Int. Database Engineering and Applications Symposium (IDEAS 2000), Yokohama, Japan, September 2000

Keeping Web Pages Up-To-Date With SQL:1999

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 (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. 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 provoking less user acceptance. In this paper, we present iWebDB/DG, the document generator of our integrated Web Content Management System. It guarantees up-to-date Web documents without on-the-fly generation. The approach is based on the extensibility infrastructure of object-relational database systems. By utilizing DB triggers and so-called user-defined functions, documents can automatically be generated by the DBMS whenever data is updated.

1. Introduction

From the beginnings of the World Wide 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, i.e., dynamic page generation, exist: CGI, Web server extension interfaces like NSAPI, ISAPI, or servlet API, etc. [2]. While there are differences regarding resource consumption, all possible solutions consume time for macro processing, query execution and page generation. Therefore, many Web sites resp. Web Information Systems (WISs) have switched over to periodical updates of frequently visited pages, e.g., a headline index of an electronic newspaper, or a site index. But this has led to a reduced topicality of accessible information as well as a large set of update scripts for generating the documents.

Our key objective for Web page maintenance is to provide information actuality and to reduce the high resource consumption at the same time. iWebDB/DG is designed to overcome both the traditional periodical update and, in part, the on-the-fly page generation approach. Based on DB triggers, iWebDB/DG keeps track of data updates. By exploiting the extensibility infrastructure of object-relational database systems (ORDBMSs, [7, 13]) and based on its configurable document generator integrated into the DBMS, iWebDB/DG is capable to keep Web pages up-to-date. Extensibility allows for user-defined types (UDTs), user-defined functions (UDFs) written in a 3GL like C or Java, and user-defined access methods that enable to implement and use own index structures as well as so-called abstract tables [6] supporting access to external data, e.g., other DBMSs or file systems. Some of these features, i.e., UDTs and UDFs, as well as triggers are defined in the SQL:1999 standard [5]. iWebDB/DG is one component of iWebDB [8, 9], our Web Content Management System which is considered to be a framework that can be adapted to a specific environment. While iWebDB exploits the full extensibility infrastructure, it is sufficient for iWebDB/DG, in principle, to employ the functionality included in SQL:1999. Therefore, using SQL:1999 and a standard ORDBMS, Web pages can be kept consistent w.r.t. the current "state of the world" without the need for on-the-fly generation.

In this paper, we first give a short overview of iWebDB. In Section 3, we present iWebDB/DG, the document generator of iWebDB, discuss the underlying concepts and its functionality, give examples, and show iWebDB/DG-based solutions for some relevant Web-related problems. In Section 4, we give implementation details and discuss results of first performance tests. Then, in Section 5, we compare iWebDB/DG with related work.

2. iWebDB - an overview

In the following, we give a short overview of iWebDB [8, 9]. As depicted in Fig. 1, iWebDB currently consists of six modules, four of them being DBMS extensions, so-called DataBlades (Informix), Extenders (IBM), or Catridges (Oracle). iWebDB/Doc provides the base functionality (types, tables, and functions) for Web Content Management (WCM). Based on abstract tables[1] iWebDB/ED (external data) integrates external data sources. Available data can be queried using SQL. iWebDB/eXtract offers enhanced

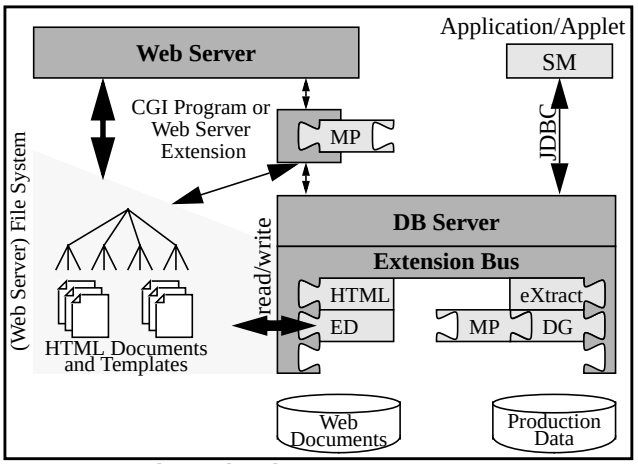

Fig. 1: iWebDB Components

search functionality and special index structures. iWebDB/SM (site manager) is a GUI-based tool for site management provided to all so-called information providers. iWebDB/MP is a macro processor library and designed to be the foundation for building gateway programs for Web-based DB access. It offers an extensible macro processor for embedding special 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, the document generator. It is detailed in the following section.

3. Updating Web documents

Having introduced iWebDB so far, we now present iWebDB/DG, the document generator itself. We discuss the iWebDB/DG concept and the application areas the module is designed for. Then, we give an overview of its functionality. Later on, we briefly discuss some examples on how it can be used for speeding up Web document access.

3.1. Overview

Instead of generating Web pages on user request or periodically, iWebDB/DG uses the concept of DB triggers and the extensibility of ORDBMSs to pre-generate documents after data modifications. In the case that a page is composed of data stored within a single DB table, an update/insert/delete trigger can be created on the table to invoke the document generator that has been registered as a user-defined function, e.g.:

```
CREATE TRIGGER docGenerate
AFTER INSERT ON news REFERENCING NEW AS n
(EXECUTE PROCEDURE DocumentGenerator
('INSERT','AFTER','news',n.id));
```

While this is a very simple scenario, later on we show how iWebDB/DG can be used for documents that are composed of data stored in several tables. Moreover, further mechanisms controlling page generation, such as conditions and invocation counters, are introduced.

After getting invoked, first, the document generator (see Fig. 2) determines the affected configuration files (see "dependency checker"). Configuration files are used to set up the generation environment, i.e., to set file and path information of the page to be generated, to load configuration parameters etc. After a configuration file has been loaded ("file loader"), it is processed by the macro processor. Then, based on the current settings, a document template is loaded ("template loader"), processed, and the resulting output is written to its destination.

```
BEGIN document generator (trigger type,
      table name, row id)
   CALL dependency checker
      RETURNS list of affected conf. files
   FOR EACH affected configuration file
      CALL configuration file loader
      RETURNS configuration file
      CALL configuration file processor
      RETURNS settings,templ. name,doc. name
      IF generate = on
         BEGIN IF
            CALL template loader
            RETURNS document template
            CALL macro processor
            RETURNS document
            CALL document writer
         END IF
   END FOREACH
END DOCUMENT GENERATOR
```

Fig. 2: Document Generator Algorithm

Since the DG creates pages without Web user interaction, the generator can only be parametrized by DB data. Therefore, generated documents cannot be customized to a special Web user and cannot react to form input, e.g., search data. But macro files can be parametrized to generate data-dependent output in several formats. For example, using iWebDB/DG, a homepage for every new staff member can be written to a file "name.html" where name is the new member's name. Homepage layout may depend on the availability of a picture, an address, group membership, etc.

iWebDB/DG's application area covers all kind of documents without time dependencies and without specific user input such as form data or user identification. Thus, apart from Web applications, most Web pages are covered.

3.2. Functionality

After having described the basic concept of iWebDB/DG, we now explain its functionality by discussing the most relevant aspects.

Integrated document generator: iWebDB/DG has been integrated into the DBMS as a user-defined function.

1. While there are ongoing standardization efforts for SQL/MED [6] to define abstract tables, the Informix Dynamic Server [3] offers virtual tables.

An alternative would be to realize an external generator application to be invoked by the triggers. However, as a consequence, this external application would consume additional resources and require more maintenance efforts. In comparison, both approaches (UDF and external application) require an extensible DBMS to integrate the document generator resp. a caller function invoking the generator. While an external application can be invoked from caller functions of different (extensible) DBMSs, the approach of designing the generator as a UDF means that each DBMS may incorporate its own document generator, or triggers may be used to call a centralized generator either directly or via JDBC. If iWebDB/DG is realized as a UDF of a single DBMS, things are kept simple and the resource consumption is low.

File storage: Serving as a framework to be adapted to specific environments, iWebDB has to support different requirements. While some users store macro and configuration files in the file system, in other environments these files are managed by a DBMS. Furthermore, a Web server can request the HTML pages from the file system or a DB. Thus, the DG has to support file-based as well as DB-based storage and retrieval. Support for both storage options can be achieved by offering macros (tags) tailored to the specific requirements for each option. But this would complicate both the creation and maintenance of configuration files and the DG component itself. Moreover, additional macros may be required to adapt the DG to other environments.

To cope with different requirements, iWebDB/DG can be extended by user-defined document loaders (see "file/ template loader" in Fig. 2), i.e., Web site administrators can employ their own document loader adapted to the specific environment. Furthermore, using the built-in SQL-based document loader, not only files stored within the DB can be accessed, but also templates stored in the local file system can be loaded by utilizing iWebDB/ED.

Analogous to the file loaders, the document writer can be adapted to the specific environment. While the built-in writer is based on SQL, users can employ their own document writer, e.g., to store the previously generated documents on a remote machine via ftp.

Dependency management: As briefly mentioned in the conceptual overview, the document generator is called by a trigger expecting the table name as a parameter. Thus, only a single trigger for a table/operation pair is required and the number of triggers is kept low. SQL:1999 offers two different trigger modes: FOR EACH STATEMENT and FOR EACH ROW. Regarding Web applications, 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 user. Therefore, supporting both modes by the dependency management can reduce the number of document updates. If the document generator is called by a "row trigger", the id of the changed row is passed on to it, and, e.g., only a specific person's homepage is updated.

While for small Web sites dependencies between tables and documents resp. configuration files can be easily specified and a simple dependency management, e.g., with directly specified table/document dependencies, is adequate, huge Web sites need a more sophisticated dependency model, e.g. [11]. To support both environments and to be adjustable to new requirements, iWebDB/DG can be configured to use tailored dependency checkers. Therefore, an adequate dependency manager can be employed for each Web site.

Condition-based generation: If one trigger would be employed for each configuration/template pair, conditions could be directly specified in the WHEN clause of the trigger. Because the dependency checks are performed inside the document generator itself, conditions cannot be specified within the trigger. However, document generation may depend on the evaluation of conditions. Thus, iWebDB/DG has to facilitate the evaluation of conditions within the configuration file. Fortunately, by processing configuration files with iWebDB/MP, all offered macros can be utilized. Therefore, not only conditions can be processed, but also additional configuration files can be included, SQL statements can be processed, etc.

Normally as a default, document generation is performed after the configuration file is processed. Using the trigger's WHEN clause, document generation could be disabled by applying a specific condition. By evaluating the conditions during configuration file processing, iWebDB/DG has to offer an option to disable the page generation within the configuration file as well.

To realize special conditions, such as generation "on only every fifth invocation" or "a minimum interval of 10 minutes between two page updates", iWebDB/DG provides special tables to centrally store and access such information.

Bulk generation: If iWebDB/Doc is used for document management, i.e., to store templates or document fragments, iWebDB/DG can be employed to update all affected documents after modifications have been performed. However, depending on the employed model, specifying dependencies for each template resp. fragment is not very useful and may be costly. Therefore, generation of multiple documents controlled by a single configuration file can be performed.

While output files and templates can be specified several times within a configuration file, so far, the generator is invoked at most once after processing the configuration file has been terminated. Thus, in order to specify the DG invocation within the configuration file and to provide this call with the corresponding context, e.g., within a loop that re-

trieves template names and output file names from a DB, a special macro is provided.

Another issue with bulk generation is the data-dependent adaptation of document templates. While in some environments a homogeneous layout is required, others prefer an individual style. To adapt configuration files and templates to the data, the INCLUDE tag can be utilized for composing such files on-the-fly in a data-dependent fashion. Thus, users can provide their own files to be included, or product-specific layouts can be applied. Moreover, rules for document naming or other common components can be stored in separate files, keeping configuration files simple and enhancing maintainability.

Error handling: Today, the availability and the quality of a (professional) Web site are very important and have direct impact on the user acceptance. However, if a Web site is updated automatically and if several components have to interact, various types of errors can occur. Furthermore, the DG is executed within the DBMS without any (administrator) interaction. Therefore, iWebDB/DG has to provide at least a facility to report errors in order to enable monitoring. However, in the iWebDB approach, the document generator is invoked within an updating transaction. Interrupting the generator would cause the transaction to be aborted. If an error message is produced and written instead of the document, the content of the outdated but accessible document would be overwritten; the document would not have any useful content for the Web user. To prevent such situations, iWebDB/DG offers configuration parameters to specify a script to be executed in the error case. Thus, error handling can be centralized and, e.g., administrators can be notified by e-mail or a log file can be written.

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

3.3. Examples

After having presented the concepts and the functionality of iWebDB/DG, we want to give application examples and to discuss iWebDB/DG-based solutions for two common Web-related problems. The latter can be categorized as follows:

- *listing*: Information is listed in a single document. The page may give an overview to a specific topic and links to further documents, e.g., a list of all products, of all (student/staff) homepages, etc., or the document may only contain a pre-specified share. For instance, the latest 30 headlines, all first level directories/links, all items of a specific category, etc.
- *description*: The document gives more or less detailed information to a specific element, e.g., a product, a person, a project, etc.

iWebDB/DG supports both categories, where documents may be composed of input from several DB tables. As an example for a Web page listing, our “news ticker” solution is illustrated in Fig. 3. Based on counter, timestamp, and conditions, page generation may be delayed for a given time. Only if prioritized headlines occur, the document is generated directly. Based on condition evaluation and the INCLUDE tag, user-specific components for the configuration file and the template may be inserted. According to this example, the page layout for product descriptions may be adapted depending on the product category.

```
Configuration File:
<VAR NAME=”DG.OUTNAME”>news.html</VAR>
<VAR NAME=”DG.OUTPATH”>/</VAR>
<VAR NAME=”DG.OUTTABLE”>webDocsFS</VAR>
<IF COND=”($DG.COUNTER > 5)&&($ENV.pri < 3)”>
   <VAR NAME=”DG.GENERATE”>false</VAR>
</IF>
Document Template:
<HTML>...
<SQL STMT=”SELECT FIRST 30 id,headline,tstamp
   FROM news ORDER BY tstamp DESC;”>
   <A HREF=”http://.../n?id= <PRINT
   NAME=”SQL.1” />”><PRINT NAME=”SQL.2”/></A>
</SQL>
...
<SQL STMT=”SELECT title,url FROM services
   WHERE service<>’news’;”>
   <A HREF=”<PRINT NAME=”SQL.2” />”>
   <PRINT NAME=”SQL.1”/></A>
</SQL>
...</HTML>
```

Fig. 3: News Ticker Example

4. Implementation and performance aspects

In the following, we will discuss some implementation aspects of iWebDB/DG and present results of first performance tests. As the other components, the generator has been integrated into the Informix Dynamic Server [3] as a so-called DataBlade.

iWebDB/DG is implemented as a Java-UDF on top of the Java-based macroprocessor library iWebDB/MP. As mentioned in the conceptual overview, most functionality can be derived from the MP and the ED modules of iWebDB. Both the generator and the check function are implemented as static methods of the same class, DG, and conform to the upcoming SQLJ standard, part 1 [12].

Performance results: Our primary implementation goal was to build a running macro processor resp. document generator within a short time frame. Therefore, optimizations

and embellishing features have been left. Different tests with SQL statements embedded in both configuration file and template have been performed. Reading the input files 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. Most time was spent for the macro processor and its Java routines. However, based on this number, some thousand page updates per day can be accomplished by a small installation. Therefore, a small iWebDB/DG installation should be adequate for managing medium-sized WISs or a pool of small WISs.

5. Related work

iWebDB/DG is intended to replace the periodic update approach and partly the dynamic Web page generation approach. It increases the overall Web server performance and improves response times. Because this is a very hot topic for all Web sites, there are many other approaches with the same objective. However, to the best of our knowledge, there is no approach comparable with iWebDB/DG. The Informix Web DataBlade [4] only integrates a macro processor as a (C-based) UDF into the DBMS to be used for dynamic page generation. The TIScover approach described in [10] uses a pre-generation technique as well. In contrast to iWebDB, all components are realized as operating system 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 may occur.

Due to the on-the-fly generation of HTML documents, 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 entire pages [1], e.g., images, static headers and footers, etc. Moreover, 3-tier system architectures with an application server pool have been proposed for load balancing reasons. However, despite substantial efforts to shorten page delivery times, fetching a document which resides in the file system, as performed in iWebDB/DG, should always be faster.

6. Conclusions

In this paper, we have presented iWebDB/DG, a document generator introducing a new approach for Web page creation: Web page generation controlled by *and* integrated a the DBMS. Instead of generating pages on-the-fly on user request or periodically by scripts, Web documents are created after changes to the DBMS (pre-generation). Thus, in contrast, up-to-date Web documents can be provided without the need for a complex system composed of an application server and its components, or by maintaining a large set of scripts resp. services for periodical page creation. Furthermore, in comparison to on-the-fly generation, page delivery times can be significantly reduced because only the local file system is accessed. Since ORDBMSs are getting more mature and become the database management standard, the approach introduced by iWebDB/DG is the way of maintaining up-to-date Web documents without the need for a complex Web server environment and high system requirements. Furthermore, triggers and Java-based UDFs required for the basic iWebDB/DG functionality are already supported by most commercial (O)RDBMSs and are included in the SQL:1999 standard. Therefore, using standard DB technology, Web pages can be kept up-to-date.

More information about iWebDB as well as a demonstration of iWebDB/DG are available online at *http://www.ordbms.de/iWebDB/*

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*, Version 9.20, Informix, Inc., 1999.

[4] *Informix Web DataBlade Module*, 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] Kim, W.: *Object-Relational - The unification of object and relational database technology*, UniSQL White Paper, 1996.

[8] 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.

[9] Loeser, H., Ritter, N.: *iWebDB - Integrated Web Content Management based on Object-Relational Database Technology*, in: Proc. of IDEAS'99, Montreal, Canada, August 1999.

[10] 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.

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

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

[13] Stonebraker, M.: *Object-Relational DBMSs - The Next Great Wave*, Morgan Kaufman, 1996.