In: Proc. 8th Int. Workshop on Database and Expert Systems Applications (DEXA'97), Toulouse, 1997, pp. 295-303.

Supporting Adaptable Technical Information Systems in Heterogeneous Environments

- Using WWW and ORDBMS -

Theo Härder, Henrik Loeser, Nan Zhang
University of Kaiserslautern, Department of Computer Science,
P.O. Box 3049, D-67653 Kaiserslautern, Germany
email: {haerder | loeser | zhang}@informatik.uni-kl.de

Abstract

Due to the ever rising competition in nearly all industrial sectors, there is a strong need for integrated information systems (ISs) that will store, retrieve, manipulate and exchange information efficiently to support all activities in an enterprise. Moreover, such an IS should not be a one-size-fit-all or a permanent solution; rather, the adaptability must be pursued so that the system can deal with all possible changes resulting from new technologies or new application demands. In this paper, we present our first experience with developing such an integrated IS for a technical application domain. From the data processing point of view, the environment to be managed consists of distributed client/server systems embodied by heterogeneous hardware and software components; unconventional data types such as multimedia data and HTML statements are to be managed; and effective and robust database (DB) support should be provided. To meet the challenges posed by these advanced applications, new information techniques, such as World Wide Web (WWW or Web) and object-relational database management systems (ORDBMSs), are employed to facilitate a platform-independent DB access via a versatile user interface, as well as powerful DB services with high extensibility.

1. Introduction

Although we have used DBMSs for more than 20 years in industrial domains to build up technical ISs supporting mission-critical design and development as well as production and maintenance work, we are still unable to shape truly integrated systems which can cover as many aspects of the requirement spectrum as possible. Such technical ISs have to provide adequate functions for storage and management, retrieval and update, as well as exchange and dissemination of a wide variety of information. They are necessarily exposed to distributed and to some degree autonomous client/server environments whose "natural appearance" is formed by heterogeneous hardware and software components (linked together by componentware). Indeed, while manual control procedures and semi-automated information exchange dominate its use in CAD environments, a technical IS is often not more than a conglomerate of relational or object-oriented DBs, files, text-processing documents, image/photo repositories, electronic mail systems, specialized CAD systems, etc., which is nowadays more and more extended by individual and unorganized Web usage.

To improve this situation, we started the RITA project [7] which is carried out in cooperation with a leading German manufacturer of automobile seats. Currently we are implementing a subsystem supporting the design and test of prototypical technical objects. A fundamental ingredient of the system is information management requiring effective and efficient DB support. To meet all challenges of this advanced application, new information techniques such as ORDBMSs and the ubiquitous Web should be employed to provide adequate DB services and to facilitate the platform-independent DB access.

As we know, there will be no single-step solutions towards integrated technical ISs, but only continuous improvements consisting of many comparatively small steps. On the other hand, in the competitive and ever-changing world with which every enterprise is confronted today, managing a huge amount of information in a system that is open, flexible, and extensible enough to withstand all possible changes will be essential for an enterprise to stay on the leading edge or even survive. These changes may be imposed by the emergence of new data types (e. g., the enormous interest in multimedia contents), by the new processing requirements accompanying the introduction of new data formats, by the advancement of domain-specific techniques (e. g., the adoption of new methods and equipments for seat testing), or by the development in computer hardware and software areas (e. g., upgrading to or setting in a new DB product or a new platform). Therefore, one of our main concerns by designing and implementing such an IS is the adaptability of the resulting system.

In this paper, we are focusing on the specific problems of connectivity in heterogeneous settings together with some DB support issues (such as extensibility of data types and functionality), which we feel are the core aspects in this regard. In particular, we investigate which role can be played by HTML documents, the HyperText Transfer Protocol (HTTP), and finally the huge information repository of the Web itself to improve technical ISs. We do not claim or want to prove that in such environments all DB or information retrieval/exchange problems can be solved by using the Web techniques.

In the next section, we will at first outline the various information management tasks which we face in our RITA project. Then, Section 3 concentrates on enabling DB access in a heterogeneous environment. In Section 4, the specifics of Web connection as well as data modeling issues are investigated in order to manage the dynamics in application scenarios. The potential of intelligent Web usage, that is, the automatic provision of information services supporting the work of individuals or groups, is discussed in Section 5. Finally, in Section 6, we conclude this paper and give an outlook on future work.

2. RITA - a Heterogeneous, Adaptable IS for Technical Applications

The goal of our project RITA is to build an integrated IS to handle and/or support the technical applications involved in all essential processes of the enterprise. RITA will act as an information repository that manages data and experience of earlier development projects and test suites, and helps to optimize the preparation and execution of new tests by supporting the engineers in presenting the data in a suitable manner. Another aspect is to integrate relevant information of new projects in the knowledge pool. By establishing such kind of IS, the coordination and information exchange should be supported, resulting in shorter development cycles, making the achievement of goals more secure, and reducing development costs.

An integrated IS with these tasks and objectives can be functionally decomposed in many parts. It often manages huge amounts of complex data to be shared among independent or cooperative processes, in DBs. Moreover, it should also provide ways for the user to interact with the system, as well as ways for different components of the system to communicate with one another in a decentralized heterogeneous setting. Therefore, data modeling and manipulation, workflow management and control, as well as information processing in a heterogeneous client/server environment are three main aspects of the system. Moreover, the seamless integration into an existing infrastructure of the enterprise constitutes another important endeavor.

Typical characteristics centering on the role of RITA as an information repository can be summarized as below:

a. Large volumes of data with diverse structures should be stored and managed. There is a variety of complex data types, including texts and reports, sketches of test plans, photos and videos of test progresses, etc. Moreover, data structures range from simple flat structures (part catalog) to heterogeneous compound ones (CAD objects). Finally, dynamic integration and use of newly defined data types should be allowed in the IS.

b. Workflows and decision-making have to be supported. Therefore the full functional spectrum of an industrial-strength DBMS including ad-hoc querying must be guaranteed.

c. Object-oriented approaches are necessary for coping with the complexity in developing such a technical IS. In particular, to realize most of the sophisticated application logic related to, e. g. complex object processing, native interfaces to object-oriented programming languages such as Java or C++ will be of great use.

d. An open, heterogeneous computing environment based on Client/server architectures is now increasingly popular for IS. Consequently, it is also a great challenge to bring about succinct and platform-independent DB access in such a setting, to facilitate convenient interaction with the system, to provide a single and versatile catalog of local or global information repositories together, as well as to manage users through a uniform interface and communication service.

e. A characteristic group of the client/server applications are CAD applications with advanced modeling and high bandwidth demands on DB servers. Since they have to rely on data shipping techniques, such applications require efficient client-side data management support (object caching, consistency control, etc.).

f. Most importantly, to keep up with ever rising competition and ever varying circumstances, flexible reaction to changes is essential. An integrated technical IS with long-term goals should empower users to dynamically alter data presentation, functional requirements, application logic, and other aspects w.r.t. the environment.

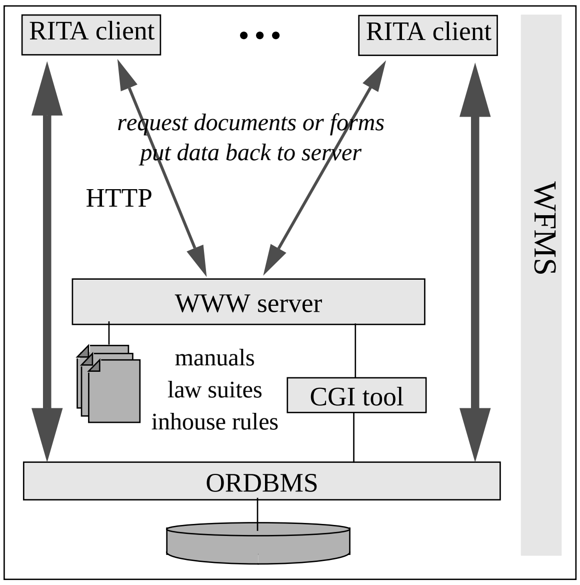


Fig. 1: The architecture of RITA

Two significant advancement trends of the information technology, which are expected to make it possible to build larger ISs with features listed above, can be attributed to the emergence of the ORDBMS technique and the WWW technique. The former promises the fulfillment of requirements *a*, *b*, and *c* through its strong modeling power, the facilities of user-defined types and functions, support for object-orientation properties without sacrificing SQL and other mission-critical DB features such as robustness. And the latter can be exploited to achieve simple DB connectivity through HTML and CGI (Common Gateway Interface). Besides, it facilitates friendly, homogeneous, and tailorable browser interfaces, efficient Web services for local and also global information catalogs (e. g., ISO, Bookmarks, Hotlinks, etc.), as well as DB-based user group management for communication and authentication services (point *d*). Although ORDBMSs incorporate advanced modeling facilities, their availability at the client side remains an open problem which cannot be dealt with in this paper (point *e*). However, ORDBMSs offer the possibility to store HTML statements in DBs as data types, hence, these two techniques can be combined together, and their own flexibility can be employed to achieve more adaptable information repositories (point *f*).

The architecture of RITA based on these techniques is briefly illustrated in Fig. 1, while more detailed discussion will be made in the subsequent sections.

3. Supporting a Heterogeneous Environment

As stated above, one of the main problems in establishing a technical IS is to provide support for a heterogeneous environment, i. e., to support different hard- and software platforms. Currently, we evaluate two different approaches for maintaining DB access w. r. t. their adequacy for our needs. The first one is the “classical” use of a programming language in conjunction with a platform-independent graphical user interface (GUI) toolkit, e. g., wxWindows [18]. The second one is to employ WWW-based DB access in order to have a platform-independent tool, i. e., the WWW browser, for data displaying. In the following we consider the pros and cons of both. After that, the ideas and the work to be done for employing the chosen technique will be described briefly.

3.1 Conventional client/server approaches

So far, client/server approaches are chosen to structure the application when DB access is involved. These approaches are typically based on traditional programming languages (e. g., C or C++) together with a platform-independent GUI toolkit and have some drawbacks concerning the implementation and maintenance of an application in a heterogeneous environment:

- A program with all its modules must be built (compiled and linked) on various platforms. That is, the graphical environment must be supported by the GUI toolkit. Moreover, libraries for DB access and communication as well as a compiler supporting the different libraries or language styles must be present for each platform. Hence, there are a lot of prerequisites to be met in order to really begin to write a program. Furthermore, a significant amount of the coding effort has to be spent for the interface part instead for the actual functionality.
- The installation and maintenance of a program is not only a time- and manpower-intensive task, but also raises the question about availability of testbeds for each environment during the development phase.
- The GUI with all its components is hard-coded into the application, therefore, is difficult to be adjusted to dynamics or changes, such as changes to the DB schema during application development.

However, the use of such conventional techniques for DB access in a heterogeneous environment has some advantages, too. These include the facility of implementing complex algorithms for the application using existing libraries, and the perhaps higher execution performance.

3.2 WWW-based techniques

For heterogeneous environments, the Web offers some rather opposite properties - accessibility and ease of use, but possibly compromising performance. Since the integration of the Java Virtual Machine (JVM) into WWW browsers, there exist two techniques for WWW-based DB access: CGI programs and Java applets.

CGI-based gateways

The CGI enables the WWW server to transfer data submitted from a WWW browser to a program residing on the server (see Fig. 2, ❶), to execute it (❷), and to transmit the resulting output (❺) back to the browser (❻). Such a program located at the server side can be a DB client which establishes a DB connection, submits a query to the DB server (❸), gets the result (❹), and converts it to an HTML page (❺) which then is transferred back to the browser (❻).

While in the past CGI programs for DB access had to be programmed in C or Perl, nowadays every DB vendor offers a special tool for this purpose. These tools are comfortable and provide a mixture of HTML and SQL, but they cannot overcome some principal shortcomings of the CGI when accessing DBs [11, 15]:

- As we have seen, six steps are needed to get a result from a DB server back to the browser. Each time a DB is accessed, a connection has to be established and access authorization has to be acquired. Since the DB server and the WWW server must be contacted for each query issued to the DB server, the WWW server is likely to become a bottleneck.
- Because of the statelessness of HTTP used for the communication between the WWW browser and the server, only single-query transactions are possible.
- In the case of failures, there may appear in-doubt situations, since the user cannot be sure whether or not the query execution was finished, e. g., connection failures may affect a query before (❷ or ❸) or after its execution (❹ to ❻). Hence, transaction properties (ACID) cannot be guaranteed even for single-statement updates [6]. Therefore, this update option may only be used if the user is willing to deal with in-doubt situations and if acceptable DB states can be reached. This means that such an approach is only tolerable for simple data structures without consistency constraints across objects or for less important data resources (without system-enforced consistency control).
- Another problem of CGI-based tools is their restricted access to only one DB server. In some cases, in conjunction with special products like IBM DataJoiner [9] or UniSQL/M [20], other DB servers can be accessed indirectly, causing further performance problems.

Nevertheless, CGI-based DB access has advantages, too:

- By using HTML forms and tables a GUI for data input and display can be developed in a straightforward way. Input forms and the resulting output pages can be created on-the-fly, providing the opportunity for dynamic reactions.
- To establish a DB frontend, a single HTML page is sufficient for

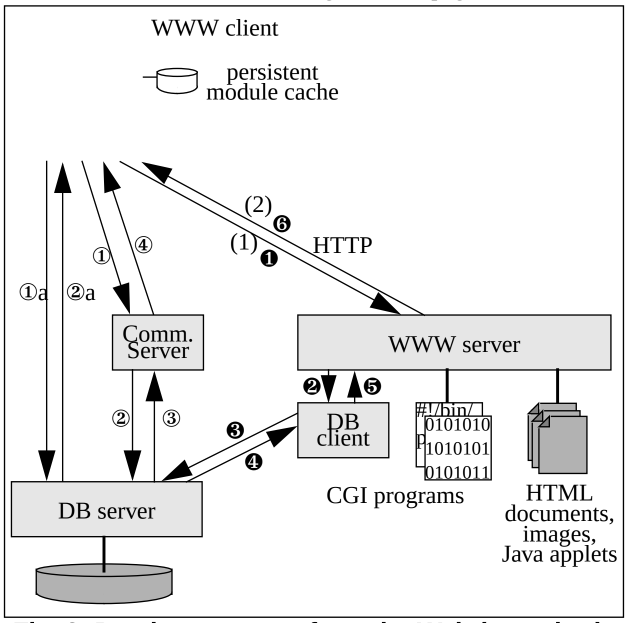


Fig. 2: Database access from the Web (overview)

the transfer to the browser. Hence, apart from a WWW browser and the server's address, there are no prerequisites to access a DB in a distributed and heterogeneous environment.
- Because of having only CGI programs and HTML pages both of which reside on the WWW server, all changes can be done in a central place, having minimal administration costs.

If low bandwidth and user-responsible consistency control is tolerable for DB applications in a heterogeneous environment, this approach saves budget and time [11]. Due to the mentioned strengths, CGI-based DB access is the right choice for low-frequency access to simply-structured, low-volume DBs, e. g., for our application inserting measurement data of tests three or four times a day or browsing data of previous tests.

Using Java

The programming language Java offers new opportunities for accessing a DB from a WWW browser. An applet, i. e., a Java program embedded in an HTML page and transferred to the browser using a normal HTTP connection ((1) and (2), see Fig. 2), is able to open a connection to a DB server residing on the WWW server's machine. A "trusted applet", an applet which is cryptographically signed by a certified institution and accepted by the user, may connect to a DB server located even anywhere. Several different approaches exist to access a DB from an applet [15]:

- The first one requires the use of the JDBC (Java DataBase Connectivity, [5]), an SQL-based call level interface (CLI) defined in the Java language. Some JDBC drivers enable the applet to directly connect to the DB server (①a and ②a), others use a special Communication Server located between the applet and the DB server (① to ④), e. g., IBM's DB2 JDBC driver.
- Another technique is based on proprietary CLIs, e. g., MsqlJava [4] for mSQL or J/OCI [21] for Oracle.
- Using an applet as a CORBA (Common Object Request Broker Architecture) client, embodies a conceivable third approach of getting data from a DB server. In this case, the Communication Server in Fig. 2 will be an ORB (Object Request Broker).

Beside some drawbacks like longer loading times of applets in comparison to normal HTML pages or the loss of HTML as an easy-to-maintain user interface, Java offers a number of advantages for accessing DBs:

- Because of the direct DB connection, it is possible to have "long" transactions, i. e., consisting of more than one query.
- The user is no longer in-doubt about the fate of DB queries due to the direct DB connection and the now possible 2PC (Two-Phase Commit).
- It is possible to support application logic at the client side, i. e., the WWW server will be relieved from certain processing tasks. For example, lists or tables may be redisplayed or reordered without contacting the server again.

Hence, Java is adequate for implementing WWW-based complex and data-intensive applications, i. e., applications incorporating workflow control, traversing a DB, or performing complex data manipulation.

3.3 Remark and outlook

Due to the weaknesses of "traditional" programming languages concerning the provision of DB access in a heterogeneous environment, WWW-based techniques are the right choice in RITA for specific DB applications relying on query shipping. Because of the different application characteristics in RITA, CGI-based as well as Java-based solutions must be employed.

For simple and low-frequent applications, e. g., for performing test data input or browsing in a requirement catalog, HTML and CGI programs are adequate. By supplying manuals and law suites as HTML documents, "ubiquitous" browsers provide a homogenous user interface to the whole IS for the dissemination and retrieval of such kinds of information. If workflow control or application logic, e. g., for complex data manipulation, are needed at the client side, Java programs can be integrated in HTML pages, and the same user interface, the WWW browser, can be used for data input and presentation.

4. Supporting Dynamically Adaptable Applications

As we have seen, both approaches accessing DBs from the WWW support a uniform user interface in a heterogeneous environment. In this section, we discuss how to provide the means for dynamic adaptation to changes in our IS through the joint facilities of WWW techniques and ORDBMSs.

4.1 Using WWW techniques

In Section 2, we have outlined the requirements for RITA, including the run-time support for new data types such as new test suites. To provide adequate tools for the dynamic creation of new data types as well as the input and display of instances of these new data types, we will exploit the WWW-based techniques presented in last section.

Creating new types

The specification of new data types by means of WWW is a non-trivial task, since the number and the types of new attributes required are usually unknown in advance. As a consequence, using only a single static HTML form is not sufficient. For this problem, we have considered two possible solutions. The first one is to create new types in an attribute-by-attribute manner using chained HTML forms. The second one employs a Java program for this purpose.

Using chained HTML forms requires either to alter the new type step by step or to pass all user input from one form to the next and then to create the new type with a single operation. Since the step-by-step connection of the WWW server and the DB server may result in dubious situations, the first alternative is unsuitable for our demands. On the other hand, the second way only needs a WWW server connection for each step, but the new attributes specified in previous steps must be integrated into the current page, i. e., a CGI tool must convert the attribute descriptions to HTML's "hidden value tag", integrating the data invisibly into HTML pages.

Employing a Java program offers opportunities such as graphically aided user interaction, and, apart from only a single DB access at the end of the creation process to insert the data, a server-independent construction of types, as well as a dynamically changeable GUI. Hence, creating a new type can be done by selecting a base type, specifying the attribute's name and clicking an "*Add*" button. If all attributes are specified, only "*Create*" must be pressed to built a new type.

For example, to create a new type for the "static backrest inclination test", we have to select "float" as base type for the different angles of inclination and the load, "image" for the pictures, and "text" for the special observations. After having specified each attribute needed, a data type for the new test suite can be created.

Data input and display

Data input and display for instances of dynamically created types can be realized using HTML forms and pages. With a fixed number of attributes, a template for HTML forms consisting of input fields for each attribute can be generated during the type creation process. In addition to this, an HTML page, extended with SQL statements, is needed to insert the data using a CGI tool. To provide data display, another generated and SQL-enriched HTML page for the CGI tool must be produced during the type creation. This page contains a selection statement and a data output section for each attribute, based on a template for each base type.

In the case of our new test suite, the data input form consists of text input fields for the angles and the load, a text area for the observations, and an input field for the path to the picture. The data display page may be composed of small units consisting of a label for and the value of an attribute for text-based types, as well as pictures embedded into the HTML page.

4.2 Using ORDBMSs

So far, we have discussed that the WWW offers a promising means for a platform-neutral and user-friendly style of DB access in heterogeneous environments, where information is distributed both throughout an enterprise (the Intranet) and all around the world (the Internet). Moreover, there is an ever-broadening spectrum of information resources, and users want a system that handles all the data types involved in the operational processes of the enterprise. As an example, the RITA system should provide more than textual descriptions of test progress and test patterns: there are photos and videos recording the course and the status change of the vehicle seats, graphics designed for seat constructions and spatial locations, as well as composite multimedia documents reporting the final test results. The Web supplies on one hand an arena to present such mix-media information, on the other hand, it requires effective DB support to manage this information. An increasing demand for new types of information and corresponding ways to manage them is posed on DBMSs.

Why choosing an ORDBMS?

Traditional relational DBMSs (RDBMSs) do not support advanced data types natively. Since they consider only character strings and numbers, data types that cannot neatly fit into tables, such as image, audio, full text, and HTML, are not understood by RDBMSs. Specific data types as well as the operations on them cannot be dynamically customized to individual application domains. As a result, although the proposed idea concerning the WWW technique can give our IS some degree of flexibility in the heterogeneous environment, the full expectation, however, cannot be met by only using RDBMSs to support the underlying data storage and management.

In contrast to RDBMSs, object-oriented DBMSs [16] exhibit much stronger modeling power which stems from a rich type system. However, supporting application-specific, advanced data types is not the sole requirement which is essential in our setting. Most of the conventional DB features (e. g., integrity constraints, fine-granule concurrency control, content-based query and query optimization, as well as robustness issues such as recovery and backup) are still critical.

To allow users to model and manipulate unconventional data effectively and to use object-oriented technologies for application development without losing the benefits of SQL and all the commercial-strength DBMS features, recently almost all of the top DB vendors have redirected the strategies to object-relational and are extending their DB server architectures accordingly. Challenges posed by adaptable ISs are expected to be met by the emerging object-relational technique [3,13,14,19] for several reasons:

- Support of a rich and extensible set of data types for technical information management, such as full-text, multimedia, temporal, and spatial data.
- The ability to enable dynamic WWW connections: The ability to define HTML as a new data type allows the DB server to natively understand, search on, and dynamically generate HTML pages from the underlying DBs directly.
- Support for user-friendly interaction and consistency control: Today, active rules have become one of the main properties of the object-relational technique. In the presence of more freedom in user interaction with the system, more attention should be paid on the consistency and integrity of stored data.
- A flexible and extensible architecture which is easily tailored according to domain-specific demands: Generally, the architecture contains two kinds of components: a core engine, which provides, among others, the ability to create new data types, new functions, and new access methods; and a series of pre-built or user-developed modules (such as DataBlades, Extenders, or Snap-Ins), which are collections of data structures with functions that manipulate them, and optionally new access methods. As a result, it is unnecessary to worry about, e. g., new data formats that emerge dynamically and continually in an enterprise and thus need to be included and maintained.

HTML pages as data type

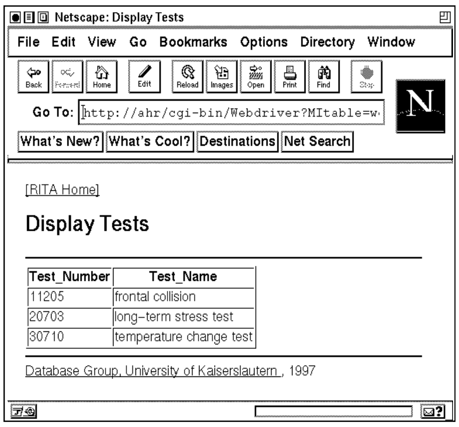
In Illustra [12], e. g., HTML pages are of the data type defined by the Web DataBlade, whose kernel is the *WebExplode* function executing inside the Illustra DB server.

This function is called through a client DB application interface — *Webdriver*. It selects an *Application Page* (an HTML page containing embedded SQL statements) stored in the DB. *WebExplode* parses the extracted pages, executes the embedded SQL queries, and formats the resulting HTML page, which is then transferred back to the client browser program by *Webdriver* through the Web server.

```
<TITLE>Display Tests</TITLE>
<A HREF=”/RITA/”>[RITA Home]</A><BR>
<H1>Display Tests</H1>
<HR NOSHADE>
<?MISQL SQL=”set schema RITA;”><?/MISQL>
<TABLE BORDER>
<TR><TH>Test_Number<TH>Test_Name</TR>
<?MISQL SQL=”select test_number, test_name
from tests where project_number=1997006;”>
<TR><TD>$1</TD><TD>$2</TD></TR>
<?/MISQL>
</TABLE>
<HR NOSHADE>
<A HREF=”http://www.uni-kl.de/AG-Haerder/”>
Database Group, University of Kaiserslautern
</A>, 1997
```

Fig. 3: An Application Page stored in DB

Due to the length limit, we can only present a simplified example. Fig. 3 illustrates an *Application Page* with the corresponding output of the Browser shown in Fig. 4.


Netscape: Display Tests
File Edit View Go Bookmarks Options Directory Window
Back Forward Home Edit Reload Images Open Print Find Stop
Go To: http://ahr/cgi-bin/Webdriver?MItable=w
What's New? What's Cool? Destinations Net Search

[RITA Home]

Display Tests

Test_Number	Test_Name
11205	frontal collision
20703	long-term stress test
30710	temperature change test

Database Group, University of Kaiserslautern , 1997

Fig. 4: Browser output

In this way, many problems inherent in previous generations of Web connections are solved, including difficulties with the management of complex data types and the need to write proprietary CGI access and management code. Moreover, since the *Application Pages* are kept in the DB, they can be easily retrieved or manipulated using the powerful functions provided by the Web DataBlade module.

4.3 Comments

Our primary goal is to implement an adaptable IS that is flexible enough to keep up with the rapid and unpredictable changing of requirements on how data are used and managed. Therefore, a DBMS that can natively support new data types, enables dynamic and interactive Web connections in all platforms, and allows users to easily custom the functionality and express specific application logic, is crucial in today’s competitive world. ORDBMSs, though without a generally accepted standard definition at the time being, embody a promising potential to build advanced ISs satisfying more complex requirements. The modeling potential as well as the shortcomings of ORDBMSs are investigated in another paper [23] based on our practical experiences with the RITA project.

It should be noted that current ORDBMSs, in their infancy, cannot yet meet all the promises and challenges. Type extensibility provides a fundamental facility to manage changes. However, a real adaptable IS is faced with more complicated situations than just to be able to define application-specific data types and functions on-the-fly. Primary requirements in technical domains include, among others, efficient support for large and complex objects. In such a context, managing changes needs extensive study of semantic-rich data relationships and genuine set-valued attributes, efficient support of schema evolution and object migration, as well as management of versions and configurations. Our work in this direction can be reflected by an ongoing project ORIENT [22] aiming at integrating relationship semantics into an ORDBMS and another project being planned to enhance ORDBMSs with mature version concepts. All these efforts rely on the extensibility inherent in ORDBMSs.

5. External WWW Usage in Technical Information Systems

Thus far, we have primarily discussed the “internal usage” of HTML and HTTP providing a platform-independent means for information representation and transfer in heterogeneous environments. However, we did not yet check the original ideas and objectives of the Web itself and how they could be exploited to improve the “external information exchange” of technical information systems.

5.1 Personalize the Web

The WWW or Internet already connects vast information resources dispersed around the world and is increasingly perceived as a single, global data repository offering all users simple means of authoring as well as uniform, simple, and (sometimes) fast access. Millions of persons, organizations, and institutions have built the Web which currently incorporates hundreds of millions of pages of various kinds of information. We strongly believe that the potential of the Web has to be tapped to broaden the spectrum of services in technical ISs. Moreover, this provision of services has to be tailored to the specific tasks and professional activities.

All pages of information just referenced by their URLs are equally proximate to a user. Hyperlinks can group “some” of the related information to enable a kind of clustering and to reduce the user’s decision space. Web browsers as cross-platform, multimedia information displays and command generators are ubiquitous. Together with these browsers they simplify somewhat search and access of information pages, however, their accessibility is completely impersonal. Users often complain that the information accessed in the Web is badly organized, irrelevant, outdated, or even wrong. On the other hand, they quickly loose track and get lost if they follow chains of hyperlinks to locate the desired information. The standard "help" of the browser software such as "Bookmarks" or "Hotlists" (to keep track of frequently used URLs for fast relocation) is quite static and will quickly get out of control as these lists grow and their entries become obsolete. A number of reasons such as slow network or connection speeds, helplessness or inability to locate information already found, insufficient means to manage and organize retrieved information [17] require more appropriate user support for accessing the Web. We believe that the provision of Web services has to be shifted from manual management to automatic approaches thereby tailoring the Web for individual users - at least in professional environments.

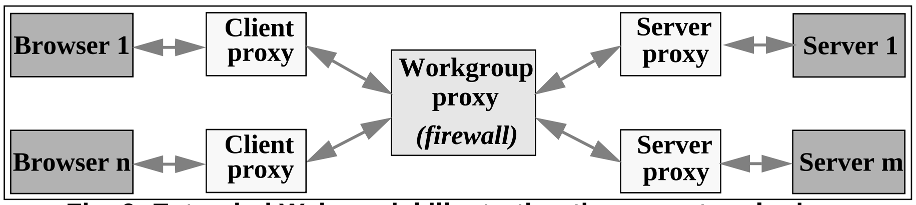


Fig. 6: Extended Web model illustrating the proxy topologies

5.2 Proxy topologies

For this purpose, the simple Web model resulting in a very tight linkage of Web browser and Web server has to be adjusted to allow for suitable user assistance. Although the fundamental communication mechanism of the Web - HTTP whose simple and direct request-response protocol between browser and URL server is stateless - cannot be changed, the Web model could essentially be improved concerning our needs by introducing programmable intermediaries between browsers and servers. A first decoupling is achieved by the use of a so-called proxy which serves as a mediator for the user's Web transactions as illustrated in Fig. 5.

In this scenario, a client issues an initial request to the proxy which performs the Web transaction on behalf of the client. Note, the proxy's response is passed back to the client. Usually, the original response is not modified, but, in principle, a proxy can produce arbitrary responses. Nowadays, a proxy is typically used to provide page caching for individual users or groups of users in an organization; on the other hand, it can incorporate a one-way firewall for intranet security. Since each request of a browser is directed to the related proxy, it can check for cached pages or the authorization of the user before further actions are taken.

This rudimentary form of control and service optimization could be greatly expanded by evolving the idea of a mediator between browser and server. Some architectural approaches for establishing such a mediator are recently proposed in the literature. [2] proposes a proxy server shell called OreO that can be used to modify the HTTP stream between a client and a Web server. The proxy server consists of a number of cooperating agents which can carry out various tasks as kind of an assistant of the user. The

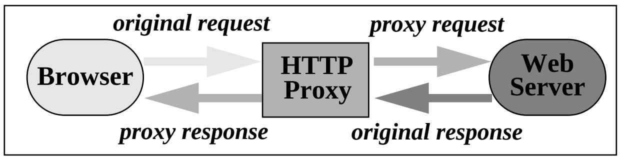


Fig. 5: Simple Web model extended by an intermediary

group of agents and their connectivity has to be configured at system start-up which makes this approach somewhat static, whereas the WBI approach (Web Browser Intelligence, [1]) can dynamically connect agents for specific tasks based on data contained in HTTP requests and responses. In our proposal, we adhere to this mediator architecture where the provision of useful Web-related services is performed by a number of cooperating agents acting between a user's Web browser and the Web.

To sketch the full potential of the mediator idea, we would like to point out the entire range of proxy topologies. As shown in Fig. 6 we can distinguish between client, workgroup, and server proxies.

The client proxy is responsible for all Web transactions of a single user, whereas a server proxy observes/handles all requests to a server. Finally, the workgroup proxy controls all server accesses of one workgroup which may be a suitable place for building a firewall. Note, not all of these proxy types may be mounted in a particular IS, but, in principle, each of these proxies can be implemented by a collaborating set of agents, as outlined above.

5.3 WBI architecture

In our special environment, we include a number of agents simplifying and tailoring the Web access to the individual needs where these agents perform on behalf and as assistants of the Web user. To replace the normal proxies by such agent-based assistance systems, we have adopted the key ideas of the WBI approach [10]. WBI consists of four types of agents:

- Monitor agents track and record user actions to provide information for other agents. For this purpose, they observe the request/response stream and perform, without altering it, actions based on the stream, e. g., monitoring page contents or deriving usage patterns.
- Editor agents intercept the communication stream, modify requests or responses, and forward their own requests/responses. Hence, they can connect to either the request part or to the response part of the stream. Examples for such modifications include inserting additional related links or information, highlighting text, modifying URLs, or adding annotations to a response page for a browser.
- Generator agents receive requests and convert them into responses by using any other resource, e. g., a local storage system, a standard form, or the Web. For example, a default service is to pass a URL request on to the appropriate Web server, to retrieve the response, and to pass it back to the browser. Likewise, a cached page is located in a local store and immediately delivered back.
- Autonomous agents react to trigger conditions (e. g., events or time intervals) independent of the usual request/response stream. Such an agent terminates after having performed its task which

could be some housekeeping actions (e. g., to detect obsolete links and find their new URLs) or some "active exploration" of the Web to find new or refreshed information.

These agents are dynamically created and linked together by a system kernel. To keep persistent data, they are using an ORDBMS to store information of HTML as well as other advanced types. Typically, a combination of agents of different type is constructed to assist given requests/responses. To enable their proper activation, each agent is registered in the kernel component with its trigger rules for activation. Such triggers can refer to times or intervals as well as to specific situations when particular servers are accessed or specific document types are received (e. g., only *.com* URLs or HTML documents, respectively). Based on such trigger rules and collaborating agents, our model of Web usage could be enhanced in the following way.

5.4 Automated index of viewed pages

Retrieval of previously-viewed pages is often frustrating, since it is hard to remember and to relocate such pages. The current use of bookmarks only allows some imperfect support, because these handles are unorganized, static, and difficult to maintain. Many situations such as obsolete links or source document updates are not properly treated. WBI agents, however, could greatly improve this situation by automating a local index capturing the personal history of accessing pages. For example, to provide easy-to-use and up-to-date information a monitor extracts text from pages accessed and records it in a DB. An editor adds an appropriate query form to the browser interface to allow for keyword search. Hence, a generator can then search the DB, access the Web, and produce the requested results. Note, the search of the personal history limits the search space to 10^3 - 10^4 pages as compared to a Web search of 10^{10} - 10^{12} pages and improves dramatically relevance and precision. Furthermore, an autonomous agent could regularly check the cached pages for timeliness or adjust URLs.

5.5 Maintenance of cached library

In technical environments, frequent references occur to all sorts of catalogs (related to parts or manufacturers) and standards (ISO, ANSI, DIN). Of course, fast access to their latest WWW editions, including all corrections and changes, is highly desirable, which could be best achieved by a local cache of this "distributed digital library". Since large volumes of data are involved, a workgroup library would be most economical.

Library retrieval and maintenance could be improved by a similar use of agents as discussed in the previous section (keyword search, detection of obsolete hyperlinks, cache refresh of updated pages). Furthermore, unknown documents which may contribute to the work of the group could be searched as some kind of automated background activity. As illustrated in a scenario in Fig. 7, pages of interest could be located by observing Web usage patterns. Monitoring (❶) the page accesses of individuals or of the group (frequency, contents) could lead to the identification of clusters of retrieved pages and to the extraction of keywords (❷). With these hints, autonomous agents could search for related documents (❸) to be added to the "cached library". To ensure the "importance and relevance", a user can confirm suggestions (❹) concerning these extensions. Finally, generator and editor agents could add links or annotations to responses ❺ to communicate amendments for updated or new information sources to the group members.

In a similar way, access to highly volatile and frequently changing information could be provided for the group. This concerns newsletter, information of professional organizations and activities as well as a calendar of events or a whiteboard.

Another important issue is the authoring and publication of the research results or other achievements of the group. Editors could provide a uniform presentation and inform about changes or replacement of requested reports. Since bibliographic and state information of research reports evolves over time (draft, submitted, accepted, published, when and where), an autonomous agent could make sure that all requests are always directed to the current version of the report.

5.6 Enhancement of convenience and performance

Obviously, Web assistants could take over a substantial share of the user's routine work. For example, electronic mail management could greatly benefit from the use of agents which can tailor the mail handling, organization, and retrieval to the needs of individuals. With the given framework, a mail DB could be easily integrated giving multi-dimensional access by time, hierarchically organized subjects, contents, etc. Likewise, advanced use of monitors and generators could also speed up the user's work as well as internal processing. For frequently used chains of hyperlinks, this access behavior could be detected and the combined use of these agents makes a shortcut available to the user. Monitoring of traffic speeds could be exploited in various ways. Internally, there are typically several transactions performed per requested Web page (HTML, graphics (GIF)). Hence, expensive, but less important transfers could be delayed or automatically switched off. Moreover, WWW access could greatly benefit from the knowledge about current traffic conditions by propagating warning signals to the user, selected use of servers, or dynamically routing

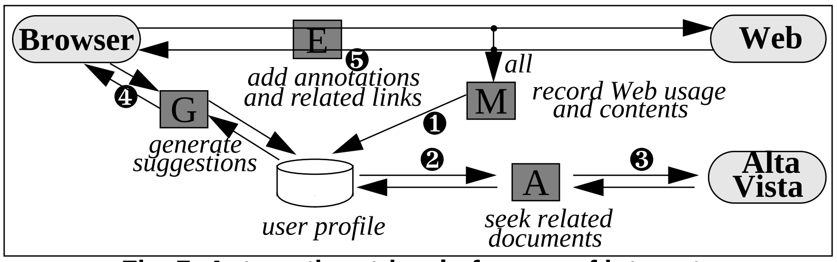


Fig. 7: Automatic retrieval of pages of interest

transactions.

6. Conclusions and Outlook

The WWW and ORDBMSs are driving forces of a tremendous change on how information is organized, stored, processed, and exchanged. They enable the construction of advanced ISs with more effective data management facilities as well as more convenient communication mechanisms both between the user and the system and also between various components in heterogeneous environments. In particular, they allow for low-bandwidth, low-budget solutions for a spectrum of DB-based services in a technical IS. The fast development times are a product of the simplicity of HTML and HTTP mechanisms and because of the fact that most heterogeneity issues are hidden even in the development process. Furthermore, the internal information services can be easily extended and enhanced by the information resources of the Web. However, its potential problems caused by information updates and unorganized growth should be fixed by automated agents controlling changes, growth, and newly created information resources while assisting the individual users or workgroups. We have taken advantage of this potential by developing an integrated IS for technical applications, with explicit focus on adaptability and flexibility concerning changes of the application requirements and the environment.

Moreover, the Web (and its mechanisms) is not a panacea for providing or mediating DB services in heterogeneous environments, but its potential for transparent and platform-independent information exchange may help a lot to alleviate our heterogeneity-induced problems in technical ISs, when applied appropriately. For example, handling of highly structured objects consisting of checkout/checkin, data transport and DB integration problems are not solvable by simple cross-platform Web techniques [8]. While DB-integrated client-side data processing remains an open problem area, we expect major improvements for the server/client connectivity, data transport across platforms, as well as DB access in heterogeneous environments and more intelligent use of the information resources of the Web.

Although these issues are beyond the limits we can explore in this paper, further investigation and practice should surely be done to make full use of the new technologies and also to achieve more satisfactory results.

References

[1] Barrett, R., Maglio, P. P., Kellem, D. C.: *How to Personalize the Web*, in: Proc. CHI'97, New York, ACM Press, 1997.

[2] Brooks, C., Mazer, M. S., Meeks, S., Miller, J.: *Application-specific Proxy Servers as HTTP Stream Transducers*, in: Proc. 4th Int. World Wide Web Conference, 1995.

[3] Chamberlin, D.: *Using the New DB2: IBM's Object-Relational Database System*, Morgan Kaufmann, 1996.

[4] Collins, D.: *An MsqlJava Tutorial*, University of Queensland, http://www.minmet.uq.oz.au/msqljava/tutorial.html, 1996.

[5] Hamilton, G., Cattell, R.: *JDBC: A Java SQL API*, Version 1.10, SUN Microsystems Computer Company, Oct. 1996.

[6] Härder, T., Reuter, A.: *Principles of Transaction Oriented Database Recovery*, in: ACM Comp. Surveys 15: 4, 1983, 287-317.

[7] Härder, T., Thomas, J.: *RITA — ein rechnergestütztes Informationssystem für technische Anwendungen* (in German), ITG-Fachbericht 137 (STAK'96), Munich, March 1996, 111-126.

[8] Hardwick, M., Spooner, D. L., Rando, T., Morris, K. C.: *Sharing Manufacturing Information in Virtual Enterprises*, in: Comm. of the ACM 39:2, 1996, 46-54.

[9] IBM: *DataJoiner: A Multidatabase Server - Version 1*, Whitepaper, 1995, http://www.software.ibm.com/data/pubs/papers/

[10] IBM: *Web Browser Intelligence - Agent Software*, http://www.networking.ibm.com/wbi/wbisoft.htm.

[11] IBM: *Net.Data Programming Guide*, http://www.software.ibm.com/data/net.data/docs, Feb.1997.

[12] *Illustra User's Guide (Release 3.2)*, Illustra Information Technologies, Inc., 1995.

[13] ISO/IEC CD 9075 Committee Draft, Database Language SQL, Jim Melton (ed.), July 1996.

[14] Kim, W.: *Object-Relational — The Unification of Object and Relational Database Technology*, Whitepaper, UniSQL Inc., Austin, 1996.

[15] Loeser, H.: *Datenbankanbindung an das WWW - Techniken, Tools und Trends* (in German), in: Proceedings of GI-Fachtagung "Datenbanken in Büro, Technik und Wissenschaft" (BTW'97), Informatik aktuell, Ulm, March 1997.

[16] Manola, F. (eds.): *Object Model Features Matrix*, ANSI X3H7-93-007v10, Feb. 1995.

[17] Pitkow, J. E., Kehoe, C. M.: *Emerging Trends in the WWW User Population*, in: Comm. of the ACM 39:6, 1996, 106-108.

[18] Smart, J.: *User Manual for wxWindows 1.65: a portable C++ GUI toolkit*, Artificial Intelligence Applications Institute, University of Edinburgh, Aug. 1995.

[19] Stonebraker, M.: *Object-Relational DBMSs — The Next Great Wave*, Morgen Kaufmann Publ., Inc., 1996.

[20] *UniSQL Server Multidatabase Support User's Guide*, UniSQL, Inc., 1996.

[21] *J/OCI Gateway - Product Information*, Vincent Engineering, http://www.vincent.se/Products/JOCIGateway/JOCIGateway.html, 1996.

[22] Zhang, N., Härder, T., Thomas, J.: *Enriching Object-Relational Databases with Relationship Semantics*, in: Proc. 3rd Int. Workshop on Next Generation Information Technologies and Systems (NGITS'97), Israel, June 1997.

[23] Zhang, N., Härder, T.: *On Modeling Power of Object-Relational Data Models in Technical Applications*, in: Proc. 1st East-European Symposium on Advances in Databases and Information Systems (ADBIS'97), St. Petersburg, Sept. 1997.