality (Bieber et al. 1997; Conklin 1987; Nielsen 1995). Hypermedia structuring functionality includes local and global information overviews; node, link and anchor typing; as well as keywords, attributes and metadata on all of these. Navigation functionality includes structure-based query, sophisticated history-based navigation and bi-directional linking. Annotation functionality includes adding user-declared links, comments and bookmarks to dynamically-generated documents and displays.

What distinguishes these DHE modules from similar modules in other Web applications and other non-Web hypermedia systems is the dynamic nature of the applications we are considering. Our target applications generate documents and screens in response to user queries and other dynamic prompts.

Therefore the screens and documents are not static, but "virtual"; they must be generated every time they are needed, and regenerated upon demand within each of these modules. Thus documents on a guided tour, pointed to by a bookmark or link, or found during structural search may not exist until the user actually selects the anchor representing it (Bieber & Kacmar 1995). We do this by maintaining sufficient parameters about each document or screen to create the SQL queries or other commands to regenerate them. These commands are not always the same as the original command that generated the document in the first place. (Bieber 1990) and (Bieber 1995) discuss this regeneration in more detail.

DHE should be able to streamline a company's software development efforts in several ways. It automatically supplements the organization's applications with hypermedia links, structuring, navigation and annotative functionality, and metadata. It also implements inter-application linking, as the university department example shows. Mapping rules can point to any accessible application. DHE also can speed the development of applications. Developers can offload link management, navigational structures (such as guided tours), user preference management and other features to DHE.

Several approaches exist for integrating hypermedia functionality into primarily non-hypermedia information systems. These include employing hypermedia data models (Campbell and Goodman 1988; Halasz and Schwartz 1994), hypermedia toolkits (Anderson 1996), link services (Pearl 1989; Davis et al. 1992; Anderson 1997), hyperbases (Leggett and Schnase 1994), hypermedia development environments (Nanard and Nanard 1995a; Marshall and Shipman 1995; Akscyn et al. 1988), open hypermedia systems (Whitehead 1997; Wiil 1997; Grønbæk & Trigg 1999), and independently executing hypermedia engines, such as DHE.

Hypermedia engines execute independently of an application with minimal modifications to it, and provide the application’s users with hypermedia support. Few approaches provide transparent hypermedia integration as our engine does. Notable projects include Microcosm’s Universal Viewer, Freckles and the OO-Navigator and SFX.

Microcosm's Universal Viewer (Davis et al. 1994) and Freckles (Kacmar 1993, 1995) seamlessly support an application’s other functionality but provide only manual linking. OO-Navigator comes the closest to our approach, providing a seamless hypermedia support for computational systems that execute within a single Smalltalk environment (Garrido and Rossi 1996; Rossi et al. 1996). This approach meets our goal of supplementing Smalltalk applications with hypermedia support without altering them. Our approach, however, applies to both object-oriented and non object-oriented applications.

SFX's engine is very similar to DHE, but it only serves one specific environment. SFX dynamically generates anchors within the reference section of academic papers being displayed on the Web. Selecting these will lead to the original work within bibliographic databases (Van der Stemple, 1999a,b,c). DHE, in contrast, provides a generalized approach for linking and additional hypermedia functionality for most analytical applications.

In the sections that follow we describe various ways that we can support database management systems and applications, and the ways that databases support our implementation of hypermedia.

3. Hypermedia Support for Existing Database Systems

Developers can retrofit existing database applications to work with DHE. This section begins by describing such an integration. Then we describe the different kinds of links and metadata that DHE provides, and conclude with a brief description of a system we have integrated with the DHE.

3.1 Integrating Database Applications with DHE

One of the first tasks in providing hypermedia support is to intercept messages between the computational and user interface (UI) portions of the application (Bieber & Kacmar 1995). In the case of a Relational Database Management System (RDBMS), the application comprises only a computational portion. The RDBMS provides a standard way of requesting computational services (i.e., store, retrieve and analyze data) by means of Structured Query Language (SQL) statements. The Relational Database Wrapper Module we describe below does this and provides its own interface for entering SQL queries and displaying their results.

Database applications build a customized interface and possibly a larger set of functionality around a RDBMS. Database applications, therefore, are responsible for their own UI. Database applications send SQL statements to their RDBMS. The RDBMS then executes these statements and returns the results of these statements to the application, which then customizes and displays them.

If the UI displays are easy to parse, i.e., a developer could easily figure out which elements are in each display screen and how to pass back database and application commands to the application, then one could write an application wrapper that intercepts all displays and redirects them to DHE. This would satisfy our goal of providing automated hypermedia functionality with minimal change to the application. DHE's UI could display the enhanced screens with hypermedia anchors as shown in Figure 1.

In general, Web-based database applications (Internet storefronts, catalogs, etc.) have a clear distinction between their UI and the database; usually the UI uses a middleware to communicate with the database. The DHE may supplement or replace this middleware. However most legacy database applications (client server or mainframe based) usually don't have such a clear distinction between their UI and the underlying database; most often the data presentation is commingled. Integration with DHE could be much more difficult in this case.

We have developed a service module, the Relational Database Wrapper Module (RDWM), which accepts requests to execute SQL statements on the underlying database, and retrieves metadata for a database element. It also provides a UI allowing users to execute SQL statements and view results, metadata and all the relationships among the data affected by the SQL statement. This UI provides a view of the data stored in the database enhanced by metadata, hyperlinks and additional hypermedia functionality.

The RDWM passes through states shown in Figure 3. We next briefly describe each of these.

<Place Figure 3 about here>

Sending Startup Messages

Because of DHE's distributed nature, all modules must register themselves with the DHE Message Manager at startup time, so that the Message Manager knows which modules are currently active and able to receive and process requests.

Receiving Request Messages

A module waits until it receives a request to perform a service. The RDWM either interacts with the RDBMS, retrieves metadata for an element or generates an input form for users to enter SQL statements.

Extracting and Executing SQL statements

The RDWM uses an XML parser to extract the action to be performed and the SQL statement (if any) from the request message. If the request is to execute an SQL statement then the UI Wrapper will have embedded the actual SQL in a request message. (This is its default procedure for user input forms.) The RDWM then uses a persistent pool of JDBC (Java Database Connectivity) Connection Objects, to execute an SQL statement on, or retrieve metadata from the RDBMS.

Identifying an Element of Interest

Once the result of the SQL statement has been passed to the RDWM for display, the RDWM parses the results. It uses the SQL query itself to determine what elements are in the results. The following twelve types of elements exist in the RDBMS context:

Columns	Tables	Indices
Stored Procedures	Catalogs	Schema
Drivers	Users	User Rights
Table and Column Privileges	JDBC Types	The RDBMS Product Instance itself

Any instance of the above types can be uniquely identified, have metadata, and have one or more relationships associated with it. If any user might be interested in exploring that type of object in terms of its metadata or relationships, the RDWM would mark each of its instances as an "element of interest", and therefore a potential anchor. Marking an item involves providing its unique identifier and its element type. (Later the Mapping Rules Module will use the element type to find relationships and metadata for a given element. If an element has at least one relationship (link) or piece of metadata, then the Mapping Rules Module will specify that the UI Wrapper make it into an anchor.)

Using the Uniform Resource Identifier (URI) syntax (Berners- Lee et. al. 1998):

```
<scheme name>:<scheme-specific-part>
```

the RDWM defines its element identification scheme as follows:

```
dhe:rdwm:<element type>:<RDBMS Name>:<Database Name>:<element specific part>
```

For example, the column named "student_id" in the table "student" in the "Oracle" database on the host "logic.njit.edu" would have the following URI:

```
dhe:rdwm:column:oracle:logic.njit.edu:student:student_id
```

Because of the case insensitive nature of SQL statements this scheme too, is case insensitive.

Figure 4 shows an example marked up query result.

<Place Figure 4 about here>

3.2 Structural Links and Metadata for Database Applications

Continuing on with the RDBW states described in Figure 3, we look at the basic structural links and metadata DHE provides database applications.

Relationships between Elements

DHE specifies relationships based on the element type. The twelve types of relationships noted above are each interrelated. Figure 5 shows these "structural" interrelationships. DHE could write a mapping rule for each, which DHE could then use to generate a link for each of its instances containing the appropriate SQL command to generate the contents of the link's endpoint. Figure 1 shows some of these links. Wan (1996; Wan & Bieber 1997) gives several mapping rules (called "bridge laws") for relational databases.

<Place Figure 5 about here>

Each relationship depicted in Figure 5 is reflexive, e.g., if a column has an "in table" relationship with a table, then the table has a relationship called "has columns" with the column.

Retrieving Metadata

The RDWM retrieves metadata on demand. It is passed the URI of the element whose metadata is being asked for. It then uses the Java Database Connectivity (JDBC) API to retrieve information relevant to the element. The current version of the DHE uses the relational schema to retrieve metadata. Future versions of the DHE will use dedicated metadata repositories, and data dictionaries to retrieve additional metadata as well.

The RDWM uses the Resource Description Framework (RDF) to model database metadata. RDF defines metadata in terms of "resources", where each resource is any entity that can be uniquely identified, i.e., has a URI (Lassila et. al 1999). Hence all "elements of interest" are resources and have associated metadata. Resources and elements are thus interchangeable in the RDWM context and the element identifier corresponds to its URI.

For example the following segment in RDF/XML serialization syntax represents metadata about a column called "Description" in the table "Department" on the Oracle instance on "logic":

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description
about="dhe:rdwm:column:jdbc:oracle:thin:@logic:1521:logic40:DEPARTMENT:DESCRI
PTION">
<ORDINAL_POSITION>2</ORDINAL_POSITION>
<CHAR_OCTET_LENGTH>500</CHAR_OCTET_LENGTH>
<SQL_DATETIME_SUB>0</SQL_DATETIME_SUB>
<TABLE_SCHEM>MOHAN</TABLE_SCHEM>
<DECIMAL_DIGITS></DECIMAL_DIGITS>
<COLUMN_NAME>DESCRIPTION</COLUMN_NAME>
<NUM_PREC_RADIX>10</NUM_PREC_RADIX>
<NULLABLE>1</NULLABLE>
<COLUMN_SIZE>500</COLUMN_SIZE>
<TABLE_CAT></TABLE_CAT>
<COLUMN_DEF></COLUMN_DEF>
<REMARKS></REMARKS>
<IS_NULLABLE>YES</IS_NULLABLE>
<DATA_TYPE>12</DATA_TYPE>
<BUFFER_LENGTH>0</BUFFER_LENGTH>
<SQL_DATA_TYPE>0</SQL_DATA_TYPE>
<TABLE_NAME>DEPARTMENT</TABLE_NAME>
<TYPE_NAME>VARCHAR2</TYPE_NAME>
</rdf:Description>
```

```
</rdf:RDF>
```

3.3 Enhanced Links Through a Database Schema Mapper

Most database applications provide no contextual information about the underlying schema of the database from which query results were retrieved. The DHE utilizes a dedicated Database Schema Mapper Module to add value to database applications by making this information explicit.

Figure 6 shows a preliminary snapshot of our Database Schema Mapper. When complete, its three frames will subdivide the top, main frame of our system shown in Figure 1. The leftmost division of the main frame contains the database query results as before. The middle division shows the relational (conceptual) schema behind the query results. The rightmost division shows the original, non-normalized entity-relationship schema corresponding to the query results.

<Place Figure 6 about here>

The Database Schema Mapper (DSM) runs in conjunction with the Relational Database Wrapper Module. As mentioned earlier the RDWM provides metadata by examining the relational schema and returns attributes such as names of columns, data types etc. To this metadata the DSM adds schematic information for the values retrieved from a database query.

The Entity-Relationship schematic information about a particular database has to be entered one time through a user interface by a system developer or administrator at the time the system is being designed or integrated with DHE. At runtime when a query has been issued, the DSM checks its internal database to see if the schematic information is available. If it is, then it allows the user the ability to view the underlying E-R schema as well as the relational schema of the database from which the query result was retrieved.

The DSM receives messages either from the RDWM or from the User Interface Wrapper. The RDWM passes query result sets through the DSM to add schematic information. The DSM parses the original query to get the table name. It then queries its own internal database to see if it contains schema information for that table. If so, the DSM generates the E-R and relational schemas, marks whichever elements of interest each contains, and sends a message to the UI Wrapper to display these together with the regular query results.

The UI Wrapper sends the DSM messages when a user follows a DSM-related link. Assume, for example, that the user selects a table and chooses a link to highlight that table in the relational schema. The mapping rule corresponding to that link sends the appropriate command to the DSM. The DSM must follow its internal mapping from the RDWM URI scheme to the DSM URI scheme to identify the corresponding table element in the relational schema. Then the DSM creates a new display where it indicates that the UI Wrapper should highlight certain elements.

When parsing the RDWM's query results, the DSM marks the following as elements of interest in the corresponding E-R schema, and includes the properties shown as parameters:

1. E-R Database Schema
    - Name (of the database in the DSM internal database)
2. Entities
    - Name

- Type (Weak Entity or Normal Entity)

3. Relationship
    - Name
    - Type (identifying relationship or weak entity relationship)
4. Attributes
    - Name
    - Type:
        - Composite or Simple
        - Multivalued or Single-Valued
        - Stored or Derived
        - Key or not

The mapping rules capture the relationships among each of these elements. Given any particular database, entity, relationship or attribute in the E-R schema, the DSM will find the corresponding database, entities, relationships and attributes related to it in the E-R schema. Additional mapping rules will find the corresponding elements in the relational schema and query result currently displayed, and vice versa.

When parsing the RDWM's query results, the DSM marks the following as elements of interest in the corresponding relational schema, and includes the properties shown as parameters, with corresponding relationships:

1. Relational Database Schema
    - Name
2. Relation
    - Name
3. Attribute
    - Primary Key
    - Foreign key
4. Referential Integrity Constraint
    - Relation name where the foreign key resides
    - Relation name where the foreign key references
    - Attribute name of the foreign key in the residing relation
    - Attribute name of the primary key of the referenced table

3.4 NJ DOT Freight Database Application.

The New Jersey Department of Transportation has an extensive database that contains commodity (coal, vegetable oil etc.) flow information between various counties in New Jersey and various zones in the northeastern United States. A rudimentary web interface to this database exists; we have created a DHE module that supplements this system, provides enhanced metadata, exposes the various interrelationships between the "elements of interest" in the system, and provides additional functionality.

Architecture.

The Freight Database Wrapper is a DHE module that acts as the wrapper to the Freight application. Like other application wrappers described in previous sections, the Freight Database Wrapper provides a gateway to application specific commands. Users may view the system through the DHE's user interface

and view reports on commodity flows between counties and zones, via a set of menu items and mapping rules.

The Freight Database Wrapper may also use the RDWM to completely bypass the existing web based freight system and access the underlying database directly. This serves a twofold purpose:

- Metadata that is currently not provided by the system can be extracted directly from the database by means of SQL statements executed by the RDWM.

- New mapping rules maybe formulated, these mapping rules would correspond to SQL statements that would be executed on the freight database, thus providing additional functionality not available in the existing system.

The Freight Database Wrapper may also be used in conjunction with the Database Schema Module, providing users with a view of the internal schema of the freight system's database, which would benefit analysts wishing for a deeper understanding of the freight application's structure.

4. Tightly Integrating Database Applications

Prior sections discussed integrating database applications created independently of DHE. In this section we describe how database applications could be developed more quickly if designed to take advantage of DHE's infrastructure. We conclude by analyzing a system we are building using the DHE and describe the benefits the DHE provides to this system.

In this role the DHE will provide access to a relational database for applications that need it. All requests to a database i.e., SQL statements that need to be executed on a database will be routed to the RDWM, which will execute the statement and return the hypermedia enriched results to the application.

To integrate with DHE, an application normally routes all database access requests to its application wrapper. In this case, because the application is being developed from the ground up, this wrapper maybe a part of the application itself. The wrapper (or wrapper portion of the application) would pass a DHE-formatted XML message to the RDWM to perform any database services requested by the application—usually the execution of an SQL statement or retrieval of metadata from the RDBMS being wrapped by the RDWM.

RDWM would still be responsible for marking up any query results, passing the hypermedia-enriched document is then routed back to the application wrapper. Figure 7 sketches this architecture. Figure 8 provides a state diagram for the information flow.

<Place Figure 7 about here>

<Place Figure 8 about here>

At this point the Application Wrapper may take the enriched document and translate it to the application's native User Interface, and display it there. However, should the application developer decide to use DHE's user interface instead of writing its own, then the Application Wrapper could add any additional content and pass the final display document to the Mapping Rules Module via the Message Manager. If it adds additional, non-database elements, then the application wrapper should mark these up too, as described in §3.1, so they also may be made into anchors. Each of these additional elements types

will require its own mapping rules for determining links and metadata. Many application commands can be moved into the link's mapping rules.

The application could also take advantage of other DHE services, such as the menu manager for displaying application specific menus and the user preference manager for managing users' sessions, login, profiles etc., not to mention the other hypermedia functionality that all applications receive.

4.1 Case Study

This section describes a proposal to develop a student paper review information system using the DHE.

Problem Description

As part of their coursework for the introductory IS Principles Class at the New Jersey Institute of Technology, students must review a published paper. In part students would email the instructor the bibliographic reference to the article they would like to review and the instructor would approve or reject the request. If rejected, the student would have to resubmit his/her request with a new article. If approved the student would review the article and email it to the instructor. The instructor would then grade the review. Because of the volume of email an instructor receives, it became difficult for him or her to quickly approve or reject a request, because he or she must go through previously approved requests and make sure that this article has not been approved for review by another student.

To ameliorate this situation we propose to develop a DHE module that would allow a student to check the articles already approved to ensure his or hers was not already chosen, request approval, check the approval status, post a link to the review once completed and check his or her grade. The instructor would be able to view pending requests, approve a request, view submitted reviews, assign grades, view reports on approved articles, submitted reviews etc. The instructor would also be able to perform administrative tasks such as registering students and deleting old reviews.

Architecture

The paper review application module itself does not have to manage customized menus or user authentication issues; the DHE provides this service. Once a user has logged in and been authenticated, any subsequent requests (view grades, approve a review request, etc.) will always contain the username and the group the user belongs to. The application can thus perform or reject the action based on the privileges the user may have. The application developer may also register customized menus with DHE's menu manager module based on user groups; thus users belonging to the "student" group will not see menus for administrative tasks.

The DHE provides database access to this application. For example when the instructor wishes to view a report on all approved articles not yet submitted, he or she would click on the appropriate menu item and a message would be issued to the application module with a request to generate this report. The application module would then issue a message to the RDWM containing the appropriate SQL statement, the RDWM would execute the statement and return a document containing the results of the statement, with the "elements of interest" marked up. At this point the application module may simply forward this message to the User Interface Wrapper and it will be displayed in the instructor's browser with link anchors automatically embedded within them. The instructor may click on this anchor to get more metadata for this element, and view a list of links, generated from the set of mapping rules for its element type. For example, if the element is an article, the metadata may be the complete bibliographic reference to the article. One mapping rule might generate a list of students who have requested this article. A second could provide a link to the article itself.

The DHE thus speeds up the development process by freeing the developer from having to program user management, menu management, database access and above all hyperlink creation. All the developer would have to do is:

- Create a set of display screens (without links) with any elements marked up with their element types and internal identifiers.

- Register a set of mapping rules corresponding to the additional functionality desired (i.e. view a list of students who have requested a specific article, view a list of pending approval requests for a given section etc.).

- Formulate a set of SQL statements corresponding to each of the main commands underlying the mapping rules (i.e. retrieve grades for a student, display submitted reviews etc.)

- Write a DHE-compatible Module that receives requests for each of the main commands and sends the appropriate SQL statement to the RDWM.

5. Integrating Data Warehousing

This section proposes integrating a data warehousing application within the DHE infrastructure, one of our future research topics. This would provide hypermedia support to data warehousing applications, as well as facilitate linking among data warehouses and other applications.

A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management decision making (Inmon 1996). A data warehouse comprises a repository of information built using data from diverse, and often departmentally isolated, application systems within an organization so this data can be modeled and analyzed by managers (Johnson 1999; Inmon 1996). Data warehouses usually are customized for a particular enterprise. Most vendors offer platforms on which enterprise data warehouses (or smaller datamarts) may be built.

A data warehouse architecture integrates a metadata repository that contains:

- Administrative metadata: source databases and their contents; gateway description; warehouse schema, view and derived data definitions; dimensions and hierarchies; predefined queries and reports; data mart locations and contents; data partitions; data extraction, cleaning, transformation rules, default values; data refresh and purge rules; user profiles, user groups; security

- Business metadata: business terms and definitions; ownership of data; charging policies

- Operational metadata: data lineage (history of migrated data and sequence of transformation applied); currency of data: active, archived, purged; monitoring information: warehouse usage statistics, error reports, audit trails

The DHE maybe used to offer a complete end–to-end solution with the added benefit of obtaining hypertext functionality without any additional effort.

The data warehouse has two broad functions:

- Accessing the data from the data warehouse

- Loading the data from the operational systems into the data warehouse

Data warehouses are typically used for on-line analytical processing (OLAP). The key structure of a data warehouse always contains some element of time and some dimension hierarchies. OLAP queries are complex. They involve grouping and aggregation. A single OLAP query can lead to several closely related queries (Chaudhuri et al. 1997). The visualization of an OLAP query result using DHE will involve links between data from one hierarchy level to the other and links SQL subqueries contained in the OLAP query. In addition, an OLAP query can result in a large collection of data with several dimensions. In the rest of this section we concentrate on the loader module.

To load data into the data warehouse a loader module will have to be designed and developed. The Loader Module will be like any other DHE module, and will perform the following functions:

Map data from Operational Systems to the Data Warehouse

The operational systems store data in their own structure, encoding etc. This has to be mapped to the data warehouse's format which is consistent across all operational systems.

Extract Metadata from Operational Systems

Metadata is the road map or blueprint to the data in the data warehouse, and needs to be operational. Also, metadata needs to be preserved for analysis once it has been loaded (Gardner 1998). Metadata may include:

- The structure of the operational data
- Relationships in the operational data
- Other user-defined metadata

Eliminate Noise

Operational data may contain data irrelevant to the warehouse (i.e., noise), which needs to be eliminated before loading.

The Loader Module is supplied a template (an Extended Stylesheet Language or XSL stylesheet) that maps data from the operational systems' format to the data warehouse's format. It processes the template and maps data from the operational system to the data warehouse. Any data not specified in the template is noise and will be eliminated.

Once the extraction process is complete, the Loader Module sends a message to the RDWM containing the data to be loaded as well as the metadata. The RDWM then loads this into the data warehouse. Figure 9 describes the extraction process.

<Place Figure 9 about here>

An argument could be made that a Loader Module is not required and the DHE is simply used to access data from the warehouse. However this approach would not allow the DHE to retrieve metadata from the operational systems. Moreover a complete end-to-end solution requires that we provide a Loader Module.

6. Research in Hypermedia and Databases

Several techniques have been proposed recently for the integration of hypertext and databases. Some of them address the issue of building hypertext structures over existing databases to provide more direct navigation through hyperlinks.

Hara and Botafogo (1994) use an SQL-like data definition language to map single relations or relational views to node types. A node type is similar in nature to an entity type, i.e., it models a real world object or concept in the hypertext relational and ER schemata defined over the database contents. Its specification includes the correspondences between relation attributes and node fields, as well as presentation information. At run time, a node type produces two kinds of nodes: a composite one for the whole relation, and a number of nodes corresponding to the tuples of the relation. The same language is used to define link types among node types. A WHERE-clause is used to constraint the creation of links during navigation.

In a similar approach, Falquet et al. (1995, 1998) offer a declarative language to produce databases views composed of node and link schemas, accessed through the WWW. Each node schema is based on one object class or a set of inter-related object classes. The content of the node is composed of a subset of the attributes of the class(es). Foreign keys to other classes constitute link types to the corresponding nodes. Two kinds of links are supported: *reference* links are indented to offer navigation structure within the nodes, while *inclusion* links are indented to create nested structures (part-of relationships). In addition, the specification of the relational view includes presentation information. The above definitions form the input to a cgi-script that produces HTML pages for the end-user. DHE would enhance the existing views specified through the database application.

The above approaches leave the original client application intact, introducing a new interface that provides hypermedia-based interaction with the database. On the contrary, DHE overlays linking facilities within the original user interface application by means of user interface wrappers.

Domenicus (Constantopoulos et al. 1996) is a hypermedia engine developed over a repository management system, called Semantic Index System (SIS). Domenicus offers hypermedia functionality (such as alphabetic lists, subject catalogs, guided tours, query cards, hyperlinks, image annotations, bookmarks and history), based on predefined queries over the information objects and their structure, managed by SIS. *Presentation Card Specifications* are executed at run-time to present objects or classes of objects stored in SIS, while *hyperlink classes* dynamically produce links during navigation. Different presentation models can coexist for the same repository instance, to fulfil the searching, browsing and updating requirements of different user groups. DHE provides many of these features in a generic way over any application, allowing tours and indexes to contain elements from several systems. Also, in DHE queries are only predefined to the extent that mapping rules hold skeleton queries for particular classes of database elements.

Other approaches suggest embedding database queries into HTML pages. For example, a mechanism offering cross-language variable substitution between HTML and SQL is the core of the *DB2 WWW Connection* system (Nguyen et al. 1996), which enables quick and easy construction of applications accessing relational databases from the Web. The developer creates *macros* that consist of HTML and SQL commands, written in distinct sections. The sections are tied together via variable substitution. Macros are stored at the Web server and are processed by cgi-scripts in order to get user input or produce output reports. DHE does not store single database queries in the pages displayed on the Web. Instead we generate a list of several possible links for any element from specifications in the mapping rules.

Instead of providing hypertext functionality for a specific database, Geldof (1996) uses an abstract page definition language to construct templates embodying presentation guidelines for terms of an

ontology; a conceptualisation containing objects, concepts and relationships among them, that are presumed to exist in some area of interest. Actual information sources are linked separately with the terms of the ontology, using a definition language as well. CGI-scripts in Perl are computed to dynamically generate the HTML pages returned to the user for browsing. While this approach adds a certain level of automated linking to aid navigation, DHE provides a generally larger set of links based solely on the database structure and entity-relationship schemas, as well as metadata. DHE, however, does not provide customized templates for domain-specific navigational contexts. DHE might integrate well with Geldof's approach to provide an additional level of functionality.

Moreover, many products have been released recently that aim to interface RDBMS and Web servers (Frank 1995). The solutions employed in these products require huge programming effort in SQL or a scripting language. The ease of integration with DHE depends on how easy it is to parse application displays to identify the elements of interest, and to specify the commands to return to the application in the mapping rules. If the application has an API or marks the elements in the displays (as should become the custom as XML becomes more prevalent), building the application wrapper should be relatively easy.

The approaches presented above presuppose the hypertext designer's insight into the intrinsic semantics of the relational structure. A different approach was proposed by Papadopoulos et al. (1996). Instead of relying on the relational schema of the database, a more semantically enriched Extended Entity-Relationship (EER) schema is semi-automatically produced, by incorporating a reverse engineering methodology. Hypertext views, consisting of node and link types, can be defined over the EER schema, while the SQL queries to instantiate them at run-time where automatically created, based on mapping information gathered during the reverse engineering process. Currently DHE requires people to enter the entity-relationship schemas manually to the Database Schema Mapper (see §3.3). This application could help to automate this process, and perhaps provide additional relationships, which DHE could provide for database applications.

While the hypermedia paradigm embodies an approach to structure and navigate information, it has several shortcomings. In particular, few hypermedia systems have focused on methodologies for information storage and retrieval. Database systems, on the contrary, are only concerned with storage and retrieval of information based on a formal model. They exhibit powerful methodologies for information storage, and effective indexing and querying. Furthermore they provide facilities such as transaction management, concurrency and access control as well as locking mechanisms.

Early hypermedia systems used proprietary data formats for storage (Fountain et al. 1990). Over the years, hypertext researchers, trying to satisfy the requirements of new application domains, have studied various techniques for integrating hypertext and database facilities. Depending on the purpose of such an integration, these approaches can be categorized as: managing internal hypermedia data, supporting hypermedia application design and assisting information retrieval.

Managing Hypermedia Data

Several hypermedia systems employ databases internally to hold hypermedia content and structure. Hyperbase Management Systems make heavy use of database facilities to provide persistent storage of hypermedia data (Leggett 1993). HyperDisco (Wiil et al. 1996), HyperStorM (Bapat et al. 1996) and DHM (Grønbæk et al. 1994) use database facilities to store, update and retrieve hypermedia constructs such as node, link and anchor data. Chimera (Anderson 1999) enhanced its ad hoc storage management capabilities with a relational database management system in order to handle data scalability issues. In a similar manner, DHE modules use relational databases to manage internal data such as the message log, menus and mapping rules.

Hypermedia Application Design

Many systems use database concepts in order to enable hypermedia application design. Hypermedia design methodologies (Christodoulou et al. 1998) not only help designers with the standard conceptual structuring of application designs but also focus on designing the navigational structures (links, indexes, guided tours, etc.), and thus are especially well suited to designing Web applications. RMM (Isakowitz et al. 1995, Balasubramanian et al. 2000) is the hypermedia design methodology based most strongly on the relational database paradigm. It was developed specifically to design hypermedia interfaces for relational database applications. It has an accompanying tool, RMCase that then generates the application based on the RMM design (Diás et al. 1995). RMM has seven steps:

(1) Entity-Relationship Design: models the information domain and its relationships
(2) Slice Design: how information units are sub-divided for display
(3) Navigational Design: how users will access information
(4) User-Interface Design: how information will be presented
(5) Conversion Protocol Design: how abstract constructs are to be transformed into physical-level constructs, e.g., what kind of WWW page corresponds to an index or a guided tour
(6) Run-time Behavior Design: how to populate the application with data and how to provide interaction behavior
(7) Construction and Testing: actual development of programs and testing

OOHDM (Schwabe et al. 1995, 1996) currently is the most popular hypermedia design methodology. While OOHDM does not follow the relational database paradigm as strongly as RMM, it still could be quite useful in designing the navigation over Web-based database applications.

Assisting Information Retrieval

Some systems utilize hypermedia as well as information retrieval facilities resulting in *Hypermedia Information Retrieval Systems*. Such systems provide users with the possibility of storing large amounts of textual and multimedia elements as well as building networks of semantic relationships among these elements within the database for use in retrieval (Agosti et al. 1996). Although querying and browsing are considered as complementary paradigms (Chiramella 1997), the particular difficulty in creating such systems lies in the fundamental distinction between structure and content. Hypermedia Information Retrieval Systems concentrate on providing automatic and semi-automatic conversion of text into hypertext (Salton et al. 1994; Furner et al. 1996, Golovchinsky 1997) aiming at “interactive retrieval” as well as applying information retrieval capabilities to hypermedia structures (Savoy 1996; Chiramella et al. 1996).

7. Future Research Plans

Given our basic DHE infrastructure, we can begin to research many interesting ways to take advantage of and enhance database systems and technologies.

Integrating Data Mining

Currently DHE determines links from the mapping rules. Because the person who develops the application wrapper also writes the mapping rules at the same time, the types of relationships DHE finds are known ahead of time. Data mining brings the opportunity of a new kind of dynamic linking. Data mining searches large databases for relationships and global patterns and relationships that are not immediately obvious, such as a relationship between patient data and their medical diagnosis (Holsheimer 1994).

Data mining tools discover these relationships or models at runtime as opposed to design time. Thus, DHE must request the Data Mining Tool to discover the relationships for an element of interest at runtime, and then use these discovered relationships to create hyperlinks.

Of course, in addition, the DHE could provide hypertext functionality to commercial Data Mining Tools. A commercial data mining tool could have a wrapper written for it, just as with any other application.

RELATIONSHIP-BASED DESIGN OF DATABASE

We currently are developing a relationship-based systems analysis methodology, Relationship-Navigation Analysis (RNA) to help analysts determine the interrelationships within their applications (Bieber & Yoo 2000; Yoo & Bieber 2000a, b; Yoo 2000). Once identified, each relationship could be designed and implemented as a link in the application. We have developed RNA to help in any application domain, not just databases. In future research we intend to compare RNA with a traditional database analysis technique. We believe RNA could be used as a tool to help designers make the "hidden" relationships explicit in an application, as well as identify relationships in application components outside the database contents.

8. Conclusion

While the main purpose of this chapter has been to describe our hypermedia and database research, we hope that it also sets forth a research agenda for joint research in these two fields. Database applications need to interact with their users. Hypermedia can supplement database applications in many ways, making them more effective. The Dynamic Hypermedia Engine enriches the user's experience with the power of hypermedia by automatically providing it to applications with supplemental links, metadata, navigation, and other hypermedia functionality. It also eases the development of complex database systems by automatically providing these metadata and hypermedia services.

Acknowledgements

We gratefully acknowledge funding by the United Parcel Service, the NASA JOVE faculty fellowship program, by the New Jersey Center for Multimedia Research, by the National Center for Transportation and Industrial Productivity at the New Jersey Institute of Technology (NJIT), by the New Jersey Department of Transportation, by the New Jersey Commission of Science and Technology, and by NJIT's SBR program.

References

Agosti, M. and Smeaton A. (1996). Information Retrieval and Hypertext, Kluwer Academic Publishers, Boston.

Akscyn, Robert M., Donald L.McCracken and Elise A. Yoder (1988). KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations," Communications of the ACM, 31(7), 820-835.

Anderson, K. (1996). Providing Automatic Support for Extra-Application Functionality," In Ashman, H.L., Balasubramanian, V., Bieber, M. and Oinas-Kukkonen, H. (eds). Proceedings of the Second International Workshop on Incorporating Hypertext Functionality Into Software Systems (HTF II), Hypertext '96 Conference, Bethesda, March 1996.

Anderson, K. (1997). Integrating Open Hypermedia Systems with the World Wide Web, Hypertext'97 Proceedings, ACM Press, New York, NY, 157-166.

Anderson, M. K. (1999). Data Scalability in Open Hypermedia Systems, Proceedings of the ACM Hypertext '99 Conference, Darmstadt, Germany, February 21-25, 27-36.

Balasubramanian, V., Bieber, M. and Isakowitz, T. (2000). A Case Study in Systematic Hypermedia Design, Information Systems Journal (forthcoming).

Bapat, A., Waesch, J., Aberer, K., and Haake, J. (1996). HyperStorM: an Extensible Object-Oriented Hypermedia Engine, Proceedings of the ACM Hypertext '96 Conference, Washington, DC, 203-214.

Berners-Lee, T., Fielding, R., and Masinter, L. (1998). Uniform Resource Identifiers (URI): Generic Syntax, Internet Engineering Task Force Request For Comments 2396, August 1998.

Bieber, M. (1990). Generalized Hypertext in a Knowledge-based DSS Shell Environment, Ph.D. dissertation, University of Pennsylvania, Philadelphia, PA 19104.

Bieber, M. (1995). On Integrating Hypermedia into Decision Support and Other Information Systems, Decision Support Systems 14, 1995, 251-267.

Bieber, M. (1999). Supplementing Applications with Hypermedia, Technical Report, New Jersey Institute of Technology, Information Systems Department.

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

Bieber, M., Vitali, F., Ashman, H., Balasubramanian V., and Oinas-Kukkonen, H. (1997). Fourth generation hypermedia: some missing links for the World Wide Web, International Journal of Human Computer Studies 47, 31-65.

Bieber, Michael, and Joonheee, Yoo (2000). Hypermedia: A Design Philosophy, ACM Computing Surveys (forthcoming).

Campbell, Brad and Joseph M. Goodman (1988). HAM: A General Purpose Hypertext Abstract Machine," Communications of the ACM, 31(7), 856-861.

Chaudhuri, S., and Dayal, U. (1997). An Overview of Data Warehousing and OLAP Technology, ACM SIGMOD Record, 26(1), 65-74.

Chiramella, Y., and Kheirbek, A. (1996). An Integrated Model for Hypermedia and Information Retrieval, Information Retrieval and Hypertext, M. Agosti, A. Smeaton (Eds.), Kluwer, Amsterdam, NL, 139-176.

Chiramella, Y. (1997). Browsing and Querying: Two Complementary Approaches for Multimedia Information Retrieval, Proceedings of Hypertext – Information Retrieval – Multimedia (HIM '97), Dortmund, Germany, 9-26.

Chiu, C., and Bieber, M. (1997). A Generic Dynamic-Mapping Wrapper for Open Hypertext System Support of Analytical Applications, Proceedings of the ACM Hypertext '97, Souhtampton, UK, April 1997, 218-219. (http://www.cis.njit.edu/~bieber/pub/ht97/ht97-mac.html)

Christodoulou, S., Styliaras, G., and Papatheodorou, T. (1998). Evaluation of Hypermedia Application Development and Management Systems, Proceedings of the ACM Hypertext '98 Conference, Pittsburgh, PA, 1-10.

Conklin, J., (1987). Hypertext: a Survey and Introduction, IEEE Computer, 20(9), 17-41.

Constantopoulos, P., Theodorakis, M., Tzitzikas, Y. (1996). Developing Hypermedia Over an Information Repository, Proceedings of the 2nd Workshop on Open Hypermedia Systems, ACM Hypertext '96 Conference, Washington, DC.

Davis, H., Hall, W., Heath, I., Hill, G., and Wilkins, R. (1992). Towards an Integrated Information Environment with Open Hypermedia Systems, Proceedings of the ACM Conference on Hypertext Milan, 181-190.

Davis, H., Knight, S., and Hall, W., (1994). Light Hypermedia Link Services: A Study of Third Party Application Integration, Proceedings of the Fifth ACM Conference on Hypermedia Technologies, Edinburgh, Scotland, 158-166.

Diáz, A., Isakowitz, T., Maiorana, V., and Gilabert, G. (1995). RMC: A Tool To Design WWW Applications, Proceedings of the Fourth International World Wide Web Conference, Boston, December 1995.

Falquet, G., Guyot, J., Prince, I. (1995). Generating Hypertext Views on Databases, CUI Technical Report No 101, University of Geneva.

Falquet, G., Guyot, J., Nerima, L. (1998). Languages and Tools to Specify Hypertext Views on Databases, selected papers of the International Workshop on the WWW and Databases (webDB '98), Valencia, Spain, March 27-28, 1998, Springer-Verlag LNCS 1590, 1999.

Frank, M.(1995). Database and the Internet, DBMS Magazine, vol. 8, no. 13, December 1995.

Fountain, A., Hall, W., Health, I., Davis, H. C. (1990). Microcosm: An Open Model for Hypermedia with Dynamic Linking, Proceedings of the - European Conference on Hypertext, Paris, France, 298 – 311.

Furner, J., Ellis, D., Willett, P. (1996). The Representation and Comparsion of Hypertext structures using Graphs , Information Retrieval and Hypertext, M. Agosti, A. Smeaton (Eds.), Kluwer, Amsterdam, NL, 75-96.

Gardner, S. R. (1998). Building the Data Warehouse, Communications of the ACM, 41(9), Sep. 1998, 52-60.

Garrido, A., and G. Rossi, (1996). A framework for extending Object-Oriented Applications with Hypermedia Functionality, The New Review of Hypermedia and Multimedia 2, 25-41.

Geldof, S. (1996). Hyper-text generation from databases on the Internet, Proceedings of the 2nd Intl. Workshop on Applications of Natural Language to Information Systems (NLDB '96), Amsterdam, IOS Press, 102-114.

Golovchinsky, G. (1997). What the Query Told the Link: The Integration of Hypertext and Information Retrieval, Proceedings of ACM Hypertext'97 Conference, Southampton, UK, April 1997, 67-74.

Grønbæk, K., Hem, J., Madsen, O., and Sloth, L. (1994). Cooperative Hypermedia Systems: A Dexter-based Architecture, Communications of the ACM 37(2), 64-74.

Grønbæk, K., and Trigg, R. (1994). Design Issues for a Dexter-Based Hypermedia System. Communications of the ACM, 37(2), 40-49.

Grønbæk, K., and Trigg, R. (1999). From Web to Workplace: Designing Open Hypermedia Systems, MIT Press.

Halasz, F. and Schwartz, M. (1994). The Dexter Hypertext Reference Model, (editied by K. Grønbæk and R. Trigg), Communications of the ACM 37(2), 30-39.

Hara, Y., Botafogo, R. A.(1994). Hypermedia Databases: A Specification and Formal Language, proceeding of the Databases and Expert Systems Applications Conference (DEXA '94), Springer-Verlag LCNS 856, 520-530.

Holsheimer, M., and Siebes, A. (1994). (Report CS-R9406) Data Mining, The Search for Knowledge in Databases, CWI, Amsterdam. (ftp://ftp.cwi.nl/pub/CWIreports/AA/CS-R9406.ps.Z)

Inmon, W. H. (1996). Building the Data Warehouse, Second Edition, Wiley Comp., ISBN O471-14161-5, USA, 1996.

Johnson, A. H. (1999). Data Warehousing, Computerworld, 33(49), Dec. 1999, 74-75.

Kacmar, C. (1993). Supporting Hypermedia Services in the User Interface, Hypermedia 5(2), 85-101.

Kacmar, C. (1995). A Process Approach for Providing Hypermedia Services to Existing, Non-hypermedia Applications," Journal of Electronic Publishing: Organization, Dissemination and Design.

Isakowitz, T., Stohr, E., and Balasubramanian, P. (1995). RMM: A Methodology for Structuring Hypermedia Design, Communications of the ACM 38(8), Aug. 1995, 34-44.

Lassila, Ora, and Swick, Ralph R. (Editors) (1999). Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999.

Leggett, J. J. (ed.) (1993). Workshop on Hyperbase Systems, ACM Hypertext '93 Conference, Technical Report TAMU-HRL 93-009, Texas A&M University.

Leggett, John and Schnase, John (1994). Viewing Dexter with Open Eyes, Communications of the ACM 37(2), 77-86.

Marshall, Catherine and Shipman III, Frank, (1995). Spatial Hypertext: Designing for Change, Communications of the ACM 38(8), 88-97.

Nanard, J. and Nanard, M. (1995). Hypertext Design Environments and the Hypertext Design Process," Communications of the ACM 38(8), 49-56.

Nguyen, T., Srinivasan, V. (1996). Accessing Relational Databases from the World Wide Web, Proceedings of the ACM SIGMOD '96 Conference, 529-540.

Nielsen, Jakob (1995). Multimedia and Hypertext: The Internet and Beyond, AP Professional.

Papadopoulos, A., Vaitis, M., Christodoulakis, D. (1996). Building Hypertext Interfaces to Existing Relational Databases, proceeding of the 7th Intl. Conference on Database and Expert Systems Applications (DEXA '96), Springer-Verlag LCNS 1134, Zürich, Switzerland, 276-288.

Pearl, Amy (1989). Sun's Link Service: A Protocol for Open Linking, Hypertext'89 Proceedings, Pittsburg, November 5, 137-146.

Rossi, G., Garrido, A. and Carvalho, S. (1996). Design Patterns for Object-Oriented Hypermedia Applications. Pattern Languages of Programs II. J. Vlissides, J. Coplien and N. Kerth, eds. Addison-Wesley, 177-191.

Schwabe, D., Rossi, G. (1995). The Object-Oriented Hypermedia Design Model, Communications of the ACM, 38(8), August 1995, 45-46.

Schwabe, D., Rossi, G., Barbosa, S. D. J. (1996). Systematic Hypermedia Application Design with OOHDM, Proceedings of the ACM Hypertext '96 Conference, Washington, DC, March 16-20, 1996, 116-128.

Salton, G. (1989). Automatic Text Processing: the transformation, analysis, and retrieval of Information, Reading, Addison-Wesley, MA.

Salton, G., Allan, J., Buckley, C., and Singhal, A. (1994). Automatic Analysis, Theme Generation, and Summarization of Machine-Readable Texts, Science, vol. 264, 1421-1426.

Savoy, J. (1996).Citation Schemes in Hypertext Information Retrieval, Information Retrieval and Hypertext, M. Agosti, A. Smeaton (Eds.), Kluwer, Amsterdam, NL, 99-120.

Van de Sompel, Herbert and Hochstenbach, Patrick, (1999). Reference Linking in a Hybrid Library Environment, Part 1: Frameworks for Linking, D-lib Magazine 5(4).

Van de Sompel, Herbert and Hochstenbach, Patrick, (1999). Reference Linking in a Hybrid Library Environment, Part 2: SFX, a Generic Linking Solution, D-lib Magazine 5(4).

Van de Sompel, Herbert and Hochstenbach, Patrick, Reference Linking in a Hybrid Library Environment, Part 3: Generalizing the SFX solution in the "SFX@Ghent & SFX@LANL" Experiment, D-lib Magazine 5(10).

Wan, Jiangling (1996). Integrating Hypertext into Information Systems through Dynamic Linking. Ph. D. dissertation, New Jersey Institute of Technology, Institute for Integrated Systems Research, Newark NJ 07102.

Wan, Jiangling, and Michael Bieber (1997). Providing Relational Database Management Systems with Hypertext, Proceedings of the Thirtieth Annual Hawaii International Conference on System Sciences (Wailea, Maui; January 1997), IEEE Press, Washington, D.C., Volume VI, 160-166.

Whitehead, E. J. (1997). An Architectural Model for Application Integration in Open Hypermedia Environments, Hypertext'97 Proceedings, ACM Press, New York, 1997, 1-12.

Wiil, Uffe K. (ed.), (1999). Proceedings of the 3rd Workshop on Open Hypermedia Systems, CIT Scientific Report #SR-97-01, 1997, The Danish National Centre for IT Research, Forskerparken, Gustav Wieds Vej 10, 8000 Aarhus C, Denmark [on-line] http://www.cit.dk/

Wiil, U. K., and Leggett, J. J. (1996). The HyperDisco Approach to Open Hypermedia Systems, Proceedings of the ACM Hypertext '96 Conference, Washington, DC, 140-148.

Yoo, Joonhee (2000). Relationship Analysis. Ph.D. Dissertation, New Jersey Institute of Technology, CIS Department, 2000.

Yoo, Joonhee, and Bieber, Michael (2000). Towards a Relationship Navigation Analysis, Proceedings of the 33rd Hawaii International Conference on System Sciences, IEEE Press, Washington, D.C., January 2000.

Yoo, Joonhee, and Bieber, Michael (2000). Finding Linking Opportunities through Relationship-based Analysis, Proceedings of the ACM Hypertext '00 Conference, San Antonio, TX, June 2000, ACM Press.

About the Authors

FIRAS ALLJALAD

Firas Aljallad is a Ph.D. Student and currently working for a major financial firm as a Senior Java Architect and Technical Specialist. His research interests concern distributed components, Java and Web development.

ANIRBAN BHAUMIK

Anirban Bhaumik is a graduate student in the department in Computer and Information Science at the New Jersey Institute of Technology, Newark NJ. Anirban has a Bachelor of Engineering in Chemical Engineering from the Regional Engineering College, Durgapur, India and a Master of Science in Chemical Engineering from the New Jersey Institute of Technology, Newark NJ. Anirban is employed at Concero a Austin, TX based software consulting company. When not working or studying Anirban likes to kayak on the Delaware.

MICHAEL BIEBER

Michael Bieber is Associate Professor of Information Systems in the Computer and Information Science Department at the New Jersey Institute of Technology, where he also teaches in the distance learning program. He co-directs NJIT's Collaborative Hypermedia Research Laboratory. Dr. Bieber is affiliated with the New Jersey Center for Multimedia Research and the National Center for Transportation and Industrial Productivity. He holds a Ph.D. in Decision Sciences from the University of Pennsylvania. Dr. Bieber has been performing hypermedia research since 1987, when he embarked on a research path in automating hypermedia support for analytical information systems. He is active in the hypertext community, co-organizing conference minitracks and co-editing special journal issues about hypermedia topics. He has published many articles in this and other areas.

DEEPTI DIXIT

Deepti Dixit has an M.S. in Mechanical Engineering from Maulana Azad College of Technology, Bhopal India and a Post Graduate Course in Foreign Trade from Indian Institution of Foreign Trade, New Delhi. She has extensive work experience in the field of Materials Management. She is completing her Masters in Computer Science at the New Jersey Insitute of Technology.

ROBERTO GALNARES

Roberto Galnares received his Bachelors degree from Universidad la Salle, A.C., Mexico. He has worked in the the computer and information systems field since 1980. He was awarded a Fulbright Scholarship for Ph.D. studies in Computer and Information Sciences in 1995. Currently he is conducting research on Hypermedia, Web Technologies, and Web Engineering.

APARNA KRISHNA

Aparna Kishna has a Master of Science in Computer and Information Science at the New Jersey Institute of Technology, Newark NJ and a Bachelor of Technology in Computer Science & Engineering from the Jawaharlal Nehru Technological University, Hyderabad, India. Aparna is employed at Lucent Technologies, Holmdel, N.J.

QIANG LU

Qiang Lu is a professor in the Computer Science Department at Suzhou University in the People's Republic of China. He currently is visiting at the New Jersey Institute of Technology.

VINCENT ORIA

Vincent Oria received his Diplôme d'Ingénieur in 1989 from the Institut National Polytechnique, Yamoussoukro, Côte d'Ivoire (Ivory Coast), a D.E.A in 1990 from Université Pierre et Marie Curie (Paris VI), Paris, France, and a Ph.D. in 1994 in Computer Science from the Ecole Nationale Supérieure des Télécommunications (ENST), Paris, France. He worked as a Research Scientist at the ENST Paris from 1994 to 1996 before joining the Department of Computing Science of the University of Alberta, Canada as a Post-Doctorate from 1996 to 1999. He is currently an Assistant Professor in the Department of Computer and Information Science at the New Jersey Institute of Technology. His current research

interests include multimedia databases, geographical information systems, data management issues in e-commerce and e-learning.

MANOLIS TZAGARAKIS

Manolis Tzagarakis holds a Diploma in Computer Engineering and Informatics from the University of Patras, Greece, where he currently is pursuing his Ph.D. He is collaborating with Research Unit II of the Computer Technology Institute (CTI), Greece, working on hypertext and database systems. His research interests include open hypermedia systems, hypermedia models, structural computing and database management systems.

MICHALIS VAITIS

Michalis Vaitis holds a Diploma in Computer Engineering and Informatics from the University of Patras, Greece, where he also is studying for his Ph.D. He is collaborating with Research Unit II of the Computer Technology Institute (CTI), Greece, working on hypertext and database systems. His research interests include hypermedia models and services, structural computing and databases interoperability. He is a student member of the ACM.

LI ZHANG

Li Zhang received her bachelor's degree in Biomedical Engineering from Xi'an Jiaotong University, China. She holds a masters in Electrical Engineering from Southeast University, China. Currently she is pursuing her Ph.D. in Computer Science at the New Jersey Institute of Technology.

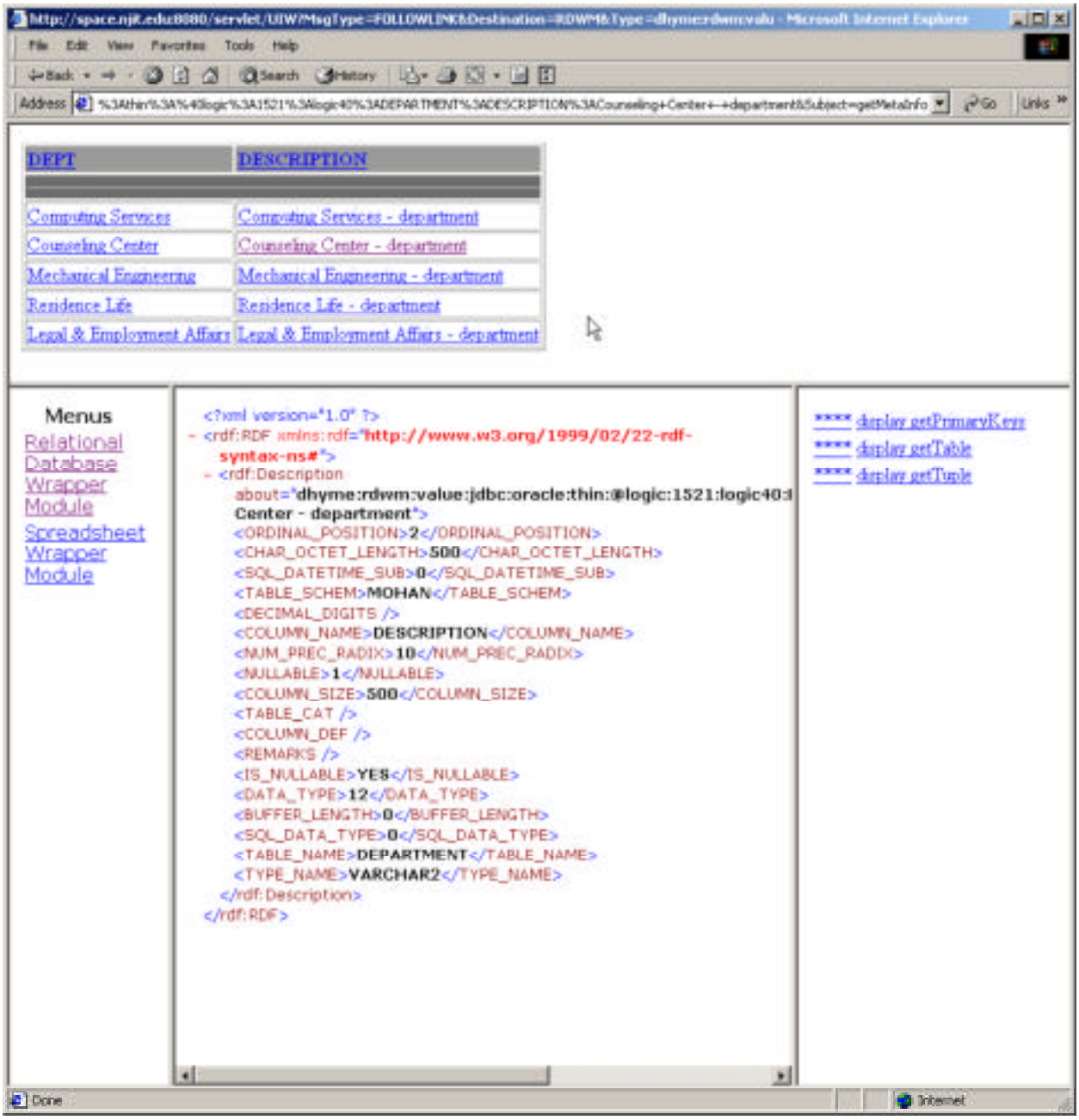

Figure 1: Screen from our Preliminary DHE Web Prototype

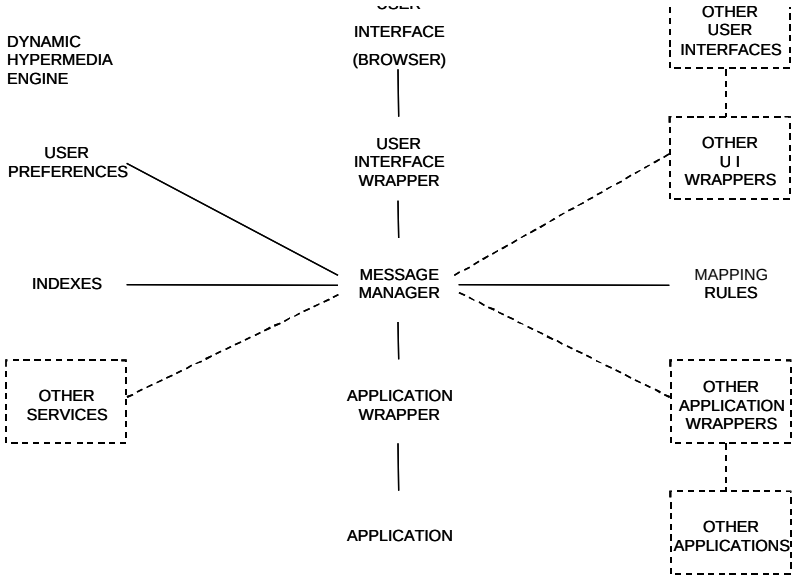

Figure 2: DHE's Logical Architecture

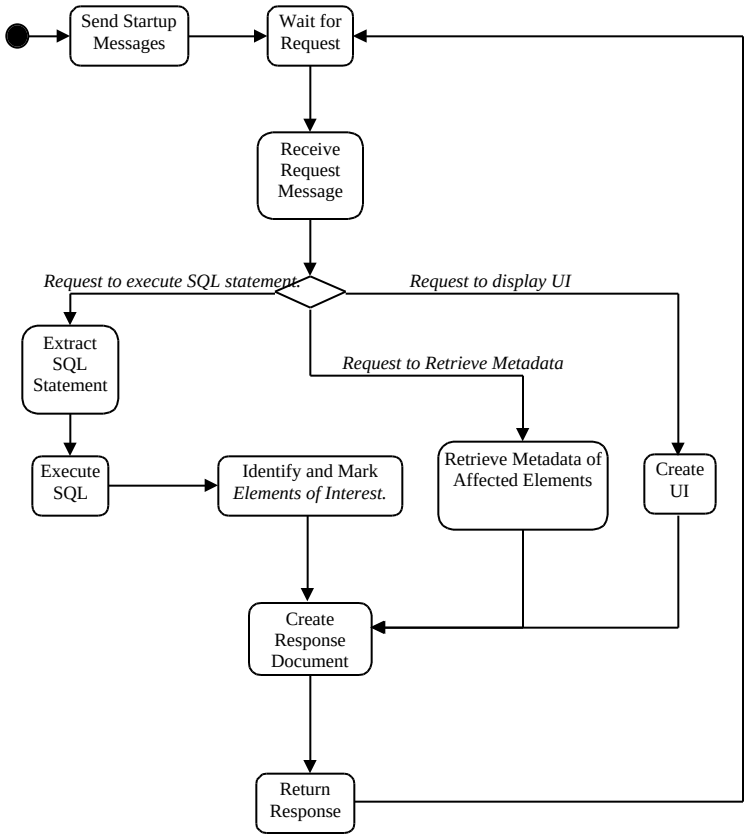

Figure 3: States of the Relational Database Wrapper Module

```
<OutDoc>

<![CDATA[<table border="0">

        <tr><td><b><#id1>DEPT</#id1></b></td><td><b><#id2>DESCRIPTION</#id2></b></td></tr>

        <tr><td colspan="2"><hr></tr>

        <tr><td><#id3>Computing Services</#id3></td><td><#id4>Computing Services -
    department</#id4></td></tr>

        </table>]]>

</OutDoc>

<List_Of_Elements>

<Element>

<Locator>#id3</Locator>

<Type>dhyme:rdwm:value</Type>

<URI>dhyme:rdwm:value:jdbc:oracle:thin:@logic:1521:logic40:DEPARTMENT:DEPT:Computing
Services</URI>

<Action></Action>

</Element>

<Element>

<Locator>#id2</Locator>

<Type>dhyme:rdwm:column</Type>

<URI>dhyme:rdwm:column:jdbc:oracle:thin:@logic:1521:logic40:DEPARTMENT:DESCRIPTION</URI>

<Action></Action>

</Element>

<Element>

<Locator>#id1</Locator>

<Type>dhyme:rdwm:column</Type>
```

Marked Up Element

Type of Marked Up Element

URI of Marked Up Element

Figure: 4 Marked up Message

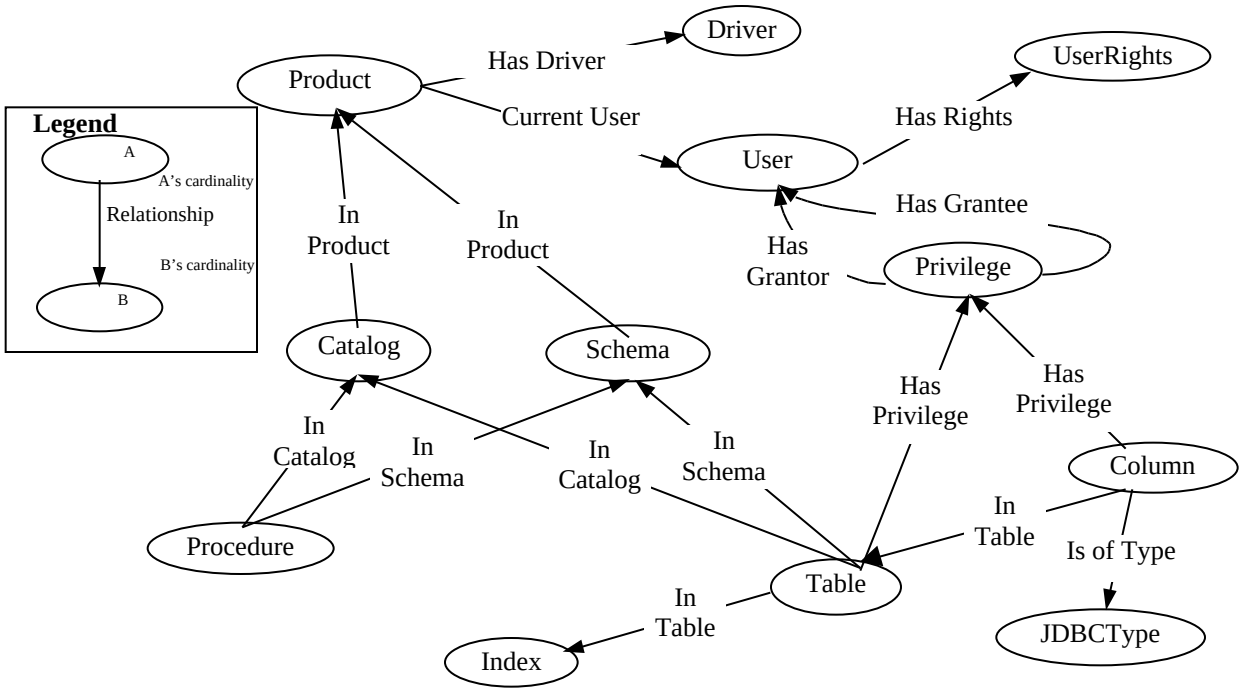

Figure 5: Relationships between Element Types

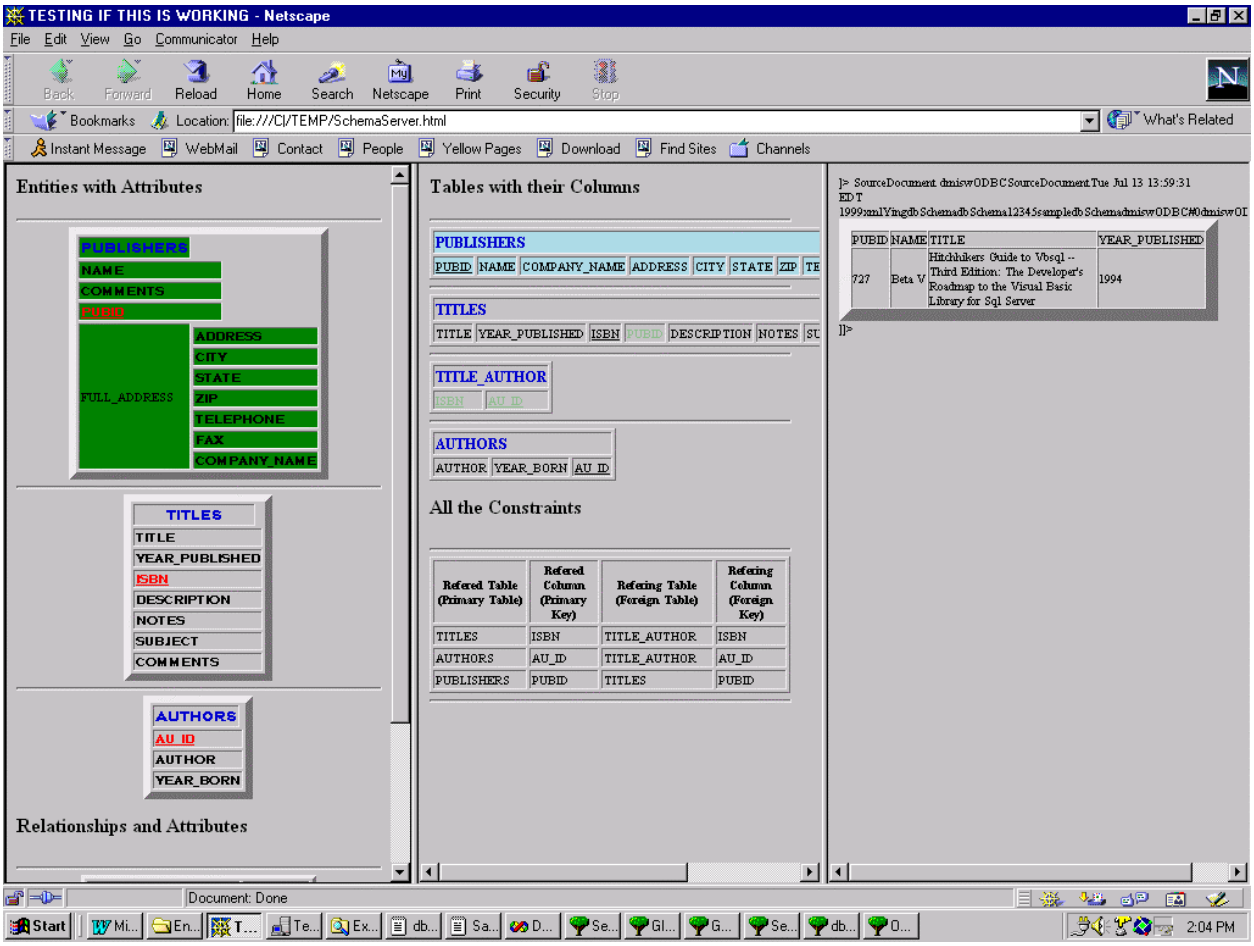

Figure 6: A Sample Relational Database Schema Mapper showing the E-R Schema, Physical Schema and Database Query Results in Three Frames

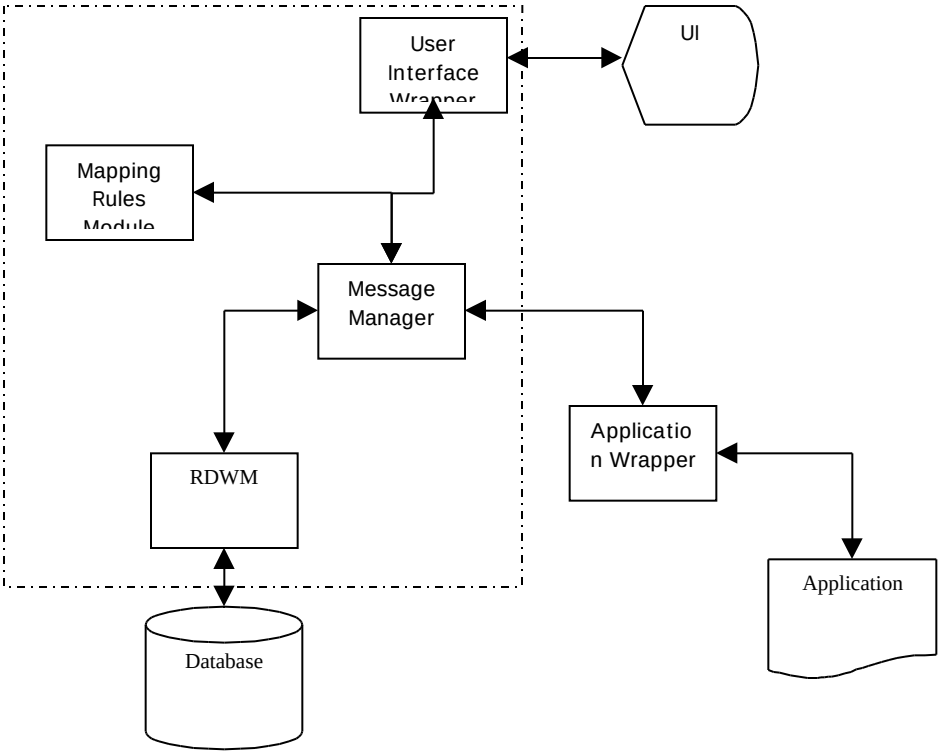

Figure 7: Architecture for Database Application in the Dynamic Hypermedia Engine[1]

[1] The double-headed arrows denote bi-directional message flow between modules/subsystems.

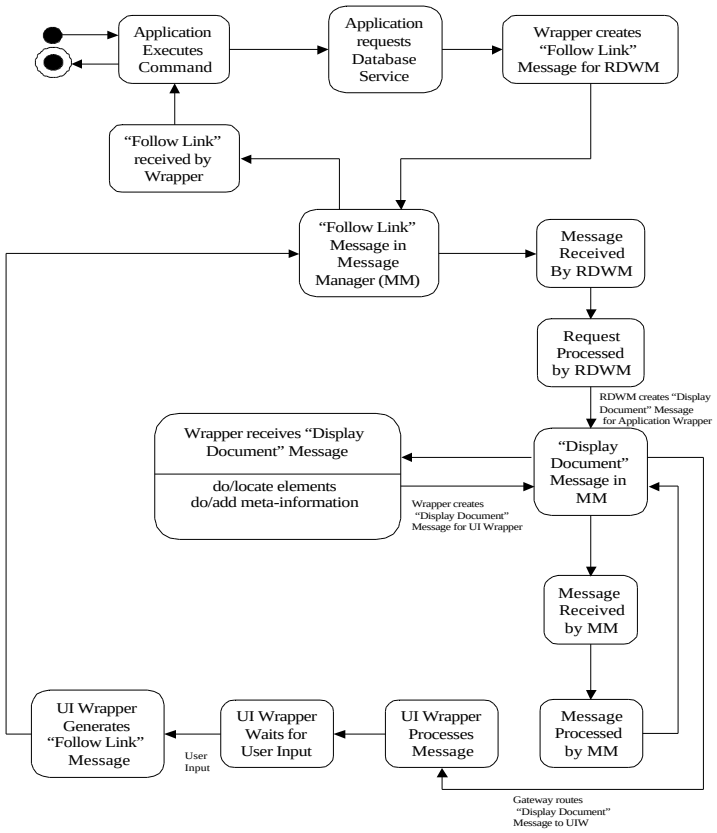

Figure 8: State Diagram for Application using the DHE

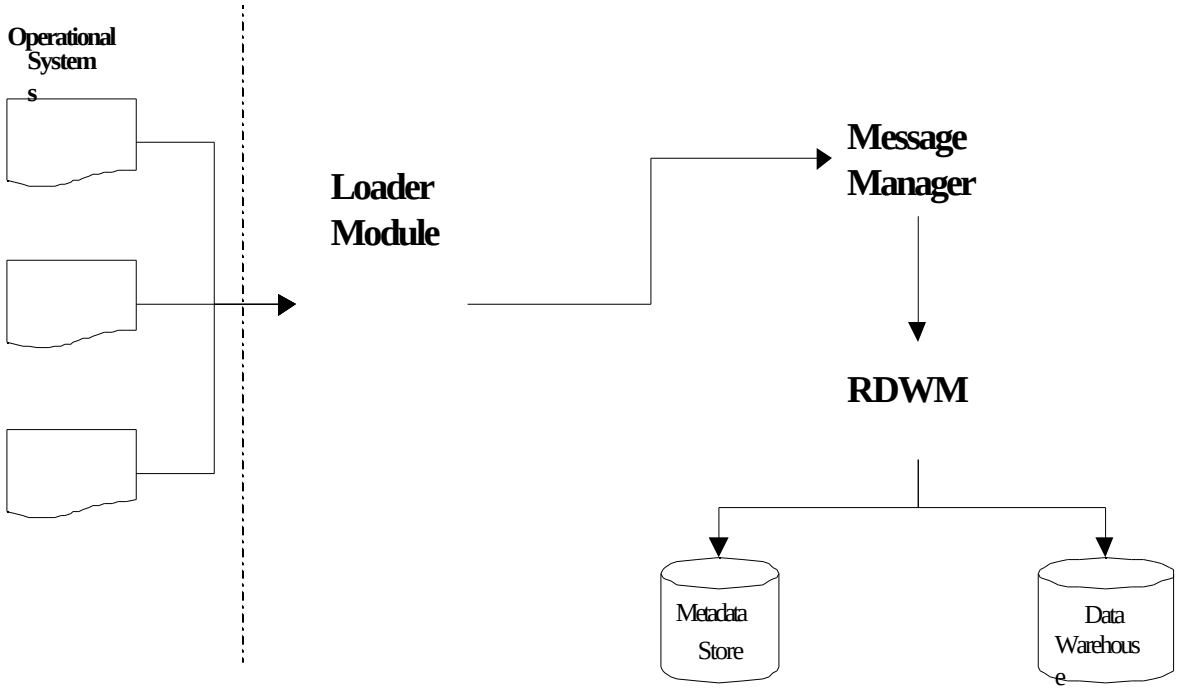

Figure 9: Data Warehouse Loader Module