A Three-Level Architecture for Distributed Web Information Systems

Markus Kirchberg[1], Klaus-Dieter Schewe[1],
Bernhard Thalheim[2], and Richard Wang[1]

[1] Massey University, Information Science Research Centre,
Private Bag 11 222, Palmerston North, New Zealand
{M.Kirchberg|K.D.Schewe|R.B.Wang}@massey.ac.nz
[2] Christian Albrechts University Kiel,
Institute of Computer Science and Applied Mathematics,
Olshausenstrasse 40, 24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de

Abstract. In this article we present a three-level architecture for distributed web information systems (WISs) based on media objects. We distinguish between a service level, a database level and an operational level. The media objects serve as a mediator between the service requested by users and the database. The operational level is based on communicating agents that realise the functionality of a distributed database.

1 Introduction

Media objects [1] support tight integration between a global data and operations level and a local interface and dialogue level in WISs. Roughly speaking a media object abstracts from the content and functionality of a web-page in a WIS including the navigation links. Using classification abstraction we define media types the instances of which are the media objects. The core of the definition is formed by a view on some underlying database schema. This view is extended by operations and by features that allow media objects to be tailored to different user needs as well as limitations arising from different end-devices or communication channels. We present a brief overview of media types in Section 2.

As media types provide a formal framework for WIS, the challenge in WIS engineering is to develop adequate system support for the storage and maintenance of media objects. In this article we address this problem and link media objects with a physical architecture, in which media objects can always be constructed and personalised at a server receiving a request from a user. In other words, the media objects will act as mediators between the service level that is available to the WIS users and a physical database level.

In case of database distribution we obtain a further separation between a global database level and an operational level, which will become available locally at each node of a network. Using this idea we exploit the well known approach of two-stack machines to realise this operational level. The major addition is

N. Koch, P. Fraternali, and M. Wirsing (Eds.): ICWE 2004, LNCS 3140, pp. 137–141, 2004.

to extend the two-stack machines in such a way that distribution, generalised remote procedure calls, parallelism and communication are supported. In Section 3 we introduce the three-level architecture for distributed WIS.

2 Media Types

A *view* V on a database schema $\mathcal{S}$ consists of a view schema $\mathcal{S}_V$ and a defining query q_V, which transforms databases over $\mathcal{S}$ into databases over $\mathcal{S}_V$.

The defining query may be expressed in any suitable query language, e.g. query algebra, logic or an SQL-variant. For our purposes, however, this is yet not sufficient, since in all these cases the query result will be a set of values. One key concept that is missing in the views is the one of *link*.

In order to introduce links, we must allow some kind of "objectification" of values in the query language. This means to transform a set $\{v_1, \ldots, v_m\}$ of values into a set $\{(u_1, v_1), \ldots, (u_m, v_m)\}$ of pairs with new created URLs u_i of type *URL*. In the same way we may objectify lists, i.e. transform a list $[v_1, \ldots, v_m]$ of values into a list $[(u_1, v_1), \ldots, (u_m, v_m)]$ of pairs. We shall talk of *query languages with create-facility*. This leads to the definition of *raw media type*.

A *raw media type* has a name M and consists of a content data type $cont(M)$ with the extension that the place of a base type may be occupied by a pair $\ell : M'$ with a label ℓ and the name M' of a raw media type, and a defining query q_M with create-facility such that $(\{t_M\}, q_M)$ defines a view. Here t_M is the type arising from $cont(M)$ by substitution of *URL* for all pairs $\ell : M'$.

Finite sets $\mathcal{C}$ of raw media types define *content schemata*. Then a database $\mathcal{D}$ over the underlying database schema $\mathcal{S}$ and the defining queries determine finite sets $\mathcal{D}(M)$ of pairs (u, v) with URLs u and values v of type t_M for each $M \in \mathcal{C}$. We use the notion *pre-site* for the extension of $\mathcal{D}$ to $\mathcal{C}$. The pair (u, v) is called a *raw media object* in the pre-site $\mathcal{D}$.

Raw media objects are not yet sufficient for information service modelling. In order to allow the information content to be tailored to specific user needs and presentation restrictions, we must extend raw media types.

Cohesion introduces a controlled form of information loss. Formally, we define a partial order $\leq$ on content data types, which extends subtyping [1]. If $cont(M)$ is the content data type of a raw media type M and $sup(cont(M))$ is the set of all content expressions exp with $cont(M) \leq exp$, then a partial order $\preceq_M$ on $sup(cont(M))$ extending the order $\leq$ on content expressions is called an *cohesion order*. Clearly, $cont(M)$ is minimal with respect to $\preceq_M$. Small elements in $sup(cont(M))$ with respect to $\preceq_M$ define information to be kept together, if possible.

Another possibility to tailor the information content of raw media types is to consider dimension hierarchies as in OLAP systems. Flattening of dimensions results in information growth, its converse in information loss.

For a raw media type M let $\bar{H}(M)$ be the set of all raw media types arising from M by applying a sequence of flat-operations or their converses to raw media

types or underlying database types. A *set of hierarchical versions* of M is a finite subset $H(M)$ of $\bar{H}(M)$ with $M \in H(M)$. Each cohesion order $\preceq_M$ on M induces an cohesion order $\preceq_{M'}$ on each element $M' \in H(M)$.

A *media type* is a raw media type M together with an cohesion order $\preceq_M$ and a set of hierarchical versions $H(M)$. A *media schema* is defined in the same way as a content schema replacing raw media types by media types. A database $\mathcal{D}$ over the underlying database schema $\mathcal{S}$ extends to a unique pre-site. Furthermore, we may extend $\mathcal{D}$ to all hierarchical versions $M' \in H(M)$ and all $M'' \succ_{M'} M'$ defined by the cohesion orders. This wide extension of $\mathcal{D}$ will be called a *site*.

Finally, we collect all media types $M_1, \ldots, M_k$ together with their hierarchical versions and types defined by the cohesion order such that $(u, v_i) \in \mathcal{D}(M_i)$ holds for a given URL u. The pair $(u, (M_1 : v_1, \ldots, M_k : v_k))$ will be called a *media object* in the site $\mathcal{D}$.

3 The Physical Architecture

Let us now look at the architecture that will support WIS based on media-objects. This architecture is illustrated in Figure 1 and we will refer to this figure to explain details of the architecture.

On the top we have the *service level* that is formed by the provision of tailored media objects to various servers. The middle level is the *database level*, which is realised by a distributed database system. The lowest level is the *operational level*. The media objects are defined by views on the database, thus mediate between the service and the database level.

The distributed database can be seen as a collection of physical objects stored on a physical storage device. These objects can be accessed through the System Buffer and Record Administration Server (SyBRAS). It provides efficient access to physical objects (synchronised by latches) and physical data independency, guarantees persistency for objects, etc.

The Persistent Object Store (POS) resides on top of SyBRAS. It provides another level of abstraction by supporting storage objects. Storage objects are constructed from physical objects, e.g., records. POS maintains direct physical references between storage objects, and offers associative, object-related and navigational access to these objects. Associative access means the well-known access via key values. Object-related access refers to direct object access using object identifiers. Navigational access is related to the propagation along physical references [6,3].

Communicating agents implement the functionality of the whole system. They integrate the processing of queries, methods assigned to objects and transactions. Furthermore, they are responsible for distributing requests to (remote) agents that store corresponding objects or process requests more efficiently.

These communicating agents are realized as two-stack abstract machines (2SAMs). Each agent consists of two levels. Lower levels link with POS and employ 2SAMs that deal with logical local objects. Higher level machines act

as coordinators for transactions and, thus, deal with logical global objects that are constructed from logical 'local' objects. On both levels multiple 2SAMs may process concurrently. In order to take advantage of concurrent processing and the distributed architecture 2SAMs can distribute requests to 2SAMs employed on the same level. However, only those machines on the higher level are equipped with an extended RPC enabling them to distribute requests to remote agents.

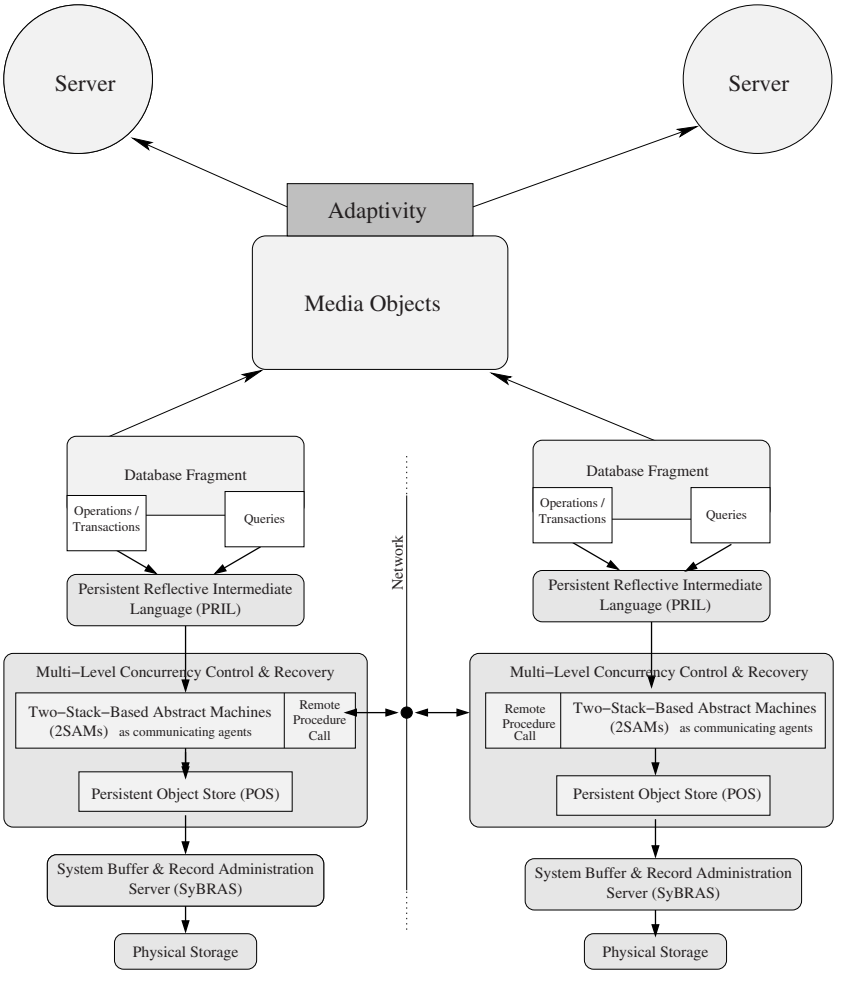

Fig. 1. General Architecture

Having these multiple object levels we can take advantage of a more sophisticated transaction management system. It is based on the multi-level transaction model [4,2] which is counted as the most promising transaction model in theory. It is combined with the ARIES/ML recovery mechanism.

Local objects do not correspond directly to logical global objects as processed by the communicating agents. Furthermore, high-level queries, transactions, object methods, etc need to be translated into code that can be interpreted by communicating agents. This conceptual gap between the logical level and the communicating agents is bridged by a persistent reflective intermediate language (PRIL).

PRIL will support linguistic reflection [5]. We provide a macro language, in which the high-level constructs in transactions such as generic update operations and the high-level algebra constructs for querying can be formulated.

4 Conclusion

The challenging engineering question in the area of WISs concerns system architectures that support WISs that are modelled as collections of media objects. In this article we presented a three-level architecture as an answer to this question. We argued that the materialisation of media objects in the underlying database reduces the problem to the integrated support of queries and operations, which both arise from the definition of media types. Our architecture is centered around two-stack abstract machines (2SAM), but the original idea of such machines has been extended in a way that communication between such machines, parallelism and distribution are supported. This defines three-levels: a service level addressing the services offered to WIS users, a database level addressing standard schema issues, and an operational level based on 2SAM as communicating agents.

References

1. Feyer, T., Kao, O., Schewe, K.-D., and Thalheim, B. Design of data-intensive web-based information services. In *Proceedings of the 1st International Conference on Web Information Systems Engineering (WISE 2000)*, Q. Li, Z. M. Ozsuyoglu, R. Wagner, Y. Kambayashi, and Y. Zhang, Eds. IEEE Computer Society, 2000, pp. 462–467.
2. Kirchberg, M. Exploiting multi-level transactions in distributed database systems. In *Proc of the Workshop on Distributed Data & Structures* (2002), Carleton Scientific.
3. Kuckelberg, A. The matrix-index coding approach to efficient navigation in persistent object stores. In *Workshop on Foundations of Models and Languages for Data and Objects* (1998), pp. 99–112.
4. Schewe, K.-D., Ripke, T., and Drechsler, S. Hybrid concurrency control and recovery for multi-level transactions. *Acta Cybernetica 14* (2000), 419–453.
5. Stemple, D., Sheard, T., and Fegaras, L. Reflection: A bridge from programming to database languages. In *Proc of the Hawaii Conf on System Sciences* (1992).
6. Zezula, P., and Rabitti, F. Object store with navigation acceleration. information systems. *Information Systems 18*, 7 (1993), 429–459.