accepted for WISE 2000, Hongkong, June 2000

Shift it to the Server!
- Let the Database Server Update Your Web Sites -

Henrik Loeser

University of Kaiserslautern, Department of Computer Science,
P.O. Box 3049, D-67653 Kaiserslautern, Germany
e-mail: loeser@informatik.uni-kl.de

Abstract

From the beginnings of the World Wide Web, Web site administrators have used dynamically generated HTML pages to provide up-to-date information, e.g., online news, stock quotes, etc. Due to the high resource consumption of dynamic page generation approaches, many sites have switched over to periodical updates of frequently visited pages, e.g., a headline index of an electronic newspaper. However, this approach has led to reduced topicality of information. In this paper, we present iWebDB, an approach for automatic ORDBMS-controlled document updates. It offers up-to-date Web documents and, at the same time, avoids the drawbacks of dynamic page generation approaches. iWebDB is based on the extensibility infrastructure of object-relational database systems. By utilizing DB triggers and so-called user-defined functions, documents can be automatically generated by the DBMS whenever related DB data has been modified.

1. Introduction

From the beginnings of the World Wide Web (WWW or Web) and the definition of the Common Gateway Interface (CGI), Web site administrators have used dynamically generated HTML pages to provide up-to-date information, e.g., online news, stock quotes, etc. In recent years, DBMSs were often used to manage the information to be included in these pages as well as the page templates. Several techniques for Web-based DB access exist, e.g., CGI, Web server extension interfaces, or servlet API [2]. While there are differences regarding the number of necessary components in the runtime environment, all possible solutions consume time and system resources for page generation leading to enhanced response times and increased hardware demands for the whole system. This extra overhead is especially critical when these tasks become hot spots in the overall processing chain.

Therefore, many Web sites have switched over to periodical updates of frequently visited pages. But this has led to a reduced topicality of accessible information, i.e., data embedded in Web pages can be inconsistent compared to the more recent state of the DB. To provide both topicality as well as lower response times and reduced resource consumption, a page caching approach is introduced in [1]. Dynamically generated documents are cached for further requests. In addition, using update notification, pages can be generated after data updates and directly written into the cache. However, this approach has been designed for a hot spot Web site. It is based on a complex runtime environment which definitely means an overkill for the requirements of most sites.

In this paper, we present a special extension of iWebDB, our DB-based solution for Web document management [7, 8]. We call this extension iWebDB/DG which provides a document generator framework for iWebDB. Both iWebDB/DG and iWebDB are based on the extensibility mechanism of object-relational database systems (ORDBMSs) and DBMS features standardized for the first time in SQL:1999 [5]. Using triggers to keep track of data updates and user-defined functions (UDFs) for dependency management and document generation, all functionality is integrated into a standard DBMS. Apart from the ORDBMS no further components are required. Thus, the DB server can keep Web pages up-to-date on its own.

The remainder of the paper is organized as follows. First, we discuss approaches for the maintenance of DB-based Web pages and their common problems. Then, in Section 3, we introduce iWebDB and its components. In Section 4, we discuss the iWebDB approach which automatically updates Web pages by using the DB server. Section 5 outlines the iWebDB implementation and briefly discusses first performance tests. In Section 6, we discuss the related work. Finally, we conclude and give an outlook on our future work.

2. Common Problems

Today, most large Web sites have been transformed from a collection of static HTML documents to DB-based sites. Documents are created from templates based on input selected from one or more DBs. Document creation is periodically performed after updates of templates resp. document components, or after updates of related DBs. Moreover, documents can be generated on-the-fly, possibly based on form input or tailored to a specific user. Therefore, Web sites not only consist of a Web server accessing the file system and delivering HTML pages to the Web client, but also of many additional components, e.g., DBMSs, application servers, etc. In addition, macro files, document components, and static HTML pages are often managed by a Web Content Management System (WCMS) and stored in a special document repository before being published on the Web server.

As already mentioned, many approaches and techniques exist to deliver topical documents to the user, all having specific merits. For the maintenance of a Web page with a news ticker, at present, two general approaches exist. One technique is to dynamically generate such a page on user request, i.e., each time a user tries to access this news ticker page a CGI script or something similar is invoked to retrieve all actual headlines out of a DB and to compose an HTML document based on this search result. The second approach is to place a static page in the Web server file system and to update this document periodically. However, frequently accessed pages, such as documents containing a site index or overview pages, are often neither created dynamically nor generated script-based by the administrator. Instead, they are edited manually and, as a consequence, are normally outdated.

2.1. Dynamic Page Creation

Depending on the expected access rates of a Web document, several techniques for dynamic page creation can be employed. They can be divided into solutions utilizing an application server, and programs or modules incorporating all needed functionality and accessing the DB directly, e.g., CGI programs, server extension modules, etc. While solutions based on an application server scale better because the load can be balanced over multiple computers, the other solutions only utilize resources when getting invoked.

However, whatever solution is employed, each has to access macro files and DBs resp. at runtime, to process these files, and to generate the document. Therefore, user requests to the Web server are answered with delay. Moreover, many resources are needed to provide a scalable system. Furthermore, runtime access to DBMSs and application servers may potentially threaten security which requires additional efforts to secure the systems.

2.2. Periodic Updates

Another approach for DB-based Web pages is to generate the documents periodically, e.g., once each hour controlled by a timed service, or manually by a Web administrator. After being invoked, the generator application reads and processes the templates and publishes the resulting documents on the Web server. All programs and components run isolated from the Web server in an own environment. Thus, neither programs nor modules on the Web server are required. Furthermore, direct connections from the Web server to a DBMS are avoided. Write access to the Web server file system is sufficient to publish the documents.

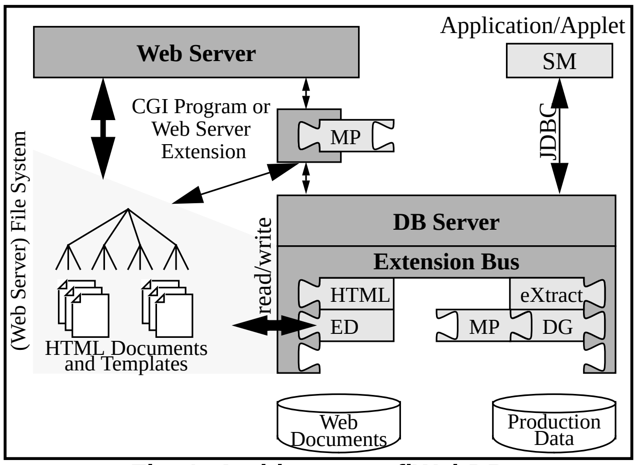

Fig. 1: Architecture ofiWebDB

However, concerning the news ticker example, important information updates may not be reflected in the Web page because the page generation has not been invoked. In contrast, page creation may be superfluous because DB updates have not occurred in the maintenance interval.

Both approaches presented as well as manual Web page control have drawbacks. What is needed is an approach providing both up-to-date documents combined with low resource requirements. Later on, we will show that our document generator, iWebDB/DG, is the solution for a large spectrum of situations.

3. iWebDB - An Overview

In the following, we give an overview of iWebDB, where we briefly describe each component. iWebDB/ED and iWebDB/MP, the components iWebDB/DG is directly based on, are presented in more detail.

As depicted in Fig. 1, iWebDB currently consists of six modules, four of them being DBMS extensions. In addition, a client application, SM, exists. The module MP is a function library that can be used for client-side as well as for server-side applications.

- ***iWebDB/Doc*** provides all base types and tables for Web Content Management (WCM), e.g., types for storing XML, HTML, Postscript, and other text documents as well as images of different formats. Moreover, corresponding functionality for managing and querying the content is offered. iWebDB/Doc is the base component of iWebDB.
- ***iWebDB/ED*** *(external data)* helps to simplify Web site administration by allowing the integration of externally stored files into iWebDB. File system data accessible via Web protocols, such as “file:”, “ftp:”, and “http:”, can be made available in relational tables via abstract tables [6], the file content in document formats provided by iWebDB/Doc. Available data can be queried using SQL. Thus, external data can seamlessly be integrated into the system.

- ***iWebDB/eXtract*** offers additional functions for analyzing documents to enhance the search capabilities provided by iWebDB/Doc. Furthermore, functionality is provided for storing extracted data in a special index structure being exploited by a search engine.
- ***iWebDB/SM*** *(site manager)* is the iWebDB tool for site management provided to all so-called information providers. Based on a graphical user interface (GUI), all administration tasks can easily be performed.
- ***iWebDB/MP*** *(macro processor)* is a function library and designed to be the foundation for building gateway programs for Web-based DB access. It offers an extensible macro processor. Macros can be embedded tags into Web documents. The library can be used to realize a fully functional client- or server-side Web database gateway (see Fig. 1). In addition, it serves as a basis for iWebDB/DG.

To complete the overview, *iWebDB/DG* is the *Document Generator* (DG) itself and is based on the iWebDB/MP library.

4. iWebDB Approach

Having outlined the common problems and solutions for providing actual documents in WISs (Web Information Systems), in the following, we introduce the iWebDB approach for automatically refreshed Web pages. First, we present the basic concepts and discuss their pros and cons. Then, we develop extensions to optimize the solutions and to allow adaptations to different requirements and, as a consequence, different environments. Finally, we discuss techniques for the error handling in DBMS-internal generation processes.

4.1. Basic Concepts

The iWebDB approach is based on the extensibility of ORDBMS, in part standardized in SQL:1999 [5]. In contrast to other solutions, all components are integrated into the ORDBMS. Using SQL triggers, iWebDB can keep track of data modifications. If an update, insert, or delete event occurs, the document generator is being invoked. As a first step, the set of configuration files affected by the data changes is computed. iWebDB does not use direct dependencies between tables and document templates, but utilizes special configuration files to set up the environment and to determine the templates to be expanded.

After the set of configuration files has been calculated, each of them is evaluated by the macro processor. In a configuration file, environment variables for template processing can be set, conditions can be checked, and template processing can be switched on or off. After the configuration file has been processed, this switch is being checked. If it is turned on, the template is being loaded and the processing environment set. Then, the macro processor is invoked again, now for the template. Finally, the document writer is called to store the resulting document at its destination. When all components are integrated into the DB server, only two system components are needed to provide up-to-date documents: the Web server in the deployment environment and the DB server in the development environment. Thus, both environments are kept as simple as possible.

4.2. Dependency Management

As briefly mentioned in the previous section, instead of directly calling the generator for a specific document, the document generator is called by the trigger with the table name as an argument. Thus, only one trigger for a table/operation pair is required and the number of triggers is kept low. However, SQL:1999 offers two different trigger modes: FOR EACH STATEMENT and FOR EACH ROW. Examining Web applications, one can see that both modes can be useful. While a list of workshop participants only needs to be updated after all changes have been performed, a homepage should be created for each new user. Therefore, supporting both modes by the dependency management can reduce the number of document updates.

While for small Web sites dependencies between tables and documents resp. configuration files can be easily specified and a simple dependency management is adequate, huge Web sites need a more sophisticated dependency model [10]. To support both environments and to be adjustable to new requirements, iWebDB can be configured to use tailored dependency checkers. Therefore, for each Web site an adequate dependency manager can be employed

4.3. Macro Processors

The document generator utilizes iWebDB/MP to evaluate the configuration files and to expand the templates to HTML pages. While iWebDB/MP provides basic features, e.g., SQL statements, flow control, etc., it lacks more sophisticated functionality which may be required for some Web sites. Furthermore, Web pages that up to now are dynamically generated are based on other tools. To provide employment of more powerful macro processors as well as to support migration from so far dynamically generated pages to ORDBMS-based page creation without template modifications, iWebDB/DG offers a special generator API and advanced dependency information. That is, other macro processors can be registered with iWebDB and used for document generation. For each document, the required processor can be specified, such that different macro processors can be utilized during a document generator invocation.

4.4. Document Output

After a document has been generated, it is written to its destination. However, this can be the (locally) accessible file system of the Web server, a remote file system accessible via FTP or HTTP. Furthermore, a content provider can be responsible for pages on multiple Web servers, e.g., a service to provide an event calendar to both a local and a regional portal. As one can see, different techniques for writing the generator output to a document are required. Additionally, this set of techniques is required in a single generator run.

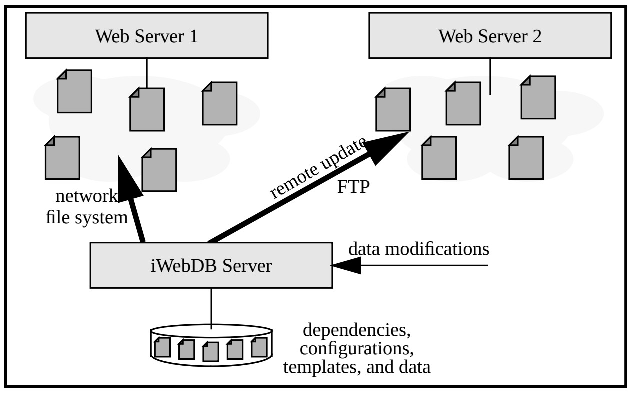

Fig. 2: iWebDB-controlled document generation

Analogously to the macro processor API, iWebDB/DG offers its so-called document writer API to support different implementations resp. techniques. To achieve this flexibility, a support module has to be registered with iWebDB/DG. Then the document destination can be selected, e.g., the regional event calendar can be published using FTP and special access rights.

4.5. Document Index

In addition to the generation of up-to-date documents, content-based search in the documents is an important requirement. Using iWebDB/ED, a document index over the Web server file system can be created. Whenever a document maintained by iWebDB is published or a Web page generated by iWebDB/DG is written to the Web server via iWebDB/ED, e.g., by inserting in or updating a table “webdocs”, the document index is being updated automatically by the ORDBMS. Using this index, up-to-date and searchable documents can be provided to the WIS users.

However, the “webdocs” table can be used to trigger the document generator as well. For instance, if a new document is being published to the Web server via “webdocs”, the new document can be included into the site map that is automatically being updated by iWebDB/DG.

4.6. Error Handling

Using iWebDB/DG, all components are integrated into the ORDBMS, pages are automatically refreshed by the DB server. However, if a Web site is updated automatically and if several internal and external modules resp. services have to interact, various types of errors may occur:

- *Syntax errors during macro processing*: Macros embedded in a template may be wrong and processing may be aborted.
- *SQL resp. DBMS exceptions*: The syntax of SQL statements may be wrong, or problems during the query processing may occur, e.g., the DB server may fail.
- *Network/file system errors*: Access to templates stored in a file system or publishing of the generated document may fail caused by, e.g., a full file system, a broken network connection, or changed and therefore wrong permissions.
- Several other situations can cause errors during document generation, e.g., a server crash.

Using a dynamical page creation approach, macro processing could be interrupted, an error message be produced and presented to the WIS user. However, in the iWebDB approach, the document generator is being invoked during an updating transactions. Interrupting the generator would cause the transaction to be aborted. If an error message is being produced and written instead of the document, the content of the outdated but accessible document would be overwritten; the document would have no useful content for the WIS user. Therefore, iWebDB/DG interrupts the document generation and catches the exception. Then, employing an UDF, an e-mail containing the error message is being sent to the administrators and the generator invocation is being terminated normally. Thus, the updating transaction can continue and is able to commit.

Employing UDFs for e-mail notification and trying to give a detailed error report, the WIS administrator is able to track the problem down and, then, to manually trigger the document generator. Meanwhile, in contrast to dynamic page generation approaches, the outdated document is accessible by the WIS user.

5. Implementation and Performance Aspects

The macro processor iWebDB/MP and the document generator itself, iWebDB/DG, have been implemented using Java. The macro processor as well as the document generator have been realized as user-defined functions (based on SQLJ Part 1 [11]) and have been integrated into an Informix Dynamic Server 2000 [3]. To facilitate installation of both modules, they have been packaged to so-called DataBlades, i.e., all UDFs, UDTs, and auxiliary tables are created resp. dropped as a whole package. However, our primary implementation goal was to build a running system in a short time frame. Therefore, optimizations and embellishing features have been left out. However, besides iWebDB/MP the Informix Web DataBlade [4] can be used for page generation as well.

To get a feeling for page generation duration, we have tested different configuration files, templates, and both macro processors. Having SQL statements embedded in both the configuration file and the template, as a first result, we detected that the page size only has a small impact on page generation times. Reading configuration files and templates over the NFS (network file system) and writing the generated documents to the local file system took up to 1 sec. per generated document on a small, non-optimized DB server using iWebDB/MP. Most time was spent for the macro processor and its Java routines. Employing the Web

DataBlade, times decreased to 0.2 sec. However, based on this number, some thousand page updates per day can be accomplished by a small installation which is sufficient for medium-sized WISs.

6. Related Work

iWebDB uses an event-based pre-generation technique to refresh Web pages after modifications to DB data have been performed. In the approach presented in [1], a cache for dynamically generated documents is used to shorten response times and to reduce runtime resource consumption. Based on a central trigger table, comparable with an event queue, pages are generated before they are requested and written to the cache. The system has been designed for a real hot spot WIS (Olympic Winter Games) with very frequent updates, and all components are in the run-time environment. Due to the complex system architecture, it is an overkill for small and medium-sized WISs. iWebDB only needs the ORDBMS itself to accomplish all document updates.

The TIScover approach described in [9] uses a pre-generation technique as well. In contrast to iWebDB, all components are realized as OS processes. Thus, administrators have to maintain a pool of continuously running services. While iWebDB/DG updates a document immediately after a related modification has occurred, TIScover checks the event queue on a periodical basis. Thus, delays between data and page updates are possible.

Due to the on-the-fly generation of Web pages, most related applications try to reduce page delivery times by caching DB connections or by maintaining a DB session pool. Thus, the costs and delays of establishing connections can be avoided. A further technique is to cache page components or whole pages, Moreover, 3-tier system architectures with an application server pool are introduced for load balancing reasons. However, despite substantial efforts to shorten page delivery times, fetching a document which resides in the file system is much faster, and, therefore pre-generation approaches like iWebDB/DG, too.

In [10], a powerful algorithm for the management and detection of update dependencies is being presented. Both page generation and the removal are being supported. In this paper, we have concentrated on ORDBMS-controlled page updates and the system architecture; a specific dependency model has not been discussed. However, by employing a sophisticated dependency model, iWebDB is also capable of removing outdated documents from a WIS.

7. Conclusions

In this paper, we have presented the iWebDB approach for automatically refreshing Web pages. In contrast to other systems, all necessary components have been integrated in an ORDBMS. The ORDBMS automatically updates Web pages whenever data modifications occur. Using a pre-generation approach, response times for documents requested by a WIS user can be kept low while at the same time resource consumption on the Web server machine can be reduced. While iWebDB exploits the extensibility infrastructure of ORDBMS, the document generator iWebDB/DG itself runs on all major ORDBMS and most standard platforms. Only support for triggers and Java-based user-defined functions is required.

iWebDB was designed to refresh DB-based Web pages when related DB data has been modified and, at the same time, to avoid the drawbacks of dynamic page generation approaches. Therefore, it should not be compared with hot spot solutions like [1]. However, a small iWebDB/DG installation is able to not only manage a small or medium WIS, but also to update documents for a pool of related WIS. Due to its open system architecture, iWebDB/DG can be adapted to different requirements, e.g., by employing tailored communication protocols, or macro processors.

More information about iWebDB as well as an iWebDB/DG demonstration are available online at *http://www.ordbms.de/iWebDB/*. You have the chance to join the iWebDB/DG-generated guestbook...

References

[1] Challenger, J., Iyengar, A., Dantzig, P.: *A Scalable System for Consistently Caching Dynamic Web*, Proc. of the IEEE Infocom’99 Conference, New York, March 1999.

[2] Florescu, D., Levy, A., Mendelzon, A.: *Database Techniques for the World Wide Web: A Survey*, ACM SIGMOD Record, Vol. 27, No. 3, September 1998.

[3] *Getting Started with Informix Dynamic Server with Universal Data Option, Version 9.20*, Informix, Inc., 1999.

[4] *Informix Web DataBlade Module, User’s Guide*, Informix Software Inc., http://www.informix.com/answers/, 1999.

[5] ISO/IEC FDIS 9075-1: *Information Technology - Database Language SQL - Part 1: Framework*, ISO, 1999.

[6] ISO/IEC JTC1/SC32: *Information Technology - Database Language SQL - Part 9: Management of External Data*, ISO, 1999.

[7] Loeser, H.: *iWebDB - An integrated Web Database on basis of object-relational database technology*, (in German), in: Proc. of BTW’99, Freiburg, Germany, March 1999.

[8] Loeser, H., Ritter, N.: *iWebDB - Integrated Web Content Management based on Object-Relational Database Technology*, in: Proc. Int. Database Engineering and Applications Symposium (IDEAS’99), Montreal, Canada, August 1999

[9] Pröll, B., Retschitzegger, W., Sighart, H., Starck, H.: *Ready for Prime Time - Pre-Generation of Web Pages in TIScover*, Proc. of the Workshop on the Web and Databases (WebDB’99), 1999.

[10] Sindoni, G.: *Incremental Maintenance of Hypertext Views*, Proc. of the Workshop on the Web and Databases (WebDB’98), 1998.

[11] *SQLJ: SQL Routines using the Java Programming Language*, http://www.sqlj.org, June 18, 1999.