

Personal Views for Web Catalogs *

Kajal T. Claypool, Li Chen and Elke A. Rundensteiner
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01609-2280
{kajal|lichen|rundenst}@cs.wpi.edu

Abstract

Large growth in e-commerce has culminated in technology boom to enable companies to better serve their consumers. The front-end of the e-commerce business is to better reach the consumer which means to better serve the information on the Web. An end-to-end solution that provides such capabilities includes technology to enable personalization of information, to serve the personalized information to individual users, and to manage change both in terms of new data as well as in terms of the evolution of personal taste of the individual user. In this work, we present an approach that allows automated creation and generation of diversified and customized web pages from an object database (based on the ODMG object model), and automated maintenance of these web pages once they have been build. The strength of our approach is its superior re-structuring capabilities to produce *personal views* as well as its generic propagation framework to propagate schema changes, data updates, security information and so on from the base source to the personal view and vice versa.

1 Introduction

Internet technologies and applications have grown more rapidly than anyone could have envisioned even five years ago, with a projection of 127% increase in Web sales for the current year alone (<http://www.internetindicators.com/facts.html>). What started out as static personal Web pages with photographs to be shared with friends and family or hastily compiled company brochures has quickly culminated into a myriad of sophisticated hardware and software applications. Companies today are moving beyond static information to provide to their users dynamic and personalized information to create new value for their stake-holders. An end-to-end solution that provides such dynamic capabilities includes technology to enable personalization of information, to serve the personalized information to individual users, and to manage change both in terms of new data as well as in terms of the evolution of personal taste of the individual user.

Consider for example the eToys web site (<http://www.etoys.com>), one of the biggest online retailer of toys for children. Currently, eToys provides a minimum two-level navigation of its web site based on *child's age* and *toy type* categorization. Sales on toys or the *hot* toys are shown as separate links and also have an age categorization. A user of the eToys site currently navigates individually through each of links. However, a frequent user of the web site often follows a set pattern of links. For example, a user with a **4 year old** would be most likely to look for toys in the age category of **4 to 6**. This user may be better served if the eToys web site collated information on sales, hot toys, and toys such as cars for **4 year olds** into one web page. Of course, since this information varies from one individual user to another where one user might target boys and another might target girls, each user would need a *personal view* of the eToys database. New toys added into the database should also be reflected in the user's personal view. Moreover, as the individual user's tastes change, the child grows older or

Copyright 1999 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*This work was supported in part by the NSF NYI grant #IRI 94-57609. We would also like to thank our industrial sponsors, in particular, IBM for the IBM partnership award. Kajal Claypool and Li Chen would also like to thank GE for the GE Corporate Fellowship and IBM for the IBM corporate fellowship respectively.

the child is no longer interested in cars, the *personal view* needs to be adapted to reflect this change in taste. Given the volume of users, 17 million US households alone (<http://www.nua.ie/surveys>), a key in providing for such dynamic behavior is a high degree of automation of both the view definition as well as the view maintenance process.

Today, technology such as *collaborative filtering*, *agent-based filtering* [AY00] are often used as mechanisms to arrive at the personal set of information. We do not look at these technologies per se but assume that one or more such technologies are available to arrive at the personalized information. We focus instead on actually serving users personalized information (database views with complex re-structuring capabilities) and on managing change on already served pages (a generic propagation framework that handles data updates as well as schema changes) by using and extending existing database (OODB) technology.

Current research efforts based on existing database technology [AMM98, FFLS97, CRCK98] have been made to automate aspects of providing dynamic views. For example, Araneus [AMM98] is an attempt to re-apply relational database concepts and abstractions to generate web sites. Strudel [FFLS97] instead has introduced a hyper-graph data model to capture the web structure and an associated web query language, StruQL. However, while these approaches focus on generating diversified web sites, they do not provide an easy to use mechanism in terms of creating the transformations for building these views. Nor do they look at the issues of maintainability in terms of reflecting the changing profile or taste, or propagating changes or additions to the existing data set.

Our Re-Web [CRCK98] system is an *easy to use, automated* web-site management tool (WSMS) based on the ODMG standard that focuses on building a *personal view* for each user and, as a second step, manages evolution as well as data changes of the *personal views*. Re-Web can translate web information to database information, re-structure and integrate information at the database level as needed and finally generate web pages from the information in the database. Exploiting the modeling power of the OO model, we have defined web semantics that allow us to map between the XML/HTML constructs and the ODMG object model constructs. We use these to do both the translation of the Web pages to the database as well as to generate the Web pages from the database. At the database level we use a flexible and extensible re-structuring facility, SERF [RCC00] to support complex re-structuring for the creation of new *personal views*. To increase the ease of Web site management, Re-Web also exploits the notion of a library of re-usable transformations from SERF [RCC00]. In particular, we define a library of typical Web restructuring and support a visual tool for building the transformations, further simplifying the web restructuring effort.

To address the maintainability of the *personal views* we also propose a generic propagation framework that can handle the propagation of information such as data update, schema changes, security information from one view to another or from the base information to the view. Thus, using this framework we can tackle in an automated fashion the updatability of *personal views* as and when new data is added to the base or existing data is updated. Moreover, as the user's profile changes either by a manual change to the profile or by some filtering mechanism, the propagation framework can be utilized to evolve the existing *personal view* to adapt to the changed profile. Once a database view is adapted to a change, the adapted web pages are automatically generated by Re-Web.

2 Personal Views

The Re-Web approach is a database centric approach. In our approach we map the ODMG object model to the XML model to automatically generate web pages based on the data in the database. Within the database itself, we use SERF [RCC00] to do complex re-structuring of the object schema as well as to create new view schemas which in turn map one-to-one to the re-structured web pages. Existing views can be maintained both in terms of data and schema changes by our propagation framework.

Generating Web Pages: Mapping ODMG to XML. For web site generation, we provide three tiers of web semantics representations. The top tier is the HTML representation, i.e., the web pages themselves, which include some visual metaphors and styles. Ideally the users navigate through the whole web site via its HTML pages in a unique and consistent fashion. The XML representation of the HTML pages forms the middle-layer, an intermediate translation between the ODMG data model is at the bottom tier and the equivalent loss-less representation of the HTML web pages.

Consider the home page of the eToys web site shown in Figure 1. A simplistic ODMG schema of the Web



Figure 1: The eToys Home Page.



Figure 2: The Personalized eToys Home Page.

site is as depicted in Figure 3¹. The database schema represents the **web-site-structure**. The structure of each web page in the given web-site, i.e., the **web-page-structure**, is given by the definition of the respective class. Each object of a class is then represented by a **web-page** with the class-driven web-page-structure. In our example, the home page for the eToys web site is modeled after the schema class eToys. Each element of a collection defined for the object is represented at the web-page level by a URL link to the specific web-page of the corresponding object. Thus, for example, in the eToys class we have a collection of objects of the type eToysByAge. For the home page, this collection is represented by the links 0-12 months, 2 year, 4 year, etc. Each of these URLs links to the web-pages 0-12 months, 1 year, etc., i.e., the representations of the objects of the class eToysByAge. Atomic literals such as the String attributes in a class are displayed as **web-items**. For example, text descriptors such as the welcomeMessage are shown on the **web-page** as plain text. A synopsis of the web-mapping is given in Table 1.

ODMG Primitives	XML Concepts	Web Constructs
Schema	set of DTDs	Web-site-structure
Type	DTD	Web-page-structure
Object	XML document	Web page
OID	URI	URL
Atomic literal	Leaf element	Web-item
Struct literal	Internal element	Structured web-item
Collection of literals	Collection of elements	List of web-items
Extent of a type	XML documents of a DTD	Web pages of a web-page-structure

Table 1: Web Semantics Mapping between ODMG, XML and Web Semantics

SERF: Re-Structuring at the Database Level. The strength of our approach lies in the fact that at the database level, we can exploit existing services such as complex re-structuring to provide the same rich capabilities for the web pages themselves. A re-structuring or a view desired for the web-site is translated to an equivalent transformation template at the database level which then is responsible for the production of the view as well as the associated re-structured objects. The new view schema is translated using the ODMG to

¹This schema has been fabricated for the purpose of our example here, and does not reflect the eToys schema.

XML mapping (Table 1) to an equivalent XML view.

We provide the SERF framework [RCC00] to enable *user-customized* and possibly *very complex* database schema transformations thereby supporting a *flexible, powerful* and *customizable* way of generating web-pages. A declarative approach to writing these transformations is to use OQL together with a view mechanism. OQL, a declarative language, can express a large class of view derivations and also has the expressive power for realizing any arbitrary object manipulations to transform objects from one type to any other type. Moreover, OQL can invoke any required system methods for supporting the view mechanism.

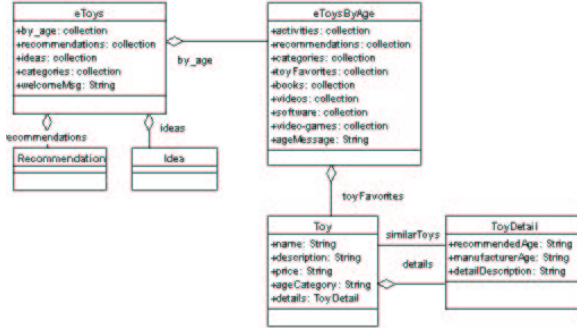


Figure 3: The ODMG Schema for the Web-Page in Figure 1.

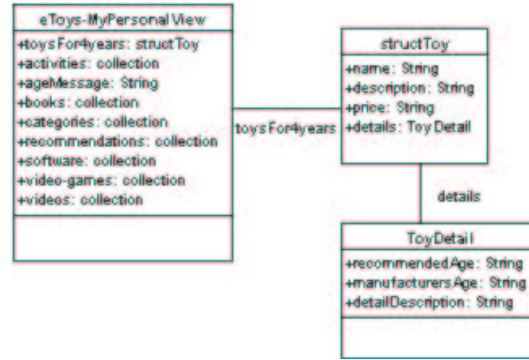


Figure 4: The ODMG Schema for the Web-Page in Figure 2.

Consider, for example, that instead of the generic existing **eToys** home page, **eToys** desired a home page more geared towards the individual user, a user who wants to view only the recommended toys for 4 year olds². Moreover, consider that the user wants to see the brief description of all recommended toys on one web-page rather than having to click to get to the toys. Figure 2 shows a representation of the personalized web page. To accomplish this, we create a new view schema at the database level. Thus, instead of having the generic **eToys** schema, we have a personalized **PeToys** view schema. The **PeToys** schema contains information only on the recommended toys for 4 year olds. Moreover, the **eToys-MyPersonalView** class contains an *inlined* description of all the toys along with additional pertinent information. This information for a toy is stored in a structure. Figure 4 shows the new view schema while Figure 5 shows the OQL transformation used to accomplish the re-structuring, i.e., to produce the new view schema.

However, writing these transformations for the re-structuring of the database is not a trivial task as the view definition queries can be very complex. Similar to schema evolution transformations, it is possible to identify a core set of commonly applied transformations [RCC00]. For example, flattening a set of objects such that they appear as a list of web-items rather than a list of URLs is a common transformation that can be applied for different web site schemas. Thus in our framework we offer *re-use* of transformations by encapsulating and generalizing them and assigning a name and a set of parameters to them. From here on these are called *view transformation templates* or *templates* for short. Figure 6 shows a templated version of the transformation in Figure 5. This template can be applied to *inline* the information linked in by a reference attribute (URL at the XML level) into the parent class (web-page) itself.

We have also proposed the development of a library of such templates. This is an open, extensible framework as developers can add new templates to their library, once they identify them as recurring. We envision that a template library could become an important resource in the web community much like the standard libraries in the programming environment.

Maintenance of Personal Views. Once the personal views are deployed, it is essential to have a maintenance process in place to provide an end-to-end solution. This is required to handle not only the data updates, for example a new toy is suitable for 4 year olds is added to the database (data update - base to view), but also to deal with changing tastes, such as the user is now interested in toys for a 2 year old in addition to the toys

²The company would of course need to have a process in place that allows the user to specify a profile or uses a filtering mechanism to generate the profile before they proceed to actually building these personal web pages.

```

// define a named query for the view
define ViewDef (viewClass)
  select c
  from viewClass c
  where c.age-category = "4year";

// create view eToys-MyPersonalView from the eToysByAge class
// and create struct structToy from the Toy class

create_view_class (eToysByAge, eToys-myPersonalView, ViewDef(eToysByAge ));

create_view_struct (Toy, structToy);

// add an attribute structToy to the view
add_attribute (eToys-MyPersonalView, structToy,
              collection<structToy>, null);

// get all the objects of class
define Extents (cName)
  select c
  from cName c;

// for each of the eToys-MyPersonalView object, find its referred Toy
// objects via the toyFavorites attribute and convert them into a
// a collection of structToy.
define flattenedCollection (object)
  select (structToy)p.*
  from Toy p
  where exists (p in object.toyFavorites);

for all obj in Extents (eToys-MyPersonalView )
  obj.set(obj.structToy, flattenedCollection(obj));

```

Figure 5: SERF Inline Transformation.

```

begin template convert-to-literal (Class mainclassName,
  String mainViewName,
  Attribute attributeToFlatten,
  String structName)
{
  // find the class that needs to be flattened given the attribute name
  refClass = element (
    select a.attrType
    from MetaAttribute a
    where a.attrName = $attributeToFlatten
    and a.classDefinedIn = $mainclassName );

  // Create the view class
  create_view_class ($mainclassName, $mainViewName, View ($mainclassName));

  // flatten refClass to a struct
  create_view_struct ($refClass, $structName);

  // add a new attribute to hold the struct
  add_attribute ($mainViewName, $structName, collection<$structName>, null);

  // get all the objects of class
  define Extents (cName)
    select c
    from cName c;

  // convert a collection of objects to a collection of structures.
  define flattenedCollection (object)
    select ($structName)p.*
    from $refClass p
    where exists (p in object.$attributeToFlatten)

  for all obj in Extents ($mainViewName)
    obj.set(obj.$structName, flattenedCollection(obj));

  // remove the attributeToFlatten attribute
  delete_attribute ($mainViewName, $attributeToFlatten);
}

```

Figure 6: SERF Inline Template.

for a 4 year old (schema change - view to base), or to deal with a new feature such as a special 25% sale on popular items every customer (schema change - base to view). All of this information needs to be propagated from the base schema to all of the view schemas in the system or conversely may need to be propagated from the view schema to the base schema. Moreover, with e-commerce there is an additional dimension of security. The personalized views unlike the home page should not be visible to the world. Thus a subset of security permissions needs to be attached to each personal view. This also may need to be propagated from the base to the view in the case where the base is the keeper of all security information, and from the view to the base in the case of the user wanting additional individuals to share the same web-page.

```

define [propagation/re-write] rule ruleName for class :
on event
in direction
when condition(s)
do [instead] action(s)
precedes [rule]+

```

Figure 7: Syntax of a Rule.

```

define re-write rule rule1 for project :
on add-attribute(C,a,t,default)
in up
when
do re-write-query(existingQuery, a) &&
propagate-to-derivedNodes
precedes rule3

```

Figure 8: A Sample Rule.

As a solution we present an extensible propagation framework. To allow for the diversity in the type of information and the direction of propagation, we have designed a rule-based, propagation framework as opposed to constructing hard-coded algorithms to handle the same. Rules for our framework can be expressed using a rule-based extension of OQL, PR-OQL. These rules are specified for each node in the derivation tree. They also specify the direction in which the information is propagated, **up** from the base to the view or **down** from the view to the base. For example, a text message that announces to the customers a special 25% sale on the top ten toys can be added to the base class via a schema change `add-attribute(eToys, saleMsg, String, 'Special sale on top ten toys')`. Using the rules in our framework, this change is propagated to all the defined views in an automated fashion. Figure 7 shows the syntax of the rules defined in our framework while Figure 8 shows a sample rule.

3 Summary

We have developed a prototype for the Re-Web system, a Java-based system (JDK1.1.8 and Java Swing). We make use of the LotusXSL and IBM XML parser for the generation of web pages from underlying databases. The system has been implemented and tested on WinNT and Solaris. Figure 9 gives the general architecture of Re-Web. The Re-Web system will be demonstrated at Sigmod 2000 [RCC00].

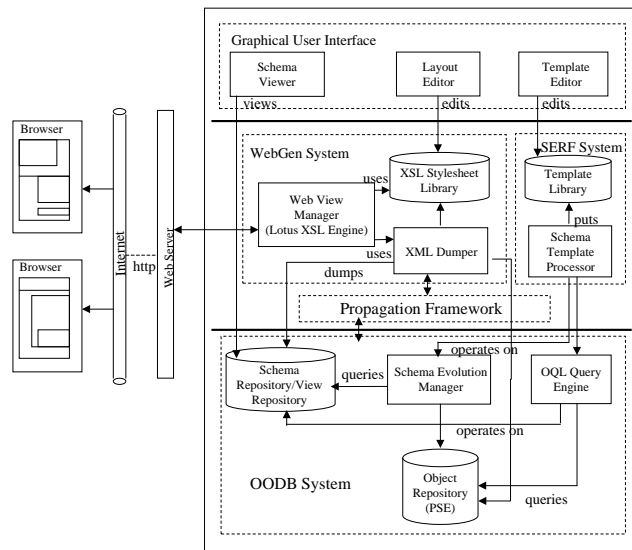


Figure 9: Architecture of the Re-Web System.

In summary, here we have given an overview of our Re-Web system and its prominent features such as:

- Generation of XML/HTML web-pages from an ODMG schema.
- Personal Views using re-structuring and view capabilities at the database level (SERF).
- Maintenance of Personal Views using an extensible rule-based propagation framework.

References

- [AGM⁺97] Serge Abiteboul, R. Goldman, J. McHugh, V. Vassalos, and Y. Zhuge. Views for Semistructured Data. In *Workshop on Management of Semistructured Data*, pages 83–90, 1997.
- [AMM98] P. Atzeni, G. Mecca, and P. Merialdo. Design and Maintenance of DataIntensive Web Sites. In *EDBT'98*, pages 436–450, 1998.
- [AY00] C. C. Aggarwal and P.S. Yu. Data Mining Techniques for Personalization. In *IEEE Bulletin - Special Issue on Database Technology in E-Commerce*, page to appear, 2000.
- [CR00] L. Chen and E. A. Rundensteiner. Aggregation Path Index for Incremental Web View Maintenance. In *The 2nd Int. Workshop on Advanced Issues of E-Commerce and Web-based Information Systems, San Jose, to appear, June 2000*.
- [CRCK98] K. Claypool, E.A. Rundensteiner, L. Chen, and B. Kothari. Re-usable ODMG-based Templates for Web View Generation and Restructuring. In *WIDM'98*, pages 314–321, 1998.
- [FFLS97] M. Fernandez, D. Florescu, A. Levy, and D. Suci. A Query Language for a Web-Site Management System. *SIGMOD*, 26(3):4–11, September 1997.
- [RCC00] E.A. Rundensteiner, K.T. Claypool, and L. et. al Chen. SERFing the Web: A Comprehensive Approach for Web Site Management. In *Demo Session Proceedings of SIGMOD'00*, 2000.