Towards Hypermedia Support for Database Systems

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

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 (physical) 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 that would discover interesting relationships for that element. DHE would then map each relationship to a link.

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.

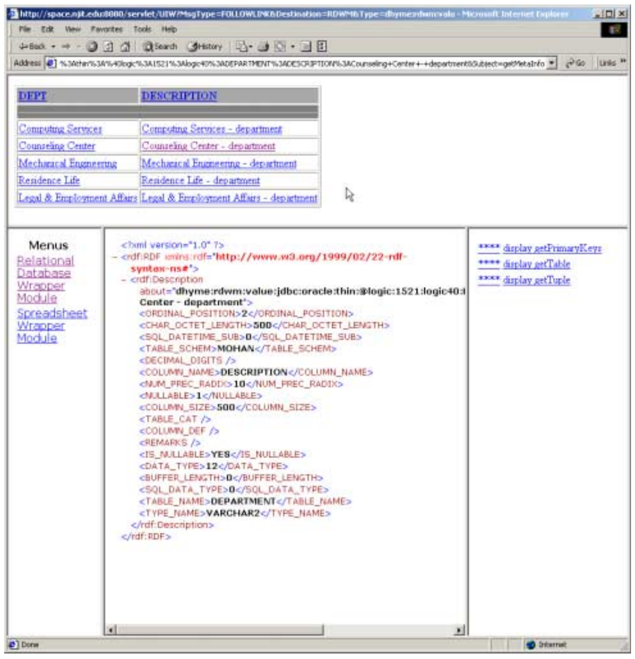
We could consider each element within a database application as a potential starting point for information exploration. Each element could have multiple links, each representing a different relationship (schema-based or otherwise). The ability to explore a piece of information in more detail could help 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.

The purpose of this paper 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. 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. We have successfully provided hypermedia support to a database application used by the New Jersey Department of Transportation and describe that as a motivating example.

This paper proceeds as follows. After introducing the Dynamic Hypermedia Engine (DHE), we describe hypermedia support for databases. First, we show how DHE can support existing relational DBMS. Then we describe how DHE uses databases internally to support hypermedia. Next we look at integrating object-oriented databases, 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, followed by a short conclusion.

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 main 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 an 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 (unnormalized) 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 RDF 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.


http://space.njit.edu:8080/servlet/UIW?MsgType=FOLLOWLINK&Destination=RDWM&Type=dhyme:rdwm:valu - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search History
Address %3Athin%3A%40logic%3A1521%3Alogic40%3ADEPARTMENT%3ADESCRIPTION%3ACounseling+Center+-+department&Subject=getMetaInfo
Go Links
Menus
Relational Database Wrapper Module
Spreadsheet Wrapper Module
Done
Internet

DEPT	DESCRIPTION
Computing Services	Computing Services - department
Counseling Center	Counseling Center - department
Mechanical Engineering	Mechanical Engineering - department
Residence Life	Residence Life - department
Legal & Employment Affairs	Legal & Employment Affairs - department

```
<?xml version="1.0" ?>
- <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
  syntax-ns#">
  - <rdf:Description
      about="dhyme:rdwm:value:jdbc:oracle:thin:@logic:1521:logic40:
      Center - department">
      <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 />
      <COLUMN_NAME>DESCRIPTION</COLUMN_NAME>
      <NUM_PREC_RADIX>10</NUM_PREC_RADIX>
      <NULLABLE>1</NULLABLE>
      <COLUMN_SIZE>500</COLUMN_SIZE>
      <TABLE_CAT />
      <COLUMN_DEF />
      <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>
```

**** display getPrimaryKeys
**** display getTable
**** display getTuple

Figure 1: Screen from the Preliminary DHE Web Prototype

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 for each. 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://space.njit.edu:8001). 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.

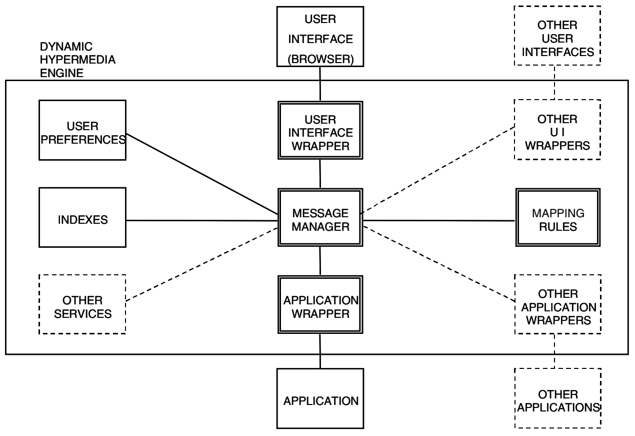


Figure 2: DHE's Logical Architecture

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_ uses mapping rules to map 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.

The current DHE prototype implements mapping rules as a record in a database table. A mapping rule record contains the type of the element, the system that owns this element type, the relationship and the system that is responsible for navigating this relationship; the target system (which maybe different from the owner system). The relationship, which is the heart of the mapping rule, is a parameterized command that can be understood by the target system, the mapping rules module sends this command to the target system, which executes it and returns the response back to the UI Wrapper. For example a mapping rule for the Freight System (see 3.3) would contain the following information:

- Element Type: `dhyme:fdwm:stcc`
- Owning System: `FDWM` (Freight Database Wrapper Module)
- Command: `getSICode`

where `getSICode` is a command (which obtains the Standard Transportation Code for a given item) that can be executed by the Freight Database Wrapper Module.

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 Functionality: 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 our applications. 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.

Several approaches exist for integrating hypermedia functionality into primarily non-hypermedia information systems (Bieber 1999). These include employing hypermedia data models, hypermedia toolkits, link services, hyperbases, hypermedia development environments, open hypermedia systems, 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 (Davis et al. 1994), Freckles (Kacmar 1993, 1995), the OO-Navigator (Garrido and Rossi 1996; Rossi et al. 1996) and SFX (Van der Stemple, 1999a,b,c).

Microcosm's Universal Viewer and Freckles 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. This approach meets our goal of supplementing Smalltalk applications with hypermedia support without altering them. Our approach 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. 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.

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 by describing a database application we have successfully provided hypermedia support to.

3.1 Integrating Database Applications

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 back to the application, which then customizes and displays them.

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 of 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 the following states.

Receiving Request Messages

A DHE 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 a user input form for display for users to enter SQL statements.

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/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.)

Relationships between Elements

DHE specifies relationships based on the element type. The twelve types of relationships noted above are each interrelated. 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.

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 physical 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.

3.2 Enhanced Links Through a 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.

The DB Schema Mapper (DSM) runs in conjunction with the Relational Database Wrapper Module. As mentioned earlier the RDWM provides metadata by examining the physical 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 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.

3.3 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 U.S. A rudimentary web interface to this database exists; we have created a DHE module that supplements this system and provides enhanced metadata, exposes the various interrelationships between the "elements of interest" in the system and provides additional functionality.

Architecture

The Freight Database Wrapper (FDW) is a DHE module that acts as the wrapper to the Freight system. Like other application wrappers described in previous sections, the FDW 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 FDW 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 FDW may also be used in conjunction with the DSM, providing users with a view of the internal schema of the Freight system's database.

4. Tightly Integrating Database Applications

Prior sections discussed integrating database applications 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. 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 a 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.

At this point the Application Wrapper may take the enriched document and translate it to the native 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 then 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 and the user preference manager, not to mention the other hypermedia functionality that all applications receive.

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. As enterprise-wide data warehouses grow in size and complexity, discovering information from these data warehouses will become complex and involved. We expect that a DHE-based data warehouse, offering hypertext functionality will ameliorate this situation, and allow data warehouse builders to concentrate on the business processes that generate and retrieve information.

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.

The DHE maybe used top 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 (through on-line analytical processing - OLAP)
- Loading the data from the operational systems into the data warehouse

Accessing the data will proceed in an analogous way to accessing regular database contents. 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
- Extract Metadata from Operational Systems
- Eliminate Noise

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.

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

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.

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

Hara and Botafogo (Hara et al 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 conceptual schema 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. (Falquet et al 1995, Falquet et al 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.

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), or mining through data generated from user activities on Web sites (the so-called "clickstream").

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

Many enterprises have investments in legacy backend systems that need to be accessed over the World Wide Web. Unfortunately this usually means either a complete retrofit of the existing system or a customized interface to the system. We believe the DHE is the first such generalized system to impart hypermedia functionality to any application without any change to it.

To integrate a legacy application with the Engine, a developer has to follow 3 quick and simple steps.

- Write one or more mapping rules
- Implement a DHE module that executes the command behind these mapping rules on the system and populates the result of this command into a DHE message
- Register the module with the Message Manager

Because of the "modular" architecture of the DHE, modules maybe added and removed at will without affecting or disrupting the rest of the Engine, which allows the developer to prototype, develop and deploy a module easily. Also, because the DHE uses XML as its internal messaging protocol the developer will find it easy to integrate with the rest of the Engine.

While the main purpose of this study 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 providing it to applications lacking it, and also eases the development of complex database systems by automatically providing these 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, and by the New Jersey Commission of Science and Technology.

References

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

Berners-Lee T., Fielding R. & 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, 1990).

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).

Chiramella, Y., and Kheirbek, A. (1996) An Integrated Model for Hypermedia and Information Retrieval, In 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, In Proceedings of Hypertext – Information Retrieval – Multimedia, (HIM '97), 9-26, Dortmund, Germany.

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

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

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

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.

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

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

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

Furner, J., Ellis, D., , Willett, P., (1996) The Representation and Comparsion of Hypertext structures using Graphs , In 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, 1996, 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", In Proceedings of Hypertext '97, 67-74, April 1997.

Hara, Y., Botafogo, R. A., Hypermedia Databases: A Specification and Formal Language, Proceeding of the Databases and Expert Systems Applications Conference (DEXA), Springer-Verlag LCNS 856, (1994), 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), 1993, 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.

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

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

Nguyen, T., Srinivasan, V. (1996). Accessing Relational Databases from the World Wide Web, Proceedings of the ACM SIGMOD Conference, 529-540.

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.

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

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

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

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

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

Van de Sompel, Herbert and Patrick Hochstenbach, (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 Patrick Hochstenbach, (1999) 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, pages 160-166.

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