

Fall 8-1-2002

# Developing a Healthcare Website with Oracle XML Technologies

Haiyan Liu  
*Dakota State University*

Follow this and additional works at: <https://scholar.dsu.edu/theses>

---

## Recommended Citation

Liu, Haiyan, "Developing a Healthcare Website with Oracle XML Technologies" (2002). *Masters Theses*. 17.  
<https://scholar.dsu.edu/theses/17>

This Thesis is brought to you for free and open access by Beadle Scholar. It has been accepted for inclusion in Masters Theses by an authorized administrator of Beadle Scholar. For more information, please contact [repository@dsu.edu](mailto:repository@dsu.edu).

Developing a Healthcare Website with Oracle XML Technologies

BY: Haiyan Liu

A project submitted in partial fulfillment of the requirement for the

Master of Science in Information Systems

Dakota State University

Year 2002



**MSIS**  
**PROJECT APPROVAL FORM**

Student Name: Haiyan Liu

Expected Graduation Date: 9/1/02

Master's Project Title: Developing a Healthcare Website with Oracle XML Technology

Date Project Plan Approved: 8/30/02

Date Project Coordinator Notified and Grade Submitted: 9/1/02

Approvals/Signatures:

Student: Haiyan Liu

Date: 8/30/02

Faculty supervisor: Mina Shan

Date: 8/30/02

Committee member: Jerry Dumas

Date: 8/30/02

Committee member: Mark Wang

Date: 8/30/02

Copies to:

Original Attached to Written Report

Copies to: Advisor, Graduate Coordinator, and Student

## **Abstract**

Taking a healthcare website as an example, this project addresses certain difficult issues involved in website development and maintenance. The three major problems confronted by current healthcare web sites are static content, underutilized data, and high development and maintenance cost. Adopting a new web design methodology, this project demonstrates how to overcome these problems and accomplish the following objectives: serve dynamic content from backend database, maximize the use of data, and minimize development and maintenance cost.

The technologies chosen to implement the project are Oracle XML technologies. Oracle XML Developer's Kit (XDK) combines the power of SQL, XML and XSLT within the XSQL Pages facility, providing an ideal framework to bring into life the clean separation of powers between XSQL page for data and XSLT style sheet for transformation.

My healthcare web site has three parts -- a personalized healthcare news portal, the online discussion forum and a personal health records organizer. Powered by Oracle9i database, it serves a dynamic content through XSQL pages, makes full use of data by means of XML data page as an interim data abstraction and XSLT to transform the data into a variety of formats, as well as saves development and maintenance cost through the separation of data, presentation and logic.

This methodology has a promising future for application. Besides the healthcare industry, it could be applied to the development and maintenance of most websites in a wide range of industries. Certainly this project is not only a demonstration but also will have its own contribution if it is put into use.

# Table of Contents

<b>1</b>	<b>INTRODUCTION.....</b>	<b>2</b>
<b>2</b>	<b>STATEMENT OF PROBLEM.....</b>	<b>5</b>
2.1	STATIC CONTENT .....	5
2.2	UNDERUTILIZED DATA .....	6
2.3	HIGH DEVELOPMENT AND MAINTENANCE COST .....	6
<b>3</b>	<b>OBJECTIVES .....</b>	<b>7</b>
3.1	GENERATE DYNAMIC CONTENT.....	7
3.2	MAXIMIZE THE USE OF DATA .....	7
3.3	MINIMIZE COST .....	7
3.4	SITE REQUIREMENTS.....	8
<b>4</b>	<b>SCOPE OF THE PROJECT .....</b>	<b>9</b>
4.1	OVERALL WEBSITE CONTENT STRUCTURE .....	9
4.2	METHODOLOGY .....	12
4.3	IMPLEMENTATION .....	14
4.3.1	<i>Use XSQL Pages to Serve Database-driven XML .....</i>	<i>14</i>
4.3.2	<i>Maximize the Use of Data.....</i>	<i>18</i>
4.3.3	<i>Minimize Website Development and Maintenance Cost.....</i>	<i>26</i>
<b>5</b>	<b>CONCLUSION .....</b>	<b>32</b>
	REFERENCES .....	33

## APPENDIXES

- A. ENTITY RELATIONSHIP DIAGRAM
- B. CODE
- C. WORK BREAKDOWN STRUCTURE/GNATT CHART

# 1 Introduction

Numerous health-related websites have sprouted during the last couple of years to captivate on the macro trend in the healthcare industry. Based on their target market and operational characteristics, these websites can be roughly divided into three categories:

- Websites affiliated to hospitals or research institutes. Designed for healthcare professionals, they allow doctors to tap into files and medical records and also provide communication with other hospitals.
- Online medical consulting service websites. Companies such as [www.mediconsult.com](http://www.mediconsult.com), provide consultation with top medical experts. You can ask specific questions and get an answer from a specialist in a few days.
- Consumer-focused healthcare information websites. Their goal is to provide objective, credible and trusted source of healthcare information that helps people play an active role in managing their own health.

Good health is indisputably one of the most basic needs for human beings. To stay healthy, we need access to quality healthcare information. Doctor visits are, however, sometimes difficult or even frustrating to schedule. Healthcare websites provide an ideal alternative for people to obtain healthcare information. The 24/7 nature of the World Wide Web makes healthcare websites an accessible and affordable virtual doctor for anyone who has Internet access. The number of U.S. Internet users who go online to look for information about health topics has grown from 54 million in 1998 to 110 million in the latest (March 2002) survey (Greenspan, 2002).

However, most healthcare websites today are facing three common problems: static content served from file servers, inflexible output formats of web pages due to underutilized data, and high development and maintenance cost due to intermingling of data, presentation and logic. The first problem can be addressed by using databases to serve as the backend repository for the content. The latter two problems stem from the mixture of data, business logic and presentation in HTML pages. To solve these problems, we need a new approach to web development.

My project, a consumer-focused healthcare information website, powered by Oracle9i database and Oracle XDK, provides a viable solution to these problems. It comprises three parts, a personalized healthcare news portal, online discussion forums and an online health records organizer. Once the user logs into the website, he is embraced with dynamic content. He can customize the category and number of headlines available in the news portal; read, post and search through messages in the discussion forums; and add personal health records in the online health record organizer section. The website content is dynamically served by querying and updating the Oracle9i database. In addition, the user has the flexibility of switching between multiple views for the same news content—news box view, news table view, SQL file view, and raw XML view. The list of views can go and on to meet the needs of different users. This formatting flexibility and extensibility is made possible by the clean separation of data and presentation between XML and XSLT. Moreover, by decoupling data and presentation, the whole process of web page development and maintenance is more modularized and simplified.

This report has the following five parts: part one gives a brief introduction of the background and overview of this project; part two identifies the major problems in the development and maintenance of current websites; part three presents the objectives of this project; part four introduces the adopted methodology based on Oracle XML technologies and the implementation in detail; the last part summarizes this paper and points out the promising application of this research.



## 2 Statement of Problem

### 2.1 Static Content

Experts agree that appealing Web sites need a healthy mix of the three c's – content, commerce, and community. Content is of the most critical importance to the success of a website.

Today the vast majority of content available on the Internet is served statically from file servers. Health care websites are no exception. Most web sites comprise of a large number of static pages from file servers. Databases have begun to serve as the repository for this content, but have been largely restricted to transactional or inventory data (Scardina, 2002).

While static pages are simple to create, fast to return and fast to download, this “Write Once, Read Many” content has many inherent drawbacks (Wardley, 2000).

- Fixed Content: the content is fixed and cannot be customized for different users.
- No Interaction: it's impossible to incorporate some run-time data such as the results of a search engine or database query, or respond to any other kind of dynamic event such as the provision of CGI parameters.
- Difficult to Maintain. While it is not impossible to manually edit a small number of static files, “this method doesn't scale well and tends to lead to inconsistencies between pages.” (Wardley, 2000) As a result, static content is time consuming to maintain and is easily out-dated.

## **2.2 Underutilized Data**

People have different needs for the same data. Some of our users like the default presentation style the way it is, but some of them may prefer to see another style, and some may want the data page delivered in other formats such as raw XML file, simple text file, or pdf file, etc. It is necessary for a website to provide users with the option of switching between different views of the data.

Most healthcare websites today only deliver data in HTML format, which results in a chasm between what users need from a website to what it can offer. Therefore, data retrieved from database is underutilized and has a short life.

## **2.3 High Development and Maintenance Cost**

Common approaches of developing websites end up embedding HTML in the code (such as CGI script) or embedding code in HTML (such as PHP, ASP), all of which mix data, presentation and business logic together in the same page. The disadvantage of these approaches is that “presentation is interleaved with business logic, such that they cannot be easily reused separately... It completely locks out non-programmers from web authoring, and it requires significant effort to change the look of a web site generated this way.” (Burke, 2000).

It is the high cost of development and maintenance that prevents healthcare websites from serving ordinary people in its effort to narrow the digital gap between the requirements and modern technologies.

## **3 Objectives**

### **3.1 Generate Dynamic Content**

“The alternative to the static content is dynamic content generation where the HTML page (or other format) is created on-the-fly whenever a request is received from a particular URL.” (Wardley, 2000)

Dynamic content has clear advantages over static content. Web pages generated dynamically are more responsive as you can incorporate relevant dynamic data as per visitors' request. Web pages returned to visitors are also customized according to their preferences. In one word, dynamic content helps to retain visitors by enhancing “stickiness” to a website.

### **3.2 Maximize the Use of Data**

In order to make our website popular with a broad range of users, we need to fully utilize the information available on our website. As different users process information in different manners, it is imperative for us to have our data repurposed for delivery in a variety of formats to meet the needs and wants of individual users. In a sense, this is about increasing the life of data so that data is utilized to the fullest extent.

### **3.3 Minimize Cost**

Cost minimization is another benchmark of whether a website is successful or not. In the process of development and maintenance, affordability and profitability are the main concerns of developers and enterprises. The approaches that minimize website-designing

cost can not only improve the competitiveness of developers and enterprises but also supply a low-cost access to better serve customers. Therefore, we need a highly structured approach that builds flexibility and maintainability into a site from the start, an approach that separates data, business logic and presentation, as well as provides cross-site consistency and greatly reduces the maintenance burden.

### **3.4 Site Requirements**

Upon identifying the problems and objectives, it became clear to me that I needed an architecture that can:

- Dynamically generate data from the database and customize the data according to a user's preference.
- "Divide and conquer". Divide the whole mass into two separate constituents – data and presentation-- and address each part separately;
- "Integrate and deliver". Use a framework to integrate the two separate parts and deliver the final page.

## **4 Scope of the Project**

### **4.1 Overall Website Content Structure**

My website consists of three parts:

- A personalized healthcare news portal

Access to the latest news and information is one of the primary reasons for a person to visit a website and has become an integrated part of any healthcare website.

- Online discussion forums

Online discussion forums are a popular way to post questions on various topics and share opinions, interests and concerns with other people. The sense of belonging to an online community helps to build loyalty to a website and therefore has been implemented by many websites.

- A personal health records organizer

This part provides tools and utilities to organize all the personal health records.

The personalized news portal offer individual users healthcare news in categories they are interested in and display the news in a manner that conforms to their preferences. For each category, the front page displays only the latest headline news up to the number of headlines a user prefers to read, as shown in Figure 1. From there, they can drill into a more detailed and comprehensive list of all the news in a particular news category. Users are allowed to reset their preferences and the display of news headlines will change accordingly. I'll explain the mechanism in the next section.

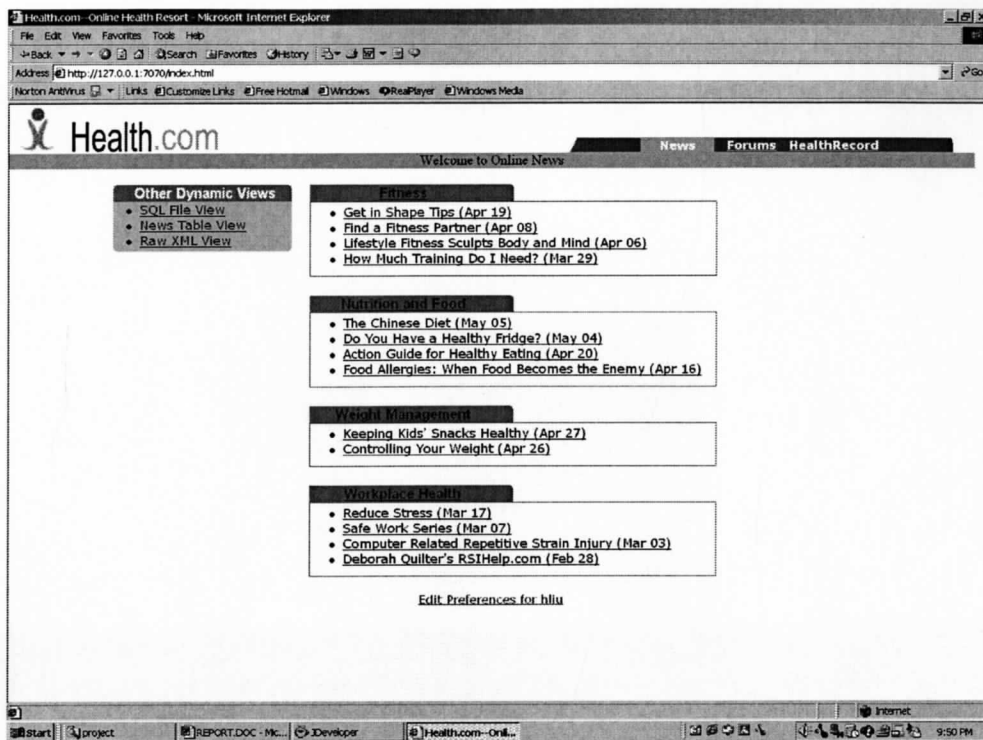


Figure 1: Screenshot of front page for personalized news portal

As shown in Figure 2, users start by looking at a list of discussion forums organized around interesting areas in the healthcare field. From there, they can drill into a particular forum to see what topics are being discussed; they can get an overview of all active topics across all the forums; or they can search for a particular word or phrase and browse through the results. While in a particular forum, users can browse the topics as well as drill in to see what replies have been posted for any topic. If a user logs in by providing a user ID, he can post new topics and post replies to existing topics.

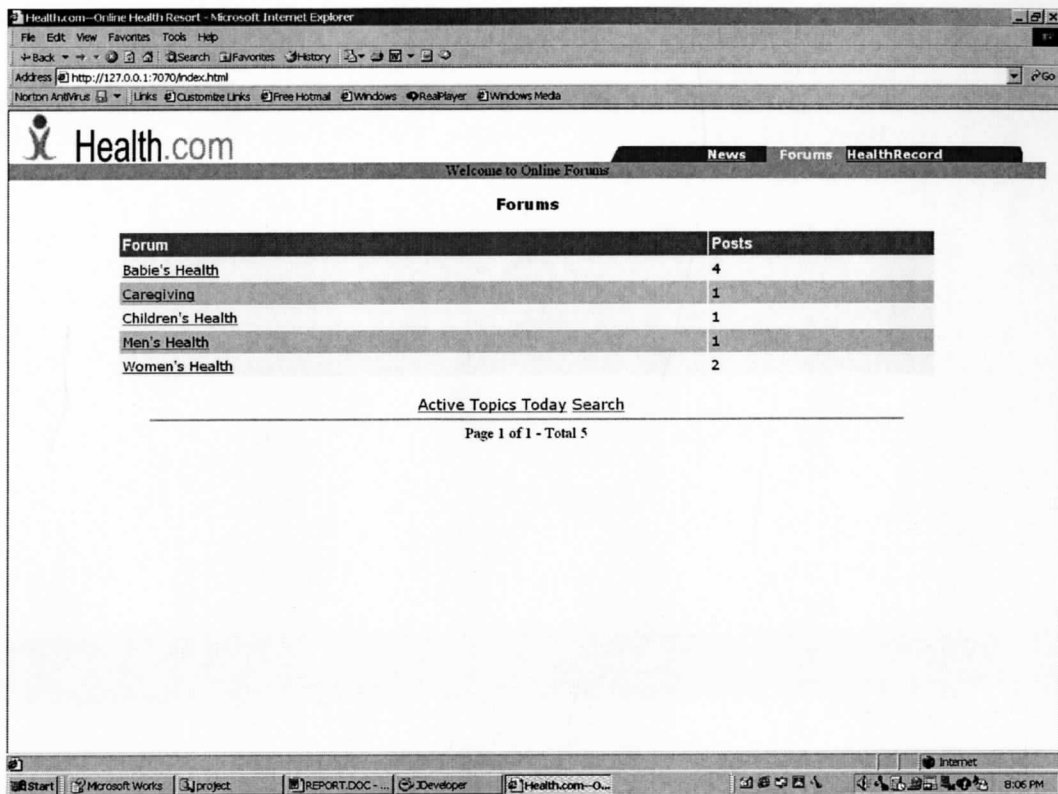


Figure 2: Screenshot of front page for online forums

The online health record organizer provides tools and utilities to centrally manage all the personal health records for a user. As shown in Figure 3, the front page has two boxes. The navigation box on the left allows users to drill into a particular subject area, where they can view the existing records as well as add more records to the online repository. The feature box on the right serves as a brief introduction to all the features available here in this part.

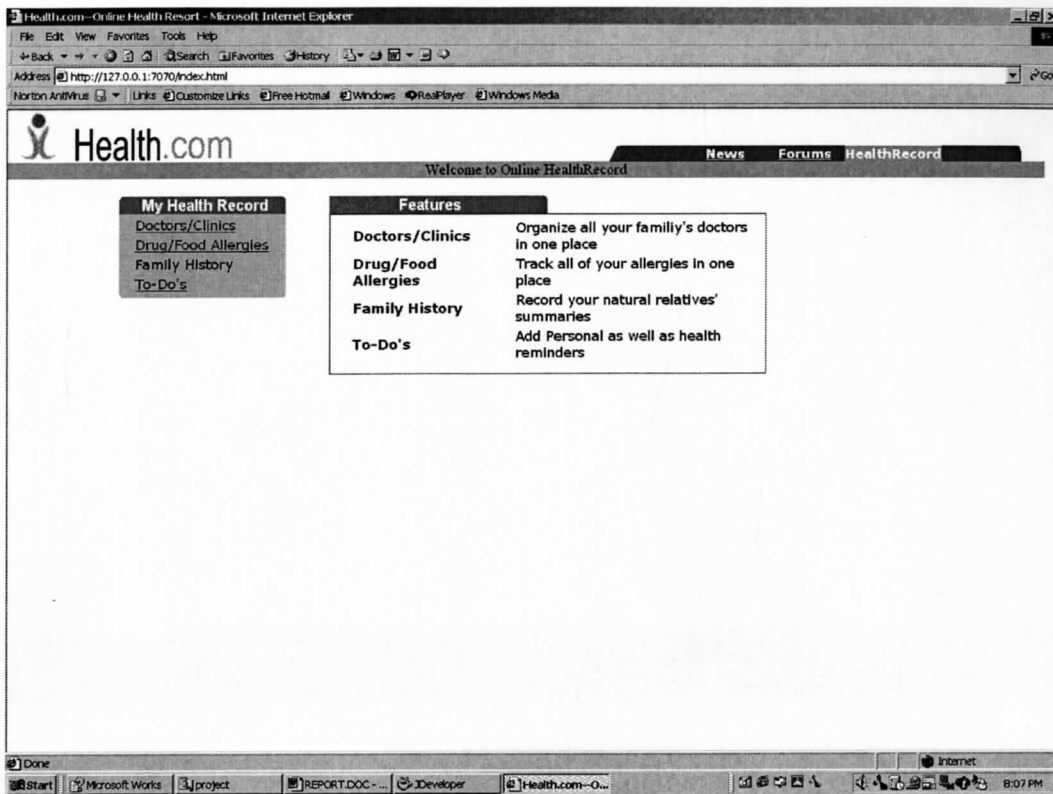


Figure 3: Screenshot of front page for health records organizer

## 4.2 Methodology

After analyzing the site requirements, I chose Oracle9i XDK to develop the website. The Oracle XML Developer's Kit (XDK) contains the basic building blocks for reading, manipulating, transforming and viewing XML documents. Moreover, the Oracle XDK is freely downloadable to 1.7 million developers that use the Oracle Technology Network. Unlike many shareware and trial XML components available on the Internet, the Oracle XDKs are fully supported and come with a commercial redistribution license.

To provide a broad variety of deployment options, the Oracle XDKs available for Java, JavaBeans, C, C++ and PL/SQL. Oracle XDKs consist of the following components:



- XML Parser: supporting Java, C, C++ and PL/SQL, the components create and parse XML using industry standard DOM and SAX interfaces.
- XSLT Processor: transforms or renders XML into other text-based formats such as HTML.
- XML SQL Utility: supporting Java, generates XML documents, DTDs and Schemas from SQL queries.
- XSQL Servlet: combines XML, XQL, and XSLT in the server to deliver dynamic web content.

As shown in Figure 4, Oracle XML technologies provide an ideal solution to:

- Dynamically generate data from the Oracle9i database and customize the data according to a user's preference.
- "Divide and conquer": a design methodology in which the definitions of data, presentation and logic are largely separate, facilitating their implementation by separate groups of people as well as their reuse as separate, independent software entities. In this way, we are able to maximize the use of data and minimize development and maintenance cost.
- "Integrate and deliver". Oracle XSQL Pages provides the publishing framework to integrate the separate parts and deliver the final pages to the client.

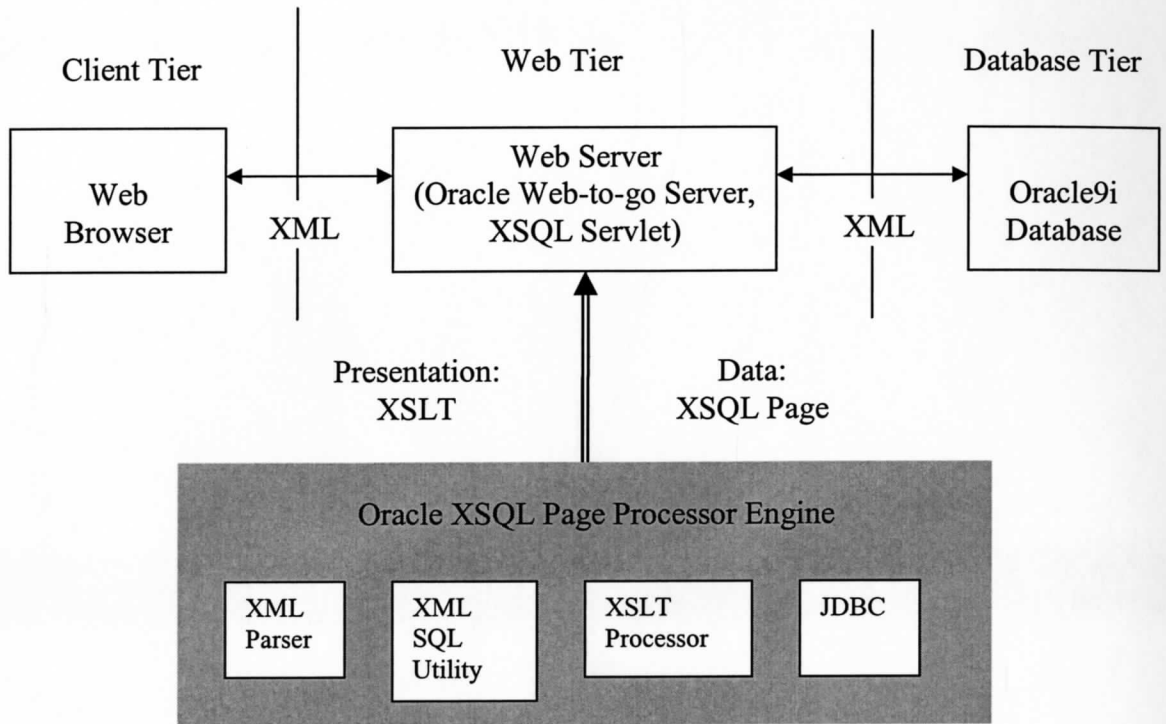


Figure 4: Overview of the three-tier website architecture

## 4.3 Implementation

### 4.3.1 Use XSQL Pages to Serve Database-driven XML

Oracle has taken significant steps to integrate XML technologies into its products. It now offers, among other things, an XML parser, an XSLT processor, an 'XML SQL' utility that facilitates moving data between XML and native database formats. XSQL Pages facility is an XML publishing framework that coordinates all these components.

The XSQL servlet interprets document written in a special XML vocabulary. Conventionally given a .xsql extension, these xml documents use `<xsql:query>` tags

where you want dynamic XML query results to be included. Based on what it sees in a document, the servlet will typically go off and issue SQL commands to some database, and then turn the results into XML for the client.

As shown in Figure 5, this website comprises three major XSQL pages, i.e., the home pages for the three functional parts. Driven by the Oracle9i database backend, they provide dynamic data for the users. In addition, there is a top navigation bar that is also dynamically generated from Site.xsql.

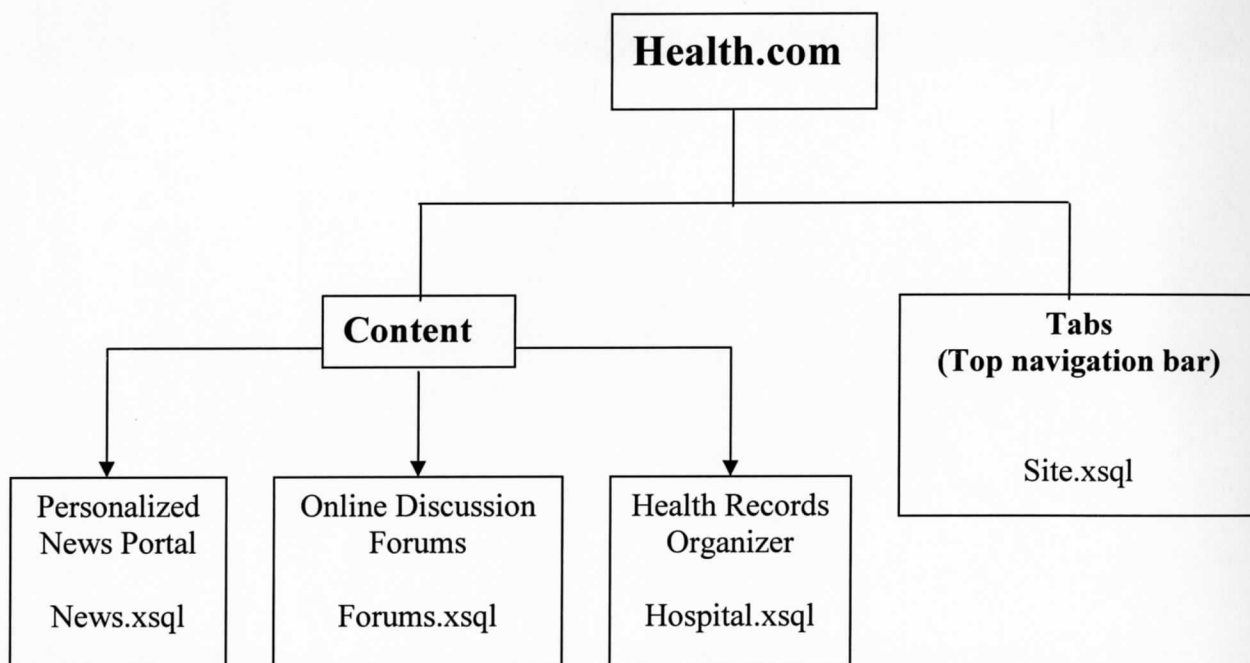


Figure 5: Overall Site Structure of Database-served XSQL Pages

## Updating Database

Users can edit their preferences by selecting news in certain categories and specifying how many number of headlines to read per category, as shown in Figure 6.

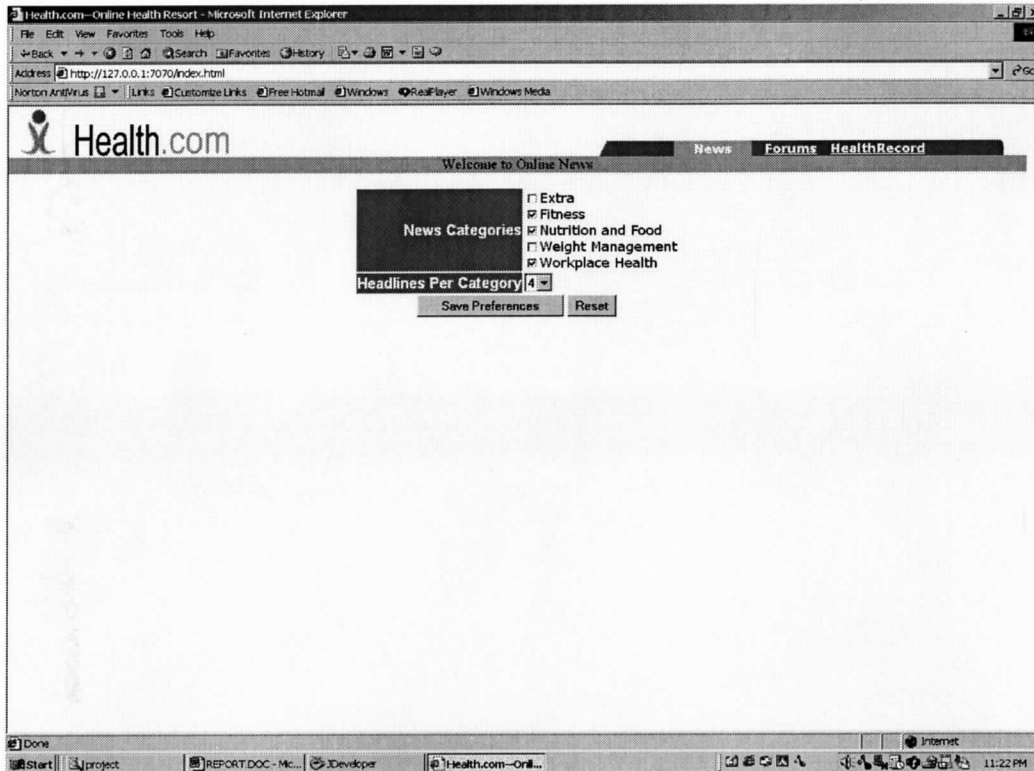


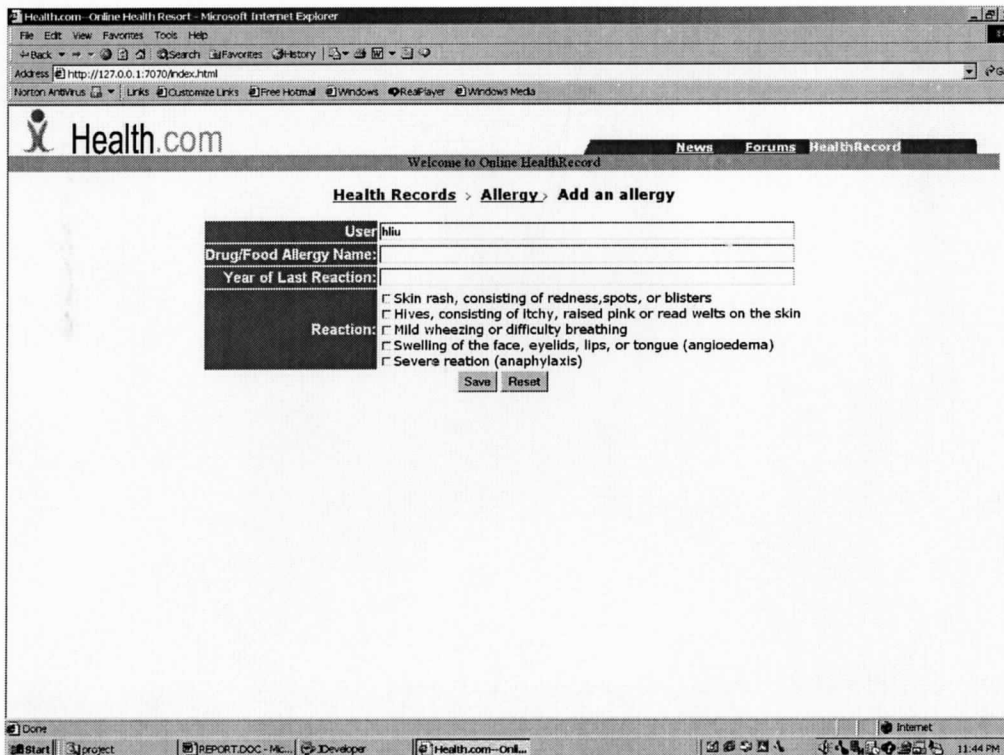
Figure 6: User preferences form to customize the look of news headlines

After users click the submit button, the posted form information is transformed into the appropriate ROWSET structure for the `user_prefs_view`, and then an INSTEAD OF INSERT trigger performs the actual insertions and updates to the preferences tables underneath the cover.

## Insert into Database

Similarly, users can insert new records into any online health record organizer, as shown in Figure 6.

After users click the submit button, the posted form information is transformed into the appropriate ROWSET structure for the user\_new\_allergy view, and then an INSTEAD OF INSERT trigger performs the actual insertions to allergy and user\_allergy tables underneath the cover.



The screenshot shows a Microsoft Internet Explorer browser window displaying the Health.com website. The page title is "Health.com" and the URL is "http://127.0.0.1:7070/index.html". The page content includes a navigation menu with "News", "Forums", and "HealthRecord". Below the menu, there is a breadcrumb trail: "Health Records > Allergy > Add an allergy". The main form contains the following fields and options:

User:	<input type="text" value="hlu"/>
Drug/Food Allergy Name:	<input type="text"/>
Year of Last Reaction:	<input type="text"/>
Reaction:	<input type="checkbox"/> Skin rash, consisting of redness, spots, or blisters <input type="checkbox"/> Hives, consisting of itchy, raised pink or red welts on the skin <input type="checkbox"/> Mild wheezing or difficulty breathing <input type="checkbox"/> Swelling of the face, eyelids, lips, or tongue (angioedema) <input type="checkbox"/> Severe reaction (anaphylaxis)

At the bottom of the form are two buttons: "Save" and "Reset".

Figure 7: Form to insert into the user\_allergy table

### **4.3.2 Maximize the Use of Data**

XML lends itself well to transforming data to a variety of formats with the assistance of XSLT (the XSL Transformation standard). In this way, “XML increases the life of data because you can repurpose it as your needs emerge” (Bannerjee, 2001).

“XML documents are intended to be easily read by both people and software, but it is not expected that people interested in the material these documents contain will wish to see that tags. To publish information held in XML format, it is necessary to replace the tags with appropriate text styles. XSL (the XML Stylesheet Language) is being developed to meet this requirement, with the assistance of XSLT (the XSL Transformation standard).” (Bradley, 2000)

Through the use of different XSLT Style sheets, the information stored in an XML document can be formatted for display/output in a variety of ways. The XSLT processor reads both an XML document and XSL style sheet and outputs a new document (which can be in any format), as shown in Figure 6.

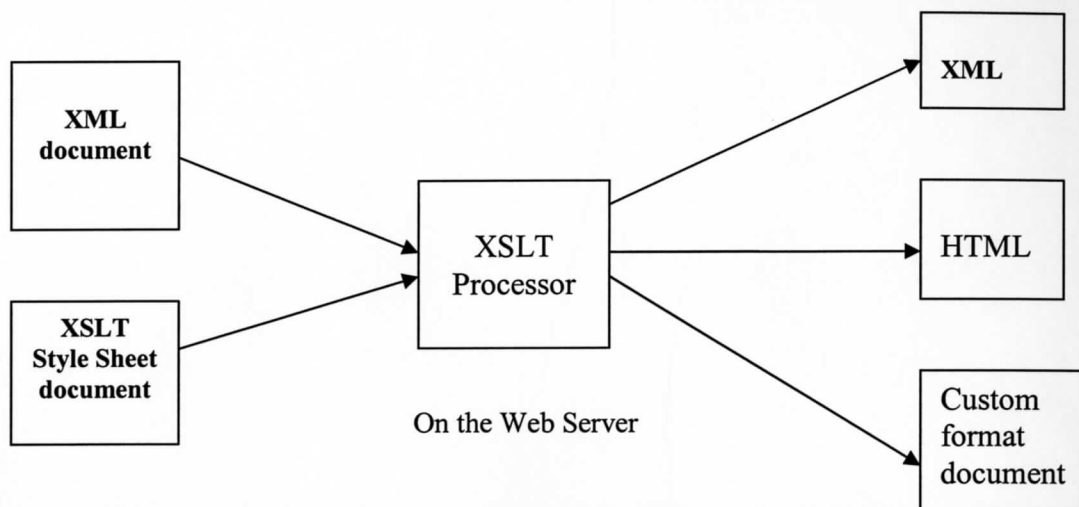


Figure 6: XSLT processor transforms a source XML document into a result document of any format

### **View selection: letting the user decide**

By reusing the dynamically produced XML as an interim abstraction and associating it with different XSL style sheet documents, we can assemble a “data page” from multiple queries and transform this assembled XML data page into any desired output format for presentation in a browser or any of a number of XML-based formats (Muench, 2000).

This flexibility of view selection enables users to switch between different views of the data without requiring that the data be generated again in a different form from the database server. “We see XML as a method to meet business need: Give me access to diverse data sources and manipulate them many times on my desktop without a trip back to the database.” (Gougler, 2000)

With the explosion of healthcare market and the request to meet the needs of various kinds of users, the ability to assemble information from multiple sources and transform it for delivery in any format required by the user is extremely valuable.

### **Oracle XSQL Pages: the Publishing Framework**

Oracle XSQL Pages provide the framework to publish and repurpose the same data using output styles specific to each scenario, which enables me to generate multiple views for the same data page and therefore meet the needs and requirements of a broad range of users. A simple process of the mechanism is:

- Modularize a query that occurs in multiple pages into a reusable XSQL page that can be shared by many pages;
- Reuse the dynamically produced XML as an interim abstraction;
- Leverage the flexibility of XSLT transformations to make the database-driven News portal content available in numerous XML, HTML, and text formats;

I'll use the structure of the Personalized News Portal to illustrate how I created a basic data file with the XSQL query and use `<xsql:include-xsql>` to include this file in a set of files, each pointing to a different style sheet to achieve a different output format.

As shown in Figure 7, users of the news portal have the luxury of switching between four views when they read the latest news:

- NewsContent.xsql



- News.xsql
- News\_as\_sql.xsql
- News\_as\_table.xsql

NewsContent.xsql retrieves the news of different categories from the database and is used as an interim abstraction for the “datagrams”. Simply include a tag named `<xsql:include-xsql>`, we are able to create a new XSQL page that reuses the dynamic XML content produced by NewsContent.xsql. This mechanism delivers customized data in a variety of ways for users and therefore significantly increases the life of data.

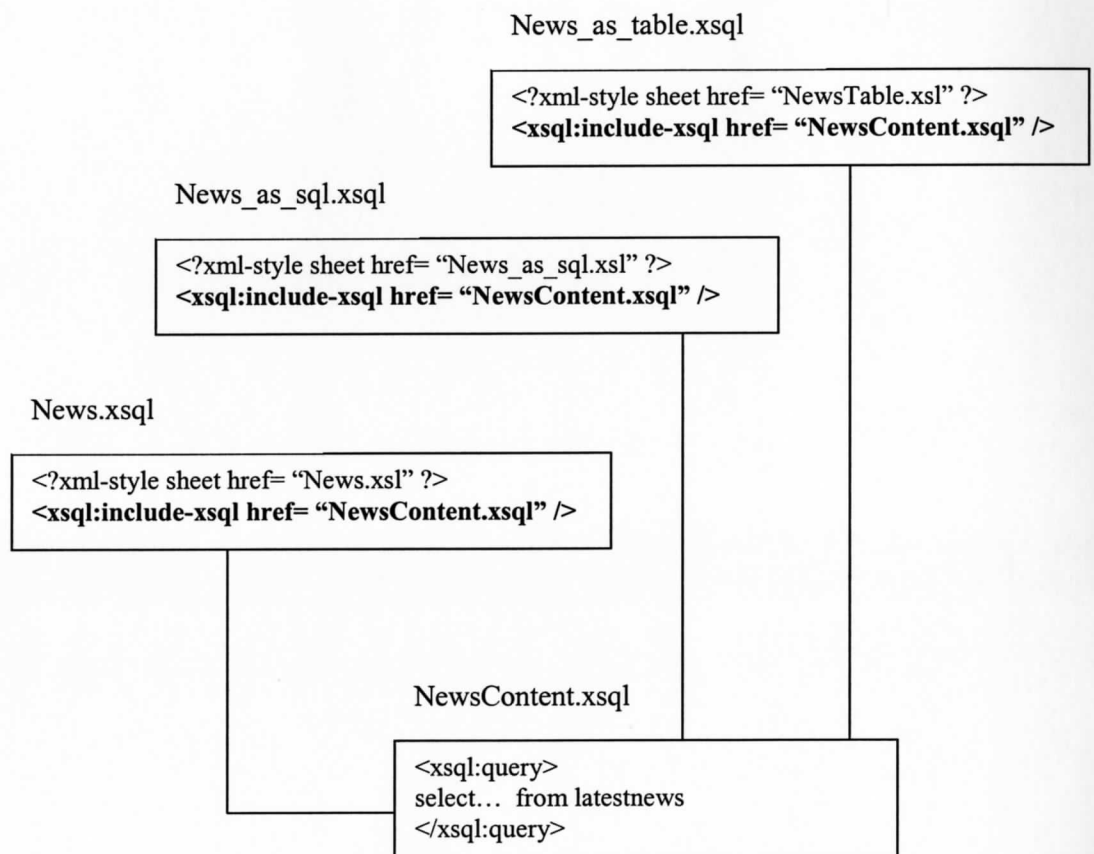


Figure 7: Rendering data in multiple formats with multiple style sheets

The basic steps are as follows:

- NewsContent.xsql is the basic data file that uses XSQL query to produce the XML news information for the news portal. It encloses a SQL query in a pair of <xsql:query> and </xsql:query> tags to dynamically retrieve news content from the Oracle9i database. Besides serving as the “datagrams” to feed other dynamic views, NewsContent.xsql itself also serves as a RAW XML data view for those users who wish to have a look at the hierarchical structure of the document.
- News.xsql includes NewsContent.xsql and transforms it with News.xsl style sheet into an eye-catching HTML page which displays news of different categories in trendy rounded boxes. As shown in Figure 8, this is my default display format for the news portal.

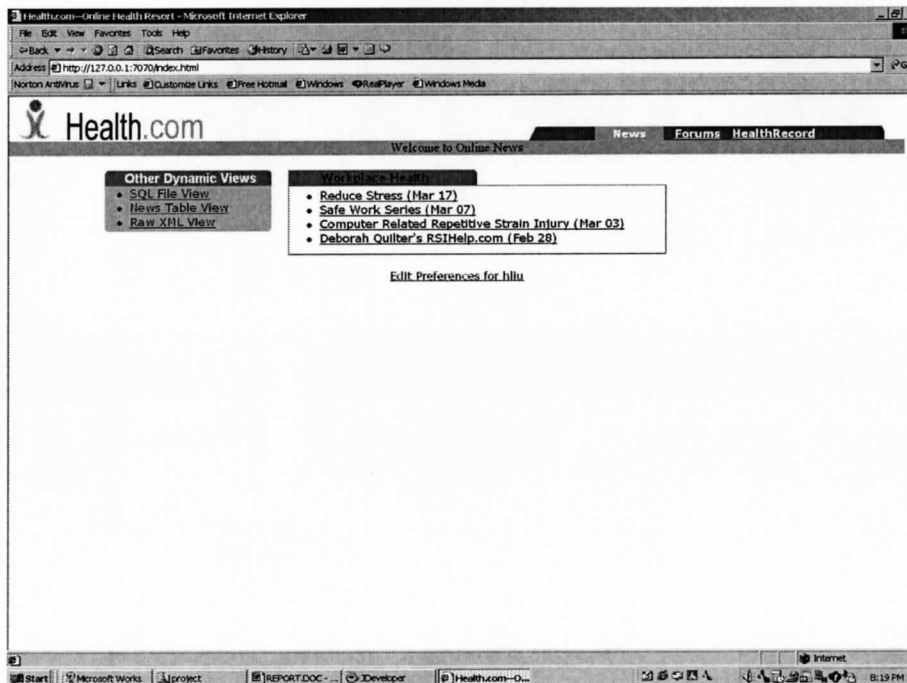


Figure 8: News.xsql – News Box View

- News\_as\_table.xsql includes NewsContent.xsql and transforms it with NewsTable.xsl style sheet into an HTML page which displays news of different categories in tables, as shown in figure 9. This output format is designed for users who prefer to see everything laid out in a plain table.

A noticeable difference between this view and the news box view is the Description field. In the previous view, this field is coded as a “title” attribute of HTML <a> tag, and therefore not displayed until you linger your mouse over the link. But in this view, the Description is put down in the front.

Title	Source	Description	Timestamp
Reduce Stress	<a href="http://www.lifelines.com/Keep/spc.html">http://www.lifelines.com/Keep/spc.html</a>	A how-to programs with immediate results, for managing stress, controlling habits, and improving self-confidence.	Mar 17
Safe Work Series	<a href="http://www.christie.ab.ca/safeworkseries/">http://www.christie.ab.ca/safeworkseries/</a>	A collection of generic and customizable health and safety print and computer based multimedia training (CBT) materials.	Mar 07
Computer Related Repetitive Strain Injury	<a href="http://www.engr.unl.edu/ee/eeshop/rsi.html">http://www.engr.unl.edu/ee/eeshop/rsi.html</a>	As more and more work, education and recreation involves computers, everyone needs to be aware of the hazard of Repetitive Strain Injury to the hands and arms resulting from the use of computer keyboards and mice.	Mar 03
Deborah Quilter's RSIHelp.com	<a href="http://www.rsihelp.com/">http://www.rsihelp.com/</a>	co-author of 'Repetitive Strain Injury: A Computer User's Guide', and an expert/consultant/lecturer in the field.	Feb 28

Figure 9: News\_as\_table.xsql – News Table View

- News\_as\_sql.xsql includes NewsContent.xsql and transforms it with News\_as\_sql.xsl style sheet into a SQL text file. This output format, as shown in Figure 10, provides a handy way for DBAs to populate their own database table.

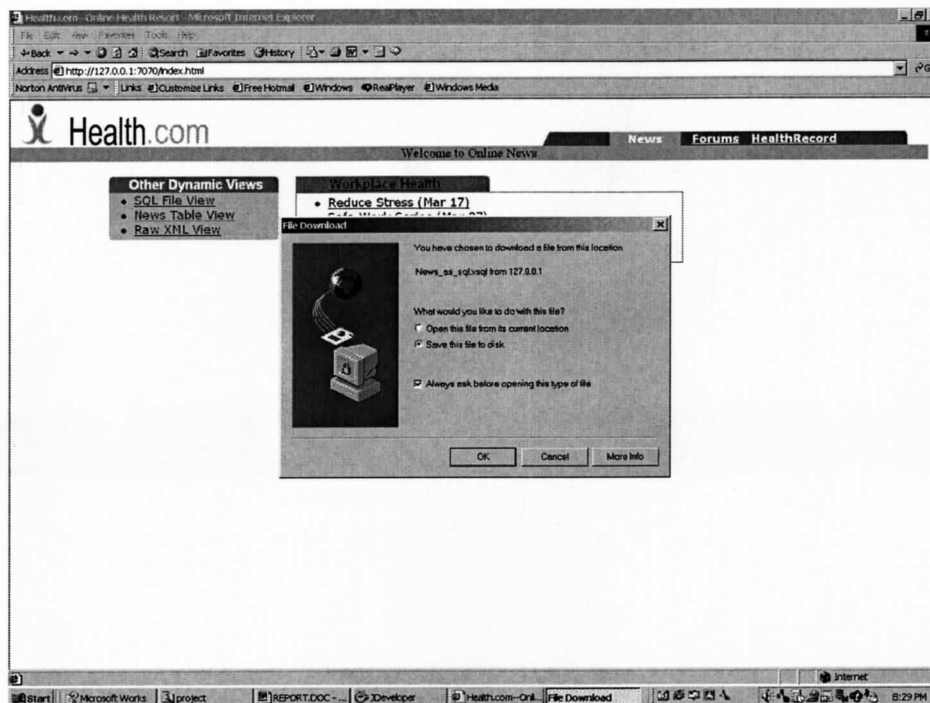


Figure 10: News\_as\_sql.xsal—SQL File View

Click on the “SQL File View”, you are prompted with a dialog box for file download. You can choose to “open the file from this current location” or “save this file to disk”. Either way, you’ll get the following dynamically generated SQL file.

```
insert into site_newsstory values(
/* title */ 'Reduce Stress',
/* url */ 'http://www.lifelines.com/Keep/spc.html',
/* description */ 'A how-to programs with immediate results, for managing stress,
controlling habits, and improving self-confidence. '
);
```

```
insert into site_newsstory values(
/* title */ 'Safe Work Series',
/* url */ 'http://www.christie.ab.ca/safeworkseries/',
/* description */ 'A collection of generic and customizable health and safety print
and computer based multimedia training (CBT) materials.'
);
```

```
insert into site_newsstory values(
/* title */ 'Computer Related Repetitive Strain Injury',
/* url */ 'http://www.engr.unl.edu/ee/eeshop/rsi.html',
/* description */ 'As more and more work, education and recreation involves
computers, everyone needs to be aware of the hazard of Repetitive Strain Injury to the
hands and arms resulting from the use of computer keyboards and mice.'
);
```

```
insert into site_newsstory values(
/* title */ 'Deborah Quilter's RSIHelp.com ',
/* url */ 'http://www.rsihelp.com/',
/* description */ 'co-author of 'Repetitive Strain Injury: A Computer User's Guide',
and an expert/consultant/lecturer in the field.'
);
```

### **Performance Gain**

By modularizing the basic XSQL query page and enclosing it into other pages, we not only increase the life of data but also increase the efficiency of data processing. “The XSQL page processor is designed to handle the nested page request efficiently, so using `<xsql:include-xsql>` does not incur additional HTTP requests or additional database connections if the nested pages use the same named connection as the enclosing page.” (Muench, 2000) The use of XML in the application to display different views for the data boosts performance in many ways, including:

- Content is local to the web server, therefore decreasing network traffic.
- Less load on the database tier since your database is queried only once for the same “datagram”.

- “Your HTML will be created and displayed faster because your application won’t have to query the database and transfer the data over the network over and over again.” (Justin Gunther, 2001)

### **4.3.3 Minimize Website Development and Maintenance Cost**

Besides increasing the life of data, the separation of data and presentation of XML pages also leads to operational simplicity and therefore significantly saves development and maintenance cost of web pages.

Nothing binds the XML input to XSLT style sheet, so there is a very clear separation between data and presentation. Leveraging the dynamic duo of XML to represent rich data structures independent of presentation details and XSLT to transform the data into any other XML, HTML, or text-based output format, the whole process of web page development and maintenance is more modularized and simplified.

The design process starts from differentiating the dynamic pieces of HTML from static ones. The static content will become part of one or more XSLT style sheets, while the dynamic content must come from an XML data source. The mapping logic of XML to XSLT is handled by an XSLT file that iterates through the XML file for the transformation. In this way, we can separate our web site into 3-tier design architecture:

- Presentation, which is stored into generic and specific XSLT layout files
- Logic, stored in XSLT style sheets
- Content, stored in XML data sources

The modularized structure of the web pages attests to this point. As illustrated in Figure 11,

- Presentation:
  - TitledBox.xsl and TitledBox2.xsl are the base-level XSLT named templates that display title and contents inside “rounded boxes”. We can call them from anywhere throughout the entire site.
  - These templates define the two parameters Title and Contents at the top, and right in the middle of the HTML table that contains the layout and images that achieve the rounded box effect.
  - NewsBoxes.xsl, ViewBox.xsl, LoginBoxes.xsl, Nav.xsl and Features.xsl are the second-level XSLT named templates that build on TitledBox.xsl and TitledBox2.xsl by invoking template parameters and passing the appropriate values for the Title and the Contents parameter.
- Logic:
  - News.xsl is the top-level XSLT style sheet that puts everything together for the personal news portal. It is linked with News.xsql to transform this XML file and produce the final look of the news portal home page.
  - HeathRecord.xsl is the top-level XSLT style sheet that puts everything together for the online health record organizer. It is linked with healthrecord.xsql to transform this XML file and produce the final look of the health record home page.
- Content:

- News.xsql is the XML data source for news portal home page.
- HeathRecord.xsl is the XML data source for health record home page.



# News Portal

News.xml: top-level framework that puts everything together

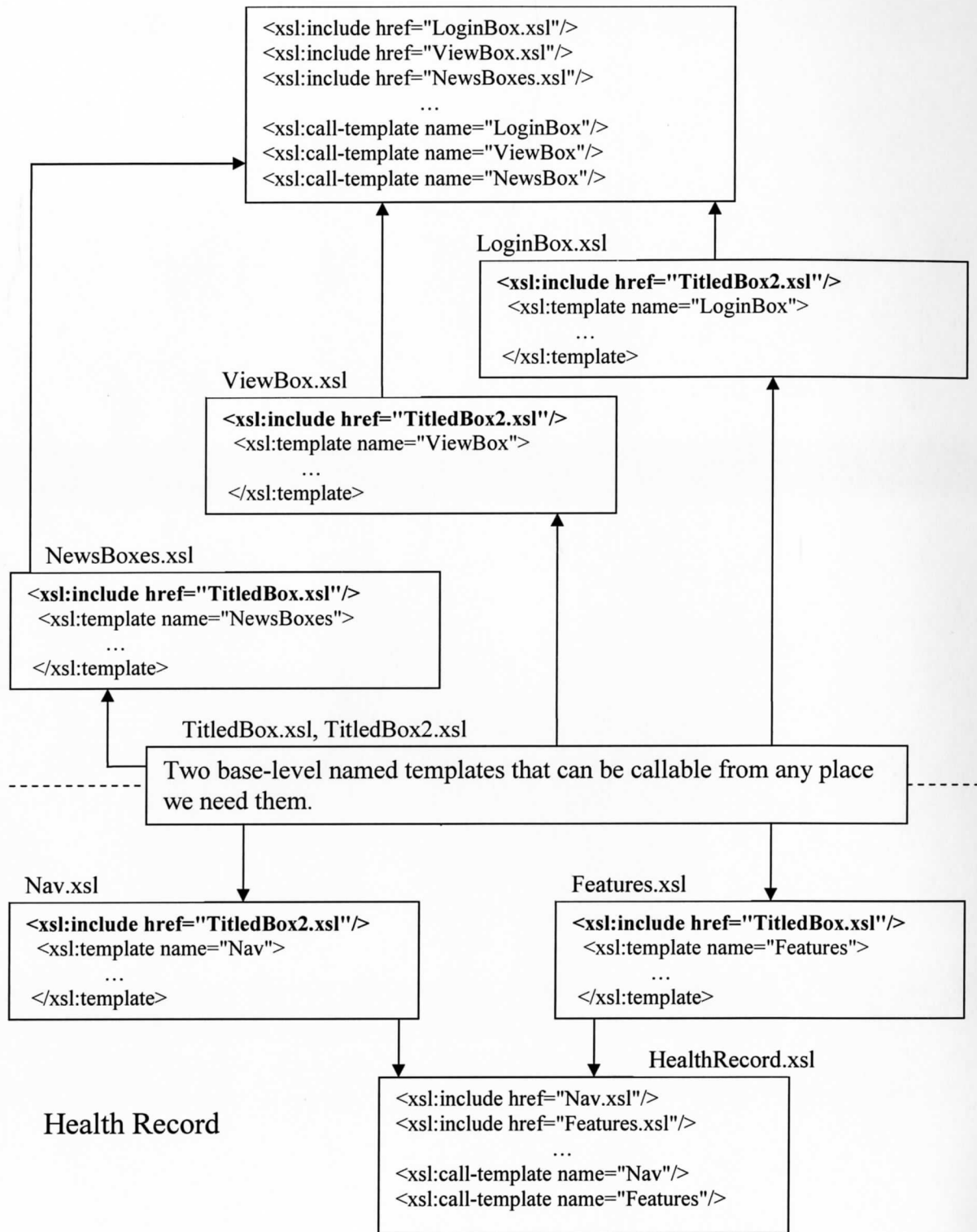


Figure 11: Modularize web page design by using XSLT named templates

This new XML+XSL approach has clear advantages over traditional approaches and brings about the following benefits:

- Parallel Development.

Separation of data from presentation allows programmers and graphic designers to work on different pieces in parallel. Now for the first time, graphic designers don't have to be software engineers in order to update the design in a web page. The XSLT can be developed completely independently of any database code by using static XML files. In other words, the "constituent parts are clearly segregated", "From a project management perspective, this allows every team member to work in parallel, rather than overloading the servlet guru while the remainder of the team struggles to contribute." (Burke, 2000)

- Reusability

One important feature of XSLT is to modularize web page design by building reusable XSLT named templates. An obvious benefit of using templates is the fact that you can reuse the template and so save a lot of redundant work. Another benefit realized is that templates can be created and maintained by non-technical personnel because templates are simply presentation pages written in HTML.

- Maintainability and Cross-site Consistency

As the templates TitledBox and Titled2Box are reused to build the other presentation layers, it is rather easy to create another look of the whole website without affecting any source code of the data pages. The use of named templates also greatly improves the cross-site consistency given that all pages will contain *exactly* the same look

allows these templates to be defined in one place so that they can be easily changed and re-applied.

## 5 Conclusion

XML makes the concept of dynamic document a real possibility. By combining XML with XSLT and SQL, information can be repurposed to suit individual needs, which maximizes the use of data. The separation of data from presentation and business logic also enables a whole new approach to web site development and maintenance, which greatly modularizes the process and minimizes the cost.

It is beyond dispute that XML represents the future of Web development. Relational database is by no means going away in the new era. Oracle XML technologies are well suited for building dynamic XML web sites.

The future enhancements for this project might include the following:

- Serve the same data differently depending on what kind of device makes the request: browser, cell phone, PDA, etc.
- Acquire XML-based information over the Web and store it in our database. In this way, our database is automatically updated to store the latest information.

## References:

Bannerjee, Sandeepan, 2001. "Running Your Enterprise with Oracle 9i", Oracle Open World Conference.

Bradley, Neil, 2000. *The XML Companion*, London, Great Britain: Addison-Wesley.

Bosak, Jon, 1997. "XML, Java, and the future of the Web," [www.ibiblio.org/pub/sun-info/standards/xml/why/xmlapps.html](http://www.ibiblio.org/pub/sun-info/standards/xml/why/xmlapps.html).

Greenspan, Robyn, 2002. "Cyberchondria is Spreading," [cyberatlas.internet.com/markets/healthcare/print/010101\\_115101100.html](http://cyberatlas.internet.com/markets/healthcare/print/010101_115101100.html).

Gunther, Justin, 2001. "Using XML to Supercharge Website Performance," [www.sql-server-performance.com/jb\\_xml\\_performance.asp](http://www.sql-server-performance.com/jb_xml_performance.asp).

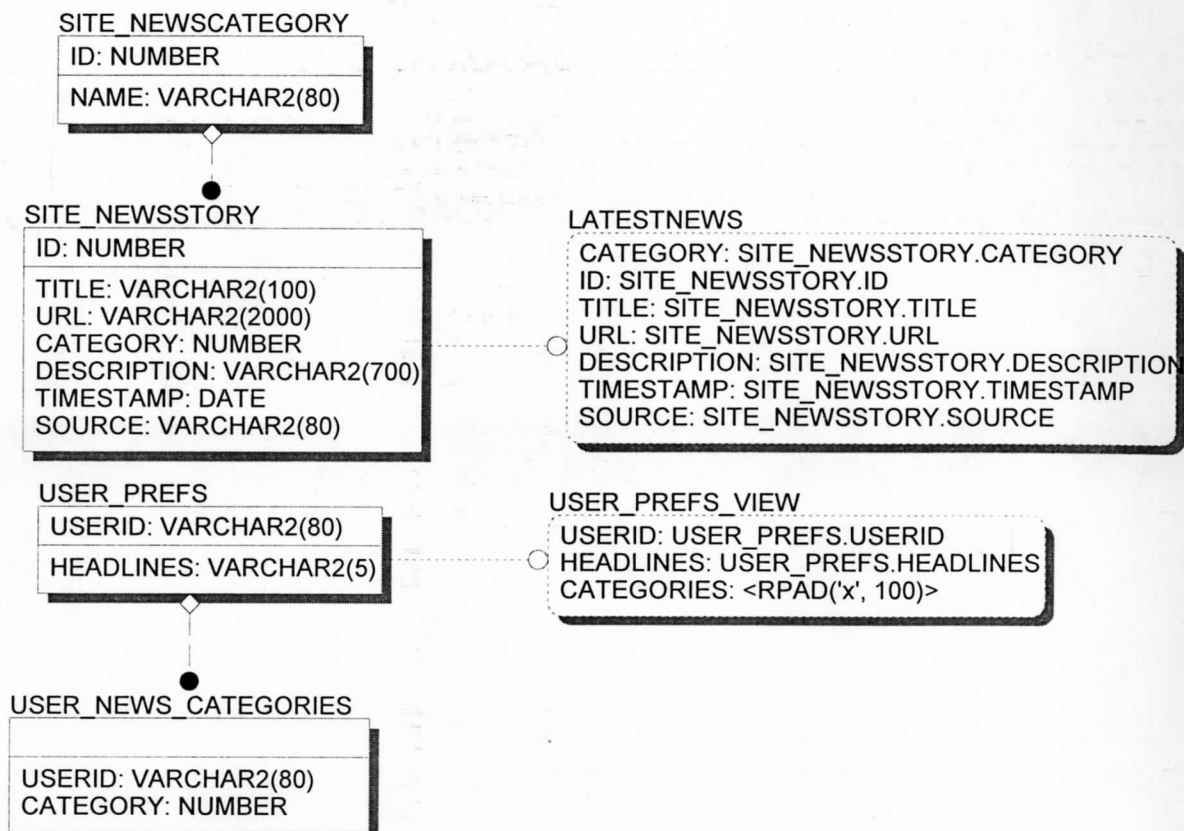
Karpinski, Richard, 2000. "Databases, Tools Push XML into Enterprise," [www.internetwk.com/news1198/news111698-3.htm](http://www.internetwk.com/news1198/news111698-3.htm).

Marchal, Benoit, 2000. *Applied XML Solutions*, Sams Publishing.

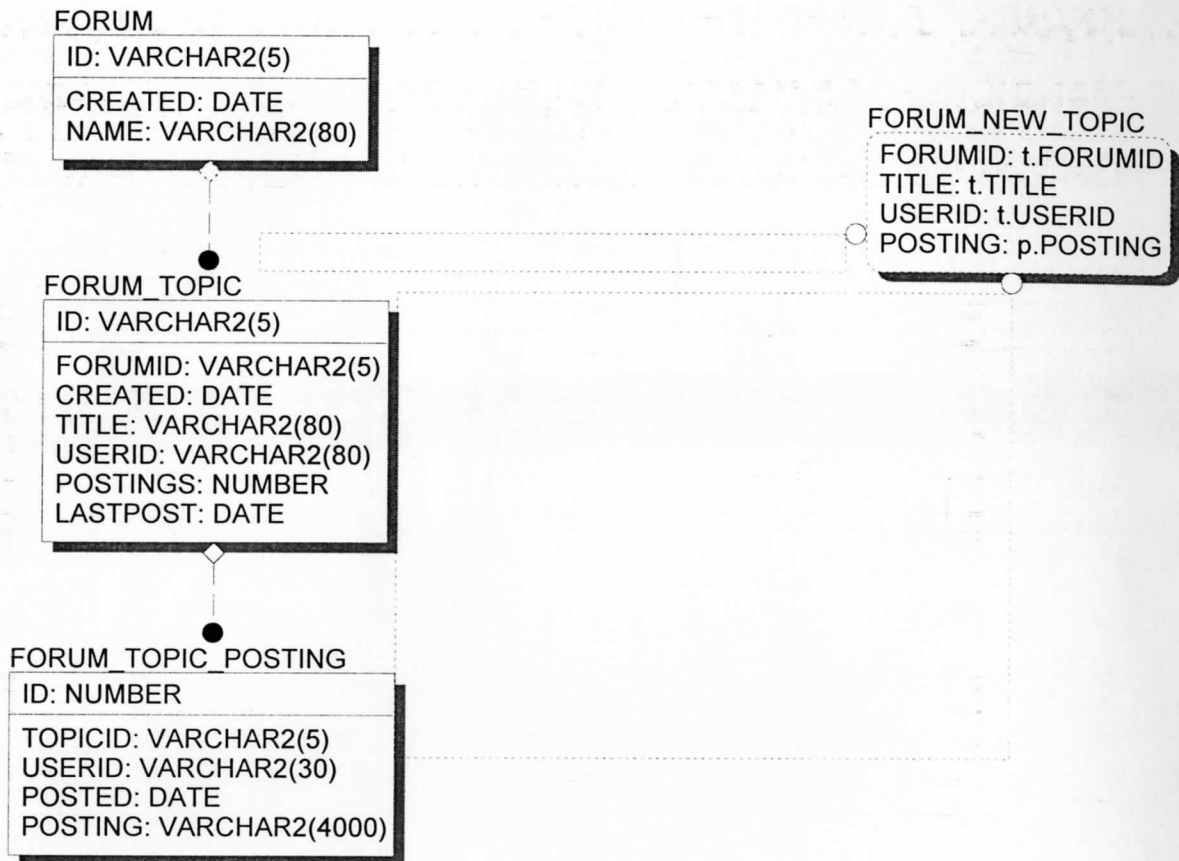
Muench, Steve, 2000. *Building Oracle XML Applications*, Sebastopol, CA: O'Reilly.

Wardley, Andy, 2000. "Building and Managing Web Sites with the Template Toolkit", [template-toolkit.org/tpc4](http://template-toolkit.org/tpc4).

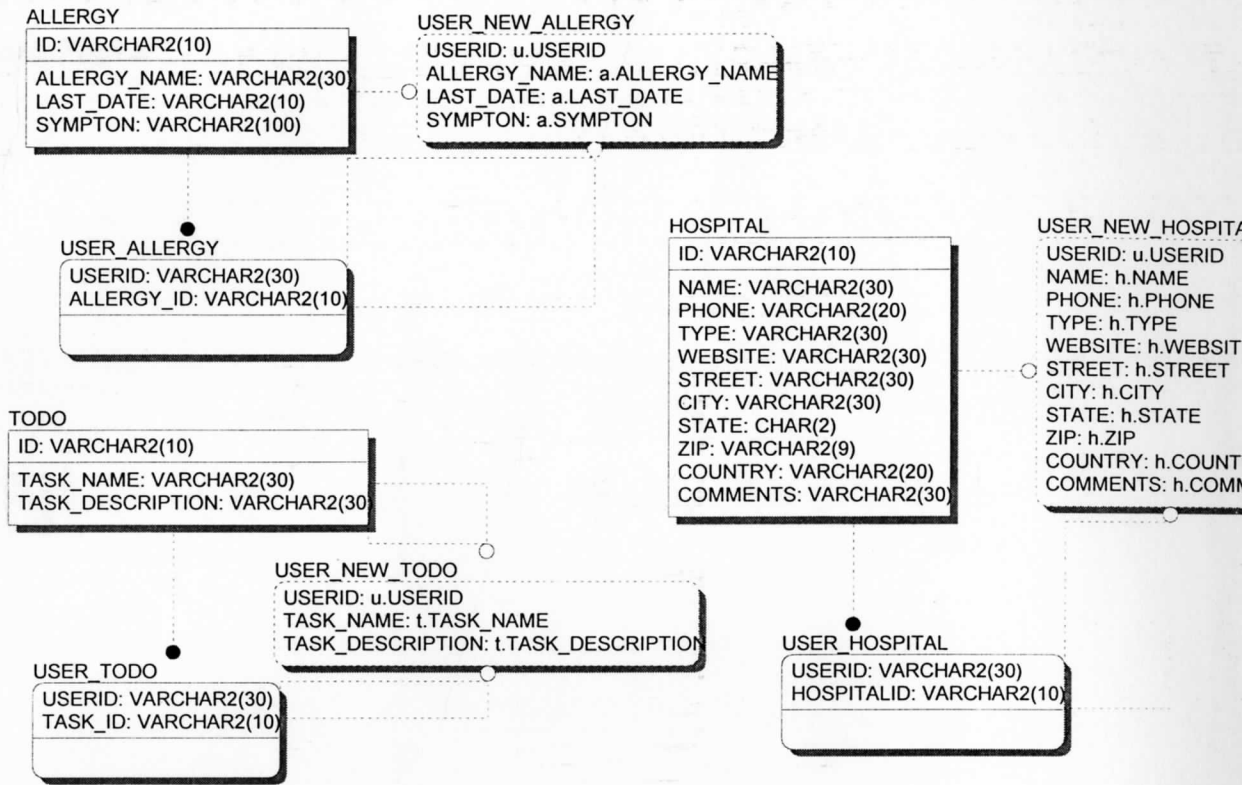
## Appendix A: Entity Relational Diagram



ERD for Personalized News Portal



ERD for Discussion Forums



ERD for Health Record Organizer



## addAllergy.xsql

```
<?xml version="1.0"?>
<!-- addAllergy.xsql: Create form for adding allergies -->
<?xml-stylesheet type="text/xsl" href="HospitalStyle.xsl"?>
<page connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <breadcrumbs>
    <label url="allergy.xsql"> Allergy </label>
    <action> Add an allergy </action>
  </breadcrumbs>
  <dataform target="allergy.xsql" submit="Save">
    <item type="text" name="userid" label="User" size="70">
      <xsql:include-param name="forumuser"/>
    </item>
    <item type="text" name="allergy_name" label="Drug/Food Allergy Name:" size="70">
    </item>
    <item type="text" name="last_date" label="Year of Last Reaction:" size="70">
    </item>
    <item type="checkboxlist" name="sympton" label="Reaction:">
      <ROWSET>
        <ROW>
          <VALUE>Skin rash, consisting of redness, spots, or blisters</VALUE>
          <DISPLAY>Skin rash, consisting of redness, spots, or blisters</DISPLAY>
        </ROW>
        <ROW>
          <VALUE>Hives, consisting of itchy, raised pink or read welts on the
skin</VALUE>
          <DISPLAY>Hives, consisting of itchy, raised pink or read welts on the
skin</DISPLAY>
        </ROW>
        <ROW>
          <VALUE>Mild wheezing or difficulty breathing</VALUE>
          <DISPLAY>Mild wheezing or difficulty breathing</DISPLAY>
        </ROW>
        <ROW>
          <VALUE>Swelling of the face, eyelids, lips, or tongue (angioedema)</VALUE>
          <DISPLAY>Swelling of the face, eyelids, lips, or tongue (angioedema)</DISPLAY>
        </ROW>
        <ROW>
          <VALUE>Severe reation (anaphylaxis)</VALUE>
          <DISPLAY>Severe reation (anaphylaxis)</DISPLAY>
        </ROW>
      </ROWSET>
    </item>
  </dataform>
</page>
```

## addHospital.xsql

```
<?xml version="1.0"?>
<!-- addTodo.xsql: Create form for adding todo -->
<?xml-stylesheet type="text/xsl" href="HospitalStyle.xsl"?>
<page connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <breadcrumbs>
    <label url="hospital.xsql"> Doctors/Clinics </label>
    <action> Add Doctors/Clinics </action>
  </breadcrumbs>
  <dataform target="Hospital.xsql" submit="Save">
    <item type="text" name="userid" label="User">
      <xsql:include-param name="forumuser"/>
    </item>
    <item type="text" name="hospitalName" label="Doctor/Clinic name:">
    </item>
    <item type="text" name="phone" label="Phone numer:">
    </item>
    <item type="list" name="type" label="Type of Clinic/Hospital:">
      <ROWSET>
        <ROW><VALUE></VALUE><DISPLAY>Select the type of Clinic/Hospital</DISPLAY></ROW>
        <ROW><VALUE>Outpatient Clinic</VALUE><DISPLAY>Outpatient Clinic</DISPLAY></ROW>
        <ROW><VALUE>Urgent_Care Clinic</VALUE><DISPLAY>Urgent_Care Clinic</DISPLAY></ROW>
        <ROW><VALUE>Doctor's Office</VALUE><DISPLAY>Doctor's Office</DISPLAY></ROW>
        <ROW><VALUE>Hospital</VALUE><DISPLAY>Hospital</DISPLAY></ROW>
        <ROW><VALUE>Labortory</VALUE><DISPLAY>Labortory</DISPLAY></ROW>
        <ROW><VALUE>Radiology Center</VALUE><DISPLAY>Radiology Center</DISPLAY></ROW>
        <ROW><VALUE>Emergency Room</VALUE><DISPLAY>Emergency Room</DISPLAY></ROW>
        <ROW><VALUE>Other</VALUE><DISPLAY>Other</DISPLAY></ROW>
      </ROWSET>
    </item>
    <item type="text" name="website" label="Website address:">
    </item>
    <item type="text" name="street" label="Street Address:">
    </item>
    <item type="text" name="city" label="City:">
    </item>
    <item type="list" name="state" label="State:">
      <ROWSET>
        <ROW><VALUE></VALUE><DISPLAY>Choose a state</DISPLAY></ROW>
        <ROW><VALUE>AL</VALUE><DISPLAY>Alabama</DISPLAY></ROW>
        <ROW><VALUE>AK</VALUE><DISPLAY>Alaska</DISPLAY></ROW>
        <ROW><VALUE>AZ</VALUE><DISPLAY>Arizona</DISPLAY></ROW>
        <ROW><VALUE>AR</VALUE><DISPLAY>Arkansas</DISPLAY></ROW>
        <ROW><VALUE>CA</VALUE><DISPLAY>California</DISPLAY></ROW>
        <ROW><VALUE>CO</VALUE><DISPLAY>Colorado</DISPLAY></ROW>
        <ROW><VALUE>CT</VALUE><DISPLAY>Conneticut</DISPLAY></ROW>
        <ROW><VALUE>DE</VALUE><DISPLAY>Delaware</DISPLAY></ROW>
        <ROW><VALUE>DC</VALUE><DISPLAY>District Of Columbia</DISPLAY></ROW>
        <ROW><VALUE>FL</VALUE><DISPLAY>Florida</DISPLAY></ROW>
        <ROW><VALUE>GA</VALUE><DISPLAY>Georgia</DISPLAY></ROW>
        <ROW><VALUE>HI</VALUE><DISPLAY>Hawaii</DISPLAY></ROW>
        <ROW><VALUE>ID</VALUE><DISPLAY>Idaho</DISPLAY></ROW>
        <ROW><VALUE>IL</VALUE><DISPLAY>Illinois</DISPLAY></ROW>
        <ROW><VALUE>IN</VALUE><DISPLAY>Indiana</DISPLAY></ROW>
        <ROW><VALUE>IA</VALUE><DISPLAY>Iowa</DISPLAY></ROW>
        <ROW><VALUE>KS</VALUE><DISPLAY>Kansas</DISPLAY></ROW>
        <ROW><VALUE>KY</VALUE><DISPLAY>Kentucky</DISPLAY></ROW>
        <ROW><VALUE>LA</VALUE><DISPLAY>Louisiana</DISPLAY></ROW>
        <ROW><VALUE>ME</VALUE><DISPLAY>Maine</DISPLAY></ROW>
        <ROW><VALUE>MD</VALUE><DISPLAY>Maryland</DISPLAY></ROW>
        <ROW><VALUE>MA</VALUE><DISPLAY>Massachusetts</DISPLAY></ROW>
        <ROW><VALUE>MI</VALUE><DISPLAY>Michigan</DISPLAY></ROW>
        <ROW><VALUE>MN</VALUE><DISPLAY>Minnesota</DISPLAY></ROW>
        <ROW><VALUE>MS</VALUE><DISPLAY>Mississippi</DISPLAY></ROW>
        <ROW><VALUE>MO</VALUE><DISPLAY>Missouri</DISPLAY></ROW>
        <ROW><VALUE>MT</VALUE><DISPLAY>Montana</DISPLAY></ROW>
        <ROW><VALUE>NE</VALUE><DISPLAY>Nebraska</DISPLAY></ROW>
        <ROW><VALUE>NV</VALUE><DISPLAY>Nevada</DISPLAY></ROW>
        <ROW><VALUE>NH</VALUE><DISPLAY>New Hampshire</DISPLAY></ROW>
        <ROW><VALUE>NJ</VALUE><DISPLAY>New Jersey</DISPLAY></ROW>
        <ROW><VALUE>NM</VALUE><DISPLAY>New Mexico</DISPLAY></ROW>
        <ROW><VALUE>NY</VALUE><DISPLAY>New York</DISPLAY></ROW>
      </ROWSET>
    </item>
  </dataform>
</page>
```

## addHospital.xsql

```
<ROW><VALUE>NC</VALUE><DISPLAY>North Carolina</DISPLAY></ROW>
<ROW><VALUE>ND</VALUE><DISPLAY>North Dakota</DISPLAY></ROW>
<ROW><VALUE>OH</VALUE><DISPLAY>Ohio</DISPLAY></ROW>
<ROW><VALUE>OK</VALUE><DISPLAY>Oklahoma</DISPLAY></ROW>
<ROW><VALUE>OR</VALUE><DISPLAY>Oregon</DISPLAY></ROW>
<ROW><VALUE>PA</VALUE><DISPLAY>Pennsylvania</DISPLAY></ROW>
<ROW><VALUE>PR</VALUE><DISPLAY>Puerto Rico</DISPLAY></ROW>
<ROW><VALUE>RI</VALUE><DISPLAY>Rhode Island</DISPLAY></ROW>
<ROW><VALUE>SC</VALUE><DISPLAY>South Carolina</DISPLAY></ROW>
<ROW><VALUE>SD</VALUE><DISPLAY>South Dakota</DISPLAY></ROW>
<ROW><VALUE>TN</VALUE><DISPLAY>Tennessee</DISPLAY></ROW>
<ROW><VALUE>TX</VALUE><DISPLAY>Texas</DISPLAY></ROW>
<ROW><VALUE>UT</VALUE><DISPLAY>Utah</DISPLAY></ROW>
<ROW><VALUE>VT</VALUE><DISPLAY>Vermont</DISPLAY></ROW>
<ROW><VALUE>VI</VALUE><DISPLAY>Virgin Islands</DISPLAY></ROW>
<ROW><VALUE>VA</VALUE><DISPLAY>Virginia</DISPLAY></ROW>
<ROW><VALUE>WA</VALUE><DISPLAY>Washington</DISPLAY></ROW>
<ROW><VALUE>WV</VALUE><DISPLAY>West Virginia</DISPLAY></ROW>
<ROW><VALUE>WI</VALUE><DISPLAY>Wisconsin</DISPLAY></ROW>
<ROW><VALUE>WY</VALUE><DISPLAY>Wyoming</DISPLAY></ROW>
<ROW><VALUE>AA</VALUE><DISPLAY>Outside USA</DISPLAY></ROW>
</ROWSET>
</item>
<item type="text" name="country" label="Country:">
</item>
<item type="textarea" name="comments" size="40" label="Comments:">
</item>
</dataform>
</page>
```

## addTodo.xsql

```
<?xml version="1.0"?>
<!-- addTodo.xsql: Create form for adding todo -->
<?xml-stylesheet type="text/xsl" href="HospitalStyle.xsl"?>
<page connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <breadcrumbs>
    <label url="todo.xsql"> To-Do's </label>
    <action> Add To-Do </action>
  </breadcrumbs>
  <dataform target="todo.xsql" submit="Save">
    <item type="text" name="userid" label="User">
      <xsql:include-param name="forumuser"/>
    </item>
    <item type="text" name="task_name" label="Add Task Name:">
    </item>
  </dataform>
</page>
```

## Allergy.SQL

```
drop table user_allergy;
drop table allergy;
drop sequence allergy_id;
create sequence allergy_id;

create table allergy(
  id          varchar2(10),
  allergy_name varchar2(30),
  last_date  varchar2(10),
  symptom    varchar2(100),
  constraint allergy_id primary key (id)
);

create or replace trigger allergy_id_trig
before insert on allergy for each row
begin
  select allergy_id.nextval
  into :new.id
  from dual;
end;
/

create table user_allergy(
  userid      varchar2(30),
  allergy_id  varchar2(10),
  constraint user_allergy_pk primary key (userid, allergy_id),
  constraint user_allergy_fk foreign key (allergy_id) references allergy (id)
);

create or replace view user_new_allergy as
select u.userid, a.allergy_name, a.last_date, a.symptom
  from user_allergy u, allergy a
 where u.allergy_id = a.id;

create or replace trigger user_new_allergy_ins
instead of insert on user_new_allergy for each row
declare
  newAllergyId VARCHAR2(10);
begin
  insert into allergy(allergy_name, last_date, symptom)
  values (:new.allergy_name, :new.last_date, :new.symptom)
  returning id into newAllergyId;
  insert into user_allergy(userid, allergy_id)
  values (:new.userid, newAllergyId);
end;
/
commit;
```

## allergy.xsql

```
<?xml version="1.0"?>
<!-- allergy.xsql: show allergy information for a user -->
<?xml-stylesheet type="text/xsl" href="hospitalStyle.xsl"?>
<page connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <!--insert the new allergy into user_new_allergy view-->
  <xsql:insert-request table="user_new_allergy"
    transform="transAllergy.xsl"/>
  <xsql:delete-request table="user_new_allergy"
    transform="transAllergy.xsl"/>
  <xsql:set-cookie name="forumuser" value="{@youremail}"
    ignore-empty-value="yes" only-if-unset="yes"/>
  <xsql:set-page-param name="forumuser" value="{@youremail}"
    ignore-empty-value="yes"/>
  <xsql:include-param name="forumuser"/>
  <!-- Include request params so Actions can check for forumuser cookie -->
  <breadcrumbs>
    <label> Allergy </label>
  </breadcrumbs>
  <data>
    <xsql:query max-rows="{@paging-max}" skip-rows="{@paging-skip}">
      select allergy_name, last_date, symptom
      from allergy a, user_allergy u
      where a.id=u.allergy_id
      and u.userid= '{@forumuser}'
    </xsql:query>
  </data>
  <actions>
    <link page="addAllergy.xsql" label="Add an Allergy" >
      <xsql:include-param name="forumuser"/>
    </link>
  </actions>
</page>
```

## Forums.xml

```
<?xml version="1.0"?>  
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">  
  <xsl:import href="ForumStyle.xml"/>  
</xsl:stylesheet>
```

## Forums.xsql

```
<?xml version="1.0"?>
<!-- Forums.xsql: Home Page Showing all Discussion Forums -->
<?xml-stylesheet type="text/xsl" href="ForumStyle.xsl"?>
<page connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <xsql:set-cookie name="forumuser" value="{@userid}"
    ignore-empty-value="yes" only-if-unset="yes"/>
  <xsql:action handler="Paging" rows-per-page="5">
    select count(id) from forum
  </xsql:action>
  <breadcrumbs/>
  <actions>
    <link page="TodaysActiveTopics.xsql" label="Active Topics Today"/>
    <link page="Search.xsql" label="Search"/>
  </actions>
  <data>
    <xsql:query skip-rows="{@paging-skip}" max-rows="{@paging-max}">
      select f.id as h_id,
             f.name as "Forum",
             sum(t.postings) as "Posts"
      from forum f,
           forum_topic t
      where t.forumid = f.id
      group by f.id,f.name
      order by name
    </xsql:query>
  </data>
</page>
```



```

<?xml version="1.0"?>
<!-- ForumTopics.xsql: Show list of topics for a forum -->
<?xml-stylesheet type="text/xsl" href="ForumStyle.xsl"?>
<page connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <xsql:include-request-params/>
  <xsql:insert-request table="forum_new_topic"
    transform="InsertPostedTopic.xsl"/>
  <!-- Include request params so Actions can check for forumuser cookie -->
  <xsql:action handler="Paging" rows-per-page="5" url-params="id">
    select count(t.id) as "total"
    from forum_topic t
    where forumid = {@id}
  </xsql:action>
  <breadcrumbs>
    <xsql:query>
      select name as "forumname", id as "forumid"
      from forum
      where id = {@id}
    </xsql:query>
  </breadcrumbs>
  <actions>
    <link page="EnterForumTopic.xsql" label="Enter a New Topic" login="yes">
      <xsql:include-param name="id"/>
    </link>
    <link page="Search.xsql" label="Search">
      <xsql:include-param name="id"/>
    </link>
  </actions>
  <data>
    <xsql:query max-rows="{@paging-max}" skip-rows="{@paging-skip}">
      select t.id          as h_id,
             t.title       as "Topic",
             t.userid      as "Started_By",
             postings-1    as "Replies",
             to_char(lastpost, 'dd Mon YYYY, hh24:mi') as "Last_Post"
      from forum_topic t
      where forumid = {@id}
      order by 5 desc
    </xsql:query>
  </data>
</page>

```

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <!--
  | UtilBreadCrumbs.xsl: Transform <breadcrumbs> structural info
  | into HTML breadcrumbs presentation.
  +-->
  <xsl:variable name="i" select="'images/'"/>
  <xsl:template match="breadcrumbs">
    <center>
      <span class="breadcrumbs">
        <a href="HealthRecord.xsql">Health Records</a>#160;
        #160;
        <xsl:choose>
          <xsl:when test="//action">
            <a>
              <xsl:attribute name="href">
                <xsl:value-of select="//label/@url"/>
              </xsl:attribute>
              <xsl:value-of select="//label"/>
            </a>
            #160;
            <xsl:value-of select="//action"/>
          </xsl:when>
          <xsl:otherwise>
            <xsl:value-of select="//label"/>
          </xsl:otherwise>
        </xsl:choose>
      </span>
    </center>
  </xsl:template>
</xsl:stylesheet>
```

## HealthPageStructure.xsl

```
<!-- ForumPageStructure.xsl: Transform the Abstract Structure of a <page> -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:import href="UtilData.xsl"/>
  <xsl:import href="ColumnOverrides.xsl"/>
  <xsl:include href="HealthBreadcrumbs.xsl"/>
  <xsl:include href="UtilActions.xsl"/>
  <xsl:include href="UtilDataForm.xsl"/>
  <xsl:include href="UtilPaging.xsl"/>
  <xsl:template match="page">
    <xsl:apply-templates select="breadcrumbs"/>
    <br/>
    <xsl:apply-templates select="dataform"/>
    <xsl:apply-templates select="data"/>
    <br/>
    <xsl:apply-templates select="actions"/>
    <xsl:if test="paging">
      <hr/>
      <center><xsl:apply-templates select="paging"/></center>
    </xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

## HealthRecord.xsl

```
<?xml version="1.0"?>
<!--HealthRecord.xsl: Format the HealthRecord Home page -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" indent="no"/>
  <xsl:include href="nav.xsl"/>
  <xsl:include href="Features.xsl"/>
  <xsl:template match="/">
    <html>
      <head><link rel="stylesheet" type="text/css" href="Forum.css" /></head>
      <body class="page">
        <center>
          <table border="0" width="80%">
            <tr>
              <td valign="top" width="200">
                <xsl:call-template name="nav"/>
              </td>
              <td>&#160;&#160;</td>
              <td valign="top" width="800">
                <xsl:call-template name="Features"/>
              </td>
            </tr>
          </table>
        </center>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

## HealthRecord.xsql

```
<?xml version="1.0"?>
<!--HealthRecord.xsql Home Page for HealthRecord-->
<?xml-stylesheet type="text/xsl" href="HealthRecord.xsl"?>
<page connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <xsql:set-cookie name="forumuser" value="{@youremail}"
    ignore-empty-value="yes" only-if-unset="yes"/>
  <xsql:set-page-param name="forumuser" value="{@youremail}"
    ignore-empty-value="yes"/>
  <xsql:include-param name="forumuser"/>
</page>
```

## Hospital.SQL

```
drop table user_hospital;
drop table hospital;
drop sequence hospitalid;
create sequence hospitalid;

create table hospital(
  id          varchar2(10),
  name       varchar2(30),
  phone      varchar2(20),
  type       varchar2(30),
  website    varchar2(30),
  street     varchar2(30),
  city       varchar2(30),
  state      char(2),
  zip        varchar2(9),
  country    varchar2(20),
  comments   varchar2(30),
  constraint hospital_pk primary key (id)
);

create or replace trigger hospital_id_trig
before insert on hospital for each row
begin
  select hospitalid.nextval
  into :new.id
  from dual;
end;
/

create table user_hospital(
  userid     varchar2(30),
  hospitalid varchar2(10),
  constraint userid_pk primary key (userid, hospitalid),
  constraint user_hospital_fk foreign key (hospitalid) references hospital (id)
);

create or replace view user_new_hospital as
select u.userid, h.name, h.phone, h.type, h.website, h.street, h.city, h.state, h.zip,
       h.country,
       h.comments
  from user_hospital u, hospital h
 where u.hospitalid = h.id;

create or replace trigger user_new_hospital_ins
instead of insert on user_new_hospital for each row
declare
  newHospitalId VARCHAR2(10);
begin
  insert into hospital(name, phone, type, website, street, city, state, zip, country,
  comments)
  values (:new.name, :new.phone, :new.type, :new.website, :new.street, :new.city,
  :new.state,
          :new.zip, :new.country, :new.comments)
  returning id into newHospitalId;
  insert into user_hospital(userid, hospitalid)
  values (:new.userid, newHospitalId);
end;
/
commit;
```

## hospital.xsql

```
<?xml version="1.0"?>
<!-- hospital.xsql: show hospital information for a user -->
<?xml-stylesheet type="text/xsl" href="hospitalStyle.xsl"?>
<page connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <!--insert the new hospital into user_new_hospital view-->
  <xsql:insert-request table="user_new_hospital"
    transform="transHospital.xsl"/>
  <xsql:set-cookie name="forumuser" value="{@youremail}"
    ignore-empty-value="yes" only-if-unset="yes"/>
  <xsql:set-page-param name="forumuser" value="{@youremail}"
    ignore-empty-value="yes"/>
  <xsql:include-param name="forumuser"/>
  <!-- Include request params so Actions can check for forumuser cookie -->
  <breadcrumbs>
    <label>
      Doctors/Clinics
    </label>
  </breadcrumbs>
  <data>
    <xsql:query max-rows="{@paging-max}" skip-rows="{@paging-skip}">
      select userid, name, type, phone, website, street, city, state, zip, country,
comments
      from hospital h, user_hospital u
      where h.id=u.hospitalid
      and u.userid= '{@forumuser}'
    </xsql:query>
  </data>
  <actions>
    <link page="addHospital.xsql" label="Add A Doctor/Clinic" >
      <xsql:include-param name="forumuser"/>
    </link>
  </actions>
</page>
```

## HealthBreadcrumbs.xsl

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <!--
  | UtilBreadCrumbs.xsl: Transform <breadcrumbs> structural info
  | into HTML breadcrumbs presentation.
  +-->
  <xsl:variable name="i" select="'images/'"/>
  <xsl:template match="breadcrumbs">
    <center>
      <span class="breadcrumbs">
        <a href="HealthRecord.xsql">Health Records</a>#160;
        #160;
        <xsl:choose>
          <xsl:when test="//action">
            <a>
              <xsl:attribute name="href">
                <xsl:value-of select="//label/@url"/>
              </xsl:attribute>
              <xsl:value-of select="//label"/>
            </a>
            #160;
            <xsl:value-of select="//action"/>
          </xsl:when>
          <xsl:otherwise>
            <xsl:value-of select="//label"/>
          </xsl:otherwise>
        </xsl:choose>
      </span>
    </center>
  </xsl:template>
</xsl:stylesheet>
```



## insertForumResponse.xsql

```
<?xml version="1.0"?>
<!-- ForumTopics.xsql: Show list of topics for a forum -->
<?xml-stylesheet type="text/xsl" href="ForumStyle.xsl"?>
<page connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <xsql:insert-request table="forum_new_topic"
    transform="InsertPostedTopic.xsl"/>
  <!-- Include request params so Actions can check for forumuser cookie -->
  <xsql:include-request-params/>
  <xsql:action handler="Paging" rows-per-page="5" url-params="id">
    select count(t.id) as "total"
    from forum_topic t
    where forumid = {@id}
  </xsql:action>
  <breadcrumbs>
    <xsql:query>
      select name as "forumname", id as "forumid"
      from forum
      where id = {@id}
    </xsql:query>
  </breadcrumbs>
  <actions>
    <link page="EnterForumTopic.xsql" label="Enter a New Topic" login="yes">
      <xsql:include-param name="id"/>
    </link>
    <link page="Search.xsql" label="Search">
      <xsql:include-param name="id"/>
    </link>
  </actions>
  <data>
    <xsql:query max-rows="{@paging-max}" skip-rows="{@paging-skip}">
      select t.id          as h_id,
             t.title      as "Topic",
             t.userid     as "Started_By",
             postings-1   as "Replies",
             to_char(lastpost, 'dd Mon YYYY, hh24:mi') as "Last_Post"
      from forum_topic t
      where forumid = {@id}
      order by 5 desc
    </xsql:query>
  </data>
</page>
```

## InsertPostedReply.xsl

```
<!-- Created by GenerateInsertTransform.xsl -->
<!-- InsertPostedReply.xsl: Insert transform for forum_topic_posting table -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <!-- Transform /request/parameters into ROWSET for forum_topic_posting -->
  <xsl:template match="/">
    <ROWSET>
      <!-- XPath for repeating source rows -->
      <xsl:for-each select="/request/parameters">
        <ROW>
          <!-- Don't need this one because insert trigger handles its assignment
          <ID>
            <xsl:value-of select="ID"/>
          </ID>
        -->
          <TOPICID>
            <xsl:value-of select="id"/>
          </TOPICID>
          <USERID>
            <xsl:value-of select="userid"/>
          </USERID>
          <!-- Don't need this one because insert trigger handles its assignment
          <POSTED>
            <xsl:value-of select="POSTED"/>
          </POSTED>
        -->
          <POSTING>
            <xsl:value-of select="posting"/>
          </POSTING>
        </ROW>
      </xsl:for-each>
    </ROWSET>
  </xsl:template>
</xsl:stylesheet>
```

## InsertPostedTopic.xsl

```
<!-- InsertPostedTopic.xsl: Insert transform for forum_new_topic table -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <!-- Transform /request/parameters into ROWSET for forum_new_topic view-->
  <xsl:template match="/">
    <ROWSET>
      <!-- XPath for repeating source rows -->
      <xsl:for-each select="/request/parameters">
        <ROW>
          <FORUMID>
            <xsl:value-of select="id"/>
          </FORUMID>
          <TITLE>
            <xsl:value-of select="title"/>
          </TITLE>
          <USERID>
            <xsl:value-of select="userid"/>
          </USERID>
          <POSTING>
            <xsl:value-of select="posting"/>
          </POSTING>
        </ROW>
      </xsl:for-each>
    </ROWSET>
  </xsl:template>
</xsl:stylesheet>
```

## InsertUserPrefs.xml

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<!-- InsertUserPref.xml: Insert transform for user_prefs_view -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <ROWSET>
      <ROW>
        <xsl:if test="request/parameters/userid">
          <xsl:for-each select="request/parameters">
            <ROW>
              <USERID>
                <xsl:value-of select="userid"/>
              </USERID>
              <HEADLINES>
                <xsl:value-of select="headlines"/>
              </HEADLINES>
              <CATEGORIES>
                <xsl:choose>
                  <xsl:when test="row/categories">
                    <xsl:for-each select="row/categories">
                      <xsl:value-of select="."/>
                      <xsl:if test="position() != last()">
                        <xsl:text> </xsl:text>
                      </xsl:if>
                    </xsl:for-each>
                  </xsl:when>
                  <xsl:otherwise>
                    <xsl:value-of select="categories"/>
                  </xsl:otherwise>
                </xsl:choose>
              </CATEGORIES>
            </ROW>
          </xsl:for-each>
        </xsl:if>
      </ROWSET>
    </xsl:template>
  </xsl:stylesheet>
```

```

NAV.XSL
<!-- nav.xsl: Format left navigation using TitledBoxes -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:include href="TitledBox2.xsl"/>
  <xsl:template name="nav">
    <xsl:call-template name="TitledBox2">
      <xsl:with-param name="Title">
        <!-- Static title of a bolded "Health Record" -->
        <b>My Health Record</b>
      </xsl:with-param>
      <xsl:with-param name="Contents">
        <table>
          <tr>
            <td><a href="Hospital.xsql">Doctors/Clinics</a></td>
          </tr>
          <tr>
            <td><a href="allergy.xsql">Drug/Food Allergies</a></td>
          </tr>
          <tr>
            <td>Family History</td>
          </tr>
          <tr>
            <td><a href="todo.xsql">To-Do's</a></td>
          </tr>
        </table>
      </xsl:with-param>
    </xsl:call-template>
  </xsl:template>
</xsl:stylesheet>

```

## News.css

```
table { font-family: Verdana, Arial; font-size:11pt}
table SPAN { font-family: Verdana, Arial; font-size:11pt}
table TD { font-family: Verdana, Arial; font-size:11pt;}
table TH {      bgcolor: #0099ff;
             color: white;
             font-family: Arial;
             font-weight:bold;
             font-size: 12pt; }

table BR {font-size: 4pt}
table .date { font-family: Verdana, Arial; font-size: 9pt }
table .code { font-family: Andale Mono, monospace; font-size: 11pt }
table .title {background-color:0099ff; color:white; font-family: Arial; font-size:14pt}
table .r1 {background-color: #f7f7e7; color: black}
table .r0 {background-color: #f9f9f9; color: black}
P { font-family: Verdana, Arial; font-size:11pt}
A:hover {color:"#FF3300"}

.breadcrumbs {font-family: Verdana, Arial; font-weight:bold}
body {font-family: Verdana, Arial; background-color:white}
.actions {font-family: Verdana, Arial; background-color:#ffcc00}
.c0 {}
.c1 {}
hr {size:2; noshade; width:605; color:#ffcc00}
```

## News.sql

```
drop table site_newsstory;
drop table site_newscategory;
drop sequence site_newsstoryid;
create sequence site_newsstoryid start with 1000;
create table site_newscategory(
  id NUMBER,
  name VARCHAR2(80),
  constraint site_newscatpk primary key (id)
);
create table site_newsstory(
  id NUMBER,
  title VARCHAR2(100) NOT NULL ,
  url VARCHAR2(2000) NOT NULL ,
  category NUMBER,
  description VARCHAR2(700),
  timestamp DATE,
  source VARCHAR2(80),
  constraint site_newsstorypk primary key (id),
  constraint story_in_category foreign key (category) references site_newscategory
);
insert into site_newscategory values (
  1, 'Fitness'
);
insert into site_newscategory values (
  2, 'Nutrition and Food'
);
insert into site_newscategory values (
  3, 'Weight Management'
);
insert into site_newscategory values (
  4, 'Workplace Health'
);
insert into site_newscategory values (
  5, 'Extra'
);

commit;
create or replace view latestnews as
  SELECT category, id, title, url, description, timestamp, source
  FROM site_newsstory
  ORDER BY timestamp desc;

SELECT cat.id,
       cat.name,
       CURSOR( SELECT *
               FROM latestnews
               WHERE category = cat.id
               AND ROWNUM < 4
               ) AS stories
FROM site_newscategory cat;
```

## News.xsl

```
<?xml version="1.0"?>
<!-- News.xsl: Format the news portal page -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" indent="no"/>
  <xsl:include href="LoginBox.xsl"/>
  <xsl:include href="ViewBox.xsl"/>
  <xsl:include href="NewsBoxes.xsl"/>
  <xsl:template match="/">
    <html>
      <head><link rel="stylesheet" type="text/css" href="News.css" /></head>
      <body class="page">
        <center>
          <table border="0" width="80%">
            <tr>
              <td align="left" valign="top" width="140">
                <xsl:call-template name="ViewBox"/>
                <!-- Show LoginBox if the forumuser in page is blank -->
                <xsl:if test="/page/forumuser = ''">
                  <br/>
                  <xsl:call-template name="LoginBox"/>
                </xsl:if>
              </td>
              <td>&#160;&#160;&#160;&#160;</td>
              <td width="800" nowrap="">
                <xsl:call-template name="NewsBoxes"/>
              </td>
            </tr>
          </table>
          <!-- Show preferences link if forumuser in page is not blank -->
          <xsl:if test="/page/forumuser != ''">
            <font size="2">
              <a href="Prefs.xsql">
                <xsl:text>Edit Preferences for </xsl:text>
                <xsl:value-of select="/page/forumuser"/>
              </a>
            </font>
          </xsl:if>
        </center>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```



## News.xsql

```
<?xml version="1.0"?>
<!-- News.xsql: News delivered in rounded box format -->
<?xml-stylesheet type="text/xsl" href="News.xsl"?>
<page connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <xsql:insert-request table="user_prefs_view" transform="InsertUserPrefs.xsl"/>
  <xsql:set-cookie name="forumuser" value="{@youremail}"
    ignore-empty-value="yes" only-if-unset="yes"/>
  <xsql:set-page-param name="forumuser" value="{@youremail}"
    ignore-empty-value="yes"/>
  <xsql:include-param name="forumuser"/>

  <xsql:include-xsql href="NewsContent.xsql"/>
</page>
```

## News\_as\_sql.xml

```
<?xml version="1.0"?>
<!-- News.xml: Format the news portal page -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="text" indent="no"/>
  <xsl:template match="/page">
    <xsl:for-each select="ROWSET/ROW/STORIES/STORIES_ROW">
      insert into site_newsstory values(
        /* title */ '<xsl:value-of select="TITLE"/>',
        /* url */ '<xsl:value-of select="URL"/>',
        /* description */ '<xsl:value-of select="DESCRIPTION"/>'
      );
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

## News\_as\_sql.xsql

```
<?xml version="1.0"?>
<!-- News_as_sql.xsql: news delivered in SQL file format-->
<?xml-stylesheet type="text/xsl" href="News_as_sql.xsl"?>
<page connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <xsql:include-xsql href="NewsContent.xsql"/>
</page>
```

## News\_as\_table.xsql

```
<?xml version="1.0"?>
<!-- News_as_table.xsql: news delivered in table format -->
<?xml-stylesheet type="text/xsl" href="NewsTable.xsl"?>
<page connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <xsql:include-xsql href="NewsContent.xsql"/>
</page>
```

## NewsBoxes.xsl

```
<!-- NewsBoxes.xsl: Format news headlines as TitledBoxes -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:include href="TitledBox.xsl"/>
  <xsl:template name="NewsBoxes">
    <xsl:if test="not(page/ROWSET/ROW)">
      <xsl:text>You have not selected any news categories...</xsl:text>
    </xsl:if>
    <xsl:for-each select="page/ROWSET/ROW">
      <xsl:call-template name="TitledBox">
        <xsl:with-param name="Title">
          <!-- Title is the category name with link to details -->
          <a href="NewsCategory.xsql?id={ID}">
            <b><xsl:value-of select="NAME"/></b>
          </a>
        </xsl:with-param>
        <xsl:with-param name="Contents">
          <!-- Contents is a list of News Stories -->
          <xsl:for-each select="STORIES/STORIES_ROW">
            <li>
              <a title="{DESCRIPTION}" target="_top" href="{URL}">
                <xsl:value-of select="TITLE"/>
                (<xsl:value-of select="TIMESTAMP"/>)
              </a>
            </li>
          </xsl:for-each>
        </xsl:with-param>
      </xsl:call-template>
      <xsl:if test="position() != last()">
        <br/>
      </xsl:if>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

## NewsCategory.xsql

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="NewsCategoryBox.xsl"?>
<page id="1" connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <breadcrumbs>
    <xsql:query>
      SELECT name as "categoryname"
      FROM site_newscategory
      WHERE id = {@id}
    </xsql:query>
  </breadcrumbs>
  <xsql:action handler="Paging" rows-per-page="6" url-params="id">
    SELECT count(id)
    FROM site_newsstory
    WHERE category = {@id}
    ORDER BY TIMESTAMP DESC
  </xsql:action>
  <xsql:query rowset-element="" row-element="">
    SELECT name as "categoryname"
    FROM site_newscategory
    WHERE id = {@id}
  </xsql:query>
  <xsql:query skip-rows="{@paging-skip}" max-rows="{@paging-max}">
    SELECT title,url,description,TO_CHAR(timestamp,'Mon DD') as timestamp
    FROM latestnews
    WHERE category = {@id}
    ORDER BY TIMESTAMP DESC
  </xsql:query>
</page>
```

## NewsCategoryBox.xsl

```
<?xml version="1.0"?>
<!-- NewsCategoryBox.xsl: Format detailed news story list for a category -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" indent="no"/>
  <xsl:include href="UtilPaging.xsl"/>
  <xsl:include href="TitledBox.xsl"/>
  <xsl:include href="BreadCrumbs.xsl"/>
  <xsl:variable name="i">images/</xsl:variable>
  <xsl:template match="/">
    <html>
      <head>
        <link rel="stylesheet" type="text/css" href="Forum.css"/>
        <title>News By Category</title>
      </head>
      <body bgcolor="#FFFFFF">
        <center>
          <table border="0" width="600">
            <tr>
              <td width="600">
                <table border="0" width="100%" cellspacing="0" cellpadding="0">
                  <tr>
                    <td colspan="2" style="text-align:center"><xsl:apply-templates select="page/breadcrumbs"/></td>
                  </tr>
                  <tr>
                    <td colspan="2" style="text-align:right; font-size:7pt"><xsl:apply-templates select="page/paging"/><br/><br/>
                  </td>
                </tr>
              </table>
              <xsl:call-template name="TitledBox">
                <xsl:with-param name="Title">
                  <xsl:value-of select="page/categoryname"/>
                  <xsl:text> News in Detail</xsl:text>
                </xsl:with-param>
                <xsl:with-param name="Contents">
                  <table border="0" width="100%">
                    <xsl:for-each select="page/ROWSET/ROW">
                      <tr>
                        <td width="3%" valign="top">
                          
                          <font color="#FFFFFF">.</font>
                        </td>
                        <td width="97%">
                          <a href="{URL}">
                            <b><xsl:value-of select="TITLE"/></b>
                          </a>
                          <span style="font-size: 7pt">
                            <xsl:text>&#160;(</xsl:text>
                            <xsl:value-of select="TIMESTAMP"/>
                            <xsl:text>)</xsl:text>
                          </span>
                          <span class="tr">
                            <br/>
                            <xsl:value-of select="DESCRIPTION"/>
                          </span>
                        </td>
                      </tr>
                    </xsl:for-each>
                  </table>
                </xsl:with-param>
              </xsl:call-template>
            </td>
          </tr>
        </table>
      </center>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

## NewsContent.xsql

```
<!-- NewsContent.xsql: provides basic data for the news portal-->
<xsql:query connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <![CDATA[
    SELECT cat.id,
           cat.name,
           CURSOR( SELECT title,url,NVL(description,'N/A')as description,
TO_CHAR(timestamp,'Mon DD') as timestamp
                 FROM latestnews
                 WHERE category = cat.id
                 AND ROWNUM <= (SELECT headlines
                                FROM user_prefs
                                WHERE userid = NVL('@forumuser','DEFAULT'))
                 ) AS stories
    FROM site_newscategory cat
    WHERE cat.id IN (SELECT category
                    FROM user_news_categories
                    WHERE userid = NVL('@forumuser','DEFAULT'))
  ]>
</xsql:query>
```



## NewsTable.xsl

```
<?xml version="1.0"?>
<!-- NewsTable.xsl: Format the news portal page as tables-->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" indent="no"/>
  <xsl:include href="TableBaseWithCSS2.xsl"/>
  <xsl:include href="ViewBox.xsl"/>
  <xsl:template match="/">
    <html>
      <head><link rel="stylesheet" type="text/css" href="Forum.css" /></head>
      <body class="page" bgcolor="#000000">
        <center>
          <!-- Transform the ROWSET of query results first -->
          <xsl:apply-templates select="page/ROWSET"/>
          <!-- Show preferences link if forumuser in page is not blank -->
          <xsl:if test="/page/forumuser != ''">
            <font size="1">
              <a href="Prefs.xsql">
                <xsl:text>Preferences for </xsl:text>
                <xsl:value-of select="/page/forumuser"/>
              </a>
            </font>
          </xsl:if>
        </center>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

## NewsTableStyle.xsl

```
<!-- NewsTableStyle.xsl: Format news headlines as TitledNews -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:include href="TableBaseWithCSS.xsl"/>
  <xsl:template name="NewsTableStyle">
    <xsl:for-each select="page/ROWSET/ROW">
      <xsl:call-template name="TitledNews">
        <xsl:with-param name="Title">
          <!-- Title is the category name with link to details -->
          <a href="NewsCategory.xsql?id={ID}">
            <b><xsl:value-of select="NAME"/></b>
          </a>
        </xsl:with-param>
        <xsl:with-param name="Contents">
          <!-- Contents is a list of News Stories -->
          <tr>
            <th>Title</th>
          </tr>
          <xsl:for-each select="STORIES/STORIES_ROW">
            <tr>
              <td>
                <a title="{DESCRIPTION}" target="_top" href="{URL}">
                  <xsl:value-of select="TITLE"/>
                </a>
              </td>
            </tr>
          </xsl:for-each>
        </xsl:with-param>
      </xsl:call-template>
      <xsl:if test="position() != last()">
        <br/>
      </xsl:if>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

## PREFS.SQL

```
drop table user_stocks;
drop table user_news_categories;
drop table user_prefs;
create table user_prefs (
  userid VARCHAR2(80),
  headlines VARCHAR2(5),
  constraint user_prefs_pk primary key (userid)
);

create table user_news_categories(
  userid VARCHAR(80),
  category NUMBER,
  constraint categories_for_user foreign key (userid) references user_prefs
);
insert into user_prefs values ('DEFAULT',5);
insert into user_stocks values ('DEFAULT','ORCL');
insert into user_stocks values ('DEFAULT','MSFT');
insert into user_news_categories values ('DEFAULT',1);
insert into user_news_categories values ('DEFAULT',2);
insert into user_news_categories values ('DEFAULT',3);
insert into user_prefs values ('liuh',4);
insert into user_stocks values ('liuh','HLTH');
insert into user_stocks values ('liuh','WEBM');
insert into user_stocks values ('liuh','ORCL');
insert into user_stocks values ('liuh','INTC');
insert into user_stocks values ('liuh','AAPL');
insert into user_news_categories values ('liuh',2);
insert into user_news_categories values ('liuh',1);
commit;
```

## Prefs.xml

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:import href="UtilDataForm.xml"/>
  <xsl:template match="/">
    <html>
      <head><link rel="stylesheet" type="text/css" href="Forum.css" />
      </head>
      <body class="page">
        <xsl:apply-templates select="page/dataform"/>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

## Prefs.xsql

```
<?xml version="1.0"?>
<!-- Prefs.xsql: Create form for displaying/editing user preferences -->
<?xml-stylesheet type="text/xsl" href="Prefs.xsl"?>
<page connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <dataform target="News.xsql" submit="Save Preferences">
    <item type="hidden" name="userid">
      <xsql:include-param name="forumuser"/>
    </item>
    <item type="checkboxlist" name="categories" label="News Categories">
      <xsql:query>
        SELECT nc.id AS VALUE,
               nc.name AS DISPLAY,
               DECODE(uc.category,NULL,'N','Y') as SELECTED
        FROM site_newscategory nc,
             user_news_categories uc
        WHERE NVL('@forumuser','DEFAULT') = uc.userid (+)
              AND nc.id = uc.category (+)
              order by name
      </xsql:query>
    </item>
    <item type="list" name="headlines" label="Headlines Per Category">
      <xsql:query rowset-element="" row-element="">
        SELECT headlines as "default"
        FROM user_prefs
        WHERE userid = NVL('@forumuser','DEFAULT')
      </xsql:query>
      <ROWSET>
        <ROW><VALUE>4</VALUE><DISPLAY>4</DISPLAY></ROW>
        <ROW><VALUE>5</VALUE><DISPLAY>5</DISPLAY></ROW>
        <ROW><VALUE>6</VALUE><DISPLAY>6</DISPLAY></ROW>
      </ROWSET>
    </item>
  </dataform>
</page>
```

## Search.xsql

```
<?xml version="1.0"?>
<!-- Search.xsql: Forum search criteria form -->
<?xml-stylesheet type="text/xsl" href="ForumStyle.xsl"?>
<page connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <breadcrumbs>
    <forumname>Search</forumname>
  </breadcrumbs>
  <dataform target="ForumSearchResults.xsql" submit="Search">
    <xsql:set-page-param name="default" value="{@id}"/>
    <item type="text" name="searchFor" size="30" label="Search For"/>
    <item type="list" name="searchIn" label="In">
      <ROWSET>
        <ROW><VALUE>E</VALUE><DISPLAY>Entire Message</DISPLAY></ROW>
        <ROW><VALUE>S</VALUE><DISPLAY>Subject Only</DISPLAY></ROW>
      </ROWSET>
    </item>
    <item type="list" name="forum" label="In Forum">
      <xsql:include-param name="default"/>
      <xsql:query>
        select id as value, name as display
           from forum
        union all
        select '-1', 'All Forums'
           from dual
        order by 1
      </xsql:query>
    </item>
    <item type="list" name="daysAgo" label="By Date">
      <ROWSET>
        <ROW><VALUE>0</VALUE><DISPLAY>Any Date</DISPLAY></ROW>
        <ROW><VALUE>-1</VALUE><DISPLAY>Since Yesterday</DISPLAY></ROW>
        <ROW><VALUE>-7</VALUE><DISPLAY>In Past Week</DISPLAY></ROW>
        <ROW><VALUE>-14</VALUE><DISPLAY>In Past Two Weeks</DISPLAY></ROW>
        <ROW><VALUE>-21</VALUE><DISPLAY>In Past Three Weeks</DISPLAY></ROW>
      </ROWSET>
    </item>
  </dataform>
</page>
```

## SearchHandler\_body.sql

```

CREATE OR REPLACE PACKAGE BODY ForumSearch IS
    FUNCTION Query(forumid  VARCHAR2  := '-1',
                  daysAgo  NUMBER    := 0,
                  searchIn  VARCHAR2  := 'S',
                  searchFor VARCHAR2  := NULL) RETURN VARCHAR2 IS
        query VARCHAR2(2000);
        noCriteriaSupplied BOOLEAN := LTRIM(RTRIM(searchFor)) IS NULL;
        searchEntireMessage BOOLEAN := UPPER(searchIn)='E';
        restrictToForum    BOOLEAN := forumid > 0;
        restrictByDate     BOOLEAN := daysAgo < 0;
    BEGIN
        query :=
        'select distinct
          t.id           as h_id,
          t.title        as "Topic",
          t.userid       as "Started_By",
          t.postings-1   as "Replies",
          to_char(t.lastpost,'dd Mon YYYY, hh24:mi') as "Last_Post",
          f.name         as "Forum_Name"
        from forum_topic t,
             forum_topic_posting p,
             forum f
        where t.id = p.topicid
              and t.forumid = f.id';
        IF searchEntireMessage THEN
            query := query || ' AND ( CONTAINS(p.posting, '''||searchFor||''')>0 ||
                               ' OR UPPER(t.title) LIKE UPPER(''%'||searchFor||'%''))';
        ELSE
            query := query || ' AND UPPER(t.title) LIKE UPPER(''%'||searchFor||'%'')';
        END IF;
        IF restrictToForum THEN
            query := query || ' AND t.forumid = '''||forumid;
        END IF;
        IF restrictByDate THEN
            query := query || ' AND p.posted > SYSDATE + '''||daysAgo;
        END IF;
        IF noCriteriaSupplied THEN
            query := query || ' AND 1=2';
        END IF;
        RETURN query;
    END;

    FUNCTION Hits(forumid  VARCHAR2  := '-1',
                  daysAgo  NUMBER    := 0,
                  searchIn  VARCHAR2  := 'S',
                  searchFor VARCHAR2  := NULL) RETURN NUMBER IS
        the_cursor ref_cursor;
        the_count  NUMBER := 0;
    BEGIN
        OPEN the_cursor FOR 'select count(1) from ('||
                           Query(forumid,daysAgo,searchIn,searchFor)||
                           ')';
        FETCH the_cursor INTO the_count;
        CLOSE the_cursor;
        RETURN the_count;
    END;

    FUNCTION Find(forumid  VARCHAR2  := '-1',
                  daysAgo  NUMBER    := 0,
                  searchIn  VARCHAR2  := 'S',
                  searchFor VARCHAR2  := NULL) RETURN ref_cursor IS
        the_cursor ref_cursor;
        my_sal NUMBER := 1;
    BEGIN
        OPEN the_cursor FOR Query(forumid,daysAgo,searchIn,searchFor)||
                           ' order by 5 desc';
        RETURN the_cursor;
    END;
END;

```

## SearchHandler\_spec.sql

```
CREATE OR REPLACE PACKAGE ForumSearch IS
  TYPE ref_cursor IS REF CURSOR;
  FUNCTION Find(forumid VARCHAR2 := '-1',
               daysAgo NUMBER := 0,
               searchIn VARCHAR2 := 'S',
               searchFor VARCHAR2 := NULL) RETURN ref_cursor;
  FUNCTION Hits(forumid VARCHAR2 := '-1',
               daysAgo NUMBER := 0,
               searchIn VARCHAR2 := 'S',
               searchFor VARCHAR2 := NULL) RETURN NUMBER;
END;
```



## SITE.CSS

```
table.x { font-family: Arial; font-size:9pt}
table.x SPAN { font-family: Arial; font-size:9pt}
table.x TD { font-family: Arial; font-size:9pt}
table.x TH { background-color:#cccc99;
            color:#336699;
            font-family: Arial;
            font-weight:bold;
            font-size: 11pt; }

table.x BR {font-size: 4pt}
table.x .h { background-color:#f7f7e7 }
table.x .code { font-family: Andale Mono, monospace; font-size: 9pt }
table.x .title {background-color:white; font-family: Arial; font-size:14pt}

table.x A:hover { background: #FFA }
table.x A:visited { text-decoration: none ; color: #336699 }
table.x A { text-decoration: none; color: #336699 }
table.x .comment { font-family: Andale Mono, monospace; color: green; font-size: 9pt }

table.y { background-color:white; font-family: Arial; font-size:9pt}
table.y SPAN { background-color:white; font-family: Arial; font-size:9pt}
table.y TD { background-color:white; font-family: Arial; font-size:9pt}
table.y TH { background-color:#cccc99;
            color:#336699;
            font-family: Arial;
            font-weight:bold;
            font-size: 10pt; }

table.y A:hover { background: #FFA }
table.y A:visited { text-decoration: none ; color: #336699 }
table.y A { text-decoration: none; color: #336699 }
table.y .h { background-color:#f7f7e7 }
```

```

Site.xml
<xsl:stylesheet xmlns:xsl="http://www.w3.org/XSL/Transform/1.0">
<xsl:template match="/">
<html><head><title>XML Central</title></head>
<body>
<center>
<table border="0" cellpadding="10">
<tr>
<xsl:for-each select="website/categories/category">
<td>
<a href="Site.xml?cat={name}"></a>
<a href="Site.xml?cat={name}"><b><xsl:value-of select="name"/></b></a>
</td>
</xsl:for-each>
</tr>
</table>
<hr x=""/>
<table width="90%" border="0">
<xsl:for-each select="website/articles/entry">
<tr>
<td width="3%" valign="top"></td>
<td width="97%">
<a href="{url}"><b><xsl:value-of select="title"/></b></a>
<br x=""/>
<xsl:value-of select="description"/>
</td>
</tr>
</xsl:for-each>
</table>
</center>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

## site.xsql

```
<?xml version="1.0"?>
<!-- Site.xsql: Provide tabset structure for home page -->
<?xml-stylesheet type="text/xsl" href="tab.xsl"?>
<site elt="query" xmlns:xsql="urn:oracle-xsql">
  <!--
  | Pass the values of current "tab" parameter (if passed in explicitly),
  | the "lasttab" cookie (if set), and the name of this page to the
  | XSLT stylesheet that renders the tab bar.
  +-->
  <xsql:set-stylesheet-param name="thispage" value="site.xsql"/>
  <xsql:set-stylesheet-param name="tab" value="{@tab}" ignore-empty-value="yes"/>
  <xsql:set-stylesheet-param name="lasttab" value="{@lasttab}"
    ignore-empty-value="yes"/>
  <!-- Remember the last tab clicked in an HTTP Cookie -->
  <xsql:set-cookie name="lasttab" value="{@tab}" ignore-empty-value="yes"/>
  <!-- This is static XML that provides the tab structure -->
  <tabs>
    <tab id="News" name="News"/>
    <tab id="Forums" name="Forums"/>
    <tab id="HealthRecord" name="HealthRecord"/>
  </tabs>
</site>
```

```

TableBaseWithCSS2.xsl
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <!--
  | TableBaseWithCSS2:
  | Basic Stylesheet to format any ROWSET of ROWS into a table
  | with column headings in a generic way. Leverages Forum.css
  | CSS stylesheet to control font/color information for the page.
  +-->
  <xsl:template match="/">
    <html>
      <!-- Generated HTML Result will be linked to Forum.css CSS Stylesheet -->
      <head><link rel="stylesheet" type="text/css" href="Forum.css"/></head>
      <body><xsl:apply-templates/></body>
    </html>
  </xsl:template>
  <xsl:template match="ROWSET/ROW">
    <xsl:apply-templates select="NAME" />
    <xsl:apply-templates select="STORIES" />
  </xsl:template>
  <xsl:template match="STORIES">
    <table class="newstable" border="1" cellspacing="0" width="80%">
      <!-- table header-->
      <thead>
        <tr>
          <th>Title</th>
          <th>Source</th>
          <th>Description</th>
          <th>Timestamp</th>
        </tr>
      </thead>
      <xsl:apply-templates/>
    </table>
  </xsl:template>
  <xsl:template match="STORIES/STORIES_ROW">
    <tr><xsl:apply-templates/></tr>
  </xsl:template>
  <!-- Match any element child of a ROW -->
  <xsl:template match="STORIES/STORIES_ROW/*">
    <td><xsl:apply-templates/></td>
  </xsl:template>
</xsl:stylesheet>

```

```

TitledBox.xsl
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template name="TitledBox">
    <xsl:param name="Title"/>
    <xsl:param name="Contents"/>
    <!-- Title of the titledNews-->
    <table cellSpacing="0" cellPadding="0" border="0" bgcolor="#ffffff" width="290">
      <tr>
        <th nowrap="" width="98%" align="center" valign="center" bgcolor="#0099ff">
          <!-- Put whatever is passed as the Title parameter here -->
          <font color="white">
            <xsl:copy-of select="$Title"/>
          </font>
        </th>
        <td width="1%" align="right" valign="TOP" bgcolor="#0099ff">
          <xsl:text>&#160;&#160;&#160;</xsl:text>
        </td>
        <td width="1%" valign="top" align="right" bgcolor="#0099ff">
          
        </td>
      </tr>
    </table>

    <!--Content of the titledbox-->
    <table cellSpacing="0" cellPadding="1" border="0" bgcolor="#ffffff" width="580">
      <tr>
        <td align="middle" bgColor="#006699">
          <table cellSpacing="0" cellPadding="3" bgColor="#ffffff" border="0">
            <tr>
              <td width="1%" align="right" valign="TOP">
                <xsl:text>&#160;&#160;&#160;</xsl:text>
              </td>
              <td nowrap="" colspan="3">
                <!-- Put whatever is passed as the Contents parameter here -->
                <xsl:copy-of select="$Contents"/>
              </td>
            </tr>
            <tr>
              <td width="1%" align="LEFT" valign="BOTTOM">
                
              </td>
              <td colspan="2" height="1" width="98%">
                
              </td>
              <td width="1%" align="right" valign="BOTTOM">
                
              </td>
            </tr>
          </table>
        </td>
      </tr>
    </table>

    <!--space bar-->
    <table>
      <tr>
        <td colspan="4" height="15">
          
        </td>
      </tr>
    </table>

  </xsl:template>
</xsl:stylesheet>

```

```

TitledBox2.xsl
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template name="TitledBox2">
    <xsl:param name="Title"/>
    <xsl:param name="Contents"/>
    <table width="100%" cellspacing="0" cellpadding="0" border="0">
      <tr>
        <td width="1%" valign="top" align="left" bgcolor="#0099ff">
          
        </td>
        <th nowrap="" width="98%" align="left" valign="center" bgcolor="#0099ff"
color="white">
          <!-- &#160; is numerical character entity for Non-Breaking Space -->
          <xsl:text>&#160;&#160;&#160;</xsl:text>

          <!-- Put whatever is passed as the Title parameter here -->
          <xsl:copy-of select="$Title"/>
        </th>
        <td width="1%" align="right" valign="TOP" bgcolor="#0099ff">
          <xsl:text>&#160;&#160;&#160;</xsl:text>
        </td>
        <td width="1%" valign="top" align="right" bgcolor="#0099ff">
          
        </td>
      </tr>
      <tr>
        <td width="1%" align="right" valign="TOP" bgcolor="#ffcc00">
          <xsl:text>&#160;&#160;&#160;</xsl:text>
        </td>
        <td nowrap="" bgcolor="#ffcc00" colspan="3">
          <!-- Put whatever is passed as the Contents parameter here -->
          <xsl:copy-of select="$Contents"/>
        </td>
      </tr>
      <tr>
        <td bgcolor="#ffcc00" width="1%" align="LEFT" valign="BOTTOM">
          
        </td>
        <td colspan="2" bgcolor="#ffcc00" height="1" width="98%">
          
        </td>
        <td bgcolor="#ffcc00" width="1%" align="right" valign="BOTTOM">
          
        </td>
      </tr>
    </table>
  </xsl:template>
</xsl:stylesheet>

```

## Todo.SQL

```
drop table user_todo;
drop table todo;
drop sequence todo_id;
create sequence todo_id;

create table todo(
  id          varchar2(10),
  task_name   varchar2(30),
  task_description  varchar2(30),
  constraint todo_id primary key (id)
);

create or replace trigger todo_id_trig
before insert on todo for each row
begin
  select todo_id.nextval
  into :new.id
  from dual;
end;
/

create table user_todo(
  userid      varchar2(30),
  task_id     varchar2(10),
  constraint user_todo_pk primary key (userid, task_id),
  constraint user_todo_fk foreign key (task_id) references todo (id)
);

create or replace view user_new_todo as
select u.userid, t.task_name, t.task_description
  from user_todo u, todo t
 where u.task_id = t.id;

create or replace trigger user_new_todo_ins
instead of insert on user_new_todo for each row
declare
  newTaskId VARCHAR2(10);
begin
  insert into todo(task_name,task_description)
  values (:new.task_name, :new.task_description)
  returning id into newTaskId;
  insert into user_todo(userid, task_id)
  values (:new.userid,newTaskId);
end;
/
commit;
```

## todo.xsql

```
<?xml version="1.0"?>
<!-- todo.xsql: show todo information for a user -->
<?xml-stylesheet type="text/xsl" href="hospitalStyle.xsl"?>
<page connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <!-- insert the new hospital into user_new_hospital view-->
  <xsql:insert-request table="user_new_todo"
    transform="transTodo.xsl"/>
  <xsql:set-cookie name="forumuser" value="{@youremail}"
    ignore-empty-value="yes" only-if-unset="yes"/>
  <xsql:set-page-param name="forumuser" value="{@youremail}"
    ignore-empty-value="yes"/>
  <xsql:include-param name="forumuser"/>
  <!-- Include request params so Actions can check for forumuser cookie -->
  <breadcrumbs>
    <label> To-Do's </label>
  </breadcrumbs>
  <data>
    <xsql:query max-rows="{@paging-max}" skip-rows="{@paging-skip}">
      select task_name,task_description
      from todo t, user_todo u
      where t.id=u.task_id
      and u.userid= '{@forumuser}'
    </xsql:query>
  </data>
  <actions>
    <link page="addTodo.xsql" label="Add To-Do" >
      <xsql:include-param name="forumuser"/>
    </link>
  </actions>
</page>
```



## TopicPostings.xsql

```
<?xml version="1.0"?>
<!-- TopicPostings.xsql: Show list of postings for a given topic -->
<?xml-stylesheet type="text/xsl" href="ForumStyle.xsl"?>
<page connection="xmlbook" xmlns:xsql="urn:oracle-xsql">
  <xsql:insert-request table="forum_topic_posting"
    transform="InsertPostedReply.xsl"/>
  <!-- Include request params so Actions can check for forumuser cookie -->
  <xsql:include-request-params/>
  <breadcrumbs>
    <xsql:include-xsql href="ForumTopicLookup.xsql"/>
  </breadcrumbs>
  <actions>
    <link page="EnterForumTopicReply.xsql" label="Post a New Reply" login="yes">
      <xsql:include-param name="id"/>
    </link>
  </actions>
  <data>
    <xsql:include-xsql href="TopicPostingsQuery.xsql"/>
  </data>
</page>
```

## transAllergy.xsl

```
<!-- transAllergy.xsl: Insert transform for user_new_allergy view -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <!-- Transform /request/parameters into ROWSET for hospital -->
  <xsl:template match="/">
    <ROWSET>
      <!-- XPath for repeating source rows -->
      <xsl:for-each select="/request/parameters">
        <ROW>
          <USERID>
            <xsl:value-of select="userid"/>
          </USERID>
          <allergy_name>
            <xsl:value-of select="allergy_name"/>
          </allergy_name>
          <last_date>
            <xsl:value-of select="last_date"/>
          </last_date>
          <sympton>
            <xsl:value-of select="sympton"/>
          </sympton>
        </ROW>
      </xsl:for-each>
    </ROWSET>
  </xsl:template>
</xsl:stylesheet>
```

## transHospital.xsl

```
<!-- transHospital.xsl: Insert transform for hospital table -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <!-- Transform /request/parameters into ROWSET for hospital -->
  <xsl:template match="/">
    <ROWSET>
      <!-- XPath for repeating source rows -->
      <xsl:for-each select="/request/parameters">
        <ROW>
          <USERID>
            <xsl:value-of select="userid"/>
          </USERID>
          <NAME>
            <xsl:value-of select="hospitalName"/>
          </NAME>
          <PHONE>
            <xsl:value-of select="phone"/>
          </PHONE>
          <TYPE>
            <xsl:value-of select="type"/>
          </TYPE>
          <WEBSITE>
            <xsl:value-of select="website"/>
          </WEBSITE>
          <STREET>
            <xsl:value-of select="street"/>
          </STREET>
          <CITY>
            <xsl:value-of select="city"/>
          </CITY>
          <STATE>
            <xsl:value-of select="state"/>
          </STATE>
          <ZIP>
            <xsl:value-of select="zip"/>
          </ZIP>
          <COUNTRY>
            <xsl:value-of select="country"/>
          </COUNTRY>
          <COMMENTS>
            <xsl:value-of select="comments"/>
          </COMMENTS>
        </ROW>
      </xsl:for-each>
    </ROWSET>
  </xsl:template>
</xsl:stylesheet>
```

## transTodo.xsl

```
<!-- transTodo.xsl: Insert transform for user_new_todo view -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <!-- Transform /request/parameters into ROWSET for hospital -->
  <xsl:template match="/">
    <ROWSET>
      <!-- XPath for repeating source rows -->
      <xsl:for-each select="/request/parameters">
        <ROW>
          <USERID>
            <xsl:value-of select="userid"/>
          </USERID>
          <TASK_NAME>
            <xsl:value-of select="task_name"/>
          </TASK_NAME>
          <TASK_DESCRIPTION>
            <xsl:value-of select="task_description"/>
          </TASK_DESCRIPTION>
        </ROW>
      </xsl:for-each>
    </ROWSET>
  </xsl:template>
</xsl:stylesheet>
```

## UtilBreadcrumbs.xml

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <!--
  | UtilBreadCrumbs.xml: Transform <breadcrumbs> structural info
  |                               into HTML breadcrumbs presentation.
  +-->
  <xsl:variable name="i" select="'images/'"/>
  <xsl:template match="breadcrumbs">
    <center>
      <span class="breadcrumbs">
        <xsl:choose>
          <xsl:when test="..//forumname">
            <!--
            | Even though we're only expecting one row, this moves the
            | node to allow XPath's inside to use shorter relative
            | paths
            +-->
            <a href="Forums.xsql">Forums</a>&#160;
            &#160;
            <xsl:choose>
              <xsl:when test="..//topicname">
                <a>
                  <xsl:attribute name="href">
                    <xsl:choose>
                      <xsl:when test="..//forumname/@url">
                        <xsl:value-of select="..//forumname/@url"/>
                      </xsl:when>
                      <xsl:otherwise>
                        <xsl:text>ForumTopics.xsql?id=</xsl:text>
                        <xsl:value-of select="..//forumid"/>
                      </xsl:otherwise>
                    </xsl:choose>
                  </xsl:attribute>
                  <xsl:value-of select="..//forumname"/>
                </a>&#160;
                &#160;
                <xsl:value-of select="..//topicname"/>
              </xsl:when>
              <xsl:otherwise>
                <xsl:value-of select="..//forumname"/>
              </xsl:otherwise>
            </xsl:choose>
          </xsl:when>
          <xsl:otherwise>
            <xsl:text>Forums</xsl:text>
          </xsl:otherwise>
        </xsl:choose>
      </span>
    </center>
  </xsl:template>
</xsl:stylesheet>
```

## UtilData.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <!--
  | UtilData.xsl: Transform <data> structural info into HTML table
  |               Elements in ROWSET/ROW whose name begins with the
  |               prefix 'H_' will be hidden from display in the output.
  +-->
  <xsl:template match="ROWSET">
    <center>
      <table width="80%" border="0" cellpadding="4">
        <tr>
          <!-- If there are no ROW in the ROWSET, print a message -->
          <xsl:if test="not(ROW)"><td>No Records</td></xsl:if>
          <!--
          | Process child elements of first ROW whose names do NOT start
          | with 'H_'. This let's user hide column by aliasing it to H_NAME
          +-->
          <xsl:apply-templates select="ROW[1]/*[not(starts-with(name(),'H_'))]"
                             mode="columnHeaders"/>
        </tr>
        <xsl:apply-templates/>
      </table>
    </center>
  </xsl:template>
  <!--
  | Match ROW and create a html <tr> table row. Alternate colors by using
  | attribute value template to toggle between CSS class name r0 and r1.
  +-->
  <xsl:template match="ROW">
    <tr class="r{position() mod 2}">
      <!-- Select all children of ROW whose names do not start with 'H_' -->
      <xsl:apply-templates select="*[not(starts-with(name(),'H_'))]"/>
    </tr>
  </xsl:template>
  <!-- Match ROW child element -->
  <xsl:template match="ROW/*">
    <td valign="top"><xsl:apply-templates/></td>
  </xsl:template>
  <!-- Match ROW child element and generate column heading -->
  <xsl:template match="ROW/*" mode="columnHeaders">
    <th align="left">
      <!-- Beautify Element Names "This_That" or "This-That" to "This That" -->
      <xsl:value-of select="translate(name(),'_','-','&#160;&#160;')"/>
    </th>
  </xsl:template>
</xsl:stylesheet>
```

```

UtilDataForm.xsl
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <!--
  | UtilDataForm.xsl: Transform <dataform> structural info
  | into a data-bound HTML Form
  +-->
  <xsl:template match="dataform">
    <center>
      <form method="POST" action="{@target}">
        <xsl:for-each select="item[@type='hidden']">
          <input type="hidden" name="{@name}" value="{normalize-space(.)}"/>
        </xsl:for-each>
        <table>
          <xsl:for-each select="item[@type != 'hidden']">
            <tr>
              <th align="right"><xsl:value-of select="@label"/></th>
              <td>
                <xsl:choose>
                  <xsl:when test="@type='text'">
                    <input type="text" name="{@name}"
                      value="{normalize-space(.)}"
                      <xsl:if test="@size">
                        <xsl:attribute name="size">
                          <xsl:value-of select="@size"/>
                        </xsl:attribute>
                      </xsl:if>
                    </input>
                  </xsl:when>
                  <xsl:when test="@type='textarea'">
                    <textarea class="code" rows="5" name="{@name}">
                      <xsl:if test="@size">
                        <xsl:attribute name="cols">
                          <xsl:value-of select="@size"/>
                        </xsl:attribute>
                      </xsl:if>
                      <xsl:value-of select="normalize-space(.)"/>
                    </textarea>
                  </xsl:when>
                  <xsl:when test="@type='list'">
                    <xsl:variable name="default" select="default"/>
                    <select name="{@name}">
                      <xsl:for-each select="ROWSET/ROW">
                        <option value="{VALUE}">
                          <xsl:if test="VALUE=$default">
                            <xsl:attribute name="selected"/>
                          </xsl:if>
                          <xsl:value-of select="DISPLAY"/>
                        </option>
                      </xsl:for-each>
                    </select>
                  </xsl:when>
                  <xsl:when test="@type='checkboxlist'">
                    <xsl:variable name="name" select="@name"/>
                    <xsl:for-each select="ROWSET/ROW">
                      <input type="checkbox" name="{ $name}" value="{VALUE}">
                        <xsl:if test="SELECTED='Y'">
                          <xsl:attribute name="checked"/>
                        </xsl:if>
                      </input>
                      <xsl:value-of select="DISPLAY"/><br/>
                    </xsl:for-each>
                  </xsl:when>
                </xsl:choose>
              </td>
            </tr>
          </xsl:for-each>
        </table>
        <input type="submit" value="{@submit}"/>
        <input type="reset"/>
      </form>
    </center>
  </xsl:template>
</xsl:stylesheet>

```

**Appendix C**

**WBS and Project Schedule**

<b>WBS</b>	<b>Task Name</b>	<b>Dependency</b>	<b>Start Date</b>	<b># of Days Scheduled</b>
1	Plan the Project			
1.1	Start Project		10/1/2001	1
1.2	Determine Project Objective	1.1	10/2/2001	10
1.3	Literature Review	1.1	10/2/2001	30
1.4	Plan the Project Framework	1.2, 1.3	11/1/2001	10
1.5	Deveop WBS, Schd. &Res. Plan	1.4	11/11/2001	10
1.6	Review & Approve Project Plan	1.5	11/21/2001	10
1.7	Planning Complete	1.6	12/1/2001	1
2	Design the Project			
2.1	Study Oracle XML technologies	1.7	12/2/2001	45
2.2	Develop Project Specifications	2.1	1/16/2002	30
2.3	Prepare Necessary Hardware	2.2	2/15/2002	10
2.4	Prepare Necessary Software	2.2	2/15/2002	10
2.5	Design Complete	2.3, 2.4	2/25/2002	1
3	Develop & Test the Project			
3.1	Install Oracle9i Database and OracleXDK	2.5	2/26/2002	3
3.2	Develop Overall Site Structure	3.1	3/1/2002	20
3.3	Database Modeling	3.2	3/21/2002	10
3.4	Separate Dynamic and Static Content	3.3	3/31/2002	5
3.5	Develop HTML Mockup	3.4	4/5/2002	10
3.6	Develop Reusable XSLT Templates	3.5	4/15/2002	10
3.7	Develop XSLT Stylesheet	3.6	4/25/2002	20
3.8	Develop XSQL Pages	3.7	5/15/2002	20
3.9	Debug and Test Functionalities	3.8	6/4/2002	20
4	Write Report			
4.1	Read Report Guidelines	3.9	6/24/2002	1
4.2	Write Report	4.1	6/25/2002	50
4.3	Revise and Approve Report	4.2	8/14/2002	15
4.4	Present the Project	4.3	8/29/2002	1
4.5	Bring Report to Library for Archive	4.4	8/30/2002	5
4.6	Project Complete	4.4	9/4/2002	