Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1999 Proceedings

Americas Conference on Information Systems (AMCIS)

December 1999

Transaction Support for DataWeb Applications - A Requirement's Perspective

Gerti Kappel
University of Linz, Austria

S. Rausch-Schott
University of Linz, Austria

Werner Retschitzegger
University of Linz, Austria

Transaction Support for DataWeb Applications - A Requirement's Perspective

G. Kappel, S. Rausch-Schott, W. Retschitzegger

Institute of Applied Computer Science, Department of Information Systems (IFS)
Altenbergerstraße 69, A-4040 Linz, University of Linz, AUSTRIA
Tel: +43-732-2468-883; Fax: +43-732-2468-9308
email: {gerti, stefan, werner}@ifs.uni-linz.ac.at

Abstract

Web-based information systems no longer aim at purely providing read-only access to information in terms of static HTML pages. Rather, more and more web-based information systems store (part of) the information within a database system (DBS) and generate HTML pages on demand. Such information systems are often called *DataWeb Applications*. Different users interact with the system, and often they perform changes concurrently. As an example in the area of *electronic commerce*, consider a web-based tourism information system, where an unpredictable number of a priori unknown tourists are allowed to book various tourism facilities at the same time. Whereas financial and security issues of such electronic commerce transactions are extensively discussed in literature, very few work exists on the database transaction aspect in terms of *consistency* and *reliability* of electronic commerce transactions and of DataWeb applications in general. The objective of this paper is to discuss the specific requirements that different architectures of DataWeb applications and the web itself pose on transaction management, and to identify promising technologies for enabling web transaction services.

Architectures of DataWeb Applications

There exist several architectural alternatives for the integration of databases into a web infrastructure. They are briefly outlined in the following since they heavily influence the possibilities for transaction support (Ehmayer 1997). In the first approach, database access is provided directly at the *client-side* by means of external viewers, browser plug-ins, and more recently Java applets. This actually corresponds to traditional client-server applications, the browser being a direct client of the database and communicating by means of some proprietary database protocol. Consequently, existing transaction technology can be applied straightforwardly to enforce consistency and reliability.

In the second so-called *server-side approach* the browser communicates as usual in a web-based environment via the standardized communication protocol HTTP with the web server, which in turn is responsible for connecting to a DBS. This could be done by different means, ranging from Server Side Includes (SSI) to scripts embedded in HTML fetching data from the database to server-side application logic like CGI scripts or Java servlets, that produce HTML files. Despite the fact that the server-side approach complicates transaction support, it is widely accepted due to its seamless integration with the web philosophy. Furthermore, in contrast to the client-side approach, which is appropriate primarily for small Intranet applications because of the need to transfer code over the net, the server-side approach is more suitable for large and complex applications within the Internet. Thus, in the following we are concentrating on the challenges and issues of transactions supporting the server-side approach.

Transactional Requirements of the Web

When looking at the transactional requirements of DataWeb applications one can distinguish between *domain-specific requirements* and *web-specific requirements*. The first category originates *from the domain* where the DataWeb application is situated and has already been discussed in literature in terms of (extended) transaction models for standard and non-standard client-server applications (Ramamritham 1997). It is therefore not further considered. Web-specific requirements, on the contrary, stem from the specifics of the web and of the architecture used for realizing the DataWeb application. They are discussed in the following along with the problems that emerge.

In general, DataWeb applications based on a server-side approach demand for transactions that *span one or more web pages of a single or of several web servers* whereby transaction borders should be allowed to be specified *statically* or *dynamically* (Apers 1997). As illustrated in Figure 1, this leads to three orthogonal dimensions of requirements.

Transactions Within a Single Site

The first and simplest category of DataWeb applications requires transactions that are executed completely at a single site within a single web page, i.e., according to a single request, like the simple booking of a certain hotel room. However, transactions of the underlying DBS used to provide consistency and reliability for this scenario incorporate only the web server and the database server, but *not* the initiating client. Thus, in case of any failure at the client side or between the client and the web server, consistency and reliability are not guaranteed. Up to now, there are only few attempts towards *incorporating clients into transactions* (Little 1997).

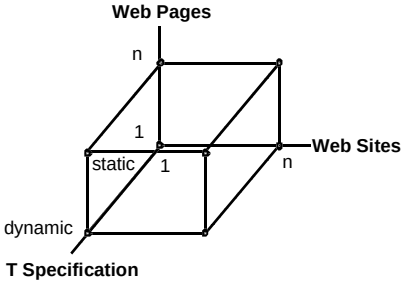

Fig. 1. Dimensions of Transactional Requirements

As soon as more than one web page located at a single site should participate within a transaction, the *transaction's context* has to be maintained somehow. This is the case in a typical shopping cart scenario, where, e.g., a holiday package is assembled out of different products each represented by another web page (Billard 1998, Pröll 1998). Since HTTP does not provide stateful communication to preserve the context, current realizations circumvent this problem by means of hidden variables, cookies or URL extensions (Ehmayer 1997).

Another problem is that, due to the *navigational nature of the web*, clients may access documents in an unexpected order. This means that, the user can hardly be enforced to request pages in a sequence required by the specified transaction. Rather, at any time the user may request a completely unrelated page just by means of its URL. The question now is what happens with a running transaction. Is it aborted? Or is it suspended and if yes how long?

A further problem for transactions spanning several web pages is the *dynamic evolution of the web*. The transaction mechanisms should be open and flexible enough, in that new web pages which are incorporated into the web site can be made part of existing transactions or of new transactions on the fly, i.e., without additional programming efforts.

Transactions Spanning Arbitrary Sites

In the most complex case, a transaction is required to span arbitrary web sites and the databases underneath. In our electronic commerce example, this would be the case if the products which are assembled to form holiday packages are not located at a single web server but rather offered by different web servers. Again, two cases may be distinguished. First, the *configuration of web servers* which are allowed to participate in the transaction could be *predefined*. This would be especially applicable for electronic shopping malls where the participating shops are known in advance. Second and more visionary, it would be even conceivable that the user *navigates to arbitrary web servers* offering products and some runtime engine checks whether the web server is capable of participating in the (global) transaction or not.

In addition to the problems discussed in the previous categories, major challenges for transaction support for this kind of applications are the *autonomy* and the *heterogeneity* characteristics of web servers. Concerning autonomy, each web server is responsible for maintaining a number of web pages and handling requests for these pages by its own, i.e., without coordinating itself with other web servers. The question arises, how to achieve proper transaction support in order to, e.g., avoid global deadlocks (Kramer 1997) while preserving the autonomy of the web servers. Considering heterogeneity, no assumptions can be made which form of transaction mechanism, if any, is supported by a certain web site. Moreover, there exist also other kinds of heterogeneity influencing the transaction mechanism, e.g., the underlying data models.

Transaction Borders

Concerning the *static specification of transaction borders*, as soon as a dedicated web page is requested, the transaction is started and remains active until a web page specified to commit or to abort the transaction is requested. With the *dynamic approach*, the transaction borders are not hard-wired into web pages. Rather, appropriate application logic decides at runtime when to start or terminate a transaction. While the static approach is simpler than the dynamic one, it lacks the flexibility of the latter. In the dynamic approach for instance, a transaction can be started or terminated depending on the runtime context in terms of the user issuing the request or the actual navigation path leading to the requested page. As a consequence, however, in contrast to the static approach, transaction borders are not known before runtime. This in turn prohibits employing conservative locking in order to prevent deadlocks.

Outlook

As investigated in this paper, DataWeb applications pose a lot of requirements on transaction processing and introduce many open problems. Our overall intention is to identify existing technologies which could satisfy the requirements on consistency and reliability for server-side DataWeb application architectures and to reveal deficiencies in their transactional support for the web. Transaction technologies that are likely to be suitable in terms of the discussed requirements are for example:

- *Extended Transaction Models*, especially the *nested transaction model* and its *variants* (Kappel 1996, Ramamritham 1997), as well as *transaction frameworks* like *JPernLite* allowing to customize the transaction model according to the domain-specific requirements of a DataWeb application (Yang 1999)
- *Distributed Database Technology*, primarily the two-phase commit protocol and concurrency protocols developed for *federated database systems* (Kramer 1997) and *multidatabase systems* (Bouguettaya 1998), including standards like *X/Open DTP* (X/Open 1993)
- *Middleware Technology* like *TP-Monitors*, OMGs *CORBA Object Transaction Service (OTS)* and Microsofts *Transaction Server (MTS)* (Cobb 1997, Kapsammer 1998)
- *Commercial approaches to transactional support for the Web* such as *WebTransactions* (Siemens 1999), *Web Transaction Server (WebTS)* (Unisys 1999), *WebSpeed Transaction Server* (Progress 1999)
- *Web-based Workflow Management Technology* (Kappel 1998)
- *Electronic Commerce Frameworks* such as IBMs *CommercePoint*, Microsofts *Internet Commerce Framework*, Sun/Javasofts *Java Electronic Commerce Framework* and CommerceNets *Eco System* (Tenenbaum 97)
- *Extensions to HTTP*, like *TIP (Transaction Internet Protocol)* (Evans 1999)

On the long run, however, not only DataWeb applications should be enabled with proper transaction support but also those applications comprising web sites without any database support behind.

References

Apers P.M.G., "*Web-Based Information Systems*", Tutorial at the EDBT Summer School '97, Italy, Sept. 1997.

Billard D., "*Multipurpose Internet Shopping Basket*", in Proc of the 9th Int. Workshop on Database and Expert Systems Applications (DEXA'98), August 1998.

Bouguettaya A., Benatallah B. and Edmond D., "*Reflective Data Sharing in Managing Internet Databases*", in IEEE Int. Conf. on Distributed Systems, May 1998.

Cobb E.E., *"The impact of object technology on commercial transaction processing"*, in VLDB Journal, 6(3), 1997.

Ehmayer G., Kappel G., Reich S., "*Connecting Databases to the Web - A Taxonomy of Gateways"*, in Proc. of the 8th Int. Conf. on Database and Expert Systems Applications (DEXA 97), France, Springer LNCS 1308, Sept. 1997.

Evans K., Klein J., Lyon J., "*Transaction Internet Protocol"*, Internet Engineering Task Force (IETF), http://www.ietf.org/html.charters/tip-charter.html, 1999.

Kappel G., Rausch-Schott S., Retschitzegger W., Sakkinen M., "*A Transaction Model For Handling Composite Events"*, in Proc. of the 3rd Int. Workshop of the Moscow ACM SIGMOD Chapt. on Advances in Databases and Information Systems (ADBIS'96), Moscow, Sept. 1996.

Kappel G., Retschitzegger W., Schröder B., *"Enabling Technologies for Electronic Commerce"*, in Proc. of the XV. IFIP World Computer Congress, Vienna/Austria and Budapest/Hungary, Aug/Sept. 1998.

Kapsammer E., Retschitzegger W., Wagner R., "*Meta Data-Based Middleware for Integrating Information Systems: A Case Study"*, in Proc. of the 4th Int. Conf. on Information Systems Analysis and Synthesis (ISAS'98), Orlando, July, 1998.

Kramer R., *"Databases on the Web: Technologies for Federation Architectures and Case Studies"*, in ACM SIGMOD Record, 26(2), June 1997.

Little M.C., Shrivastava S.K., Caughey S.J., Ingham D.B., "*Constructing Reliable Web Applications Using Atomic Actions"*, in Proc. of the 6th Int. WWW Conf., CA, April 1997.

Progress Software, *WebSpeed Transaction Server*, http://www.progress.com/internet/webspeed/, 1999.

Pröll B., Retschitzegger W., Wagner R., *"TIScover - A Tourism Information System Based on Extranet and Intranet Technology"*, in Proc. of the 4th Americas Conf. on Information Systems (AIS'98), MD, Aug. 1998.

Ramamritham K., Chrysanthis P.K., *"Executive Briefing: Advances in Concurrency Control and Transaction Processing"*, IEEE Computer Society, Los Alamitos, CA, 1997.

Siemens, *WebTransactions*, http://bs2www.mch.sni.de/webtrans/manual_e.htm, 1999.

Tenenbaum J.M., Chowdhry T.S., Hughes K., *"Eco System: An Internet Commerce Architecture"*, IEEE Computer, 30(5), May 1997.

Unisys, *Web Transaction Server (WebTS)*, http://www.marketplace.unisys.com/clearpath/software/internet/wts/index.html, 1999.

X/Open company, *"X/Open Guide: Distributed Transaction Processing: Reference Model"*, Version 2, 1993.

Yang J.J., Kaiser G.E., "*JPernLite: An Extensible Transaction Server for the World Wide Web*", to appear in: IEEE Transactions on Knowledge and Data Engineering, 1999.