Data Warehousing for Electronic Commerce

by

Jinbo Zeng

A thesis
Submitted to the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements
for the degree of

Master of Science

Department of Computer Science
Faculty of Graduate Studies
University of Manitoba
Winnipeg, Manitoba, Canada

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

Data Warehousing for Electronic Commerce

BY

Jinbo Zeng

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

Of

MASTER OF SCIENCE

Abstract

The Web has become a borderless marketplace for purchasing goods and services. The key for merchants to win in this very competitive market is to have accurate knowledge about the needs of potential customers and the ability to establish personalized services that satisfy these needs. Usually a Web server logs the activities of the visitors to the merchant's online site. From the Web server log, we can extract information that indirectly reflects the visitors' interests and the Web site's quality in serving the visitors' interests by applying data mining techniques.

The current Web server log standard, defined by the World Wide Web Consortium (W3C), reports Web usage by Uniform Resource Locators (URLs) which indicates only the location of served Web pages and often nothing about the content. This log may lead to inaccurate business analysis for data-driven Web pages and frequent updated static content Web pages.

This thesis presents a prototype implementation of a Web content usage logging system for accurate Web usage analysis for enhancing e-commerce activities.

The major design contributions in this thesis are:

1. A Web content usage logging system that provides Web administrators and business analysts the flexibility to capture Web site visitors' interests at a atomic level.

2. An object-oriented approach that can benefit from this system to log interested objects' attributes and relationships among these attributes.

3. Adopting data mining algorithms in Web object mining domain that helps to discover interesting business oriented rules among customer activities.

A prototype Web store and Web content logging system are developed and tested to measure the performance of logging mechanism and generate the Web content log for data mining. This thesis also presents techniques to extract, transform, and load (ETL) the source Web content log for data warehouse for data mining. The little overhead of real-time Web content logging can potentially save huge amount of effort in the data ETL process for data mining. This research provides a good case study for design, collect and mine e-commerce system usage data for business analysis.

Acknowledgement

First and foremost, I would like to begin by thanking my supervisor, Dr. Sylvanus Ehikioya, for taking me on as a student in the first place. I really thank him for his encouragement, appreciation, personal guidance and constructive comments during the completion of this thesis.

I would like to thank Dr. Mary Brabston (Dept. of Accounting and Finance, I.H. Asper School of Business) and Dr. Ruppa Thulasiram (Dept. of Computer Science) for serving as my thesis examination committee members.

I would also like to thank the other graduate students in the computer science department, whose companionship made graduate studies an entertaining time.

I extend my heartfelt thanks to my girlfriend, Shuhui Gu, for her love, support, and faith in me.

Finally, I would like to thank my parents, Nailin and Jieqing, and my brother, Jinhao, for their emotional support and encouragement all through.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The World Wide Web continues to grow at an astounding rate in the volume of traffic. The rapid rise of commercial Internet services and applications makes the Internet a very challenging marketplace for business. The Web has become a borderless marketplace for purchasing goods and services. The key to win in this very competitive market is knowledge about the needs of both existing and potential customers and the ability to establish personalized services that satisfy these needs.

With the explosive growth of the World Wide Web and e-commerce, discovery and analysis of data from Web sites enables business owners to understand visitors and customers' behaviors and expectations and how a Web site is used by customers. Web traffic analysis, through mining Web usage data, can greatly assist the overall business decision-making by coordinating sales and marketing efforts to turn customer information into sales.

Web sites are served by Web servers. Each Web page is referred by an Internet address, called Uniform Resource Locator (URL). When a visitor requests a Web page, the Web server usually logs the visitor's requested URL into a file while serving the Web page content to the visitor. From this file, the Web server log, we can extract information that indirectly reflects the visitors' interests and the Web site's quality in serving for visitors' interests by applying data mining techniques.

A successful e-commerce strategy should be customer-focused (including individual and business customers, and business partners). To realize the promise of one-to-one and loyalty marketing philosophies while reducing overall marketing costs by developing more effective, targeted campaigns, companies no longer want to treat their customers solely based on large groups, but they want to get up close and personal with each of them individually to improve customer relationship and satisfaction [46, 55]. New data warehouse implementations in recent years have introduced customer relationship management (CRM) modules. CRM comprises the in-depth analysis of customer behaviors and attributes to achieve complete knowledge of the customers, including their habits, desires, and needs [47, 48, 49].

A data warehouse provides a single, complete and consistent store of data captured from diverse sources [50]. Data warehousing facilitates business decision-making for enterprise-based analysis and access. It is designed for end users in a way they can understand and use in a business context, such as supporting decision support systems that are business subject-oriented (as opposed to application-oriented) [51]. Data warehousing for e-commerce involves two domains: business-transactions-related data warehousing and Web-usage-related data warehousing [6]. The former is similar to traditional business data warehousing, providing business analysis information. The latter is relatively new to data warehousing, providing analysis of customers' activities. By data warehousing Web usage data, it is possible to present reports to decision makers in a way they are familiar with in the business context through business intelligence system. This research deals with the intelligent use of data captured at Web sites about customers' behaviors for the profit maximization of the organization [52]. This is a relatively new dimension for e-commerce and data warehousing, although the pure data warehousing component is more matured. This thesis focus on (a) integrating Web users' personal profiles, business transactional data, and users' visit log into an enterprise data; and (b) the discovery and analysis of the relationship between product sales and customers' behavior by mining the integrated data. The analyzed results will provide a valuable insights into customers' behaviors and content effectiveness, thus building a more effective and more profitable e-commerce system. For example, the system can provide

important clues for analyzing sales data and visitors' behavior data of specific marketing campaigns for the business decision-maker.

1.1 Current Problems

Web servers register a Web log entry for every single access, recording some useful information, mostly the extended Server Side Includes (XSSI) variables and hypertext transfer protocol (HTTP) header content. However, this information is very limited for dynamic pages and prevents Web analyzers from providing better pattern discovery. So the analysis of dynamic page content is unable to give more precise insights into the business while most e-commerce Web sites are adopting server side script to generate Web pages dynamically. The current Web server log standard, defined by World Wide Web Consortium (W3C), report Web usage by URLs which indicate only the location of served Web pages and none about the content. This log may lead to inaccurate business analysis. Traditional Web log analysis software, such as Webalizer and WebTrends, report basic traffic information based on Web server log files. This information is typically used for Web site management purposes, meaningful only to Webmasters and system administrators who are responsible for keeping the site running. These administrators want to know how much traffic they are getting, how many requests fail, and what kinds of errors are being generated. Furthermore, there are some common problems with present Web server log mechanism, some of which may cause inaccurate data analysis for business analysis purpose:

1) Web content does not enforce a rigid format and any data type. Web content data is considered as semi-structured data or unstructured data because it is not either table-oriented as data in relational databases or sorted-graph as data in object databases [9]. This non-rigid format causes difficulties in applying current mature relational database and data mining technologies for better organization and to understand the data.

2) An increasing number of Web sites are adopting server side script to generate Web pages dynamically. The current W3C standard and extended Web server logs report Web usage by URLs, which indicate only the location of served Web pages and often no information about the content. Static pages may be acceptably analyzed based on current Web log file and Web page contents by some extra content analysis tools. But for most Web sites, it can be problematic when the page content is changed regularly. Furthermore the current logging mechanism only logs information about which server side programs are called for dynamic page requests, while the page content returned by the same server side program can be totally different. Nowadays, most successful Web sites are built on the model-view-controller paradigm, using server command pattern, in order to be extendable and maintainable while providing rich information for users [53, 21]. This pattern may only have few entrance server-side scripts or programs that can call different business logic components according to the different users' requests. In this situation, the record in the Web log will reflect the high click rate of those few entrance URLs while providing no information about users' requests and their intentions. These click-rate data may lead to inaccurate business analysis.

3) IP packet sniffers are used by some vendors, such as WebTrends, to capture the content of dynamic pages. While packet sniffers can find more data than what is usually available in normal Web logs, identifying users and sessions, and extracting logical business information are extremely difficult [4]. Furthermore, IP packet sniffers can miss most critical business requests because data are encrypted over the network between the server and client.

4) Proxies and local content caching are widely used. Sometimes Web servers may not even receive clients' requests while cached contents are displayed and there is a higher possibility for popular pages to be cached. This also can lead to skewed business analysis due to the incompleteness of the data collected [16, 25, 28, 31].

5) The tools currently available to analyze data based on groups and their behaviors (comparing to the sales which is an important reference to set the customer's priority) are difficult to extend to provide individual analysis. Currently, all commercial Web mining products cluster Web transactions mainly on request URLs retrieved from Web access log. Item-based patterns and rules can be missed when the transactions are not considered to be similar during data mining [21, 22, 23, 24, 25, 28, 31].

1.2 Research Motivation

As discussed above, the current Web servers' access logging mechanism does not support strong page content tracking and analyzing ability. In order to give an in-depth look at the data in the Web log for business analysis purpose, data mining must be introduced. Data mining is used to uncover patterns and relationships that cannot be seen in simple summary statistics by applying machine learning and discovery techniques [13]. Mining data is processed in three phases: preprocessing, patterns discovery, and patterns analysis. The results of data mining are then displayed using the visualization tools for easy understanding by business decision-makers.

Currently, all Web mining products cluster Web transactions mainly on request URLs. So item-based patterns and rules can be missed when the transactions are not considered to be similar during data mining. Simply applying data mining technologies on transaction URLs are not able to reveal the relationships among items presented in the transactions in dynamic Web content scenario. Most current Web traffic analysis tools are inadequate to report the effectiveness of specific marketing and merchandising efforts.

All these existing issues above help motivate this thesis. To provide a more efficient approach for business decision-makers with insights into visitors' activities and prospects, and how the Web sites assist their business strategies, the objective of this

research is to integrate Web usage information, Web content information and business transaction records and provide suitable data mining algorithms.

1.3 Proposed Solution

Most of current Web traffic analysis tools are not adequate to report the effectiveness of specific marketing and merchandising efforts [52]. In order to provide a more efficient approach to analyze the business running on Web sites, each Web site is modeled as a collection of groups of related objects [31] instead of a collection of pages. Object oriented design assists in achieving a better way of modeling, organizing, and interconnecting Web data. So object-oriented methodology is used in my design to build Web sites.

The objective of this thesis is to design and develop a prototype Web object content logging system and apply data mining algorithms to logged Web object content usage data for business analysis, in place of the current URL based Web traffic data mining system. This prototype includes three major parts, a Web object content logging module for Web server, an object-oriented Web Store running on the Web Server with the enhanced Web object content logging module, Web object log data mining.

- Web object content logging module.

  Current Web server logging module does not log any information about the content that a client requests. So analyzing this kind of Web log data cannot give an insight of how the Web site is serving the business. The Web server enhanced with a Web object content logging module is able to log interesting data attributes in the return pages. It supports logging for both static Web pages and dynamic Web pages.

- Object-oriented Web Store

  This Web store is designed, modeled and implemented in an object-oriented approach. Object-oriented design makes the Web site easy to maintain. In addition, it provides a better understanding of the Web site content and analysis results for

business analysts. The log data of clients' requests and responses from this Web store provides the data source for the next step, Web object data mining.

- Web object log data mining

  Data mining algorithms must be applied on the log data in order to provide analysis report for business purpose. There are two major steps in the data mining procedure, data preprocessing and pattern discovery.

  Data preprocessing

  The raw data from Web server log needs to be cleaned and transformed into the formats and database schemas that can be used by data mining algorithms necessary for pattern analysis.

  Pattern Discovery

  Different pattern discovery algorithms are developed and integrated with those from several fields such as statistics, data mining, machine learning, and pattern recognition. Current Web data mining pattern discovery algorithms are based on static pages. When applying these algorithms to dynamic pages, the results will be skewed. The data mining prototype includes the following functionality with enhanced data mining algorithms.

  - *Clustering* Web usage data and Web content objects. Clustering algorithms are used to group together a set of items having similar characteristics to perform market. These groups can be classified by using supervised inductive learning algorithms [52, 54, 56].

  - *Associate Rules, Sequential Patterns and Dependency Modeling* of Web usage data and Web content objects. These data mining techniques are used to discover possible relationships among items. The goal is to associate related information and then attempt to discover the inter-session patterns in a time-ordered set of events. This function is used to analyze customer groups' and

individuals' behaviors in-depth, and the trends as well. The analysis results can be used to build a more effective market campaign and convenient navigation [5, 6, 13, 15, 52, 54, 56].

This prototype uses XML and its related technologies, such as Stylesheet transformations (XSLT), to simplify some procedures, such as data modeling, data parsing, etc. Introducing metadata for Web objects' attributes by XML technologies greatly assists data analysis for business purposes. Using XML also provides better compatibility and easier deployment.

1.4 Contributions of the Thesis

The work presented in this thesis started with the desire to give business analysts business-oriented reports for e-commerce Web site mainly using dynamic pages for Web page content. The work reported in this thesis extends the initial ideas reported in [31].

The current Web usage analysis systems can only provide clients' requests reports that lack detail and accuracy of what clients get from the Web site. Business analysts who seek more profit from the e-commerce Web site by analyzing customers' activities and interests can hardly understand these reports.

In this thesis, a three-part Web Data Mining prototype is designed and implemented. It provides an innovative solution to give business-oriented analysis of clients' activities for e-commerce Web sites. These three parts are the Web Content Logging module, Object-Oriented Web Store and Web Object Mining system. It suggested a relatively simple solution to capture interested data attributes requested by users via a Web Content Logging module with great flexibility in the configuration of these data attributes names. And it gives the ability to handle and capture information from static Web pages as well. The object-oriented Web Store prototype gives a demonstration for how to utilize the Web object content logging module to capture business interested data. Current popular

data mining techniques and algorithms are evaluated to discover interesting patterns among the Web Object log data in the Web Object Mining system. The Web Object Mining system provides an interactive user interface to give analysis results of the Web usage data for business analysts. With better understanding of how the Web site is serving its customers, business decision-makers are able to make more profitable strategies for e-commerce.

1.5 Organization of the Thesis

Besides this introduction – Chapter 1, the remaining part of this thesis consists of six chapters, described as follows:

Chapter 2: This chapter, entitled "Enhanced Web Content Logging Mechanism", presents an innovative module for capturing interested Web page contents. The proposed architecture also provides an infrastructure for Web developer to build the front tier in an Object-Oriented approach to make the current loosing structured Web front tier easier to develop and maintain.

Chapter 3: The third chapter, entitled "Object Oriented Web Store", presents a Web Store prototype designed in Object-Oriented approach and built on the infrastructure proposed in Chapter 2.

Chapter 4: The fourth chapter, entitled "Web Content Usage Log Preprocessing", presents methodologies to clean raw content log data and import cleaned content log data to relational database without losing the tree data structure of the content log data. Keeping the tree data structure is considered very important because it gives full flexibility in choosing interesting attributes for data mining in the next step.

Chapter 5: The fifth chapter, entitled "Web Content Usage Data Mining", presents data mining of the content log data by some common data mining algorithms to show some advantages of mining structured Web content log over the traditional Web access log.

First it shows granulated content usage statistics reports. Then it shows sample mining results by the k-means clustering algorithm and the PrefixSpan sequential pattern algorithm.

Chapter 6: The sixth chapter, entitled "Implementation", gives details about prototype development, running environment, and running results, including the performance test results of the infrastructure presented in Chapter 2.

Chapter 7: This chapter contains the conclusion and a roadmap for future work.

Chapter 2

Web Content Logging Mechanism

As discussed in Chapter 1, the current Web server access logging mechanism will not give us strong page content tracking and analyzing ability. To track what is actually sent to the client, we introduce an extra module, which parses all pages just before sending them to the clients, to the Web server.

Another issue worth mentioning here is that currently the majority work of Web developers are writing programs run on the Web server side to generate Web page content dynamically. These programs process data according to users' requests and company's business logic and apply the results to Web designer's page templates. In most cases, Web developers do not start to write the programs for dynamic content pages until the page layout templates are provided by designers. And there is no software tool to import and update Web designers' templates for Web developers for their server side programs, such as Java Servlet, Java Server Page, PHP Script and Active Server Page. It is extremely difficult for the developer to do some modification when the page layout gets very complex, such as multiple levels of HTML tables, etc.

The design goal is to provide a powerful content logging mechanism, which facilitates developing new Web sites/pages. The logging procedure is "transparent" to Web developers, and the logging information module is fully configurable by the Web server administrators.

The eXtensible Markup Language (XML) emerging technology can be applied to Web development to separate page layout and page content. In my design, the Web developer has no need to care about the actual page layout. What he/she needs to do is only to add one line code to include the eXtensible Stylesheet Language (XSL) file provided by the Web designer. The Web developers only concentrate on the page content from business logics.

2.1 Technical Background

The prototype is built on a popular Web server, Apache HTTP server, and Jakarta Tomcat, which is a Java Servlet and Java Server Script (JSP) engine. Because Apache HTTP Server is an open source software, I was able to examine its logging mechanism in detail.

The Apache HTTP Server has a very small kernel, which simply handles HTTP request and respond with static content. However it is highly configurable and extensible with extra modules using the Apache module API. The Apache HTTP Server package comes with some basic modules to enhance the functionalities of the Web server. Module *mod_log_config* is one of its basic modules. This module provides some logging features of client requests. There are four directives provided by this module: *TransferLog* to create a log file, *LogFormat* to set a custom format, *CustomLog* to define a log file and format in one step, and *CookieLog* to set the filename for logging of cookies. Once module *mod_log_config* receives the HTTP request header passed from the kernel, it parses the header and writes the information into log files or databases according to customized directives. Another module, *mod_usertrack,* is an extension of module *mod_log_config*, which logs users' activity on a site using cookies. The cookie can be logged by using the text *%{cookie}n* in the configurable log format file. Analyzing the Web server's log file can generate clickstreams of visitors' activities on a site. Here is the list of the information that can be logged by these two Apache HTTP Server modules

- The common log file format [1, 2]:

remotehost	Remote hostname (or IP number if DNS hostname is unavailable, or if remote host IP DNS reverse lookup is turned off).
rfc931	The remote logname of the user.
authuser	The username by which the user has authenticated himself.
date	Date and time of the request.
Request	The request URL exactly as it came from the client.
status	The HTTP numeric status code of client's request, such as a 200 means the request is fulfilled, and a 404 means that the server has not found anything matching the URL given
bytes	The content-length of the document transferred.

- W3C Extended Log Format contains more information than the common log file format. It can log all the information contained in HTTP header with totally 47 parameters according to RFC *2616* Hypertext Transfer Protocol -- HTTP/1.1. The following are some parameters, which are supported by all major Web browsers, and relevant to Web usage analysis [3].

Method	The method of the HTTP request, could be GET, PUT, POST, etc.
Referer	Site and page that referred visitor to this page.
Cookie	Any cookies sent by the browser.
URL-QUERY	CGI arguments appended after the URL.

  Some other well-known parameters include USER-Agent, and Content-Type, etc.

A modern data driven e-commerce Web site uses programs in the server side to accept users' requests, query the database and generate Web pages for the clients. This kind of Web pages are usually called dynamic Web pages because the contents of these Web pages are generated dynamically on the contrary to static Web pages in which the contents are precompiled. Instead of writing a Web page for each product, a data driven Web site uses only few templates embedding with database query results to show product information interactively. The dynamic content solution greatly assists in managing a Web site with many pages. Some HTTP Web server vendors introduces extra modules,

usually called Web Application Server, to support dynamic content publishing using common programming languages, such as Perl, C and Java. Jakarta Tomcat provides commercial-quality dynamic Web page content server solutions based on the Java Platform. It is developed under the Apache license in an open and participatory environment. The Jakarta Tomcat works as a Java Servlet and Java Server Pages engine for Apache HTTP Server following Java Servlet Specification [41, 42, 43]. The logging facility follows the rules of Apache HTTP Server, however, it has been developed in very low priority thus has very limited functions. Although Tomcat can be run as a Web Server standalone, this solution is not suggested for production environment usually for better scalability and stability. Apache HTTP Server is recommended for handling static information, such as image files, because the native C language performs much better in low level system operations, such as I/O. To get better performance and manageability in a production environment, the Jakarta Tomcat Server is always combined with the Apache HTTP Server by setting up the Jakarta Tomcat Server as the Java Servlet and JSP Engine [44, 45]. Figure 2.1 gives an overview of how the clients' requests are handled internally with this solution. Another important reason for adopting the Apache HTTP Server to handle static information is that this architecture keeps the Web content log clean for data mining. Due to the stateless connection property of the HTTP protocol, several requests are often made as a result of a single page request, i.e., when the URL of a page is requested, several requests will be generated by the Web browser automatically to download related files, such as static image files, JavaScript files, and Style Sheet files, usually contained in that page. These requests are usually not of interest and always need to be cleaned out from the raw log in order to apply data mining algorithms more efficiently [4, 5, 6, 7]. These static file requests are now filtered out by the Apache HTTP Server from the Jakarta Tomcat Java Application Server. So the extra Content Log plug-in only captures selected users' requests. This eliminates the log requests cleaning step in the later data mining phase.

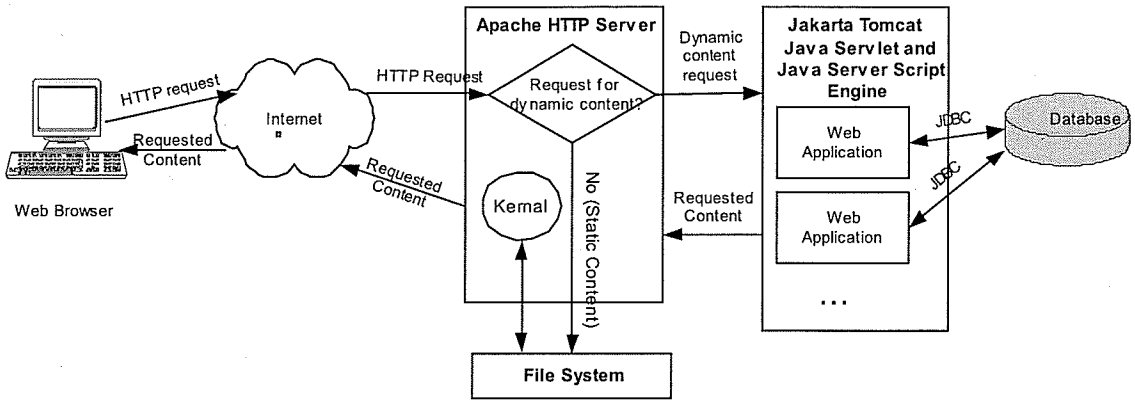

Figure 2.1 Overview of Processing Clients' Requests

2.2 Enhanced Content Logging Web Server

Web pages can be categorized into static pages and dynamic pages. Static pages are usually plain Hyper-Text Makeup Language (HTML) files saved in the Web servers file system. When a client request comes, the Web server reads the requested file from the file system and passes the content to the client. The file content does not change in this whole procedure. The same content is returned from the Web server for requests with the same Unified Resource Locators (URLs). Dynamic pages are output of server side programs. When a client request comes, the server side program generates the content for the responding Web page dynamically. The content is usually the results from database queries, and it can be different for requests with same URL.

This thesis focuses on capturing and analyzing the rich content of dynamic pages as well as historical static pages. In order to support logging the content of static Web page, a static Web page handler for Tomcat is developed as well. So the log information from Apache HTTP Server which usually only contains requests for image files in this solution can be ignored to simplify the data cleaning step in the data pre-processing stage for Web usage mining.

To support rich content logging and server side XML/XSL transformation, two plug-in modules for Jakarta Tomcat Server, the Content Transforming plug-in and the Content Logging plug-in, are developed.

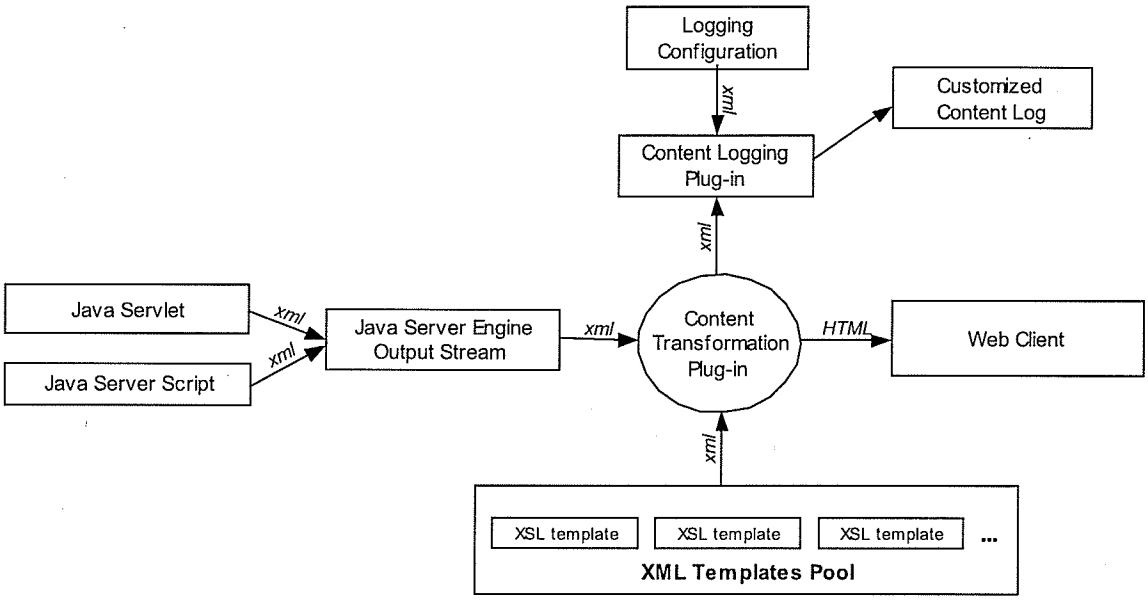

Figure 2.2. Data Flow Diagram

2.2.1 Dynamic Web Page Content Transforming Plug-in

One of most challenging things in integrating new technologies, such as Internet services (including Web services, email services, and Simple Object Access Protocol (SOAP) services, etc), into enterprise's legacy IT infrastructure is to integrate heterogeneous data formats from different applications running on different platforms. XML offers a good data inter-connection infrastructure for these applications [8, 9]. In order to provide application independent data, we must separate the information itself from its presentation, such as font type, and font size, etc. The current Web content generation is mostly based on HTML, but HTML mixes formatting tags, descriptive tags and programmable logic (both on server side and client side). The most obvious weakness of HTML is that it is not structured. It is hard to check the integrity of the data. The Content Transforming Plug-in aims to change the way Web information is created, rendered and served.

It completely separates the presentation and business logic layer while it keeps the integrity of data for business logic. It offers a different way of working of traditional dynamic content publishing which edits mixed layout control tags and contents. It uses XML to describe business data, and uses XSL transformation capabilities to merge business data and presentation format. So each layer to be independently designed, created and managed, reducing management overhead, increasing work reuse, and reducing time to market.

For Web developers that write code to query database and generate dynamic content according to the parameters' values sent by clients, the only change for building Web sites over the prototype will be that the data format of the contents of all output should be changed to formal XML instead of previous HTML. The size of Java Servlets' XML output data is less than 20% the size of regular HTML output in the Web Store prototype. Web developers will benefit from less complexity in their programs, and easier maintenance for the codes. Another advantage is to support different final output data format by simply applying different XSL templates. This feature can be used to support different kinds of clients, such as different Web browses running on PCs, workstations, and hand held facilities, SOAP and Electronic Data Interchange (EDI) requests, etc. separately with the same business data. In the XML specification, tag *<?xml-stylesheet?>* is used to indicate which XSL template the current XML file should apply for format transformation. Once the Web application output contains a line likes,

<?xml-stylesheet type="text/xsl" href="complex-06.xsl"?>

the content transforming plug-in will try to locate and XSL file according to the URL value of the *href* attribute. The XSL file can be anywhere accessible via HTTP request. After the content transforming plug-in finishes loading that template, the Web applications' initial output in XML format will be replaced by the transforming result according to the XSL template indicated by the Web application. A Web site usually has limited number of XSL templates and these templates are usually relatively small in size (few kilobytes). Considering there

may be too much overhead loading these files from hard disks or even remote Web servers each time requested, a templates pool caching all the templates that may be requested. A hash table is maintained for quick lookup of the entry memory address for each template.

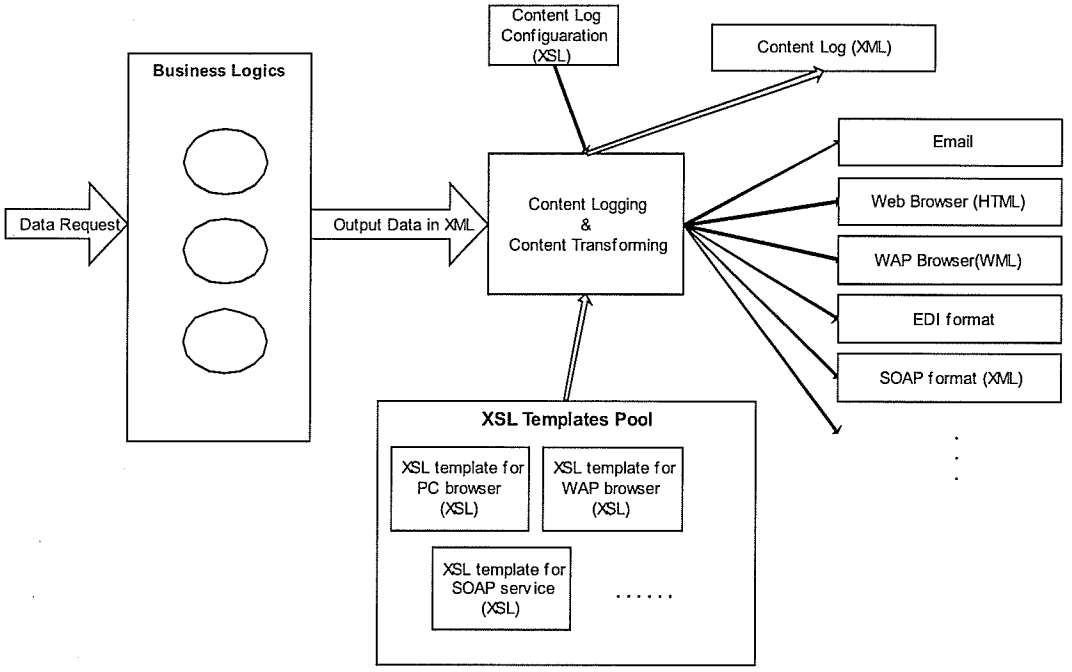

Figure 2.3 Data Transformation Diagram

2.2.2 Dynamic Web Page Content Logging Plug-in

Web pages and business-logic are getting more and more complex, so many parameters will be required to adequately log the content of these pages. These parameters may be in different realms, and are usually not well organized. A typical dynamic URL can contain user authentication information, preferred page layout information, navigation path information and data query, etc, and there is no need to put all these parameter name and value pairs in a certain sequence for parsing. These may cause some difficulty in analyzing Web utilization for business purposes.

Moreover, the current user's request logging mechanism, shown in Figure 2.4 may lead to a biased analysis because there is no way to tell what is exactly returned to the user. Although static page content can be associated with request URL to provide a content utilization analysis, the result will be incorrect if the content is updated often. The situation is even worse for Web sites using dynamic Web pages. For example, when there are many items in one category, the products are usually shown on many numbered Web pages. So only a portion of all products will be showed on each page, sorted by some attributes. Because the product database can be updated very often, adding / deleting products may result a product to be shown on a different page. So it is very likely that two requests with same URL and parameters may get different page contents.

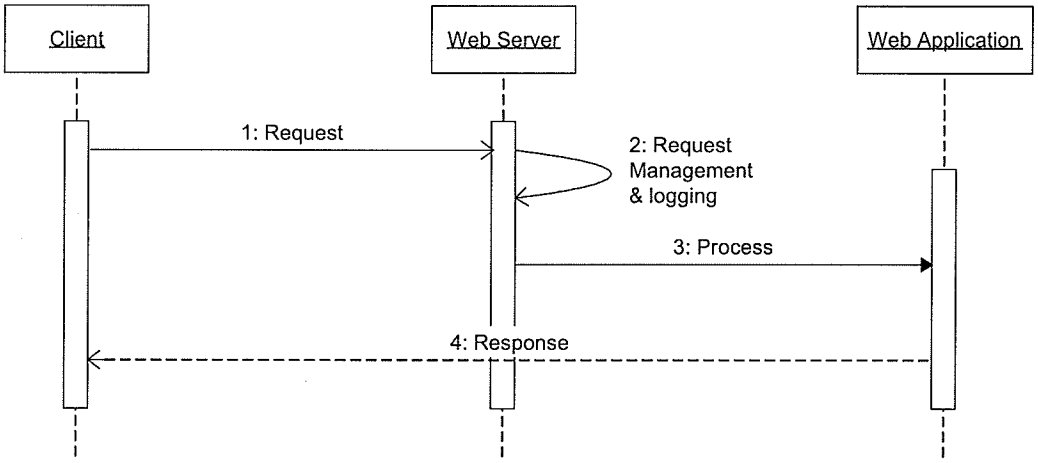

Figure 2.4 Traditional Web Logging Procedure

Figure 2.5 shows the newly designed Web Content Logging Procedure. A new content logging module is introduced which is different from all existing logging modules. As we discussed above, the traditional logging mechanism does not yield strong page content tracking and analysis ability. The Web Content Logging Plus-in is used to track what information / data is actually sent back to the clients.

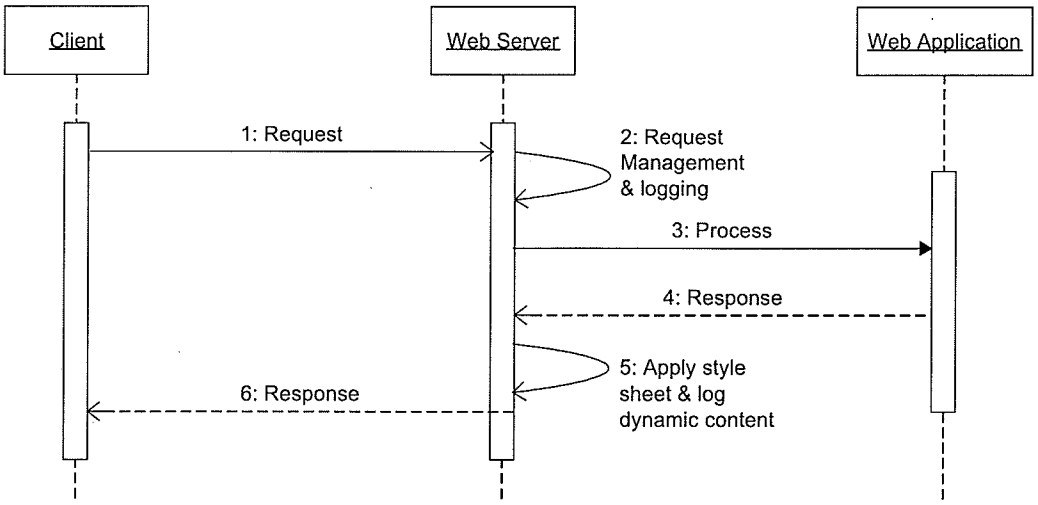

Figure 2.5 Web Content Logging Procedure

While the Content Transforming Plug-in is transforming the raw XML output from Web applications, the Content Logging Plug-in performs quick parse of the same data. It generates an abstract of the interested data according to the Content Logging configuration file, the *log.xsl*. Instead of writing an extra content parser and define a configuration file, XSL transformation is used, making this job much simpler. Any system administrator or analyst has some knowledge of XML and XSL can control what to log from the content of his interest. Although many current Web sites are still heavily based on HTML, most recent projects (for Internet, Intranet and Extranet) are adopting XML solutions to facilitate data inter-exchange and reduce the overall development and ownership cost.

After the dynamic content is generated, the Web application writes the content into an output buffer, and the Web server manages sending the content in the buffer back to the request user. The content log plug-in intercepts the content in the buffer and uses XSL transformation to generate log data for the content of each response.

The implementation is developed by Java and runs on Tomcat 4. Figure 2.6 shows the XSLTFilter class diagram, figure 2.7 shows the client request and response

sequential diagram, and figure 2.8 shows the Web content rendering and logging procedure of the implementation.

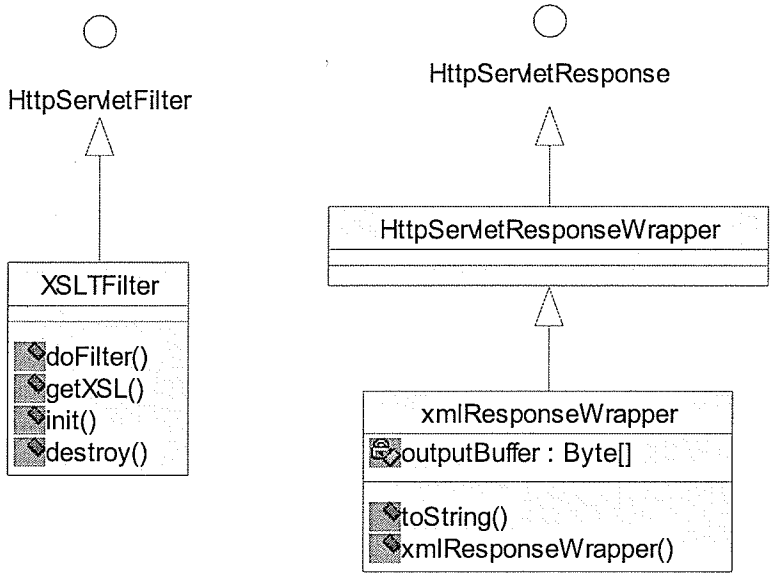

Figure 2.6 XSLFilter and xmlResponseWrapper Class Diagrams

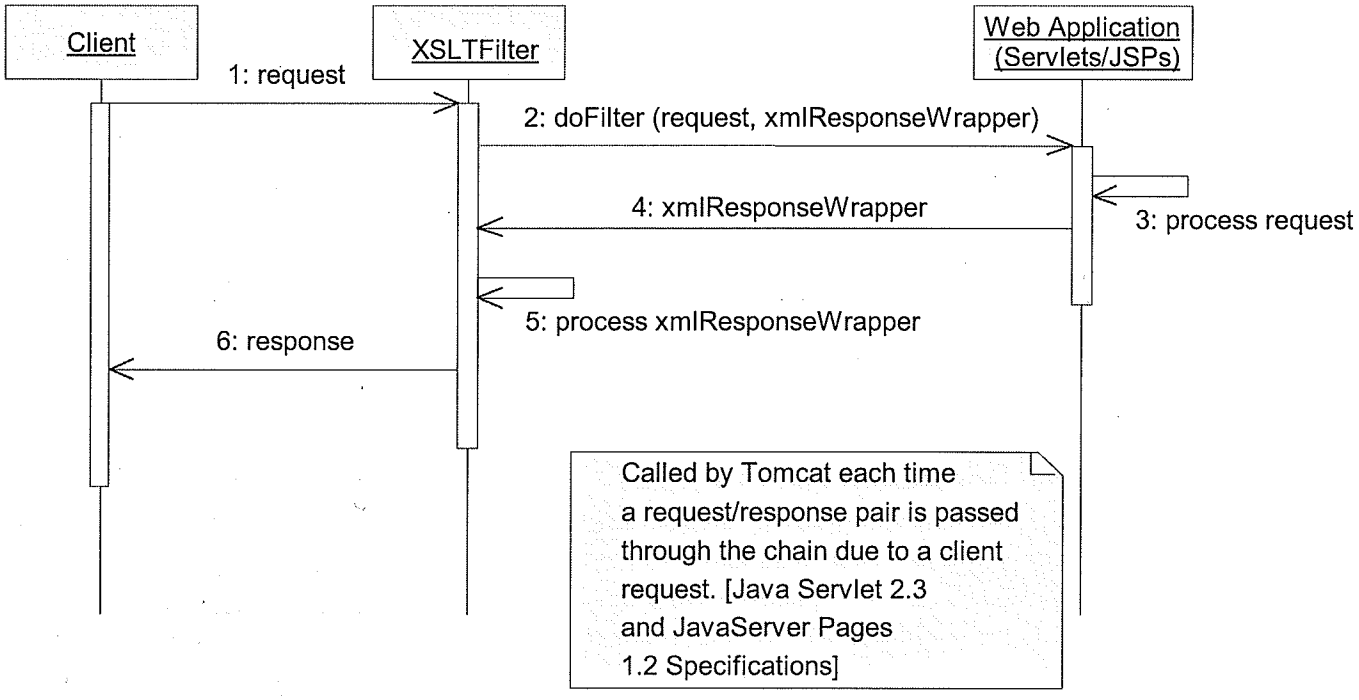

Figure 2.7 Client Request Sequential Diagram

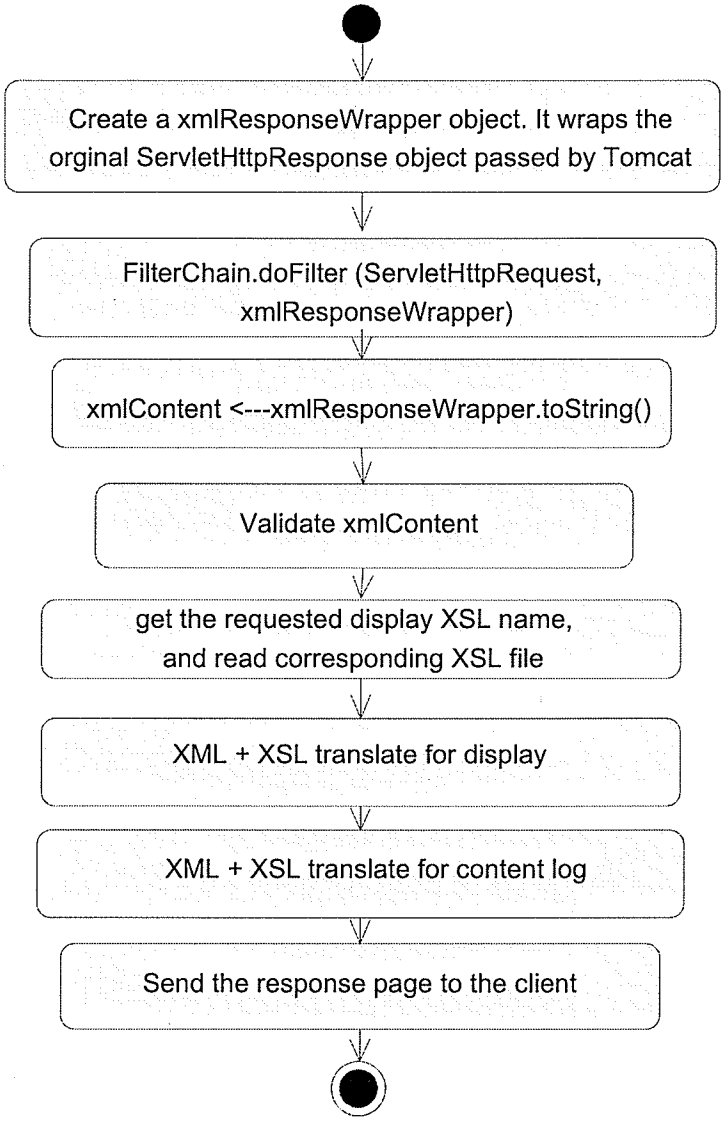

Figure 2.8 Web Content Transform Procedure

The log information is saved into relational database to allow easy data pre-processing for the log analysis. A database record insert operation has a little overhead than appending data operation on a file. The log can also be saved in a normal text file in a heavy-duty server.

The content log database keeps track of the following data:

RemoteIP	Remote client IP address
RemoteHost	Remote client hostname, if available
Timestamp	Timestamp of the output, the data type is long instead of popular SQL data type date for less overhead both in record insert and data analysis pre-processing later.
SecureConnection	If the output data will be encrypted during the transmission to the client, usually for critical data, such as credit card information, using Socket Secure Layer.
RemoteUser	Username, if the user logged in.
SessionID	A unique random string used to track users. It is used to identify a user. A user can not be identified just by an IP address, because of the use of proxies and gateways.
SessionAttr	Instead of saving all session attributes in the cookies, Jakarta Tomcat keeps users' attributes in the server associate with the SessionID. This implementation enables Web applications keep tracking users even if the cookie is disabled. It also avoids users sending fake data in the cookies.
ReqURL	Client required URL.
RequestParameters	Client submitted information query data together with the ReqURL. The Content Logging Plug-in capture users' HTTP POST data as well, while all current popular Web server log systems only capture HTTP GET data. HTTP GET is considered less secure than HTTP POST because HTTP GET method append all the parameter name and value pairs after the URL. All these information can be shown in the browser's address in plaintext.
Referer	The reference page led to this page.
TemplateName	The XSL template name requested in the output.

Content	This field is used to save the abstract of the output page content in XML format. Common XML file header tags are eliminated to reduce the data size.

2.2.3 Static Web Page Content Logging Plug-in

Although most Web sites adopt dynamic Web page technologies nowadays, a small number of Web pages are still developed as static Web pages. Static Web pages are usually developed much easier and quicker with development tools. The Web site entry page of many Web sites still use static page. Static Web pages have some advantages over dynamic Web pages. They can always be served with quicker responses, and often includes rich multimedia for more impressive presentation. Because static pages may be updated regularly, it is necessary to keep tracking the content sent to clients' as well. None of the current popular Web servers include such functions to capture the page content. So a Static Web Page Handler is developed to capture important information in static Web pages.

The current HTML specification does not have any support to describe part of a Web page's content, although it defines *TITLE* and *META* tags in the HTML page *HEADER* for description of the current page. Because XML provides a good technology to describe data, it is introduced into the static Web pages as metadata to describe the data and define the data type. XML island is a technique of inserting structured XML data inside HTML pages. Web browsers such as IE5 consider these XML elements as objects in the HTML Document Object Model. The XML data island is created with the XML tag and it does not affect the look of the page.

The Tomcat Web Application Server is configured to handle static Web page request too in order to assign session id to the client and keep tracking users requests through the whole session. The Static Web Page Handler (SWPH) basically reads the requested static Web page from the file system and sends its content to the client. During this procedure,

SWPH first indicates that the output will be in HTML instead of XML, which is used for dynamic Web pages, in the HTTP header Content-Type field. The Dynamic Web Content Log module and Dynamic Web Content Transform module ignore any output with Content-Type other than XML and route the output to Static Web Page Log module. The Static Web Page Log module parses the HTML content, recording the TITLE and META tags in the HTML header, and required XML islands in that HTML file based on fields indicated in the static Web page configure file. Adding XML islands into existing HTML pages is a minor task for most Web administrators. The data and its structure in the XML islands are only required to follow XML specifications. This gives flexibility to build the data structure and makes utilizing various XML data processing tools easy. A sample XML island describing a flash animation is given in Table 2.1 on page 26. The sample XML island with id = "xml_island_01" provides some description of the Flash animation. This kind of information in the Web log may provide a better understanding of visitors' interest. The current Web log only contains the URL of this page. It has no further knowledge of its content. For example, once the Flash animation is redesigned, the current Web log will have no knowledge of this kind of update. Thus, the final reports may lose insights in many situations. On the contrary, by analyzing the new Web Object Content Log, we are always able to find which design visitors like better.

To improve the performance of serving static pages, this Static Web Page Handler preload static Web Pages from the file system into the memory. So the content for every static Web Page request will not read from the slow I/O system unless the timestamp of the file is changed.

Another enhancement is to remove XML islands from the original content when necessary to give best backward compatibility of old Web browsers.

```
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#ver
sion=5,0,0,0" WIDTH=400 HEIGHT=275>

<xml id="xml_island_01">
<Name>intro.swf</Name>
<Version>1.0</Version>
<ReleaseDate>01/01/2001</ ReleaseDate>
<Author>J. Zeng</Author>
<Description>Spring Products Introduction </Description>
</xml>

        <PARAM NAME=movie VALUE="intro.swf">
        <PARAM NAME=loop VALUE=false>
        <PARAM NAME=quality VALUE=medium>
        <PARAM NAME=bgcolor VALUE=#000000>

        <EMBED      src="intro.swf"
                    loop=false
                    quality=medium
                    bgcolor=#000000
                    WIDTH=400
                    HEIGHT=275
                    TYPE="application/x-shockwave-flash"
                    PLUGINSPAGE=
"http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=Shock
waveFlash">
        </EMBED>
</OBJECT>
```

Table 2.1 An XML Island Sample

Chapter 3

Object-Oriented Web Store

The World Wide Web (WWW) has become the most popular platform for E-Commerce applications, far beyond traditional Electronic Data Interchange (EDI) and third party's proprietary applications. Web applications combine navigation through an electronic version of product catalog, visitor self-served profile and order management and shopping cart together. Building complex e-commerce Web applications is a time consuming task.

Web applications are becoming increasingly complex and mission critical. Object-oriented programming is a method of designing and programming based on a hierarchy of classes, and well-defined and cooperating objects. It assists in managing the complexity, and allows quickly developing and deploying scalable, reusable Web applications. The primary benefits of object-oriented programming are the reuse and high quality of code that it offers. Because reusable code is usually better written, it is more reliable and easier to debug. By building a library of tested objects, there is no need to test each object again, only to test the aggregation of the objects. This aids in rapid application development and prototyping. Using design patterns, the object-oriented approach to software development has proven to be of value in building applications of all degrees of size and complexity. A design pattern describes a proven solution to a recurring design problem, placing particular emphasis on the context and forces surrounding the problem, and the consequences and impact of the solution [27, 28]. Design patterns are generative, reusable and proven [21]. They are abstract representations of the very morphological rules which define the patterns in the world

[21]. These patterns have been designed by very experienced experts who have knowledge and insights of certain problems, and successfully used by other developers across the world for various projects. Furthermore, most contemporary programming languages (such as Java and C++) and development tools (such as IBM VisualAge for Java) are object oriented.

An application framework refers to a mature platform geared toward application development and deployment. It helps ease the process of developing large-scale applications that are scalable, secure, and maintainable. Market leading application framework suites, such as IBM WebSphere, Sun ONE Application Framework, and Microsoft .Net, all include common Web and e-commerce application modules for building Model-View-Controller design pattern based interactive applications. The Model and Controller parts are usually for business logic and interaction processing, and the View is for presenting the user interface. Windows-based (including X-Window and Microsoft Windows) applications use system visual components to build Graphic User Interface (GUI) and all these visual components are objects. Although currently most Web applications are adopting object-oriented approach in developing the backend programs, usually business logics, there is little success in applying this approach to the front-end of Web applications because most popular front-end technologies, such as HTML, Microsoft Active Server Page (ASP) and Sun Microsystems Java Server Page (JSP), are not object-oriented. HTML is loosely structured and very high tolerance for errors. This nature of HTML makes it hard to notice and trace problems in front-end programs.

Another obvious weakness of current Web solutions is that all Web pages do not have any metadata for their unstructured content. The Web application's interface is most likely going to be HTML. The HTML that is rendered by browser is just a long string, with mixed data attributes and view control tags in loose structure, instead of presenting business objects' attributes in strict format. So it is hard to analyze for content logging for business analysis purpose. Some Web application server vendors are introducing special content logging APIs to developers. The logging procedure and content are then must be

hard coded in the Web applications. Therefore, it does not provide any flexibility with poor backward compatibility.

The XML can be used to structure information. XML documents are structured which contains both content and some indication of what role that content plays. So XML document can be used to describe objects. Once the data format in data flow/exchange among the Web application(s) follows the XML specification, the Web application can then be developed in a pure object-oriented solution. This benefits not only the developers for easy debugging and code reuse while keep the quality of the software, but also the business analysts for business object analysis.

Adopting XML and its related technologies, the prototype Web store is developed in a pure object-oriented approach using multi-tier application architecture and model-view-control design pattern. Multi-tier application architecture and model-view-control design pattern based solutions are able to clearly separate the front view from the business logic in the backend, and business objects can easily be mapped to the front-end presentation. Moving mapping object attributes to XML formatted data and rendering XML data procedures from application level to the system level reduces the complexity of application and improves the stability of the whole system. A clean XML view of object data helps developers to trace problems. The object-oriented approach can greatly shortens the development period and assists the maintenance of the code of the prototype Web store. Moreover, it provides an efficient way for flexible content logging for business analysis.

3.1 Overview of the Object-Oriented Web Store

An XML-based Web store built on the Web server with Web object content rendering and logging modules is designed in an object-oriented approach to facilitate Web application development and take advantage of the Web server's content logging module for business analysis.

3.1.1 Developing an Object-Oriented Web Store

We consider the output of Web site as a collection of groups of related object rather a collection of pages. Extending object-oriented design to the front-end programs of the Web applications greatly assists developers to achieve a better way of modeling, organizing, and interconnecting Web data. When the Web page is designed using OO methodology, it not only makes easier for the Web developers to generate dynamic pages, but also makes the Web pages become well structured. In addition, the Web content and structure analyzer can derive benefits from structured contents. The content logging plug-in benefits from the object-oriented based Web site because the log configuration file can indicate exactly what objects and attributes are interesting that need to be tracked by metadata.

3.1.2 Separating the View from the Business Logic

When developing simple Web applications for a single type of client, we tend to mix the data access and data modifying logic with the logic for various client views. But as the clients need more flexibility to support other platforms, or other output formats, such as PDAs, PDF files, SOAP clients, etc, this solution becomes problematic because:

- Different versions of the same application to suit each type of client needs support must be developed.

- Since the code for views and that for data access / modification is intertwined, the code for the data access/modification is duplicated everywhere, thereby making the application almost unmaintainable. This design approach causes the unnecessary extension of the development life cycle.

Separating the presentation from business logic and application control modules enables division of developer responsibilities, and thereby produces faster time to market. Another reason for using this design pattern is to demonstrate that the content logging

module can provide an insight view of the Web usage analysis for business purpose in object level, while the current Web logging mechanism can not because requests for different purposes are sharing few URLs for this pattern.

The prototype Web site is an online store selling furnitures. All products are grouped in series according to their styles. The online store provides the following functionality:

- An overview of all series, an overview of each series which contains description and a list of all products in this series, and a detail view of each product such as name, descriptions, specifications, grades and stock information.
- User registration: It sets up users' profiles which include username, password, billing address, shipping address, email address, contact phone, and user order history.
- User login and logout: Users are required to login before checking out in order to identify the user.
- Shopping cart checking out: The Web site verifies current session and user. Only a valid registered user can checkout order. A successful order will affect stock information and user's profile.
- Editing of shopping cart: Items can be added to, deleted from, and changed in quantity in the shopping cart.
- Dynamic product recommendation: Shortcuts to some products and ad bar are generated dynamically and added into the return page.
- Editing user profile: A user can change his / her personal information, such as password, addresses, etc.

Additional function includes tracking visitor mouse activities over the Web page to find out any potential visitors' interests.

3.2 Building the Web Store in an Object-Oriented Approach

Developing a loosely-coupled Web applications, such as a Web store, using the Model-View-Controller (MVC) architecture as the design pattern is highly recommended. The

MVC paradigm is a way of breaking an application into three parts: the Model, the View, and the Controller.

Model — The heart of an application. The Model performs necessary business logic computations and maintains the state and data required. The model provides methods for appropriate external control and manipulation as well as state reporting methods to extract values from the model. It also notifies/updates known views of any state changes within the model that may affect the View.

Controller — The interface presented to the user to manipulate, control, and use the program.

View — The interface that displays information about the model to the user. To obtain change notifications or updates, a view needs to register with the model.

By applying the MVC architecture to a Web application, data access logic is separated from the data presentation logic. Figure 3.1 shows how the MVC architecture can be mapped to multi-tiered Web applications. The mapping of the MVC architecture multi-tier application follows:

- All the business logic to process the data can be represented in the Model.
- The View can access the data through the Model and decide on how to present them to the client. All the Web pages belong to View.
- The Controller can interact with the View and convert the client actions into actions that are understood and performed by the Model. The Controller also decides on the next view to be presented depending on the last client action and results of the corresponding Model action(s). A common approach in Web application is to have a centralized user request handler as the Controller. In the Java programming world, it is usually implemented as a Java Servlet.

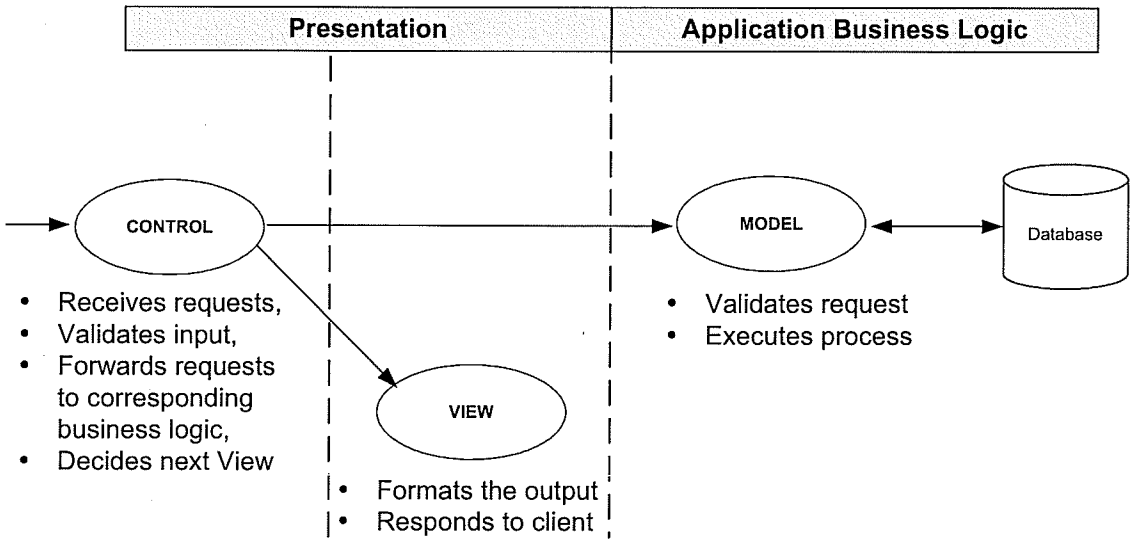

Figure 3.1 Model-View-Controller (MVC) Overall Architecture

3.2.1 The Controller

The controller is responsible for coordinating the model and view. The view depends on the controller for view selection. The controller instructs the view to use a particular page template and include what attributes of requested objects. On the other hand, the model depends on the controller for making state changes to the model. The controller accepts user's request from the view, translates them into business events based on the behavior of the application, and processes these events. The processing of an event involves invoking methods of the model to cause the desired state changes. Finally, the controller selects the page template shown in response to the processed request.

The prototype Web store has three controllers, the *Browse*, the *User*, and the *ShoppingCart,* which are written as Java Servlets. Each Controller accepts corresponding requests and determines the view of the response.

The *Browse* Controller, shown in Figure 3.2, mainly provides product information for visitors, assists them navigate through the product catalog, and provides shortcuts to promotional items.

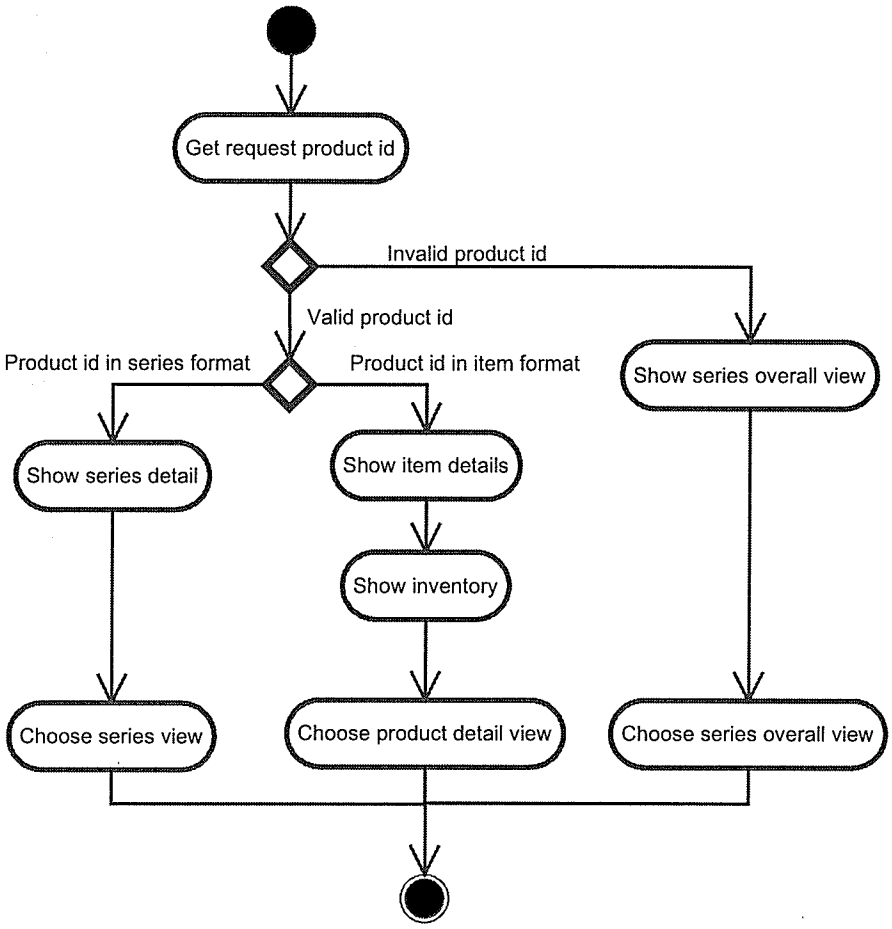

Figure 3.2 Activity Diagram of the Browse Controller

The *User* Controller, shown in Figure 3.3, accepts requests for product request, user registration, user login validation, and shopping activities.

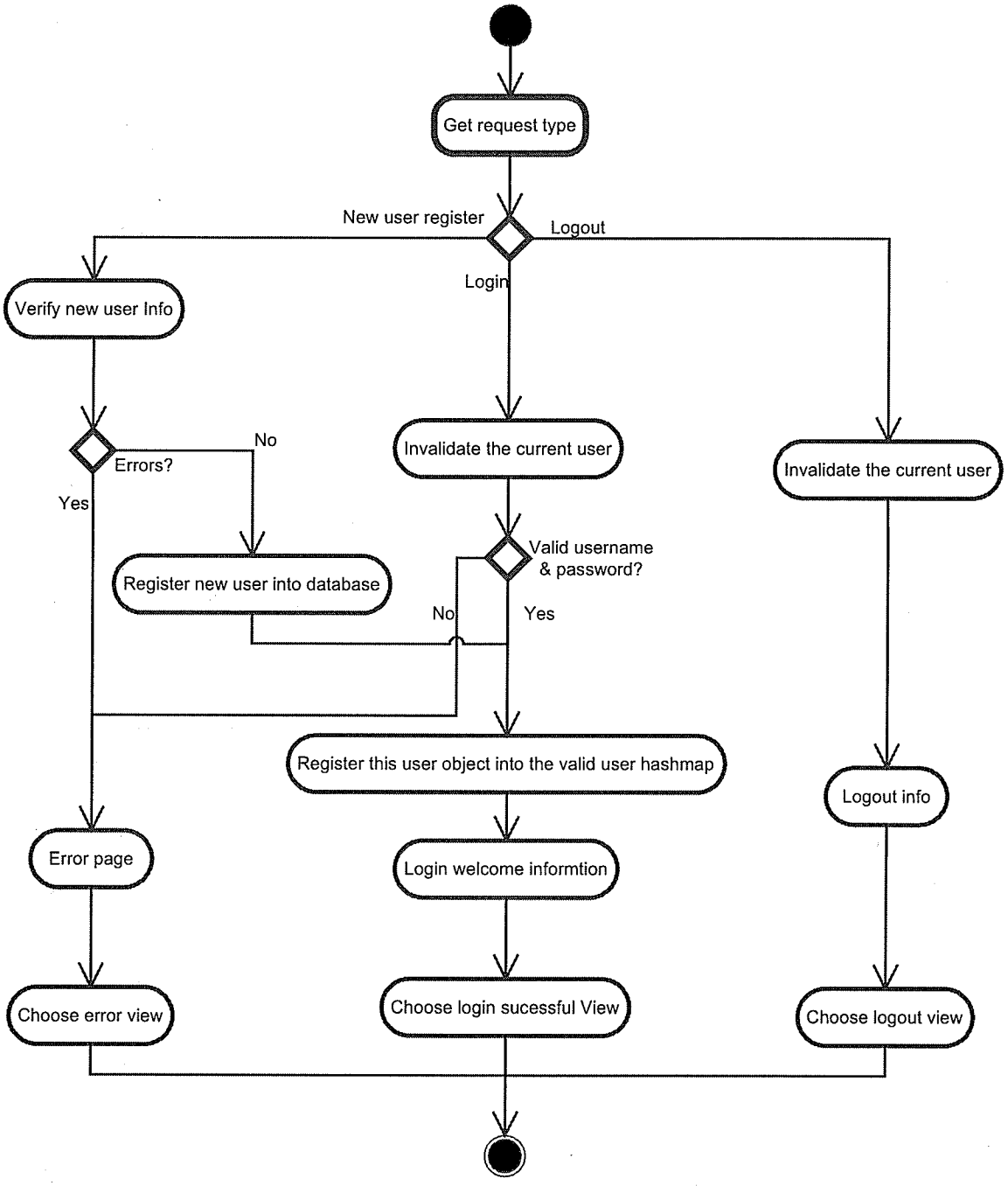

Figure 3.3 Activity Diagram of the User Controller

The *ShoppingCart* Controller, shown in Figure 3.4, focuses on managing user's shopping cart. It accepts requests for adding, deleting, and modifying the items in a user's own shopping cart, and checking out the shopping cart.

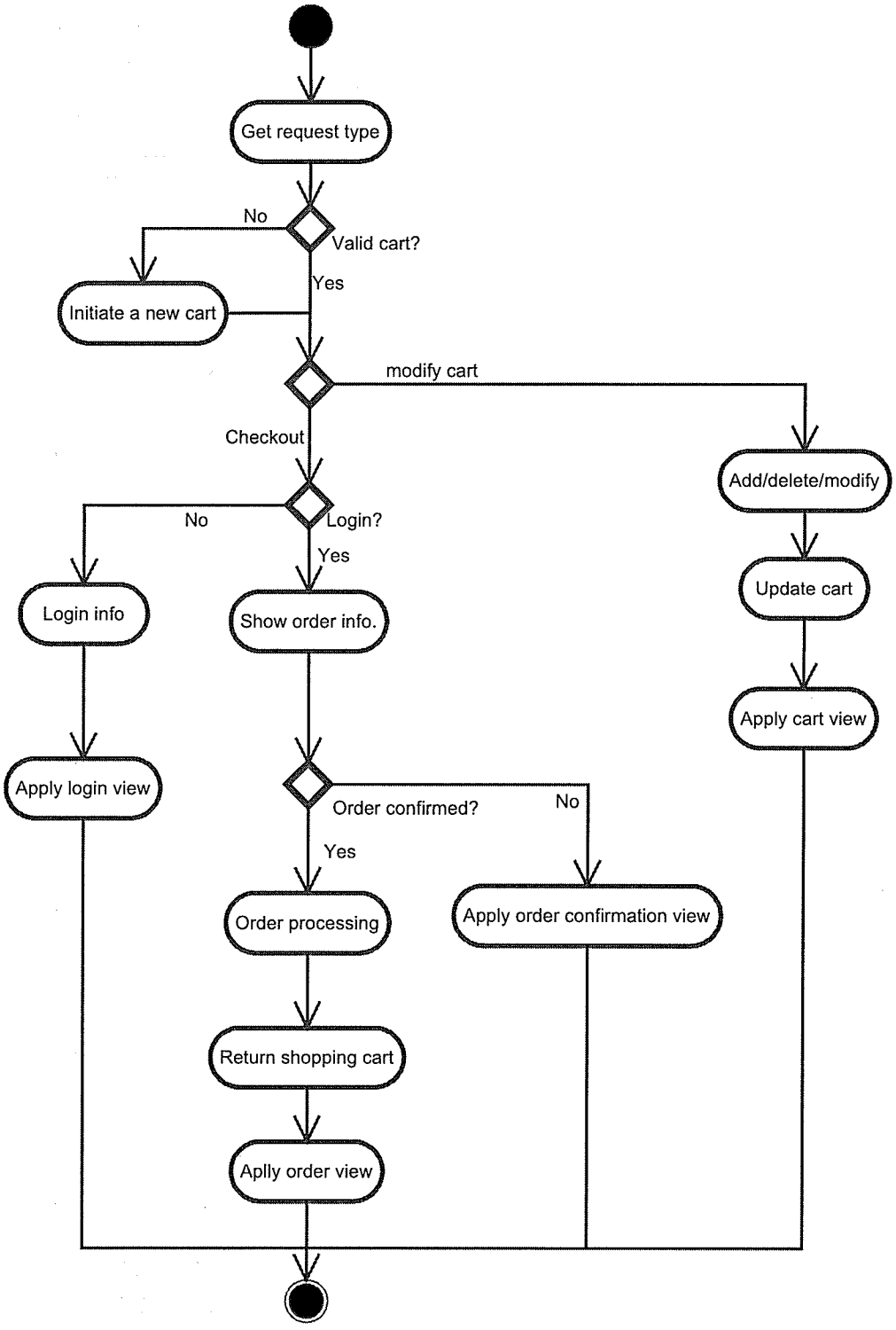

Figure 3.4 Activity Diagram of the Shopping Cart Controller

3.2.2 The Model

The model is the heart of the Web application. It represents the business logic that govern the access and modification of business objects. The Web store basically has two main objects, the Product and the Customer as showed in following UML diagrams.

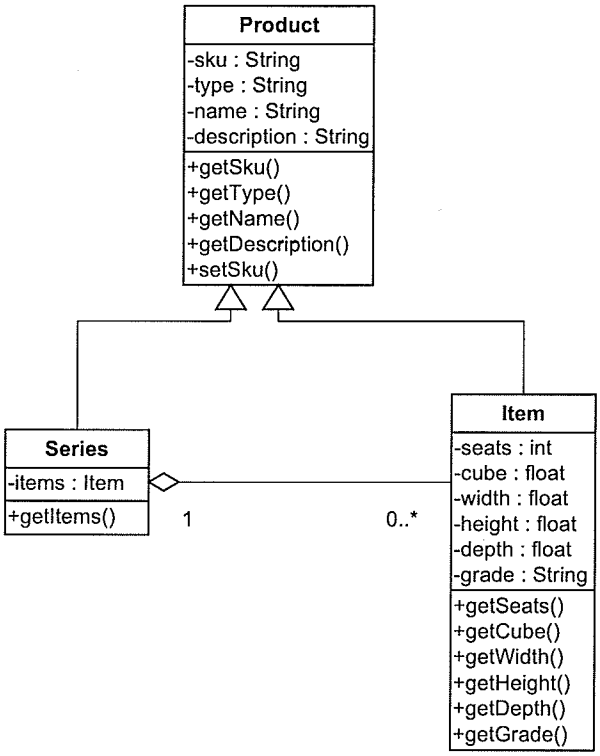

Figure 3.5 Product, Series, and Item Class Diagram

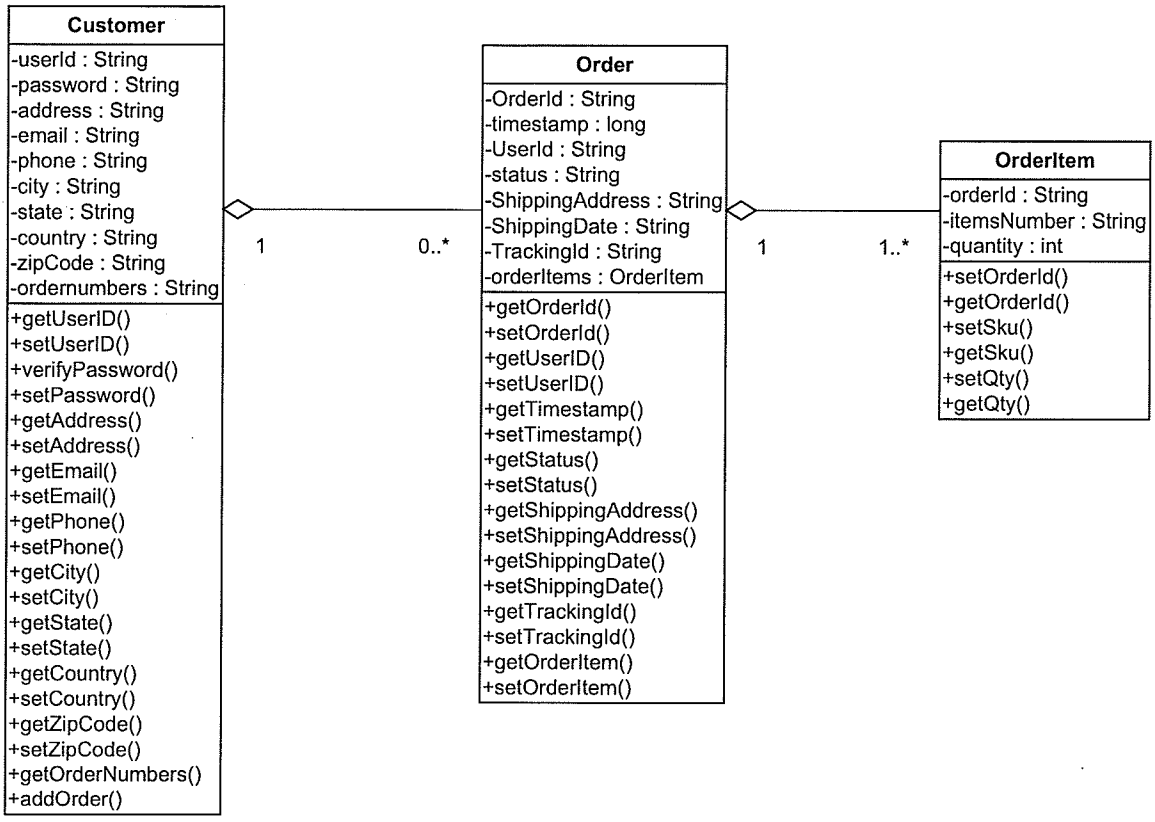

Figure 3.6 Customer, Order, and OrderItem Class Diagram

Using object-oriented approach, the Web page itself can be considered as an object too. As illustrated in Figure 3.5, the Web page is modeled as a class, called *Page*, which consists of three objects: *Header, LeftPanel,* and *MainPanel*. An e-commerce Web page usually consists of three parts:

- A page header which usually contains ad bar, navigation bar and user information,
- A site navigation panel which usually sits on the left side in the Web browser's window,
- A main panel which contains detail information for the products.

The names of these objects do not imply any view information. They can be displayed anywhere in the Web browser's window instructed by View.

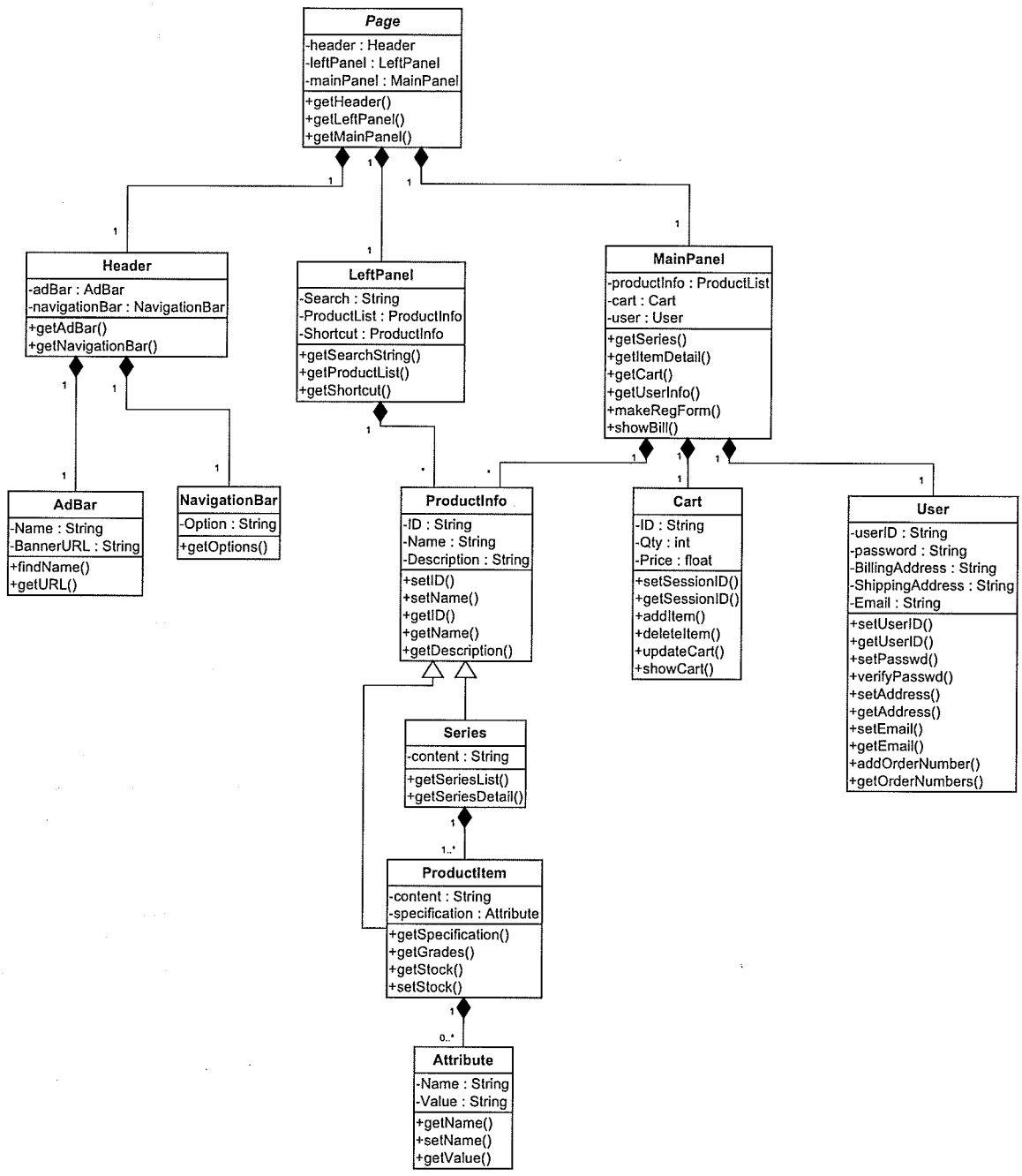

Figure 3.7 Page Class UML Diagram

Dynamic Web pages consist basically a set of objects' attributes according to users' requests. Objects' attributes used in Web application(s) can be easily be mapped into XML document. Each class contains a method which gives its attributes' values in XML format. For example, the a *Series* object can be represented in XML as follows:

```
<Series ID="40000">
        <Name>Rosemont</Name>
        <Product ID="40000-51">
                <Name>Sofa Recliner</Name>
        </Product>
        <Product ID="40000-52">
                <Name>Sofa Table Rec.</Name>
        </Product>
        <Product ID="40000-53">
                <Name>Love Recliner</Name>
        </Product>
</Series>
```

For structured data of this nature, it is easy to capture interesting attributes. As shown in the above XML sample, the data contain no user interface information. This insulation from user interface information enables Web application software developers to concentrate on manipulating business objects. The tags are considered as metadata of the value of objects' attributes. Metadata is definitional data that provides information about other data within an application or environment. So other programs, such as the content logging plug-in, can understand how to handle the data.

3.2.3 The View

3.2.3.1 Generate the View

The design of the Web pages is important so as to give visitors good impression about the pages. The structure of Web pages tends to get more complicated nowadays by introducing dynamic technologies, such as JavaScript, Web browser plug-ins, etc. Modeling Web pages helps manage the increasing complexity in Web pages. In the MVC

architecture, the View renders the contents forwarded from the Model. The Controller specifies which view should be used.

The View acts as a staging step for translating server side objects to the client side objects in Web applications. The eXtensible Stylesheet Language (XSL) provides the technology for mapping server side object to client side object.

Traditionally, the View heavily uses server side script, such as Java Server Page (JSP), and Active Server Page (ASP), as the programming language. A typical JSP or ASP program is basically an HTML file embedded with simple logic and data access tags which access objects (such as JaveBean, COM) attributes. These programs are usually a mix of client script code, cascading style sheet, HTML, and server side script code. Most of these codes are loosely structured and with less grammar rules, and each part of the code may interfere with others. This kind of code is extremely difficult to maintain and debug sometime. For example, the character '<' is a reserved character in HTML, but business data do not have such restriction in most cases. If this character is contained in the business data, the Web browser cannot parse the whole document correctly and causes the wrong presentation of the data, both in the value and the look. Most likely, improper including this character will cause the client scripts to have fatal errors, and the Web page (User Interface) could become unreadable.

To apply strict rules for better code maintenance and debugging, I separate the business data from its presentation code. The business data is in XML format. Although the final presentation code is in HTML, it is never hand coded, but transformed from the XML according to the XSL code which strictly defines how to present each object. Both XML and XSL have very strict grammar. This approach enables the developer to locate the error very quick. The XML representation of the business data provides a common interface between the View and the Model. So following this interface, the View and the Model can each be developed, tested and maintained independently.

According to the user's request, the Controller decides which View (i.e. an XSL template in this prototype) should be applied to the business objects' attributes as the output from the Model. The Web server prototype provides the infrastructure for transforming the XML data to HTML code according to the XSL template selected for the client. This transformation is transparent to Web developers and visitors. Recently, some Web browsers, such as Microsoft's Internet Explorer (IE), support this kind of transformation in the browser. To provide the maximum backward compatibility, this transformation is strongly recommended in the server side. Another issue is that Web browsers are not developed following strictly W3C's specifications of HTML, JavaScript, XSL, etc. This causes different rendering in each of them even with the same code. Currently, if a Web page includes JavaScript, the page has to contain the code for Netscape Navigator (NN) and IE together in order to work properly, increasing the complexity of the Web page. Because the Controller is able to determine the client's type, such as IE, NN, or WAP browser, it can indicate which View template to use while each template is fine tuned for each browser. XML and its related technologies are new and they face very frequent upgrades, which may not fully backward compatible sometimes. Keeping the XML-XSL transformation in server side keeps a unique transforming processor, avoiding deployment hassles and troubles caused by using different versions of the processor(s) in the clients' side.

3.2.3.2 Additional Function: Tracking Mouse Activities

Visitors tend to put mouse cursor over the spot they are interested in while surfing. The most interested link tends to be clicked within a Web page. The Web server always logs this action. By examining the Web log, business analysts can find visitors' most interested links in each page. However, visitors may also be interested in other parts of a Web page. The current Web logging mechanism can only record a single interesting spot on each Web page, missing other spots a user may be interested in.

To find other visitors' interested spots, I use the Document Object Model (DOM) technology to discover these spots. A DOM document has a logical tree structure to allow programs and scripts access and update the content, structure and style of the document

dynamically [31]. While loading an HTML page, a DOM scriptable browser creates a hidden, internal hierarchical roadmap of all the elements it recognizes as scriptable objects. So the client side script (JavaScript in my implementation) uses the paths to reach these objects through the hierarchy to communicate with these objects. Each object has its own properties. The value of these properties can be read and set dynamically.

Tracking all the mouse activities may cause too much overhead, so only mouse activities over predefined "hot object" on a Web page are captured. The Webmaster or business analyst defines the “hot object” in the XSL templates. Each "hot object" is defined by either <DIV> tag or <SPAN> tag with a unique ID. It can be in any part of the Web pages, such as hyperlinks, images, blocks of text or even a part of inline content without visitors' notice. Each hyperlink on the Web pages are generated dynamically, appending the mouse activities the of HTTP request. The Web server filters out the mouse activities from the requests and writes to log database together with other required fields, such as session ID and request timestamp, for further analysis. In my prototype, I capture MouseOver, MouseOut and MouseClick events and record the object ID once the mouse cursor stays more than the predefined number of seconds over a "hot object".

Chapter 4

Web Content Usage Log Preprocessing

Web usage mining is the application of data mining techniques on Web data, which includes Web content data and Web usage data, to identify access patterns of Web users. These information assist companies in the better understanding of both the Web sites and Web users. Knowledge from user access patterns is very useful in numerous applications: supporting business intelligence, target marketing, personalization, and Web site design in content and structure, etc. This may increase the value of the Web site for the company.

For dynamic content pages, most of the Web content data are the query results from the database. Mining the Web usage data alone is unable to provide in-depth analysis as described in the Chapter 2. This thesis provides a way to capture Web usage data and Web content data in real time using a Content Logging Plug-in. This approach integrates Web usage data with related Web content data. Therefore, reference to "Web Data Mining" in the rest of this thesis means mining the integrated data logged by the Content Logging Plug-in.

Web data mining procedure is usually divided into three distinctive phases: data preprocessing, pattern discovery and pattern analysis. Data preprocessing is necessary to clean the raw data and produce appropriate data sets for applying data mining algorithms in the pattern discovery phase. The data preprocessing phase is probably the most

difficult task in the Web mining processes due to incompleteness of the available data [32].

There are usually three major tasks in the data preprocessing phase: data cleaning, session identification, and path completion. Data cleaning is the task of removing log entries that are not needed for the data mining process. Session identification generates user session data. In this phase, the cleaned data is broken into independent data sets according to the session ids. Each of these data sets represents a series of continuous visits by a user. Path completion determines if there is any important access missed in the log due to caching.

4.1 Data Cleaning

A running Web server gets many kinds of requests. Besides forthright requests from users by typing the URLs in the browser, and clicking on hyperlinks on Web pages and bookmarks, there are hidden requests generated by browsers for Web page embedded script files and image files, and requests from various computer programs, such as search engine robots, computer virus and worms. All requests are recorded in the Web log. The data cleaning step is first to eliminate requests that will not be used for analysis. Figure 4.1 shows the data cleaning algorithm used in the implementation.

4.1.1 Eliminate Irrelevant Requests from the Web Content Log

Each entry in a Web log contains an HTTP status code to record the outcome status of a client request. The most popular values are 200 and 404. The value 200 indicates the request has been processed successfully, and 404 means the request URL is not found. Because we are only interested in successful requests, any request with HTTP status code other than 200 is deleted.

A typical commercial Web site includes client script files and many image files in each HTML file served. Usually, it is necessary to eliminate irrelevant items in a Web log to avoid negative impact on the analysis result. Thus, any requests other than an HTML page request, such as image file, script file and style sheet file, embedded in that request

page should be deleted. These kind of requests are usually static file requests and can be easily found and deleted by checking the filename suffixes, such as "gif", "js" or "css", in the request URL field. In the Content Logging Plug-in solution, all static file requests are actually handled by the Web Server - the Apache HTTP Server. In this case, the Web application content log will not record any of these information. Because the page content always includes the URL for embedded images and scripts files, in case there are any interesting images or scripts, system administrators need to set the log config file to capture these attributes and meta-data.

Some amount of requests generated by automated agents or Web spider from search engines traversing links may occur. Because this kind of agents usually have distinctive "Agent" field in the HTTP header and usually is logged, this kind of traffic can be deleted very easily as well. In addition, a Web administrator can add a Robot Exclusion file with URL "`/robots.txt`" and accessible via HTTP to specify an access policy for robots or even exclude them [35].

4.1.2 Consolidate URLs

The final step in data cleaning phrase is to normalize the s and parse all the Common Gateway Interface (CGI) data [32]. There are probably several variations for a single in the Web log. First, most Web servers are configured to direct a request for a directory, a request's URL ending with a directory name or domain name for the root directory, to a default file, usually "index.html" or "default.html". The directory request may or may not end with a slash, depending on the Web server configuration. For example, "estore.com", "www.estore.com", "www.estore.com/", and "www.estore.com/index.html", etc. are all for the same file. Another issue is that CGI data is always appended to the URL. For example, "www.estore.com/products.jsp?sessionid=abcd12345&sku=123" indicates that the URL is "www.estore.com/products.jsp" with sessionid=abcd12345 and sku=123 as CGI data. This is a popular technique for passing request parameters for dynamic content and for identifying a client-server conversation for people not accepting cookie. The s in "Refer" field need to be normalized as well.

```
Database table URIMapping has two columns: id and uri

Method cleanLog ( ) {
1       urlId <- 0;
2       while((logEntry<-readLine(logFile)) != EOF ) {
3               // delete invalid requests
4               if (logEntry.statusCode != 200 ) {
5                       delete logEntry;
6                       continue;
7               }
8               // delete supplementary files
9               if ( logEntry.URI contains ".js" ||
10                  logEntry.URI contains ".css" ||
11                  logEntry.URI contains ".jpg" ||
12                  logEntry.URI contains ".gif" ||
13                  logEntry.URI contains ".png" ||
14                  logEntry.URI contains ".swf" ) {
15                      delete logEntry;
16                      continue;
17              }
18              // remove cgi data
19              if ( logEntry.URI contains '?' )
20                      logEntry.URI <- trim logEntry.URI from '?';
21              // complete URI
22              if ( logEntry.URI ends_with '/' )
23                      logEntry.URI <- logEntry.URI + "index.html"
24              // add URIs into URIMapping table
25              if (logEntry.URI does not exist in URIMapping) {
26                      insert into URIMapping (id, uri) values (urlId, logEntry.URI);
27                      urlId ++;
28              }
29      }
30
end;
```

Figure 4.1 Algorithm for Web Content Log Cleaning

Figure 4.2 on page 48 shows a sample Web log captured by the Content Log plug-in. The Web content log plug-in is designed to facilitate data preprocessing. Compared to the W3C's Extended Common Log Format, the new Web content log adds "Session ID", "Session attribute" and "Content" columns, and strips the CGI data in the request.

ID	REMOTEIP	REQTIMESTAMP	SESSIONID	REQURI	SESSIONATTR	UESTPARAMETE	REFERER	CONTENT
1	192.168.0.1	2002-01-14-00.53.05.204000	sukunriw81	/myshop/servlet/Browse		Sku=40000	http://192.168.0.1/myshop/servlet/Logort	<MouseOverIds/><Series ID=" 40000" ><Name>Rosemor
2	192.168.0.1	2002-01-14-00.53.05.856000	sukunriw81	/myshop/servlet/ShoppingCart			http://192.168.0.1/myshop/servlet/Browse?Sku=40000&	<MouseOverIds/><Comment>Empty Shopping Cart</C
3	192.168.0.1	2002-01-14-00.53.11.453000	sukunriw81	/myshop/servlet/Browse		Sku=40000-31	http://192.168.0.1/myshop/servlet/Browse?Sku=40000&	<MouseOverIds/><Product ID=" 40000-31" ><Name>Ros
4	192.168.0.1	2002-01-14-00.53.13.005000	sukunriw81	/myshop/servlet/Buy	cart={40000-31-210=1};		http://192.168.0.1/myshop/servlet/Browse?Sku=40000-31	<MouseOverIds/><Product ID=" 40000-31-210" ><Qty>1</
5	192.168.0.1	2002-01-14-00.54.01.628999	sukunriw81	/myshop/servlet/Browse	cart={40000-31-210=1};	Sku=40063	http://192.168.0.1/myshop/servlet/Update	<MouseOverIds/><Series ID=" 40063" ><Name>Carrera
6	192.168.0.1	2002-01-14-00.54.05.861999	sukunriw81	/myshop/servlet/Browse	cart={40000-31-210=1};	Sku=40063-35	http://192.168.0.1/myshop/servlet/Browse?Sku=40063&	<MouseOverIds/><Product ID=" 40063-35" > <Name>Car
7	192.168.0.11	2002-01-14-00.54.20.614000	g3x r0gi1e1	/myshop/servlet/Buy	cart={40000-3-110=1};		http://192.168.0.1/myshop/servlet/Browse?Sku=40000-35	<MouseOverIds/><Product ID=" 40000-3-110" ><Qty>1</C
8	192.168.0.11	2002-01-14-00.55.22.636000	g3x r0gi1e1	/myshop/servlet/Checkout	cart={40000-3-110=1};		http://192.168.0.1/myshop/servlet/Buy	<MouseOverIds/><Product ID=" 40000-3-110" > <Descrip
9	192.168.0.11	2002-01-14-00.55.24.318999	g3x r0gi1e1	/myshop/servlet/Login	cart={40000-3-110=1};		http://192.168.0.1/myshop/servlet/Update	<MouseOverIds/><Username/><Password/>
10	192.168.0.1	2002-01-14-00.56.17.572000	sukunriw81	/myshop/servlet/Checkout	cart={40000-31-210=1};		http://192.168.0.1/myshop/servlet/Buy	<MouseOverIds/><Product ID=" 40000-31-210" ><Descri
11	192.168.0.1	2002-01-14-00.56.19.544000	sukunriw81	/myshop/servlet/Login	cart={40000-31-210=1};		http://192.168.0.1/myshop/servlet/Update	<MouseOverIds/><Username/><Password/>
12	192.168.0.1	2002-01-14-00.56.24.522000	sukunriw81	/myshop/servlet/CheckLogin	cart={40000-31-210=1};		http://192.168.0.1/myshop/servlet/Bill	<MouseOverIds/><ValidUser>true</ValidUser><Username
13	192.168.0.1	2002-01-14-00.56.28.367000	sukunriw81	/myshop/servlet/ShoppingCart	cart={40000-31-210=1};		http://192.168.0.1/myshop/servlet/CheckLogin	<MouseOverIds/><User>400</User><Product ID=" 40000
14	192.168.0.1	2002-01-14-00.56.30.460000	sukunriw81	/myshop/servlet/Checkout	cart={40000-31-210=1};		http://192.168.0.1/myshop/servlet/ShoppingCart	<MouseOverIds/><Product ID=" 40000-31-210" ><Descri
15	192.168.0.1	2002-01-14-00.56.32.583000	sukunriw81	/myshop/servlet/Bill			http://192.168.0.1/myshop/servlet/Update	<MouseOverIds/><OrderNumber>00008</OrderNumbe
16	192.168.0.1	2002-01-14-00.57.01.263999	sukunriw81	/myshop/servlet/Checkout			http://192.168.0.1/myshop/servlet/ShoppingCart	<MouseOverIds/><Product ID="----"><Qty>0</Qty></Pro
17	192.168.0.1	2002-01-14-00.57.01.628999	sukunriw81	/myshop/servlet/Browse	cart={40000-31-210=1};	Sku=40063	http://192.168.0.1/myshop/servlet/Update	<MouseOverIds/><Series ID=" 40063" ><Name>Carrera
18	192.168.0.1	2002-01-14-00.57.05.861999	sukunriw81	/myshop/servlet/Browse	cart={40000-31-210=1};	Sku=40063-35	http://192.168.0.1/myshop/servlet/Browse?Sku=40063&	<MouseOverIds/><Product ID=" 40063-35" > <Name>Car
19	192.168.0.1	2002-01-14-00.58.08.134000	sukunriw81	/myshop/servlet/Logout			http://192.168.0.1/myshop/servlet/Browse?Sku=40063-35	<MouseOverIds/><Message>You have been successfully l
20	192.168.0.11	2002-01-14-00.59.34.413000	g3x r0gi1e1	/myshop/servlet/CheckLogin	cart={40022-3-110=1};		http://192.168.0.1/myshop/servlet/Bill	<MouseOverIds/><ValidUser>false</ValidUser>
21	192.168.0.11	2002-01-14-00.59.34.483000	g3x r0gi1e1	/myshop/servlet/CheckLogin	cart={40022-3-110=1};		http://192.168.0.1/myshop/servlet/Bill	<MouseOverIds/><ValidUser>false</ValidUser>
22	192.168.0.11	2002-01-14-00.59.45.488999	g3x r0gi1e1	/myshop/servlet/Browse	cart={40022-3-110=1};	Sku=40029	http://192.168.0.1/myshop/servlet/CheckLogin	<MouseOverIds/><Series ID=" 40000" ><Name>Rosemor
23	192.168.0.11	2002-01-14-00.59.47.723000	g3x r0gi1e1	/myshop/servlet/Browse	cart={40022-3-110=1};	Sku=40029-1	http://192.168.0.1/myshop/servlet/Browse?Sku=40000&	<MouseOverIds/><Product ID=" 40000-1" > <Name>Rose
24	192.168.0.11	2002-01-14-00.59.49.375000	g3x r0gi1e1	/myshop/servlet/Buy	cart={40022-1-110=1,40022-3-110=1};		http://192.168.0.1/myshop/servlet/Browse?Sku=40000-1	<MouseOverIds/><Product ID=" 40000-1-110" ><Qty>1</C
25	192.168.0.11	2002-01-14-00.59.52.469000	g3x r0gi1e1	/myshop/servlet/Checkout	cart={40022-1-110=1,40022-3-110=1};		http://192.168.0.1/myshop/servlet/Buy	<MouseOverIds/><Product ID=" 40000-1-110" > <Descrip
26	192.168.0.11	2002-01-14-00.59.54.082000	g3x r0gi1e1	/myshop/servlet/Login	cart={40022-1-110=1,40022-3-110=1};		http://192.168.0.1/myshop/servlet/Update	<MouseOverIds/><Username/><Password/>
27	192.168.0.11	2002-01-14-01.00.02.333999	g3x r0gi1e1	/myshop/servlet/CheckLogin	cart={40022-1-110=1,40022-3-110=1};		http://192.168.0.1/myshop/servlet/Bill	<MouseOverIds/><ValidUser>true</ValidUser><Username

Figure 4.2 Sample Usage Log Data

Based on the log database, most tasks can be skipped. Only a URL normalization program is required in this solution. To make data mining algorithms work more efficient, the URL normalization program builds a URL index table, as shown in Figure 4.3. Each URL is mapped to a unique number, and the URL string in "Request" and "Refer" field are replaced by the corresponding number. The CGI data in "Refer" field only contains data sent via HTTP GET method while missing data sent by POST or other methods. Due to its incompleteness, the "Refer" field is not useful and thus gets truncated.

URL_ID	Request
1	/myshop/servlet/Browse
2	/myshop/servlet/ShoppingCart
3	/myshop/servlet/Buy
4	/myshop/servlet/Register
5	/myshop/servlet/Checkout
6	/myshop/servlet/Login
7	/myshop/servlet/Logout

Figure 4.3 URL Index Table

4.2 Building User Session Database

A user session is defined as a cohesive set of all of the user requests from one user across one of more Web pages during a specific time period [36, 37]. A user session identification process segments the cleaned Web log into user visit session sets. Ideally, each visit session set in the output only contains a complete series of requests in time sequence from a single visitor in a browsing or shopping visit. User session identification is one of the most important tasks in the Web Mining procedure because the data mining algorithms used in pattern discovery phase work on these data sets to discover patterns among these sets that users may be interested. Errors may lead the data mining algorithms to give biased report or even fail.

4.2.1 Logging Session Related Information

The Web content log generated by the Web Content Logging Plug-in benefits the User Session Identification process by providing system generated session id and all the attributes associated with that session id. Traditionally people use server side log data in Extended Log Format (ECLF) for Web usage data mining. However, it is usually unreliable and inefficient for data mining purposes. There are two major reasons: hard to identify unique user sessions, and incomplete request data.

First, a user session cannot be identified merely by the IP address alone because there is a many-to-many relationship between users and IP addresses. Users may share IP addresses by proxies or Network Address Translation (NAT) devices, etc. In order to identify each unique user, client-side data acquisition is proposed in [33]. These solutions push a small client monitor program to the client side, which is used to record and upload user's activities to the data acquisition server. Such programs are more efficient and provide accurate data about users' activities. This may even help to eliminate data processing phase for data mining. But the obvious drawback is that users are very reluctant to download anything to monitor theirs behaviors.

Secondly, all POST data are missed in the log file. Some proposed solutions thus have to remove all the requests with POST data during data preprocessing [32]. The HTTP protocol is based on a request/response paradigm. HTTP protocol defines an open-ended set of methods to be used to indicate the purpose of a request. A client establishes a connection with a server by sending a request to the server in the form of the identifier of the resource - the Uniform Resource Identifier (), the method to be applied to the resource, and protocol version followed by message Type (usually MIME type), client type and body content. HTTP protocol is stateless which means after the client finishes receiving response data from the server, the connection closes immediately. The HTTP server never keeps any information about the clients. So the client-server conversation status data and data inquiry parameters will be send to the server each time for each request from the client. The method to be applied to the request resource is indicated in

the Method token. For e-commerce Web site, almost of all the HTTP requests use either GET or POST method. Client request data using the GET method is appended onto the end of the action being requested. The POST method requests the destination server to accept the data entity enclosed in the request as a new subordinate of the resource identified by the request's . POST is designed to allow a uniform method to provide a block of data, such as data filled in a form, to a data-handling process, extending a database through an append operation. According to the HTTP specification, applications must not cache responses to a POST request because the application has no way of knowing that the server would return an equivalent response on some future request [32, 33]. POST data is an important part in e-commerce transactions. Almost all the user information, purchase orders, and other inquiries use POST method. Ignoring POST requests in the Web log tends to generate biased analysis reports of the Web traffic.

The Web Content Logging Plug-in is an extra module for common Web application server. It uses some basic functions of the Web application server to enhance the Web logging. These functions include session management, and user request data parsing. The log mining preprocessing is benefited from:

First, instead of identifying each user by his / her IP address, the Web content log identifies a user by a session ID. Assigning a session ID is very popular approach for uniquely identifying a visitor in e-commerce Web sites. A session ID is a random unique long string, and usually consists of at least 32 characters. The Web application assigns a session ID to the client on first visit if he does not provide a valid one in the request. To make the task of developing a securer and reliable Web application easier, Web application servers nowadays provide built-in user session management function, such as assigning and expiring session ID automatically. Applications developers can easily associate a session ID with various attributes, such as shopping cart, or to record a client's current status. The Web application server maintains the session ID in a hash table. Each session ID has a pointer to the memory of its associated attributes. Web application server expires inactive sessions regularly in order to use memory efficiently. The default session expiry period is 30 minutes, based on result by Catledhe [38], for the

optimum timeout for a session. Because the server expires old session IDs being idle for a predefined period and reassigns a new session ID even when the same client comes back, the splitting timeout sessions in traditional Web log mining can be skipped.

Another benefit of the Web Content Logging Plus-in module is to make Web application developers' work easier. Most Web application servers parse and decode requests data, and save "GET" and "POST" data into objects (Object HttpServletRequest in Java-based Web application server) before the actual Web application program being called to process the request. The object exists during the life cycle of the request, till all the HTTP connections are closed. The request data decoding and parsing function from Web application server also makes logging POST data easy in the Web content log plug-in with few simple method calls. To avoid logging POST data in a big size and other data not interested in, the Web content log plug-in reads the log configuration file to determine what attributes to log. Because the number of attributes is usually very limited, configuring what attributes need to be logged is generally an easy task.

4.2.2 Building Session Database

Based on the raw Web content log database, a session database table (as shown in Figure 4.4) is built for data mining phase. The algorithm is shown in Figure 4.5 and 4.6. There are four major steps: deleting duplicate requests, completing session visit path, building session URL table, and building session attribute table.

Users may send duplicate requests by double click, or refresh page for various reasons, such as slow response, etc. These duplicate entries are eliminated from the session. Any request having same URL, request data and referrer with its previous request in the session will be deleted.

SESSIONID	RAW_ID	REQTIMESTAMP	REQURI_ID
svkuhriw81	1	2002-01-14-00.53.05.204000	1
svkuhriw81	2	2002-01-14-00.53.06.856000	2
svkuhriw81	3	2002-01-14-00.53.11.453000	1
svkuhriw81	4	2002-01-14-00.53.13.005000	3
svkuhriw81	5	2002-01-14-00.54.04.628999	1
svkuhriw81	6	2002-01-14-00.54.06.861999	1
g3xr0gj1e1	7	2002-01-14-00.54.20.614000	3
g3xr0gj1e1	8	2002-01-14-00.55.22.636000	4
g3xr0gj1e1	9	2002-01-14-00.55.24.318999	5

Figure 4.4 Session Table

Web content caching is widely used since the early days of the Internet. The main purpose for content caching is to reduce the response time, especially for slow connections, and traffic volume. Web pages, images, and other files with the same URLs are always cached in proxy servers that are closer to the client. So when a client requests a page, the server may not even get the request. Popular pages are most likely to get cached. There is no provably correct algorithm for determining what cached pages were viewed, and the only verifiable way to track client activities and avoid being cached is to use either a software agent or modified browser in the client side [32, 33]. However, with some techniques, dynamic Web content can avoid being cached effectively by proxies or local browsers.

Some of the techniques are:

- Using
  <META HTTP-EQUIV="Pragma" CONTENT="no-cache">
  <META HTTP-EQUIV="Expires" CONTENT="-1">
  tags inside the <HEAD> tag of the HTML page tells W3C HTTP compliant Web browsers not to cache the page content.
- Any secure information delivered by Secured HTTP will not be cached by any proxies or Web browsers.
- Using URL rewriting to encode request data in the URL. So the URL will look different for each request, and thus will not be cached.

In order to provide the best quality data for data mining algorithms, all the sessions have their navigation path verified. The path completion function checks if the current request is following the link in the last page. A navigation path stack is used. This function pushes each verified request in a session into the stack in time sequence. If a request's "Referer" field in the HTTP header does not match the URL of previous request, this function keeps popping up requests in the stack and inserting the top request into the navigation path. This function quits when one of the following conditions occurs:

- The "Referer" equals to the URL of the request on the top of the stack.
- The stack gets empty, or the number of popped requests exceeds a predefined BAD_SESSION threshold. In this case the whole session is considered "bad" and gets abandoned.
- Finishes verifying the last request in a session.

All related session attributes and logged content are saved in a request attribute table as shown in Figure 4.7. All these information attached to a URL is saved with the hierarchy. In the data mining phase, business analysts can give each attribute a weight and adjust them in order to find interested patterns for different topics. Figure 4.8 shows the database schema of Web content log data.

Let Vector $S_i = \{s_1, s_2, .., s_n\}$ denote a time ordered session history as the output of this algorithm.
And s_i represents a visit record in the session. It acts as the foreign key to the *RAWID* in Web Content Log.
Let r denote the current record read from the raw Web content log.
Let *PathStack* denotes a stack.
Let *NextPath* denotes a Vector

Let *SID* $\{sid_1, sid_2, ..., sid_n\}$ denote all the distinctive SessionIDs in the original Web Content Log.
Let O_{sid} $\{o_1, o_2, ..., o_n\}$ denote all the records with SessionID=sid_i in the original Web Content Log, ordered by *RequestTimeStamp*.

```
1. For each sid_i do {
2.     for each o_x in O_sid do {
3.         if (i==1) then {
4.             CreateNewSession(o_1);
5.             continue;
6.         }
7.         if DuplicateRequest (o_x) then
8.             continue;
9.         else {
10.            nextPath <- CompletePath (o_x);
11.            if (nextPath != NULL) then
12.                Append nextPath into S_i;
13.            else {
14.                remove S_i;
15.                break;
16.            }
17.        }
18.        next;
19.    }
20.    next;
21. }
end;
```

```
Method CreateNewSession(r) {
1.    New Session S_i with initial value(r);
2.    Reset PathStack;
3.    Push r into PathStack;
4.    return;
5. }
end;
```

Figure 4.5 Algorithm for Building Session Database (1)

```
Method DuplicateRequest (r) {
1.      last <- Pop PathStack;
2.      Push last into PathStack;
3.      if (r.sessionID == last.sessionID)
                AND (r.reqestURI == last.requestURI)
                AND (r.requstParams == last.requestParams)
                AND ((r.requestTimeStamp - last.requestTimeStamp) < T_MaxDuplicate)
4.      then
5.              return true;
6.      else
7.              return false;
8. }
end;


Method CompletePath (r) {
1.      New PathVector;
2.      top <- Pop PathStack;
3.      if ( r.Referer == top.requestURI ) then {
4.              Push top into PathStack;
5.              Push r into PathStack;
6.              Append r to PathVector;
7.      else {
8.              while ( PathStack != NULL AND r.Referer != (top <- Pop PathStack).requestURI ) {
9.                      Append top to PathVector;
10.             }
11.             if (PathStack == NULL) then
12.                     PathStack <- NULL;
13.             else {
14.                     Push top into PathStack;
15.                     Append top to PathVector;
16.                     Push r into PathStack;
17.                     Append r to PathVector;
18.             }
19.     }
20.     return PathVector;
21. }
end;
```

Figure 4.6 Algorithm for Building Session Database (2)

Session svkuhriw81 :

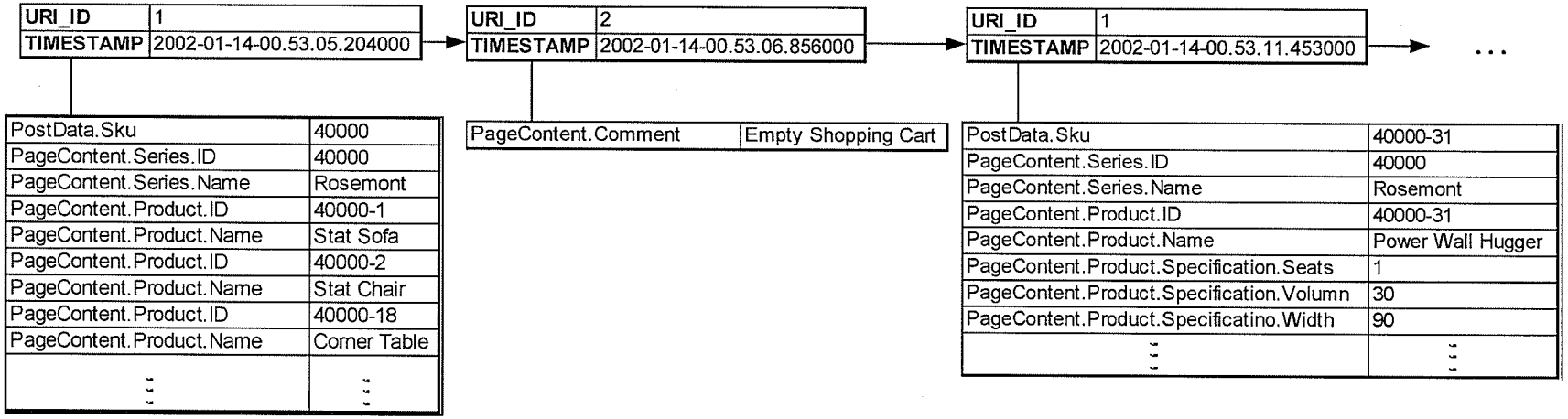

Figure 4.7 A Session Example

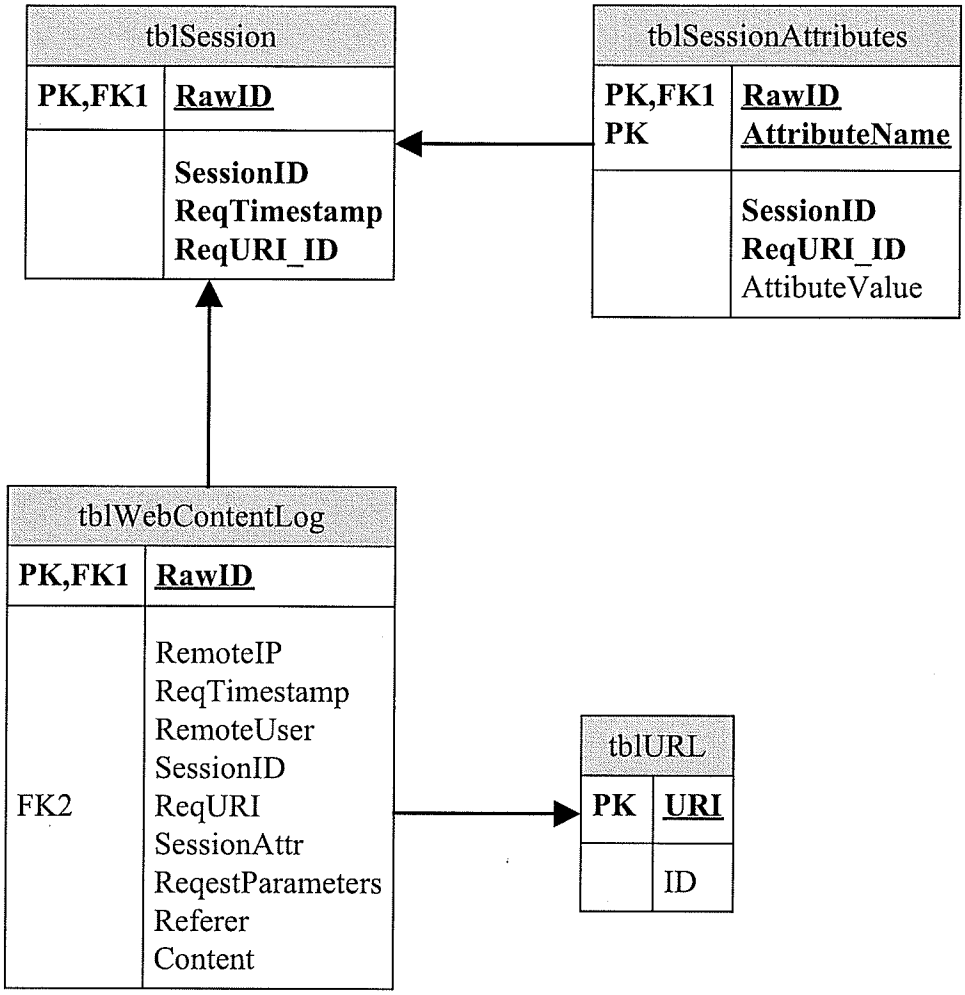

Figure 4.8 Session Database Entity-Relationship Diagram

Chapter 5

Web Content Usage Data Mining

The Web Data Mining process is to analyze large volumes of prepared Web log data in order to help organizations understand customers' behaviors, and effectiveness of current Web site structure and promotional campaigns.

5.1 Fundamental Statistics

Basic statistical analysis provides periodic reports on the overall Web site traffic and running status. Currently, there are many Web traffic statistics report tools available on the market. All of them provide similar content. One of the leading products, Webtrends [57], provides reports on following topics,

- General statistical reports and summary of activities for report period, which includes statistical results of various activities from the Web site during a designated time frame, such as total hits, total page views and unique visitors (based on unique IPs), average number of visitor, most active time, etc.
- Most requested pages.
- Technical Statistics and Analysis, which includes the total number of hits for the site, how many were successful, how many failed, and calculates the percentage of hits that failed. These reports are useful for the administrators to determine the server load and the reliability of the site, etc.
- Most popular Web browsers and operating systems used by the visitors to the site.

These reports generally give the Web administrators enough Web traffic information to maintain the server. However, these reports provide very limited information for business

purposes. Furthermore, these reports have no in-depth information about dynamic URLs. The Web Store prototype uses a single Java Servlet with URL "*http://192.168.0.10/mystore/servlet/Browse*" to handle all requests for browsing product series, series items, and detail information for each product item. Unlike the traditional Web traffic reports which generally have only statistics results for the URL "*http://192.168.0.10/mystore/servlet/Browse*", the object logging solution is able to give more details about how the Java Servlet is used to generate dynamic pages.

The Web store is developed using an OO approach, so each dynamic Web page is constructed using various objects with hierarchy. Figure 5.2 gives an example of a product item page with corresponding objects' structure shown in Figure 3.5 on page 37.

Once interesting objects' attributes are captured, statistical programs can generate various statistical reports of these attributes based on logged data. XML-enabled database simplifies the data query procedure by extracting XML elements and attributes into traditional SQL data types. The prototype implementation uses IBM DB2 with XML Extender installed.

The following sample implementation illustrates how to build attribute statistics reports. This implementation gives some statistical results of a user session attribute "cart". The Web store application assigns a Shopping Cart object to each user session. This shopping cart object is maintained in the Web server as a session attribute. Each time a customer adds a product / item into the shopping cart, the Web store application records that item number and quantity in the customer's shopping cart. The customer can delete an item, change an item and its quantity in the cart update page. The shopping cart information is recorded in XML format, a sample is given below in Figure 5.1:

```
<cart>
  <product id="40000-11-150">1</product>
  <product id="40000-42-250">3</product>
  <product id="40000-25-110">1</product>
</cart>
```

Figure 5.1 A Sample Shopping Cart XML Formatted Data

Figure 5.2 Detail Product Information Page

Studying the users' shopping carts helps in the better understanding of user purchase behaviors, such as what is the most purchased items overall, and what is the most purchased items for a given URL, etc.

IBM DB2 with XML Extender provides Xcolumn, a XML enabled column, for saving XML data and manipulating XML data using built-in system tools. To ensure the data is in good format, the system validates the data against a preloaded XML Document Type Definition (DTD) file before it writes the data into the database. In Figure 5.3, Line 20 in the DTD file states that the XML data need to have a cart element which may contain zero or more instances of product element. Line 30 indicates that the product element must have an id attribute with data type character data/string. Line 40 indicates the product element contains parsed character data (#PCDATA), i.e., data that has been checked to ensure that it contains no unrecognized markup strings. This data is used as the quantity of the product item in this implementation.

```
10 <?xml version="1.0" encoding="UTF-8"?>
20 <!ELEMENT cart (product*)>
30 <!ATTLIST product id CDATA #REQUIRED>
40 <!ELEMENT product (#PCDATA)>
```

Figure 5.3 DTD of Shopping Cart XML Data

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "e:\dxx\dtd\dad.dtd">
<DAD>
  <dtdid>cart.dtd</dtdid>
  <validation>YES</validation>
  <Xcolumn>
    <table name="cart_sku">
      <column name="sku" type="varchar(20)" path="/cart/product/@id" multi_occurrence="YES"/>
    </table>
    <table name="cart_qty">
      <column name="qty" type="integer" path="/cart/product" multi_occurrence="YES"/>
    </table>
  </Xcolumn>
</DAD>
```

Figure 5.4 DAD for Shopping Cart XML Data

DB2 database saves validated data into Xcolumn and builds indexes for the XML elements and attributes based on Document Access Definition (DAD) file. The DAD file is an XML document specifies how saved Xcolumn data are to be accessed. A DAD file uses XML Path Language (XPath) to specify an XML element or attribute location inside an XML document. XPath is a language for addressing parts of an XML document. Every location path can be expressed using the syntax defined for XPath. DB2 XML Extender provides call procedures for querying and updating XML elements and attributes via XPath. A query statement, such as:

SELECT *

FROM *xsessionattr a,*

table(db2xml.extractIntegers(a.Attributes, '/cart/product/@id')) as x

will return all the product ID in attribute Xcolomn in xsessionattr table. When using a database system without XML support, developers usually need to write programs for retrieving XML data, parsing XML data, and then querying the parsed data, etc. For other non-XML implementations on relational database systems keeping the data hierarchy structure may be very complex.

Based on the DAD file shown in Figure 5.4 on page 62, DB2 creates two side tables, CART_SKU and CART_QTY, to help index the XML elements and attributes in the Xcolumn. Figure 5.5 shows the relationship of the tables.

A database view based on table XsessionAttr, CART_SKU and CART_QTY is created to simplify the data queries for product id (sku) and qty, Figure 5.6 shows its columns with sample data. The DB2 XML Extender adds the DXXROOT_ID column to the XSessionAttr table, and the CART_SKU and CART_QTY side tables. The DXXROOT_ID column acts as the foreign key of the XSessionAttr table to associate its two side tables.

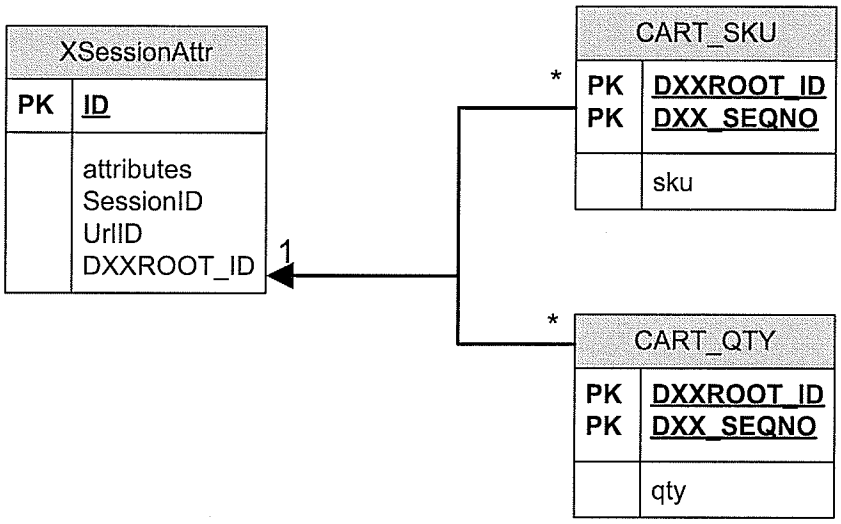

Figure 5.5 Relationship Diagram of XML-based Session Attribute Table (XsessionAttr) and its Side Tables (CART_SKU and CART_QTY)

I820SRV - DB2 - DATAPROC - ADMINISTRATOR.SESSVIEW

DXXROOT_ID	URLID	SESSIONID	DXX_SEQNO	SKU	QTY
8	12	45yi208dx1	1	40063-61-...	1
9	11	45yi208dx1	1	40063-61-...	1
10	11	45yi208dx1	1	40063-61-...	1
11	17	45yi208dx1	1	40063-61-...	1
12	11	45yi208dx1	1	40063-61-...	1
13	11	45yi208dx1	1	40063-61-...	1
14	11	45yi208dx1	1	40063-61-...	1
15	12	45yi208dx1	1	40064-31-...	2
15	12	45yi208dx1	2	40063-61-...	1
16	11	45yi208dx1	1	40064-31-...	2
16	11	45yi208dx1	2	40063-61-...	1
17	11	45yi208dx1	1	40064-31-...	2
17	11	45yi208dx1	2	40063-61-...	1
18	18	45yi208dx1	1	40063-61-...	1
18	18	45yi208dx1	2	40065-63-...	2
19	14	45yi208dx1	1	40063-61-...	1
19	14	45yi208dx1	2	40065-63-...	2
20	15	45yi208dx1	1	40063-61-...	1
20	15	45yi208dx1	2	40065-63-...	2
21	13	45yi208dx1	1	40063-61-...	1
21	13	45yi208dx1	2	40065-63-...	2

Figure 5.6 A Database View based on table XsessionAttr, CART_SKU and CART_QTY

Building a similar view for page content data can help in creating statistical reports on overall object usage (which includes the most requested products, the most popular attributes and their values, the most popular methods, etc.). The query and report can be very complex. It depends on what analysis is of interest.

As shown above, statistical reports on objects provide business analysts a better overview of the business running on the Web site than most current Web traffic statistics reports. Figure 5.7 shows a 3D product hits chart for each URL based on the logged session cart data.

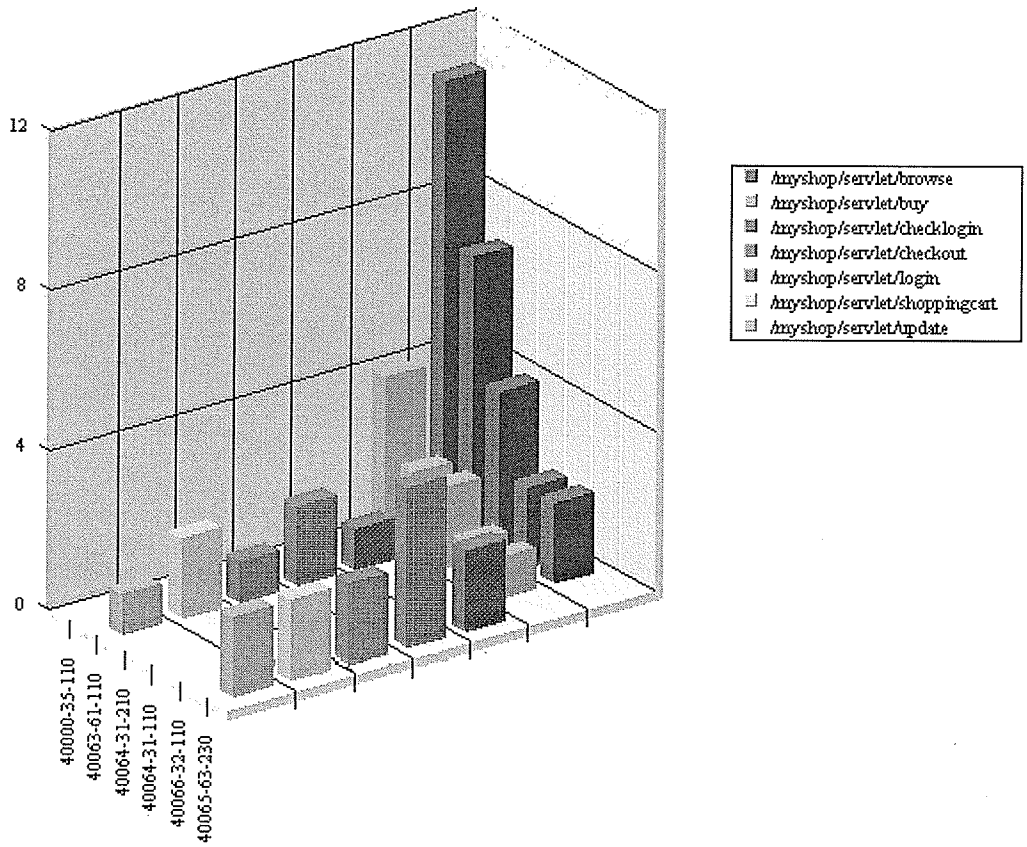

Figure 5.7 A Sample Web Object Usage 3D Chart

5.2 Data Mining

Statistics is mainly focused on analyzing numerical data. It usually provides straightforward answers via intensive analysis of small size and clean numeric data. Data mining is a new discipline lying at the interface of statistics, database technology, pattern recognition, machine learning, and other areas [75]. The essence of data mining is an attempt to discover unexpected patterns composed of conjunctive and disjunctive combinations of elements which are of interest or value to the data owners in an efficient and timely manner [74, 75].

The data mining process is iterative and interactive; any step may lead to a refinement of the previous steps. User feedback plays a critical role in the success of data mining in all stages, starting from the initial description of the data, the identification of the potentially relevant features and the training set (where necessary), and the validation of the results. Data mining is usually a complex procedure, which includes selecting proper data mining algorithms for various analysis purposes, and modeling input data, developing user interactive interface and data mining result presentation for each selected algorithm. The purposes of this thesis are to simplify the data modeling procedure, make it more efficient and provide more flexibility in choosing interesting data attributes for analyzing e-commerce Web traffic. Once a clean user session database is built, some data mining technologies can be introduced to discover popular patterns. The choice of techniques and algorithms depends on the discovery goals. Visitor grouping and path analysis are among the most interesting subjects for e-commerce Web sites. Visitor grouping uses data clustering algorithms to group visitors/sessions with similar selected attributes' value. Path analysis usually refers to mining common visiting path sequence.

The discovered patterns from data mining algorithms are also considered as hypotheses for determining if an instance belongs to one of these patterns in the future. For most data mining algorithms, usually a randomly selected subset of data, called training data, is used as input to generate hypotheses [80]. To estimate the quality of generated hypotheses, another subset of data, called test data, is used to test the correctness of these

hypotheses [80]. Each instance in the test data set gets predicted by the hypotheses and is compared to the correct one. The ratio of the correct predicted instances to the number of instances in the test data set is called accuracy.

5.2.1 Data Clustering

Clustering techniques are used to divide instances into natural groups so the instances' intra-group relationship is maximized while the inter-group relationship is minimized [72, 73, 74]. Traditional distance-based clustering algorithms [80] and probability-based clustering algorithms [80] are widely used. The heuristic distance-based clustering algorithms [80], such as k-means, usually are used in numeric domains and are hard to prevent overfitting, because no finite amount of evidence is enough to make a completely firm decision on what a given instance should be placed categorically in one cluster [80]. The probability-based clustering algorithms find the most likely set of clusters from a probabilistic perspective. A typical probability-based clustering algorithm, the Expectation-Maximization (EM) algorithm [76, 77], can select the best number of clusters for the data and calculate the cluster probabilities (which are the "expected" class values) to maximize the likelihood of the data distributions.

5.2.1.1 The Basic EM Algorithm

The EM algorithm [77] assumes all the information obtained from the observation of an event A has probability $p(A)$ with some kind of underlying probability distributions. The EM algorithm provides a generic approach in finding the maximum-likelihood estimate of normal distribution parameters from a given data set. The likelihood estimate is usually represented by probability density [76]. According to Central Limit Theorem, Gaussian Mixture Model (also known as Normal Distribution Model) is suitable as an underlying data distribution model to model the probability density of the data in most real life scenarios. The Gaussian Mixture Model has been proposed as a general model for estimating an unknown probability density function [78].

The EM algorithm defines the likelihood function as:

$$L(\theta \mid X) = p(X \mid \theta) = \prod_{i=1}^{n} p(x_i \mid \theta) \qquad (5.1)$$

where θ is the set of distribution parameters (mean μ and variance σ^2 for normal distribution), and X is the observation set, $X = \{x_1, \ldots x_n\}$, x_i is an observation instance. The likelihood $L(\theta \mid X)$ is thought of as a function of the parameter θ where the data X is fixed. The goal of the EM algorithm is to find the set of parameters θ^* to maximize the likelihood $L(\theta \mid X)$. Equation (5.2) can be derived from equation (5.1). Equation (5.2) is often used in implementation because of its simplicity of computation comparing to (5.1).

$$\log L(\theta \mid X) = \sum_{i=1}^{n} \log p(x_i \mid \theta) \qquad (5.2)$$

The likelihood $L(\theta \mid X)$ is considered to be maximized when its value converges over EM iterations, i.e. $L(\theta^p \mid X) = L(\theta^{p+1} \mid X) = L(\theta^* \mid X)$. Thus $d(\log L(\theta|X)) = 0$.

Gaussian distribution has density function $f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$ with mean μ, and variance σ^2 as parameters. Each is calculated using following formulas, respectively:

$$\mu = \frac{x_1 + x_2 + \ldots x_n}{n}$$

$$\sigma^2 = \frac{(x_1 - \mu)^2 + (x_2 - \mu)^2 + \ldots + (x_n - \mu)^2}{n}$$

The conditional probability of a cluster C_i with given instance x is

$$\Pr[C_i|x] = \frac{\Pr[x|C_i] \bullet \Pr[C_i]}{\Pr[x]} = \frac{f(x; \mu_{C_i}, \sigma_{C_i}) p_{C_1}}{\Pr[x]}$$

where $f(x; \mu_{Ci}, \sigma_{Ci})$ is the normal distribution for cluster C_i, $\Pr[x]$ is the overall probability of instance x, $\Pr[C_i] = p_{ci}$ is the overall probability of cluster C_i, and $\Pr[x|C_i]$ is the conditional probability of a instance x comes from a given cluster C_i.

The EM algorithm has two major steps. The first step, the Expectation step (E step), finds the probability p_i for each expected cluster C_i. And the second step, the Maximization step (M step), finds the best values for each cluster's distribution parameters θ^* to maximize the likelihood $L(\theta^*|X)$ of the distributions for the given data.

5.2.1.2 Finding Popular Visiting/Purchasing Patterns via EM Algorithm

Data clustering programs can build clusters based on the navigation URLs from traditional Web log. With more detail page content information logged by the Web content logging plug-in, data clustering algorithms are able to provide in-depth analysis. In real world scenario, the clustering approaches may vary in algorithms and data schemas, all depend on business requirements.

The implementation in this thesis demonstrates building clusters of the values of logged attributes using the EM data clustering algorithm. The Web content logging plug-in is implemented to log interesting attributes, such as product series number, end items number, and total value of the order in a session, in order to analyze interesting products and visitors' behaviors. The analysis results from the EM algorithm can help answer questions, such as what products lead to better sales, which visitors interested in a product are also interested in what kind of other products, etc, for target marketing.

In the Web content log, each instance represents a user session. The products id is considered as a nominal attribute of the input data of the EM algorithm, with appearance count as value, indicating if the corresponding product is shown in a user session. Assume these attributes $\{a_1, a_2, ..., a_m\}$ are independent to each other, and each attribute a_i has value set $\{v_{i1}, v_{i2}, ..., v_{il}\}$. Then the joint probability of an instance

$$P(X = \{v_{1x}, v_{2y}, ..., v_{mz}\}) = P(a_1=v_{1x}) \bullet P(a_2=v_{2y}) \bullet ... \bullet P(a_m=v_{mz}) \qquad (5.3)$$

An instance $X = \{v_{1x}, v_{2y}, ..., v_{mz}\}$ has probability p_i belonging to cluster c_i, where

$\sum_{i=1}^{n} p_i = 1$ and all clusters $C = \{c_1, c_2, \ldots c_n\}$.

Implementing the Web Content Log clustering program includes four major steps: data initialization, detecting the best number of clusters to represent the data, EM's E step to find the probability p_i for each expected cluster C_i, and EM's M step to find the best values for each cluster's distribution parameters θ^* to maximize the likelihood $L(\theta^*|X)$ of the distributions for the given data.

5.2.1.2.1 Data Initialization

The preprocessed Web content log contains clean session attributes. To find interesting attribute patterns, the cleaned session database must be preprocessed to a suitable input data for the EM algorithm. The data preprocessing module first scans the session database for attributes and their values. Because the actual number of attributes may be very large, the user can set a threshold of the observing frequency to limit the attributes to be further analyzed.

The preprocessing module scans the session database finding every unique attribute and counting the frequency it shown in each session, and the total frequency across all the sessions. These two frequencies are used to build the underlying Poisson distribution of each attribute. Poisson distribution is widely used to model the number of events occurring within a given time interval for various phenomena of discrete nature [88]. For nominal attributes with discrete values, the probability of each attribute value is the count of that attribute value divided by the sum of the count of all values. Thus,

$$P[x_i] = \frac{count(x_i)}{\sum_i count(x_i)} \qquad (5.4)$$

where $P[x_i]$ denotes the possibility of an attribute having value x_i, $count(x_i)$ denotes the total number of an attribute's value x_i appeared, $\Sigma\, count(x_i)$ denotes the sum of the total number of the attribute's each value x_i appeared.

Numeric attributes, such as the total amount a customer purchased, are assumed to have Gaussian distribution [88] as underlying distribution model. They have continuous values. The parameters of the underlying Gaussian distribution, the mean μ and variance σ^2, are calculated using following formulas.

$$\mu = \frac{w_1x_1 + w_2x_2 + \ldots + w_nx_n}{w_1 + w_2 + \ldots + w_n} \qquad (5.5)$$

$$\sigma^2 = \frac{w_1(x_1-\mu)^2 + w_2(x_2-\mu)^2 + \ldots + w_n(x_n-\mu)^2}{w_1 + w_2 + \ldots + w_n} \qquad (5.6)$$

$$\text{So } P(x_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x_i-\mu)^2}{2\sigma^2}} \qquad (5.7)$$

where x_i denotes a value of the attribute, and w_i denotes the possibility of the attribute having value x_i, and $P(x_i)$ denotes the possibility of the attribute having value x_i.

5.2.1.2.2 Finding the Best Number of Clusters to Represent the Observed Data

Even for experienced users, it is hard to tell how many clusters best represent the observed data. This implementation uses the likelihood to evaluate the "goodness" of the number of clusters. Starting with a cluster, this module calculates the likelihood of each cluster and compares the mean of the likelihood of all clusters. It continues to increase the number of clusters for the observed data until the likelihood decreases. So the number of clusters just before the last iteration is considered as the best number for the number of clusters used by EM algorithm. The algorithm is shown in Table 5.1.

To improve the "goodness" of the likelihood for the clusters generated by the generic EM algorithm, a ten-fold cross validation is used. The ten-fold cross validation randomly splits all instances in the observation data set into 10 subsets $\{X_1, X_2, \ldots X_{10}\}$ in approximately equal size. Nine of them are randomly chosen as training data $\{Y_1, Y_2, \ldots Y_9\}$ and the remaining one T is used as test data. The EM algorithm takes each training data set Y_i $(1 \le i \le 9)$ to give a pair of estimated parameters $\theta_i(\mu_i, \sigma_i)$. For each pair of estimated parameters θ_i, this module uses the test data T to calculate the corresponding likelihood $L(\theta_i \mid T)$. Finally, the mean of all nine likelihoods for a given number of clusters $L_i = \frac{\sum_{m=1}^{9} L(\theta_m \mid T)}{9}$ will be used to compare the previous mean of likelihood L_{i-1} to test if the best number of clusters C for the observation data has been reached.

During the implementation test, the actual logarithm likelihood L sometimes converges very slowly after a certain point, the running time of this module then tends to be very long. For early quit from the iteration to improve the efficiency of the program after the likelihood gets stable, this module allows users to assign a number for the maximum number of clusters (thus the maximum number of iterations) or assign the minimal difference between the current logarithm likelihood and the last one.

```
/*************************************************
 * AllInstanceSet is a set contains all instances.
 * NUM_CLUSTER_LIMIT is the max. number of clusters allowed.
 * MAX_DOUBLE_VALUE is the max double value.
 * function RandomShuffle (InstanceSet) randomly shuffle instances in a give instance
 * set.
 * function RandomSplit (InstanceSet) randomly split a big instance set into ten subsets
 * in approximately equal size.
 * function E() is the E step in EM (see Table 5.3)
 * function iterateEM(InstanceSet, NumClusters) is shown in Table 5.2.
*************************************************/
int function EstClusterNum() {
1     boolean LkDecreased ← false;
2     numFolds ← 10;
3     num_cluster ← 1;
4     NUM_CLUSTER_LIMIT ← 10;
5     curr_likelihood ← -(MAX_DOUBLE_VALUE);
6     while (LkDecreased) {
7           LkDecreased ← false;
8           RandomShuffle(AllInstanceSet);
9           subset[10] ← RandomSplit(AllInstanceSet);
10          testSetIndex ← Random (seed);
11          testSet ← subset [testSetIndex];
12          sum_likelihood ← 0.0;
13          for (i ← 0; i < numFolds; i++) {
14                if ( i = = testSetIndex )
15                      continue;
16                trainSet ← subset[i];
17                curr_likelihood ← iterateEM(trainset, num_cluster);
18                fold_likelihood ← E(testset, num_cluster);
19                sum_likelihood ← sum_likelihood + fold_likelihood;
20          }
21          mean_likelihood ← sum_likelihood / numFolds;
22          if ( mean_likelihood > curr_likelihood &&
               num_cluster < NUM_CLUSTER_LIMIT ) {
23                      curr_likelihood ← mean_likelihood;
24                      LkDecreased ← true;
25                      num_cluseter++;
26          }
27    }
28    return num_cluster - 1;
29 }
end;
```

Table 5.1 Estimate the Number of Clusters

5.2.1.2.3 The EM Iteration

The EM iteration module is the core of the whole EM cluster mining program. It iterates the E and M steps until the log likelihood of the data converges.

1. Because each observed instance has a certain amount of membership for each class, each instance correspondingly has different weighted contribution to the statistics of every cluster [79]. This implementation uses a weighted instance approach to find the parameters to maximize the likelihood. The EM iteration first randomly guesses the weight w_i for each attribute of each instance for calculating the initial distribution models.
2. Then the EM iteration module utilizes Gaussian distributions (see formula (5.5), (5.6) and (5.7) above) in the M step with observed instance data to estimate parameters $\theta_i(\mu_i, \sigma_i)$ for each attribute x_i in each cluster c_i for maximizing the overall likelihood [p222, 80] . The algorithm is shown in Table 5.2.
3. The E step, shown in Table 5.3, calculates the probability of each instance x_i belonging to cluster c_i using the estimated parameters θ_i^t from the M step. So the logarithm of overall likelihood $\log(likelihood) = \sum_m \sum_n \log p(x_m \mid c_n)$.
4. The iteration starts step 2 again for next estimate θ_i^{t+1} by the current θ_i^t until reaching the maximum overall likelihood. Theoretically, the maximum value of likelihood is 0. In practical implementation, the increase of log-likelihood tends to be negligible after very sharp change over the first few iterations, as shown in Figure 5.8. To run the program more efficiently, a maximum number of iterations and a minimal difference of the log-likelihoods between iterations are defined for earlier termination of the iteration whenever one of the conditions is reached. In this implementation, the program iterates until the difference between two successive values of log-likelihood is less than 10^{-4}. A maximum iteration number is also set to prevent endless loop when the likelihood does not converge.

```
/*************************************
 * random_guess() generates a random number between 0.0 and 1.0;
 * instance is a member in InstanceSet;
 * function M is shown in Table 5.4;
 * function E is shown in Table 5.3
*************************************/

function iterateEM(InstanceSet, num_cluster) {
1       prev_likelihood ← 0.0;
2       curr_likelihood ← 0.0;
3       weight[instance][cluster] <- random_guess(seed);
4       for (i ← 0; i < MAX_NUM_ITERATIONS; i++) {
5               M(InstanceSet, num_cluster);
6               prev_likelihood ← curr_likelihood;
7               curr_likelihood ← E (InstanceSet, num_cluster);
8               if ((curr_likelihood - prev_likelihood) < MIN_DIFF)
9                       break;
10      }
11      return curr_likelihood;
12 }
end;
```

Table 5.2 The EM Iteration

The E step implementation calculates the log-likelihood. The reason for using logarithm of the likelihood instead of likelihood is that it simplifies the calculation to sum, avoiding heavy weight multiplication of double type data according to $\log(x_1 \bullet x_2 \bullet \ldots x_n) = \sum_{i=1}^{n} \log x_i$.

1. The E step function first calculates the probability $P(X = x_i \mid C = c_x)$ of each instance i belonging to a cluster c using $P(X = x_i \mid C = c_x) = P(X = x_i) \bullet P(C = c_x)$, where $x_i = \{v_{1x}, v_{2y}, \ldots, v_{mz}\}$.
2. This implementation assumes all attributes are independent of one another. So it uses equation (5.1), $P(X = \{v_{1x}, v_{2y}, \ldots, v_{mz}\}) = P(a_1 = v_{1x}) \bullet P(a_2 = v_{2y}) \bullet \ldots \bullet P(a_m = v_{mz})$, to calculate the joint probability $P(X = x_i)$ of instance x_i. Given a value v_{lw} for instance x_i 's attribute l, the probability $P(a_l = v_{lw}) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(v_{lw} - \mu_l)^2}{2\sigma_l^2}}$, where μ_l and σ_l are part of the parameter set $\theta'(\mu_l, \sigma_l)$ from the M step.

3. The probability of cluster C_x is given by $P(C = C_x) = \frac{\sum_i P(x_i \mid C_x)}{\sum_x \sum_i P(x_i \mid C_x)}$ and it is implemented by function *estimate_cluster_probs()*
4. The probability of an instance x_i belonging to cluster c_i, given by $P(X = x_i \mid C = c_x)$, gets normalized and is used as the weight in equation (5.1) in the M step for calculating the parameters of the attributes distributions.
5. Compute the likelihood $\log(likelihood) = \sum_m \sum_n \log p(x_m \mid c_n)$, and $\lim(likelihood) = 1$.

```
/*********************************************
 * function NORMALIZE() normalize the values of elements of the input array;
 * function LOG() returns the logarithm value of the input;
 * function NormalDens() returns the probability of normal distribution with input of a
 * value, a mean, and a standard deviation.
*********************************************/
function E (InstanceSet, total_cluster)  {
1       log_likelihood ← 0.0;
2       estimate_cluster_probs();
3       for (instance ← 0; instance < total_instance; instance ++ ) {
4               for (cluster ← 0; cluster < total_cluster ; cluster ++  ) {
5                       joint_prob[instance] ← 1.0;
6                       for (attribute ← 0; attribute < total_attribute ; attribute ++ )  {
7                               joint_prob[instance] ←
                                        joint_prob[instance] *
                                            NormalDens(attr_value, mean, std_dev);
8                       }
9                       weight[instance][cluster] ←
                                joint_prob[instance] * cluster_probs[cluster];
10              }
11              likelihood ← 0.0;
12              for (i ← 0; i < total_cluster ; i ++  ) {
13                      likelihood ← likelihood + weight[instance][i];
14              }
15              log_likelihood ← log_likelihood + LOG(likelihood);
16              // normalize the weight of each instance across all clusters
17              NORMALIZE (weight[instance]);
18      }
19}
```

```
end;

function estimate_cluster_probs() {
1       cluster_prob[]← 0;
2       for (instance ← 0; instance < total_instance; instance ++ ) {
3               for (cluster ← 0; cluster < total_cluster ; cluster ++ ) {
4                       cluster_probs[cluster] ←
                                cluster_probs[cluster] + weights[instance][cluster];
5               }
6       }
7       NORMALIZE(cluster_probs);
8}
end;
```

Table 5.3 The E Step

The M step, shown in Table 5.4, computes parameter set θ_i^{t+1}.

1. The mean value of attribute a of cluster c is calculated using the equation

$$\mu_{ca} = \frac{\sum_{i=1}^{n} v_{ia} w_{ic}}{\sum_{i=1}^{n} w_{ic}},$$

where I is a set of all instances, and $I = \{1,2, \ldots,n\}$,

v_{ia} is the value of attribute a of instance i,

and w_{ic} is the weight of instance i in cluster c.

2. From equation (5.4), the variance of attribute a of cluster c

$$\sigma_{ca}^2 = \frac{w_{c1}(x_{a1}-\mu_{ca})^2 + w_{c2}(x_{a2}-\mu_{ca})^2 + \ldots + w_{cn}(x_{an}-\mu_{ca})^2}{w_{c1}+w_{c2}+\ldots+w_{cn}}$$

$$\Rightarrow \sigma_{ca}^2 = \frac{(w_{c1}x_{a1}^2 - 2w_{c1}x_{a1}\mu_{ca} + w_{c1}\mu_{ca}^2) + \ldots + (w_{cn}x_{an}^2 - 2w_{cn}x_{an}\mu_{ca} + w_{cn}\mu_{ca}^2)}{\sum_{i=1}^{n} w_{ci}}$$

$$\Rightarrow \sigma_{ca}^2 = \frac{\sum_{i=1}^{n} w_{ci}x_{ai}^2 + \sum_{i=1}^{n} w_{ci}\mu_{ca}^2 - 2\mu_{ca} \bullet \sum_{i=1}^{n} w_{ci}x_{ai}}{\sum_{i=1}^{n} w_{ci}}$$

$$\Rightarrow \sigma_{ca}^2 = \frac{\sum_{i=1}^{n} w_{ci}x_{ai}^2 + \sum_{i=1}^{n} w_{ci}\mu_{ca}^2 - 2\mu_{ca}^2}{\sum_{i=1}^{n} w_{ci}} \Rightarrow \sigma_{ca}^2 = \frac{\sum_{i=1}^{n} w_{ci}x_{ai}^2 + \mu_{ca}^2(\sum_{i=1}^{n} w_{ci} - 2)}{\sum_{i=1}^{n} w_{ci}}$$

where instance i from *1* to n

```
/*  function SQRT () returns the square root of the input value. */
function M (InstanceSet, total_cluster)  {
1       cluster_models[]← 0.0;
2       for (cluster ← 0; cluster < total_cluster ; cluster ++ )        {
3               for (attribute ← 0; attribute < total_attribute ; attribute ++ )  {
4                       for (instance ← 0; instance < total_instance; instance ++ ) {
5                               cluster_models[cluster][attribute].mean ←
                                        cluster_model[cluster][attribute].mean
                                        + instances(instance).value(attribute)
                                        * weights[instance][cluster]);
6                               cluster_models[cluster][attribute].stdDev ←
                                        cluster_model[cluster][attribute].stdDev
                                        + instances(instance).value(attribute)
                                        * instances(instance).value(attribute)
                                        * weights[instance][cluster]);
7                               cluster_model[cluster][attribute].weight ←
```

```
                        cluster_model[cluster][attribute].weight
                                + weights[instance][cluster]);
8                   }
9               }
10      }
11      for (attribute ← 0; attribute < total_attribute ; attribute ++ )  {
12              for (cluster ← 0; cluster < total_cluster ; cluster ++  ) {
13                      cluster_models[cluster][attribute].mean ←
14                              cluster_models[cluster][attribute].mean
15                                      / cluster_models[cluster][attribute].weight;
16                      // variance
17                      cluster_models[cluster][attribute].stdDev ←
                                ( cluster_model[cluster][attribute].stdDev
                                  - cluster_models[cluster][attribute].mean
                                    * cluster_models[cluster][attribute].mean
                                    * (2 - cluster_model[cluster][attribute].weight )
                                    ) / cluster_model[cluster][attribute].weight;
18                      // std dev
19                      cluster_models[cluster][attribute].stdDev ←
                                SQRT ( cluster_models[cluster][attribute].stdDev );
20              }
21      }
22}
end;
```

Table 5.4 The M Step

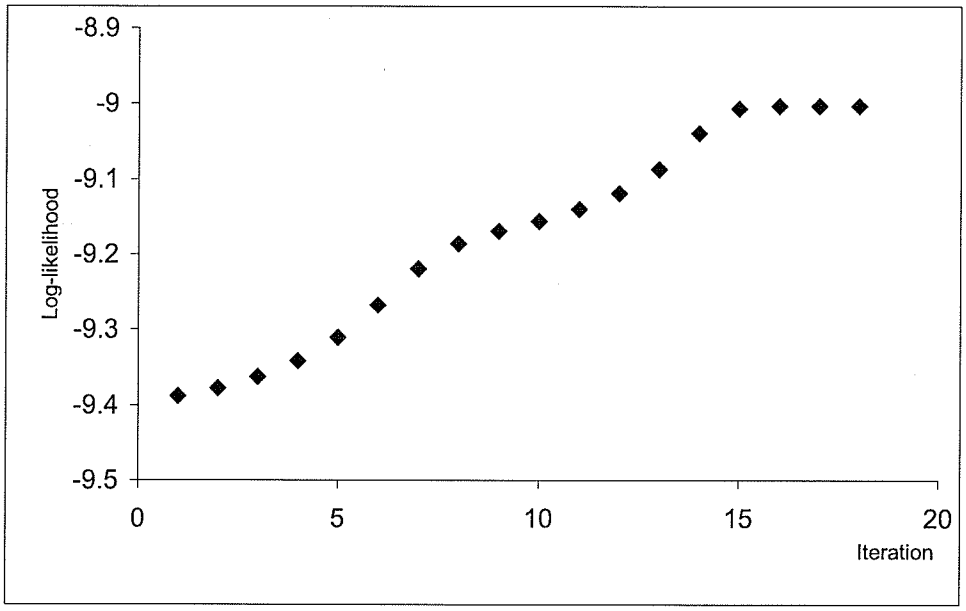

Figure 5.8 Log-likelihood over Iterations

	Cluster 0		Cluster 1	
Iteration	mean	std dev	Mean	std dev
1	74.4577	6.8451	72.6779	5.6306
2	74.8128	6.8563	72.3003	5.4622
3	75.2109	6.8762	71.8711	5.1939
4	75.6936	6.8514	71.3549	4.8312
5	76.2775	6.7257	70.7636	4.3925
6	76.9772	6.4213	70.1487	3.9489
7	77.7699	5.8863	69.6260	3.6188
8	78.4853	5.2663	69.3409	3.4735
9	79.0018	4.8515	69.3115	3.4375
10	79.4431	4.6040	69.4286	3.4572
11	79.9211	4.3611	69.5910	3.4928
12	80.4783	4.0145	69.7521	3.5211
13	81.1193	3.4754	69.8975	3.5377
14	81.7652	2.7155	70.0233	3.5514
15	82.1760	2.0763	70.1123	3.5723
16	82.2636	1.9288	70.1396	3.5860
17	82.2688	1.9227	70.1444	3.5910
18	82.2693	1.9225	70.1452	3.5921

Table 5.5 Normal Distribution Parameters in each iteration

Figure 5.8 on page 79 shows the logarithm values of the likelihood over EM iteration. As shown in Figure 5.8, the values after iteration 15 have very minor change. To improve the efficiency of the EM algorithm, the iteration stops when the difference between the current likelihood and the likelihood in previous iteration is less than a predefined number (10^{-4} in this example). Table 5.5 lists the values of mean and standard deviation of both clusters over iterations. Figure 5.9 shows the convergence of the clusters' distributions over EM iterations.

From the running results in Table 5.6 on page 82, we can learn some relationship between the click number of the product id 40000-11-050 and the total value of the purchase. If a customer sees the product 40000-11-050 six times in a session, he/she tends to purchase with mean value of 82.2693. The probability for a user clicking product 40000-11-050 six times in a session is 28.26%. And the log-likelihood is -9.00321.

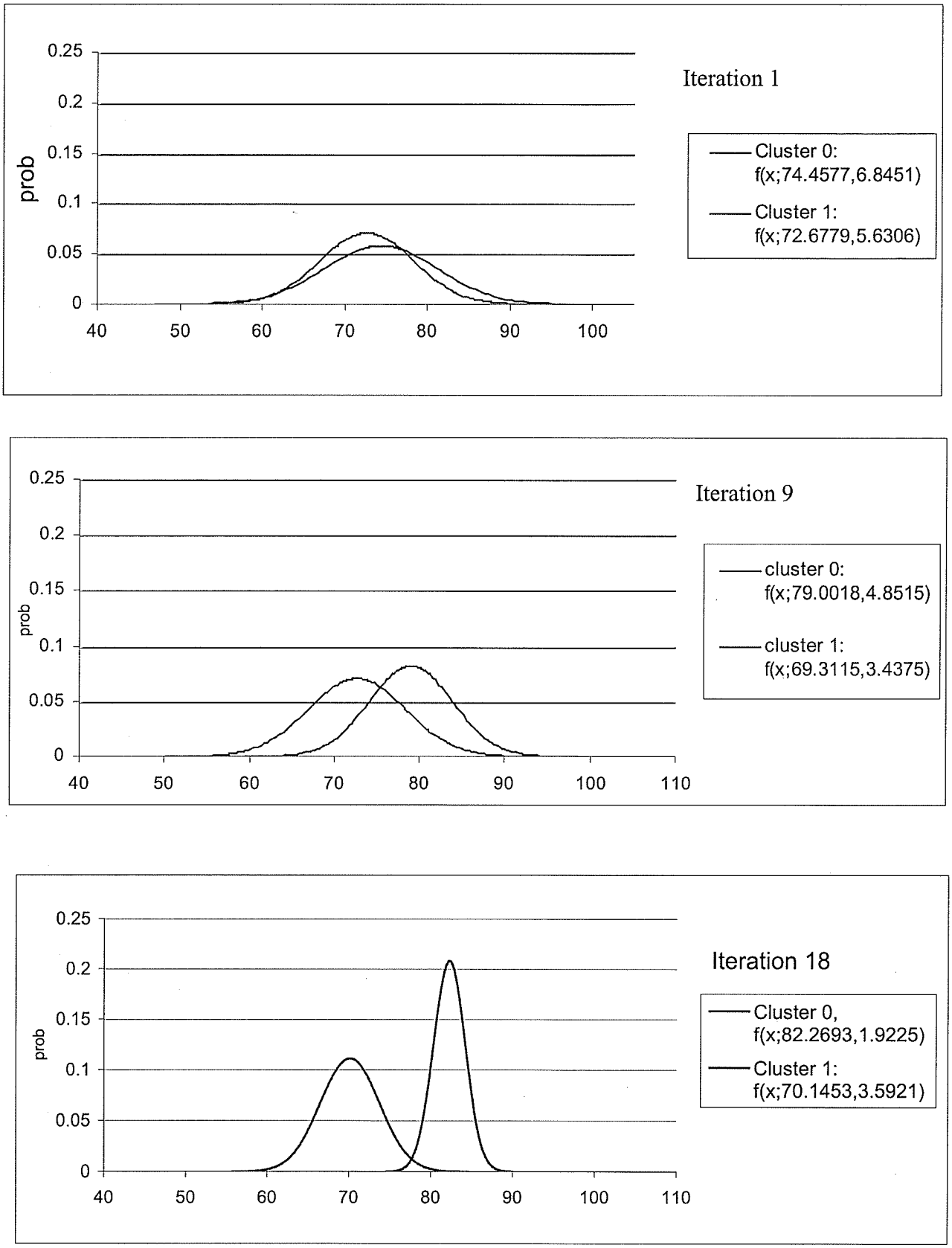

Figure 5.9 Clusters' Distributions Converges Over Iterations

```
EM Analysis result:
==============================
Number of clusters: 2


----------------------------------------------
Cluster: 0
        Instances: 4
        Prior probability: 0.2826

        Attribute: 40000-11-050
        Possion Distribution. Counts = 6.03 6.02 (Total = 12.04)

        Attribute: order value
        Normal Distribution. Mean = 82.2693 StdDev = 1.9225
----------------------------------------------
Cluster: 1 Prior probability: 0.7174
        Instances: 10
        Prior probability: 0.2826

        Attribute: 40000-11-050
        Possion Distribution. Counts = 1.97 3.98 (Total = 5.96)

        Attribute: order value
        Normal Distribution. Mean = 70.1453 StdDev = 3.5921
==============================================
Log likelihood: -9.00321
```

Table 5.6 Running Results of the EM Module

5.2.2 Sequential Pattern Mining

Sequential patterns mining was first introduced in [62]. The main purpose of mining sequential patterns is to find popular patterns of events in time series from a database which contains sets of time ordered events. Discovered frequent sequential patterns show the common changes of events and attributes' values over time. Finding evolving patterns of attributes among transactions in a time period and over time gives a better understanding of market trend. Sequential patterns mining can be used to analyze and predict market responses for promotions, seasonal movements, cyclic activities and other periodical patterns, etc. The discovery of sequential patterns also helps in target marketing aimed at groups of users based on these patterns. For e-commerce Web sites, logged customers' browsing sequences and purchasing behaviors are saved in a database and each click has a timestamp on it. Mining sequential patterns from Web log can help to reorganize the Web content to be more user friendly and efficient for target customers. Web sites' owners also can organize their business processes and logistics better to save costs once they have a better understanding of the current business processes and customers behaviours.

5.2.2.1 The Evolvement of Sequential Pattern Mining

Agrawal and Srikant [62] introduce three algorithms for mining sequential patterns: AprioriAll, AprioriSome, and DynamicSome. These algorithms are based on an association rules mining algorithm, the Apriori algorithm [61, 62]. Association rules mining finds frequent sets of items and, therefore, generates desired rules. Association rules mining, according to Agrawal et al [63], is defined as follows:

> "Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items. Let D be a set of transactions, where each transaction T is an itemset, a set of items, such that $T \subseteq I$. Associated with each transaction is a unique identifier *TID*. We say that a transaction T contains itemset X, a set of some items in I, if $X \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ holds in the transaction set D with confidence c if c% of transactions in D that contain X also contain Y."

Confidence c is defined as the percentage of transactions which contains both X and Y among the total transactions.

Mining association rules usually ignore item sequence in transactions. Items within an itemset are kept in lexicographic order.

The formal statement of mining sequential patterns is presented in [62] as follows:

> "Denote an itemset i by $\{i_1\ i_2\ \ldots i_m\}$, where i_j is an item. Denote a sequence s by $<s_1 s_2 \ldots s_n>$, where s_j is an itemset. All the transactions of a customer are together viewed as a sequence, where each transaction corresponds to a set of items, and the list of transactions, ordered by increasing transaction-time, corresponds to a sequence. Formally, let the transactions of a customer, ordered by increasing transaction-time, be T_1, T_2, ..., T_n. Let the set of items in T_i be denoted by $itemset(T_i)$. The customer-sequence for this customer is the sequence $<itemset(T_1)\ itemset(T_2)\ \ldots\ itemset(T_n)>$. A customer supports a sequence s if s is contained in the customer-sequence for this customer. The support for a sequence is defined as the fraction of total customers who support this sequence."

Generally speaking, mining association rules focuses on finding intra-transaction patterns, whereas mining sequential patterns is concerned with inter-transaction patterns [62].

Among association rules and sequential patterns mining algorithms, Apriori-like algorithms are the most popular. This kind of algorithms contains four major steps, namely:

1. Data preparation. The original transaction records in the source database is grouped by customer id and sorted in time sequence.
2. Generating candidate rules/patterns. The algorithm generates those rules/patterns that may be frequent, which is also called candidate sequences.

3. Finding large itemsets. The database is scanned to find the support for each candidate. A large itemset refers to an iemset that has higher support than the minimal support. A candidate gets pruned if it is not large.
4. Maximizing rules / patterns' length. Those rules / patterns that are contained in other super rules / patterns are pruned in this step because they already exist and should not be used to generate candidates again.

There are many apriori-like algorithms. Among them Apriori, AprioriTid, and AprioriHybrid [61], AprioriAll, AprioriSome, DynamicSome [62], and GSP [64] are most well known. The major difference among these algorithms is how to generate candidates and prune the candidates which will not lead to large itemsets more efficiently. Because the transaction database may be huge in size, improving the efficiency in these two aspects can greatly improve the performance.

The AprioriAll sequential patterns mining algorithm only has minor differences in the join step for generating candidates when compared to the Apriori algorithm. To generate candidates with length k, both algorithms take the set of all large itemsets L_{k-1} with length k-1 as input parameter in candidate generate function. In Apriori algorithm, the initial candidates C_k is generated by the algorithm shown in Table 5.7.

```
insert into Ck
    select p.item1, p.item2, ..., p.itemk-1, q.itemk-1
    from Lk-1 p, Lk-1 q
    where p.item1 = q.item1, ... p.itemk-2 = q.itemk-2, p.itemk-1 < q.itemk-1
```

Table 5.7 Join Step in Apriori Candidate Generate Function [61]

In AprioriAll algorithm, the initial candidates C_k is generated by the algorithm shown in Table 5.8.

```
insert into Ck
    select p.item1, p.item2, ..., p.itemk-1, q.itemk-1
    from Lk-1 p, Lk-1 q
    where p.item1 = q.item1, ... p.itemk-2 = q.itemk-2,
```

Table 5.8 Join Step in AprioriAll Candidate Generate Function [62]

As shown above, for same data source (assuming the database has timestamp for each transaction) the candidates in AprioriAll $C_{AprioriAll}$ is a superset of the candidate in Apriori $R_{Apriori}$. However, the final result by AprioriAll $R_{AprioriAll}$ is not always a superset of the final result by Apriori $C_{Apriori}$. This is because for each association rule discovered by Apriori algorithm, there may exist many sequential patterns and some, or even all, of them get pruned because of low support. For example, from <*a*>, <*b*>, $C_{Apriori}$ = {<*ab*>}, while $C_{AprioriAll}$ = {<*ab*>, <*ba*>}and candidate <*ab*> gets pruned because it is under the support threshold.

Generally speaking, in Web log mining domain, sequential patterns are more interesting because they not only report popular itemsets, but they also reveal underlying relationships among them. With the understanding of frequent sequential patterns, businesses can organize these Web contents, control inventory, manage logistic more efficiently, and make effective target marketing strategies and predict future trend. This thesis uses sequential pattern mining to demonstrate these benefits from Web content log.

5.2.2.2 Current Web Sequential Pattern Mining

Some Web usage mining research systems have been developed in the past few years, such as the WebMINER [67, 68, 69] and WebSIFT [52]. These systems all include mining Web usage sequential patterns from Web log. Sequential pattern mining is also used for various purposes other than for market analysis. For example, Mobasher et al [7] introduce mining combined Web usage and content for better personalization, and [70, 71] all use data mining techniques to enhance the Web caching mechanism through better prediction of next possible service request. Some problem specific sequential pattern mining algorithms, such as the WAP-mine [56] algorithm, exist for mining sequential patterns from Web log efficiently.

Although many work on sequential access patterns mining exist, most of them focus on improving the efficiency of mining sequential access patterns from Web logs. Web content mining and Web usage mining used to be treated separately. But mining Web content alone results in missing important relationship among Web objects based on their

usage, and mining usage data alone can be problematic when the content may change frequently [7]. While many Web sites are delivering services via various channels with dynamic generated content, mining integrated Web content and Web usage data can provide more meaningful analysis. Mobasher et al [7] build feature vector for each pageview and calculates the weight for each vector from usage data. By using the profile matching score which is computed from feature vector and corresponding weight, content data and usage data are integrated. Then k-means clustering algorithm is used to build clusters based on profile matching score for content recommendation purpose.

5.2.2.3 Mining Sequential Pattern from Web Content Log

Traditionally, Web access sequential pattern mining only uses access URLFs in the Web log as data source. For static pages with frequent updated content, and dynamic generated content, especially with the rapid growth in popularity of the MVC design pattern based dynamic sites, the mining results usually do not reveal much information. For example, the prototype Web site described in Chapter 3 uses a single program to generate page content to show products both in series and item levels, so all product browsing requests are represented by the same URL in the Web access log. So mining such Web access log can not reveal users' product browsing behaviours.

This thesis presents techniques to mine sequential access patterns from integrated Web content data and usage data. The goal is to demonstrate mining the integrated content and usage log, the Web content log which is introduced in Chapter 2, can help business analysts better understand visitors' interests.

Using pageview, we can distinguish the contents of different requests for same URL. A pageview p is defined as a response page requested by user. Each pageview p contains a URL l_i and a set of content features $<a_1, a_2, \ldots, a_n>$. The URL l_i is user's request URL and the content feature set includes all interesting features contained in the response page. Content features are defined and configured by site administrator to indicate interesting content to log for analysis.

The content saved in Web content log is in XML format. XML data is usually in a tree structure. To simplify the mining procedure, each attribute value in the page is mapped to a unique content id in a global dictionary. A sample is shown in Table 5.9.

XML data	Mapped Content ID
<main>	
<product>	
<sku>40000-02-120</sku>	1
<status>In stock</status>	2
<price>499.99</price>	3
</product>	
<product>	
<sku>40000-02-220</sku>	4
<status>In stock</status>	5
<price>539.99</price>	6
</product>	
<product>	
<sku>40000-02-350</sku>	7
<status>In stock</status>	8
<price>599.99</price>	9
</product>	
</main>	

Table 5.9 An XML Data to Content ID Mapping Sample

In Table 5.9, because each <status> belongs to different <product>, they are considered as different attributes, thus have different Content ID. The size of logged XML content can be huge sometimes, which can results in huge size of content IDs, and thus, causes mining procedure to be inefficient. To improve the efficiency of the mining procedure, narrowing down the size of content IDs quickly is highly recommended. Designing

different XSL stylesheets according to different business purposes gives a lot of control in subtracting interested subsets in smaller size from the original tree structure.

After mapping a pageview p to a URL l and a vector of content IDs <1, 2, …, n>, each session, as the sample shown in Figure 4.6, can be replaced in a simpler form, as shown in Figure 5.10.

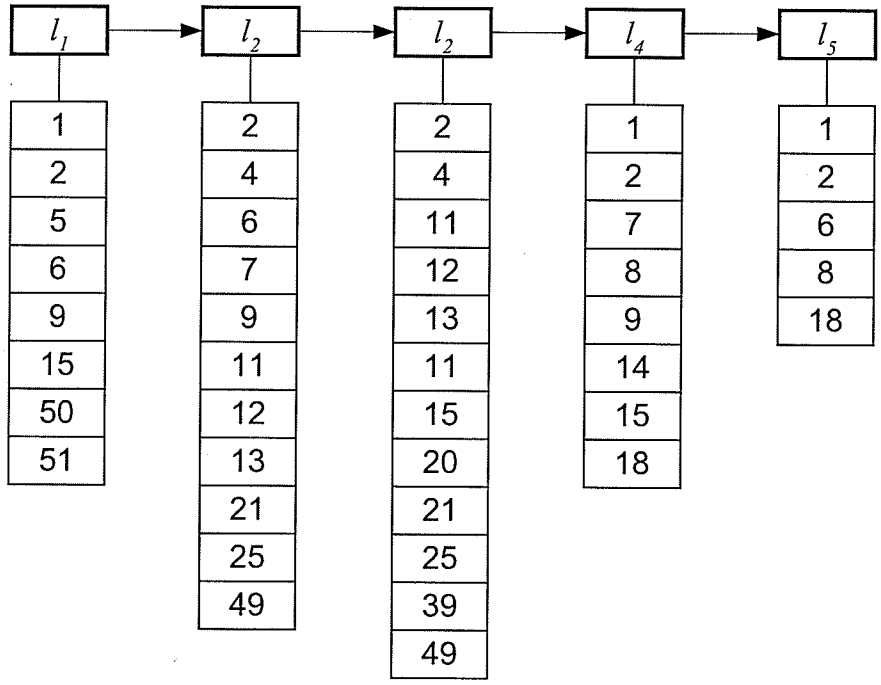

Figure 5.10 A Sample User Session Represented by Content ID

However, the content ID session database is not ready for access pattern sequential mining yet because the size of the content Ids is usually still too large to run the mining algorithm efficiently.

In the Web content mining domain, when a transaction contains an item, an itemset, or a sequence, we say this transaction support that item, itemset, or sequence, respectively. The support, *supp*, for an item, an itemset, or a sequence is defined as the fraction of total number of transactions which contains this item, itemset, or sequence, respectively [61, 62]. According to Agrawal and Srikant [61, 62], any subset of a large itemset or sequence must be large too. So by eliminating items below a minimum threshold support, the

sequential patterns mining algorithms can work much more efficiently without impacting on the mining results. The original content ID session database is scanned to build a frequent content ID session database which only contains items that have higher supports than the minimal support defined.

Mining content access sequential patterns can be formally defined as follows: Let Content Set $C = \{c_1, c_2, \ldots c_n\}$ be a set of all contents, and $supp(c_i)$ $(1 \leq i \leq n) \geq SUPPORT_{min}$. The content in each page p is represented by a page vector $p\ (c_a, c_b, \ldots c_x)$, $1 \leq a < b < x \leq n$. The sequence of contents in each page can be ignored, and thus they are listed in alphabetical order. Each content c_i can occur at most once in a page, but can occur multiple times across multiple different pages. Each vector is considered as an element in the sequence and the length of a sequence is the total number of elements.

The goal of mining Web content log is to find long sequential access patterns with high support. Mining Web content log can tell exactly what items customers are exactly interested in instead of ambiguous URLs.

A typical Apriori-like algorithm usually uses multiple-pass dataset scanning for counting candidates' supports generated from shorter frequent sequences. While mining a big dateset, such algorithm usually faces following challenges:

- When the variation of elements in the dataset gets bigger, the exhaustive search required to find all patterns makes the size of the candidate sequence set increase exponentially. This brute force approach may consume both enormous time and space. The number of subsequences that must be examined is N(N-1)/2 where N is the length of the overall sequence. For a sequence with 1000 elements, 499,500 subsequences must be examined. Apriori-like algorithm also results in difficulties at mining long sequential patterns. For example, the total number of Appriori candidates of a 100-sequence will be:

$$\sum_{i=1}^{100} c_i^{100} = 2^{100} - 1 \approx 10^{30}$$

- The algorithm starts to build candidates with length one until the length that no sequence is found above the support. In each loop, the whole dataset is scanned to find the support of each possible candidate to count the exact support.

- To improve the efficiency of mining large dataset, Apriori-like algorithm usually maps large itemsets (elements that contain large number of individual items) as single entities. By comparing two large itemsets for equality in constant time, it helps reduce the time required to check if a sequence is contained in a customer transaction. In real world, selecting items for each large itemset must be done very carefully if the user wants to have some level of details in the mining results while not losing much of the execution efficiently.

The PrefixSpan algorithm [65] mainly employs the innovative method of database projection for frequent sequences to make the candidates for next pass much less than the candidates generated by Apriori-like algorithms to make this algorithm much faster.

- By constructing frequent sequence prefix projected databases, the PrefixSpan algorithm divides the whole database into small sets which may lead to longer sequences to greatly reduce the support scanning space.
- The PrefixSpan algorithm does not generate any candidate sequence. Instead of generating and testing candidates, PrefixSpan grows longer sequential patterns from the shorter frequent ones [65].
- Projected databases keep shrinking while the length of prefix growing. When the length of a prefix increases, the number of distinct frequent patterns with same prefix decreases dramatically.

Because of the superb efficiency of the PrefixSpan algorithm even in mining large itemsets, this thesis adopts the PrefixSpan algorithm to demonstrate the benefits of sequential pattern mining on Web content access data.

The frequent content IDs are tokenized in the session access sequence database. Table 5.10 gives an example of Web contents in the session access sequence database.

Seq_id	Sequence
1	<(af)(bc)(e)>
2	<(eh)(ab)(bh)(dhj)>
3	<(ac)(aef)(abc)(dhi)>
4	<(bd)(abd)(ac)(cf)(ac)>

Table 5.10 A Sample Content Sequence Database

The PrefixSpan algorithm uses the following steps to mine frequent sequences.

Step 1: Find the supports of sequences with length = 1 and sorted in descending order. From the example in Table 5.10, a sequence only contributes once to its support, no matter how many times it appears in a whole sequence. By setting the $Support_{min} = 3$ as frequent sequence threshold, from Table 5.11 we will have frequent 1-sequences: <a>, <b>, <c>, <d>, <e>, and <f>.

Step 2. The 1-length frequent sequences are used as prefixes to partition the whole database into subsets. The example in Table 5.11 will have six subsets. Each has <a>, <b>, <c>, <d>, <e>, or <f> as prefix correspondingly.

Step 3. Recursively mine projected database constructed by each partition of sequential patterns from Step 2. Find subsets of sequential patterns. Pei et al [65] give the formal definition of projected database as follows:

> "Let α be a sequential pattern in a sequence database S. The α-projected database, denoted as $S|_\alpha$, is the collection of postfix of sequences in S w.r.t. prefix α" [65].

According to Pei et al [65], the PrefixSpan algorithm still satisfies the assumption in the Apriori algorithm - "any subset of a large itemset must be large".

Let α and β be two sequential patterns in sequence database S such that α is a prefix of β. Then

1. $S|_\beta = (S|_\alpha)|_\beta$;
2. For any sequence γ having prefix , $support(\gamma) = support(\alpha|(\gamma))$
3. The size of α–projected database cannot exceed that of S.

1 – sequence	Support
A	4
B	4
C	3
D	3
E	3
F	3
H	2
I	1
J	1

Table 5.11 Support of 1 - Sequences

```
main () {
1.      total ← total number of frequent 1-sequential patterns discovered in step 2.
2.      α[total] ← frequent 1-sequential patterns discovered in step 2.
3.      S[total] ← Projected databases for each frequent 1-sequential pattern.
4.      for (i ← 0; i < total; i++)
5.              prefixSpan(α [i], 1, S[i]);
}

function prefixSpan (α, l, S) {
1.      B ← freqItems (S, α);
2       for (i ← 0; i < B's size; i++ ) {
3.              α' ← α + B[i];
4.              construct α'-projected database S';
5.              if ( S' == null )
6.                      print α';
7.              else
8.                      prefixSpan (α', l + 1, S');
9.      }
}
end;

function freqItems (S, α) {
1.      Scan S once to find frequent items F;
2.      x ← last element in α;
3.      β ← x's prefix in α;
4.      for (i ← 0; i < F's size; i++)  {
5.              β ← F[i];
6.              if ( <β, (xb)> is a sequential pattern || <a,b> is a sequential pattern )
7.                      Add β into Set F' ;
8.      }
9.      return Set F';
10. }
end;
```

Table 5.12 PrefixSpan Algorithm [65]

Table 5.12 shows the PrefixSpan algorithm while Table 5.13 illustrates how the PrefixSpan algorithm constructs projected database and finds frequent prefixes.

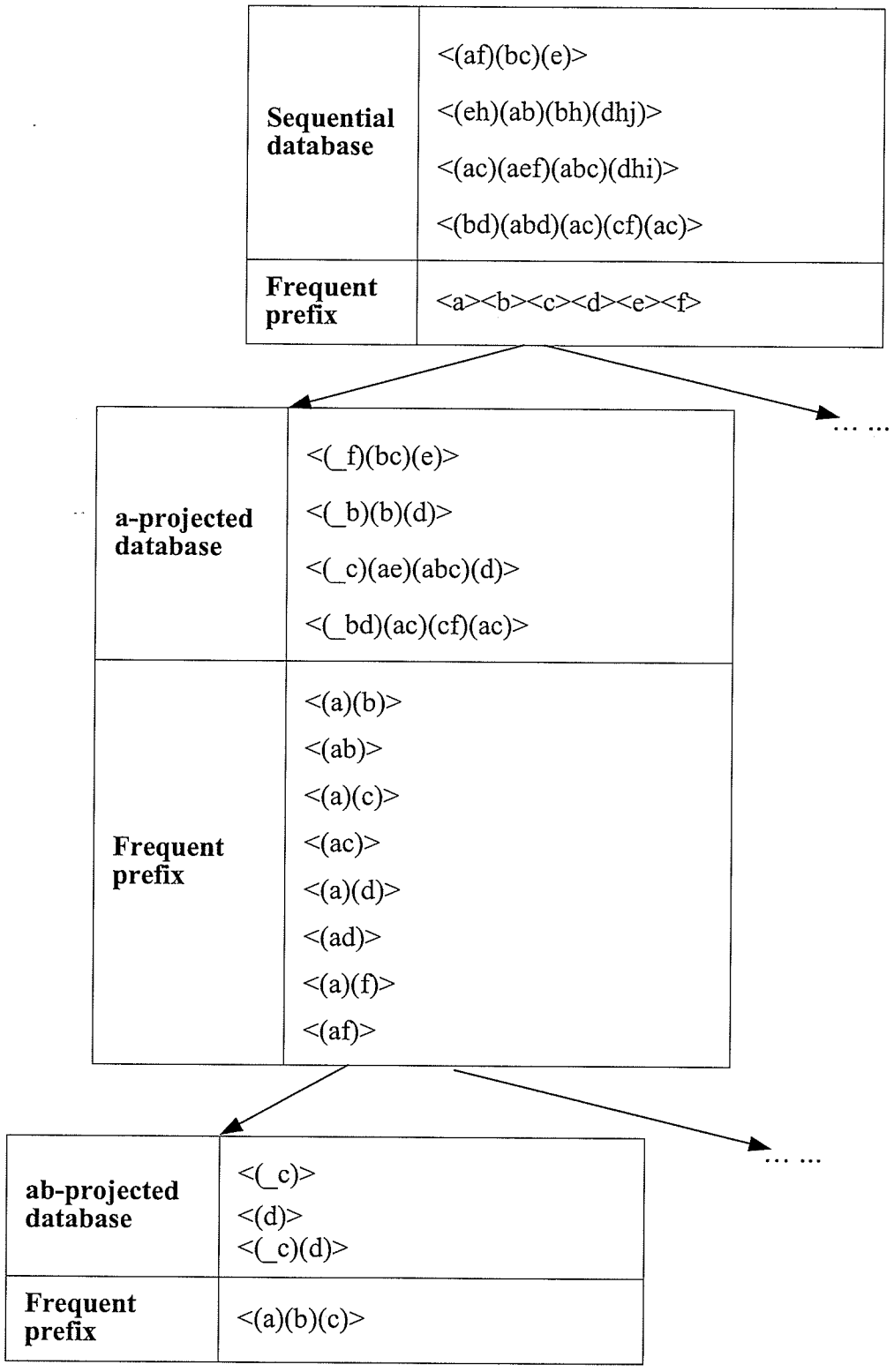

Table 5.13 Finding Frequent Sequence from the Sample Sequential Database

Table 5.14 shows the frequent sequential patterns discovered from the sample database with length greater than 3 and support equal or greater than 3.

Frequent Sequential Pattern	**Supported Sequences**
<abc>	10, 30, 40
<abd>	20, 30, 40
<afc>	10, 30, 40

Table 5.14 Frequent Long Sequential Patterns in the Sample Sequential Database

Some other sequential pattern mining algorithms have been evaluated as well. Among them, the WAP-mine algorithm in [56] and the Multi-dimensional Sequential Pattern Mining in [66] are most interesting. The WAP-mine algorithm can mine the access pattern very efficiently through building WAP access trees. But it uses URL to URL to build access trees which is limited to one to one relationship only. It can not be easily adopted to mine patterns among items from page to page. The multi-dimensional sequential pattern mining with attributes in customer profile data as the dimensions to associate these dimensions with sequential patterns. The Dim-Seq algorithm [66] mines the frequent patterns for each frequent dimension combinations.

Chapter 6

Implementation

The overall architecture of the prototype is shown in Figure 6.1. This prototype includes Web content rendering and filters for usage logging (described in Chapter 2), a Web store (described in Chapter 3), and an EM clustering program and a PrefixSpan sequential pattern mining program (described in Chapter 5).

The filters and the Web store implementation use Java as the programming language and runs on Jakarta Tomcat application server version 4.1.12. The implementation utilizes the filtering chain mechanism introduced in the Java Servlet Specification version 2.3 to achieve XML data XSL transformation for user interface rendering and content logging. An earlier version of the Web store implementation based on Jakarta Tomcat version 3.2 uses Tomcat interceptor modules for the transformation and logging. Although Servlet 2.3 filtering looks similar to existing legacy module interceptor, known as hooker, in Apache Http Server, the new Servlet 2.3 filtering mechanism is completely different architecturally.

- The previous hooked methods in interceptor modules are called on every request. Furthermore, method scoping and concurrency concerns in a multithread environment do not allow any easy or efficient sharing of variables and information between the different hooked method invocations when processing the same request [84, 85].
- The filtering mechanism in Java Servlet Specification 2.3 fully utilizes the object-oriented nature of the Java platform to make a chain of filters by nested method calls.

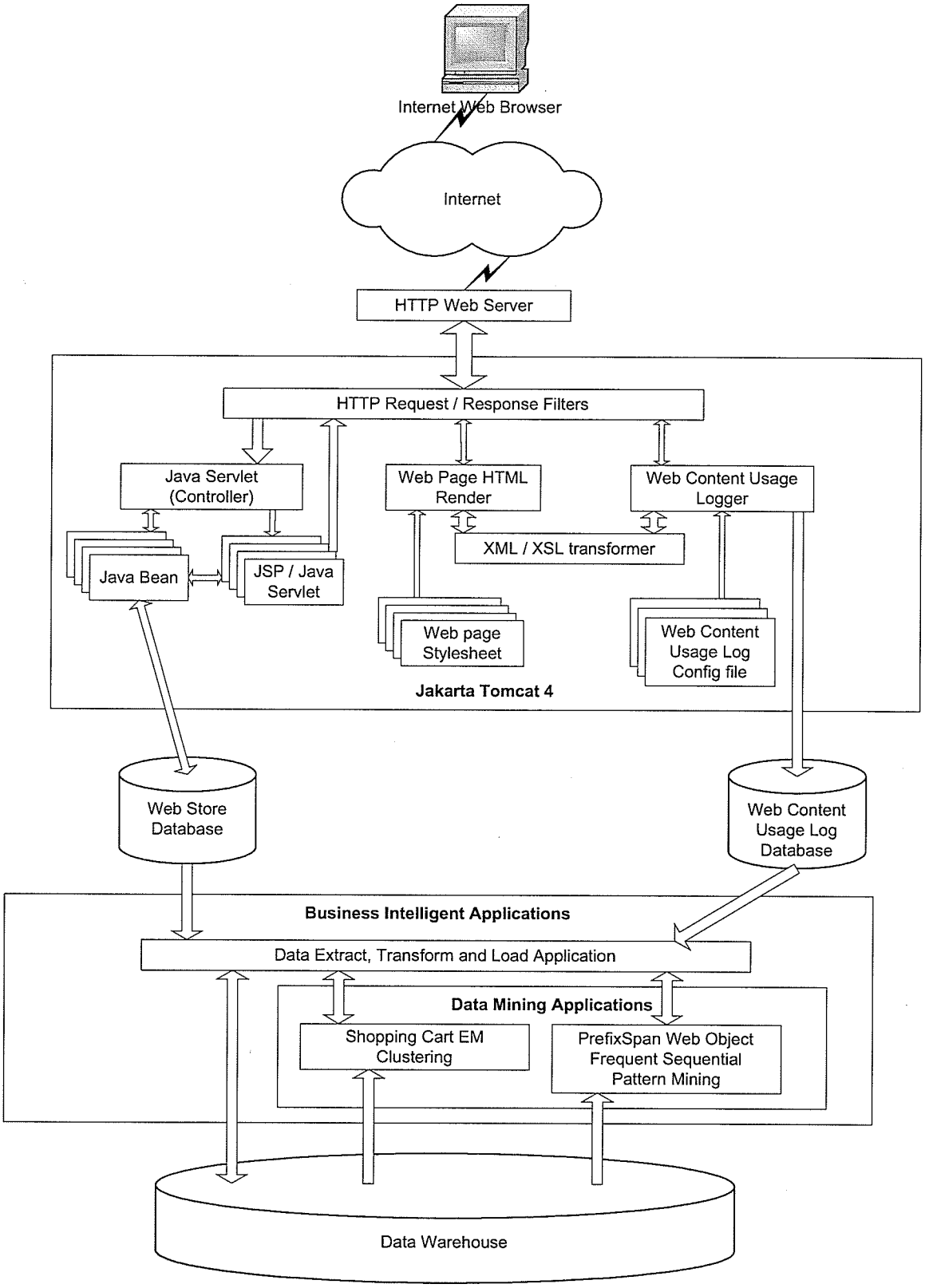

Figure 6.1 Prototype Overall Architecture

For each request, only necessary methods are called. The architecture helps in developing a more stable application with better performance.

The EM clustering program and the PrefixSpan sequential pattern mining program are also developed in Java and they both run in command line mode. DB2 is used as the backend database of the Web store, the Web content usage raw log database, and cleansed data repository (called data warehouse) for the data mining programs. To connect to DB2 database, JDBC is used as the database driver in the Java programs.

6.1 Web Store Implementation

Figure 6.2 Product Category Page

The default homepage of the prototype Web store is the product category page, as shown in Figure 6.2. This page has a product series navigation panel on the left and a main panel giving a brief introduction for each series. The visitor's status bar on the top enables him / her to check his / her shopping cart, login and logout. This page also serves as the welcome page when a visitor first enters the web store. The top visitor status bar and left navigation panel are used across all web store pages.

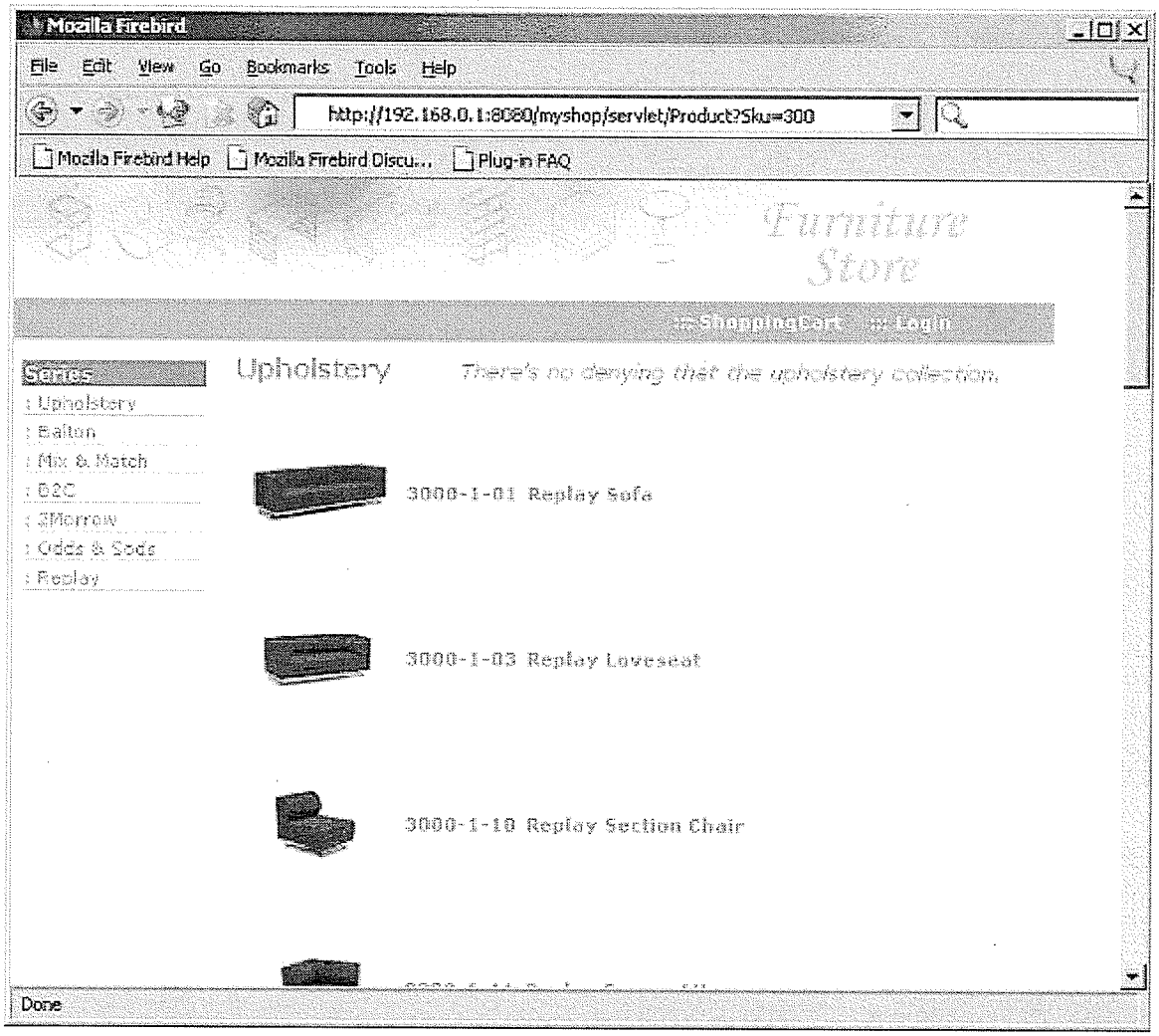

Figure 6.3 Series Detail Page

The series detail page, as shown in Figure 6.3, shows series introduction followed by a list of all items in the series a visitor selected. The list contains a thumbnail, product number, and name for each item.

Figure 6.4 Product Item Detail Page

Visitor can get all detail information about a product item from the product item detail page, shown in Figure 6.4. This page shows product item number, name, a big product image, features, options, and price. By clicking the "Buy" button, a visitor adds this item to his / her shopping cart. A visitor can always change the default quantity "1" to the number he / she wants to order.

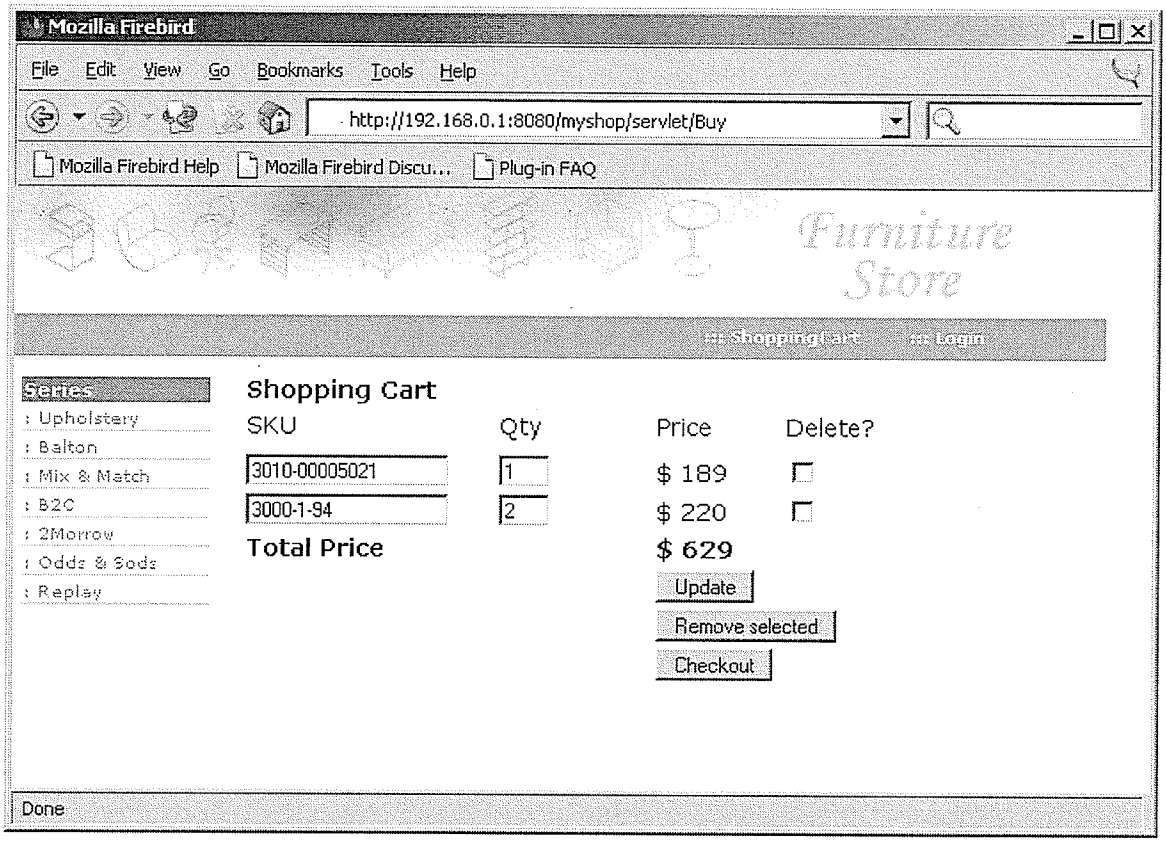

Figure 6.5 Shopping Cart Status Page

Clicking the shopping cart link on the status bar brings the visitor to the shopping cart status page. This page lists the product number, name, quantity, and unit price of all the items in the current visitor's shopping cart, and calculates the total price. Figure 6.5 shows an example of a shopping cart for a customer. A visitor can change the number in the quantity field and click "Update" button to change the quantity he / she wants to order. Checking the remove check box on the right and clicking "Remove" button removes checked item(s) from the shopping cart. To checkout current shopping cart, simply click the "Checkout" button. If the visitor has not logged in or registered, he / she

will be taken to the customer login page. Otherwise the Web store will show him / her the order review page, shown in Figure 6.6.

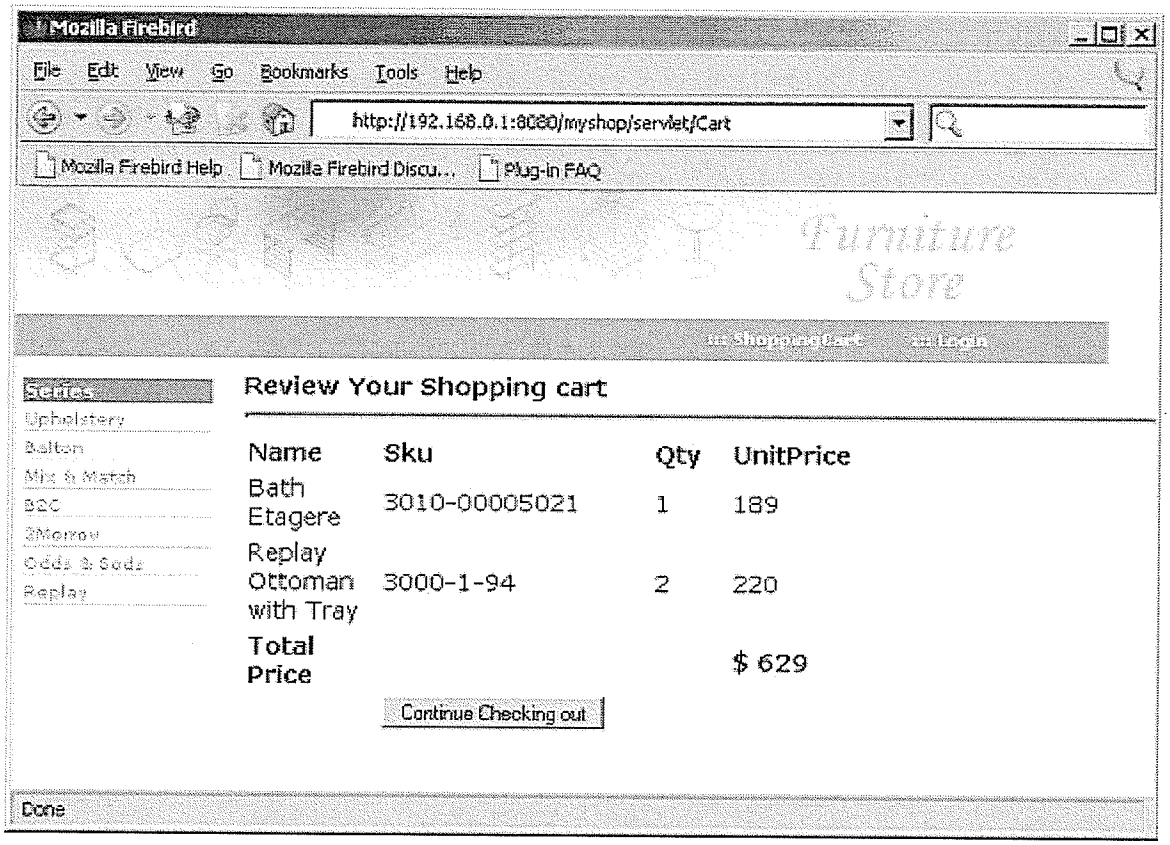

Figure 6.6 Order Review Page

Before clicking the "Place the order" button to submit the order, a customer verifies his / her billing and shipping information, the product number, name, quantity, and unit price of all the items and grand total he / she wants on this order review page.

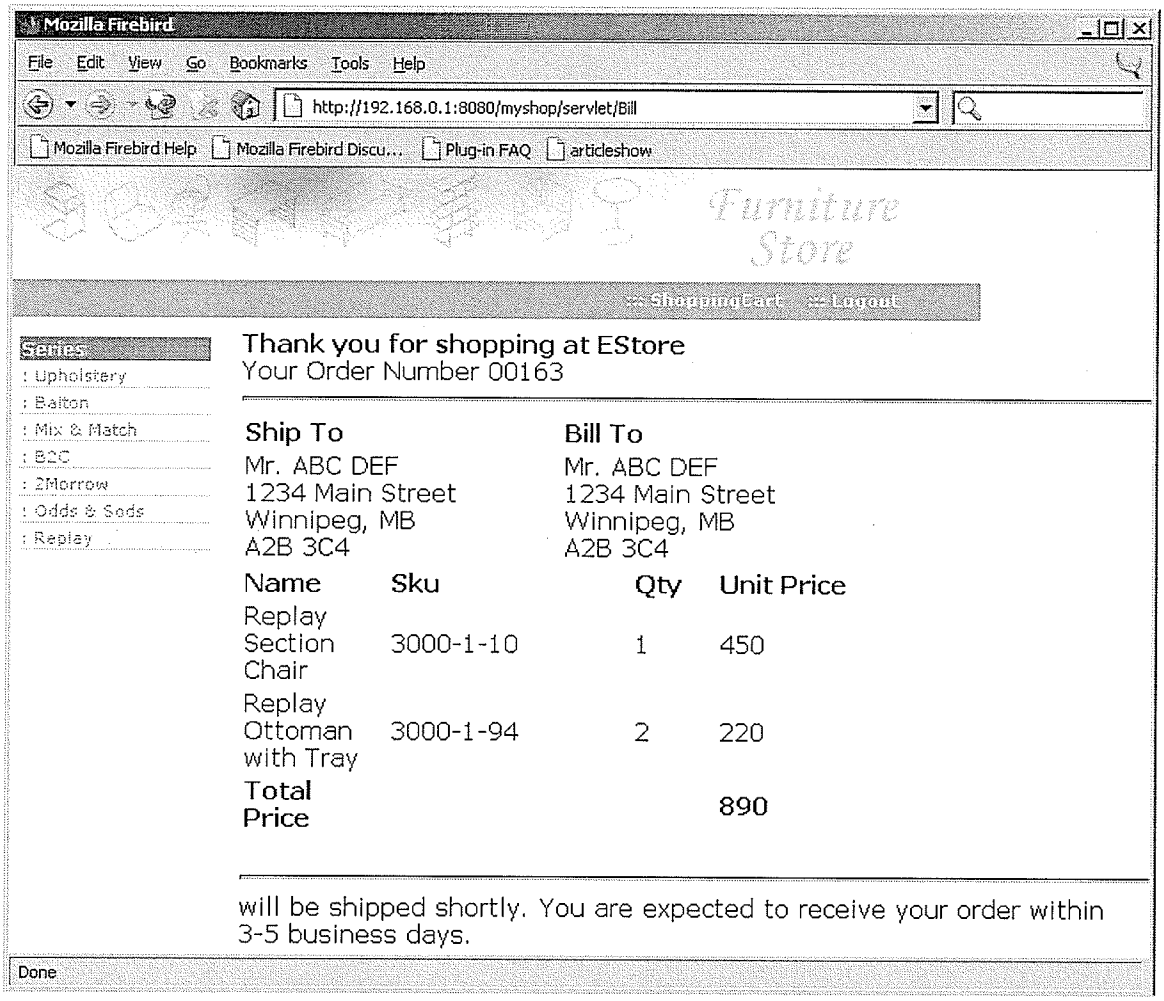

Figure 6.7 Order Confirmation Page

The order confirmation page, as shown in Figure 6.7, gives a purchase order number and shows customer's billing and shipping information, the product number, name, quantity, and unit price of all the items and the grand total of the order.

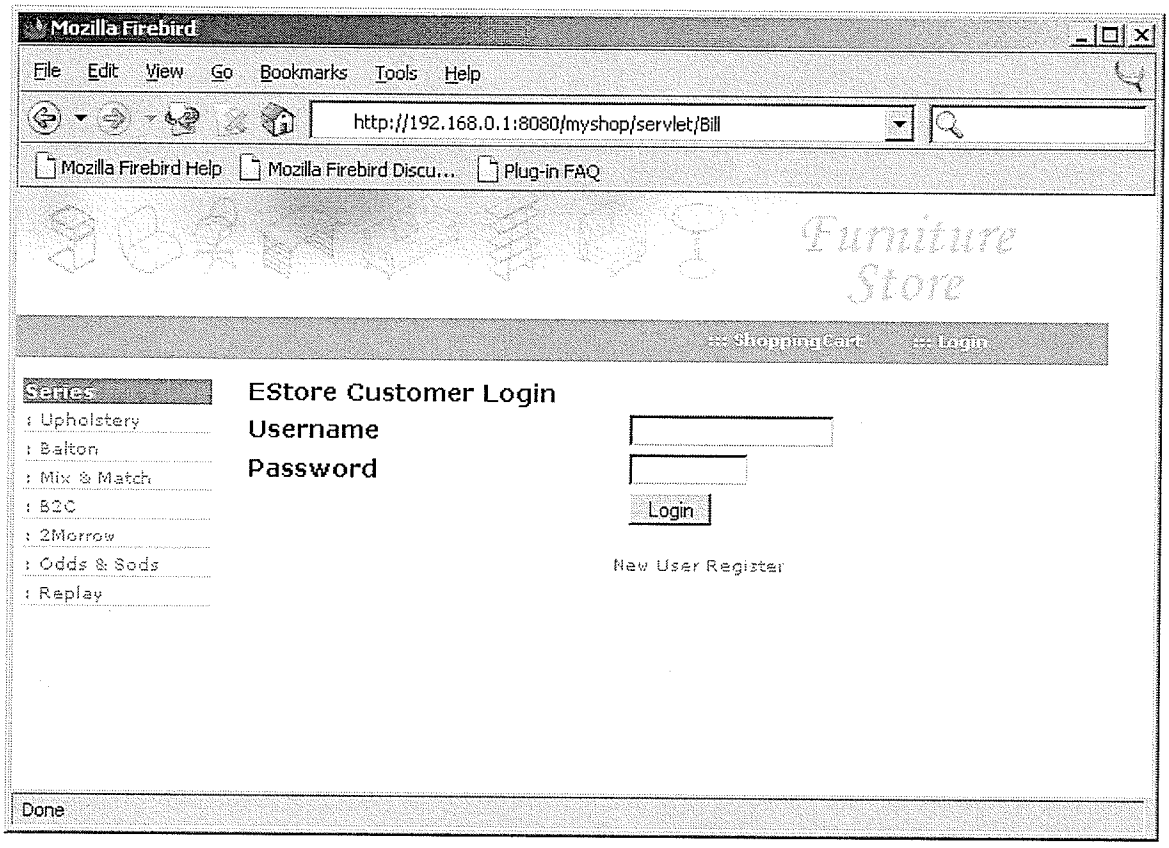

Figure 6.8 User Login Page

A customer enters username and password, and then clicks the "Login" button on the user login page, shown in Figure 6.8, to login to the Web store. If a visitor has not registered as a customer, the "Register" link takes him / her to the customer registration page. A customer must login before checking out his / her shopping cart. If the username and password are successfully verified, the Web store shows the customer a success login welcome page. If the username or password is invalid, the user login page prompts the visitor invalid username or password, and requests the customer to retry.

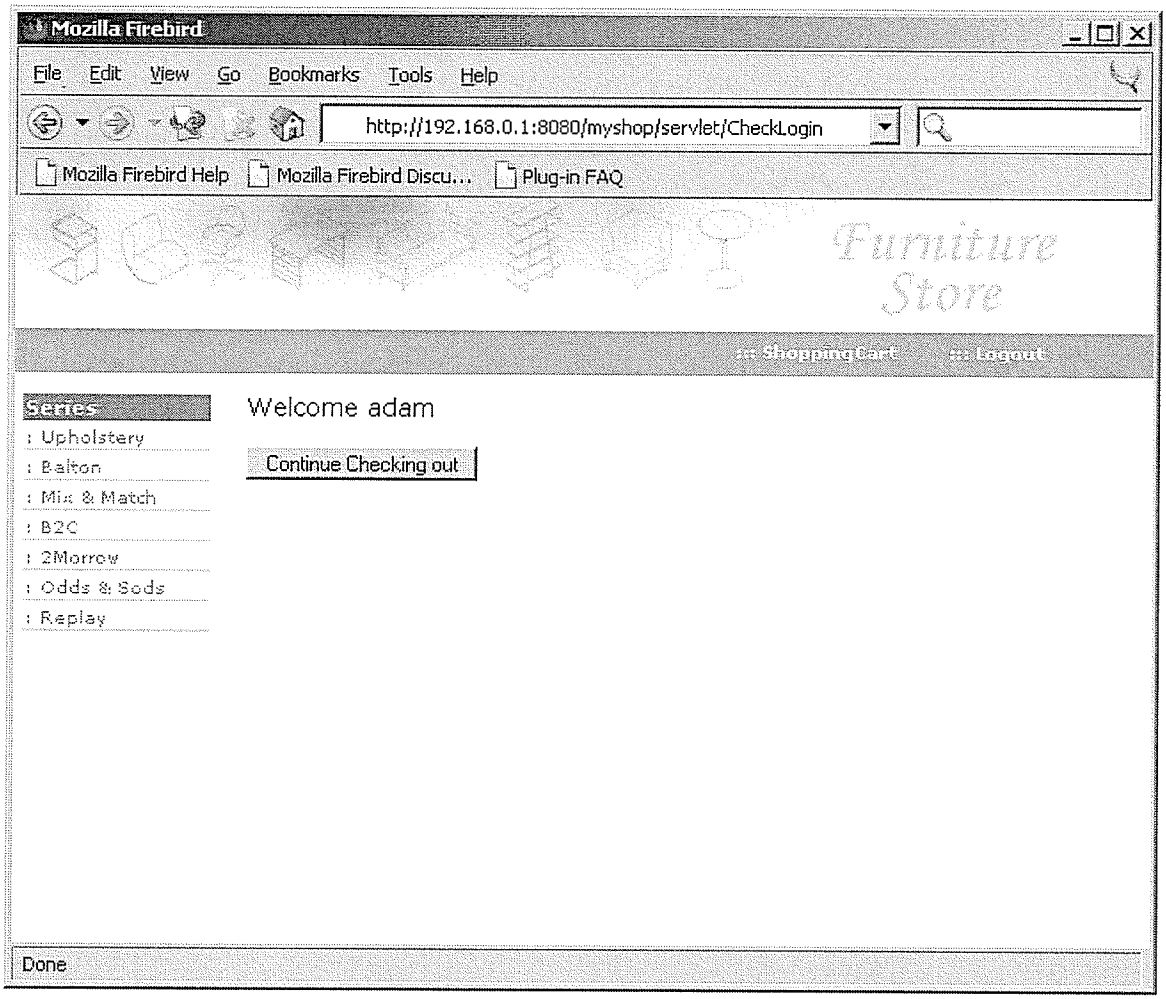

Figure 6.9 User Welcome Page

The customer welcome page, shown in Figure 6.9, confirms the customer has logged in successfully. If the customer logged in during the checkout procedure, a "continue checkout" button will be shown. Clicking this button leads the customer to the order review page.

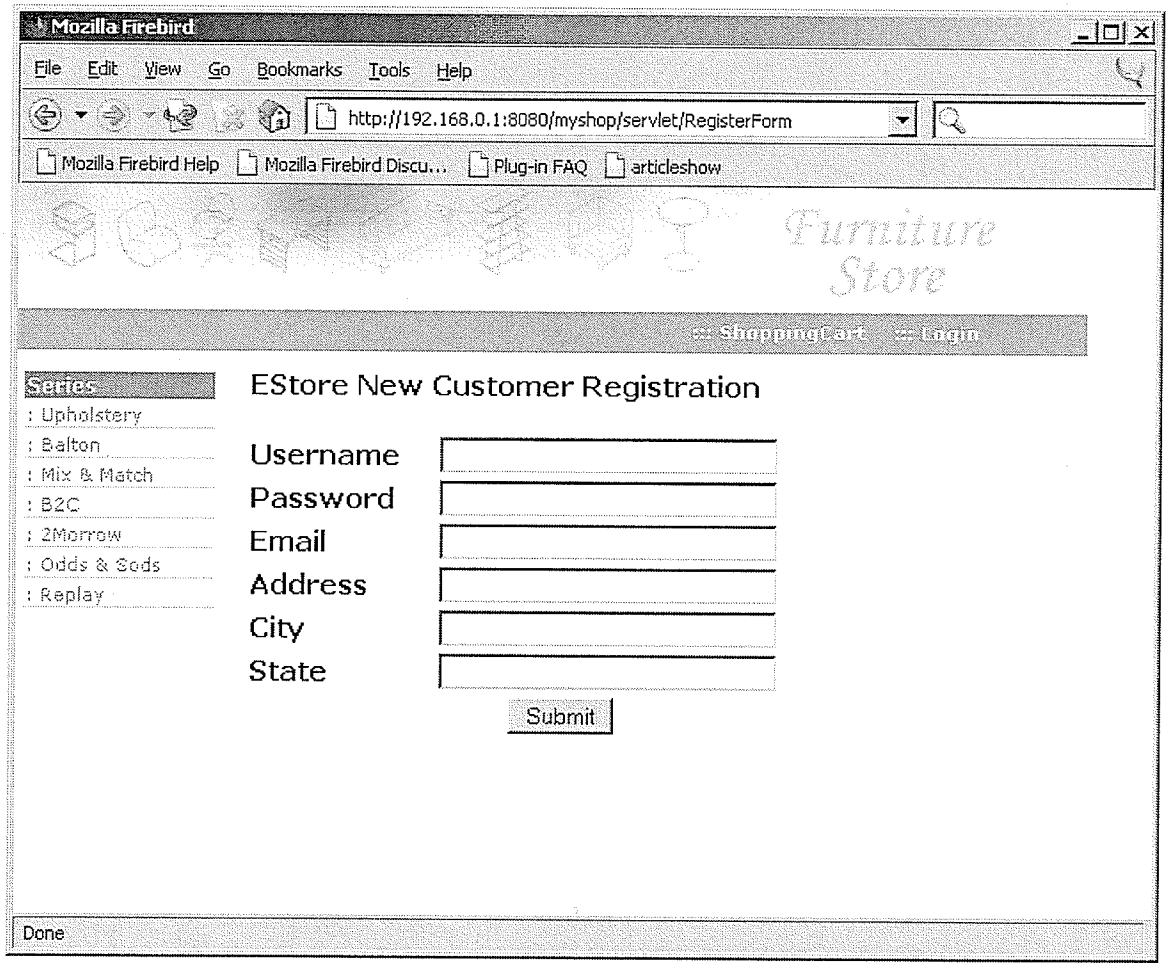

Figure 6.10 New User Registration Page

A visitor uses register form page, shown in Figure 6.10, to register as a new customer of the Web store. The visitor needs to provide a username, password, and billing information. Successful registration leads the customer to the customer welcome page.

6.2 Web Store Performance

The XML and XSL transformations for HTML output and content log are two extra procedures required from traditional architecture for building dynamic content Web page. To evaluate the cost of these two procedures, the series detail page is chosen to compare the performance of different implementations. A series detail page using traditional JSP

page for HTML format output is developed, which calls JavaBeans for database query. The JSP implementation shares three database queries for dynamic content with the product series detail page of the Web store prototype, and have very similar business logic and HTML output.

The test Web application server uses Jakarta Tomcat version 4.1.12 running on an Intel Pentium Celeron 366MHz PC with a 7200RPM and 2M cache hard disk and the database is IBM DB2 version 7.2 running on an Intel Pentium III 766MHz PC with a 7200RPM and 2M cache hard disk. The test was connected in a 100M LAN environment.

The stress performance test uses Apache JMeter version 1.9.1, an open source web performance test tool for both static and dynamic content.

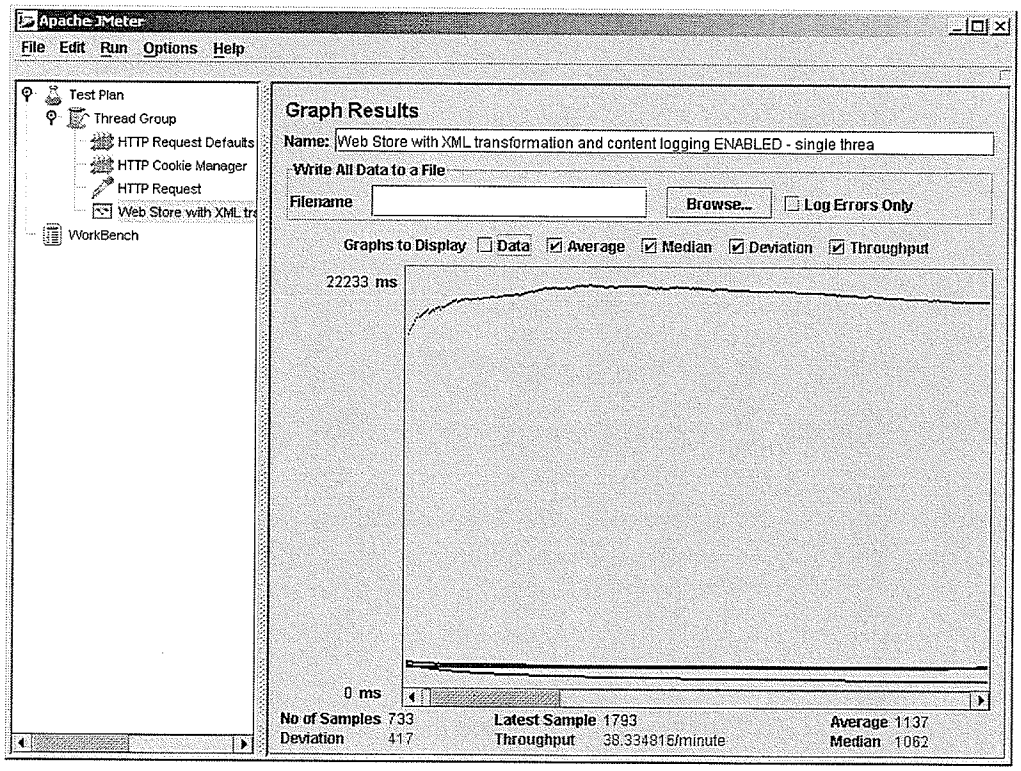

Figure 6.11 Web Request with XML & XSL Transformation Enabled, and Content Log Enabled (Single Thread Mode)

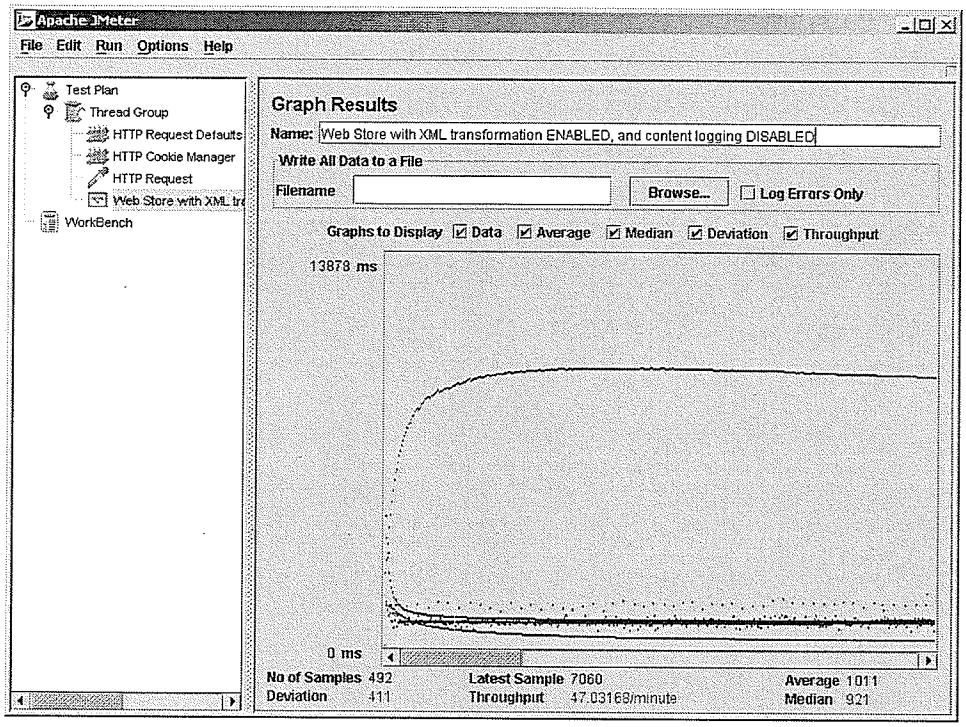

Figure 6.12 Web Request with XML & XSL Transformation Enabled, and Content Log Disabled (Single Thread Mode)

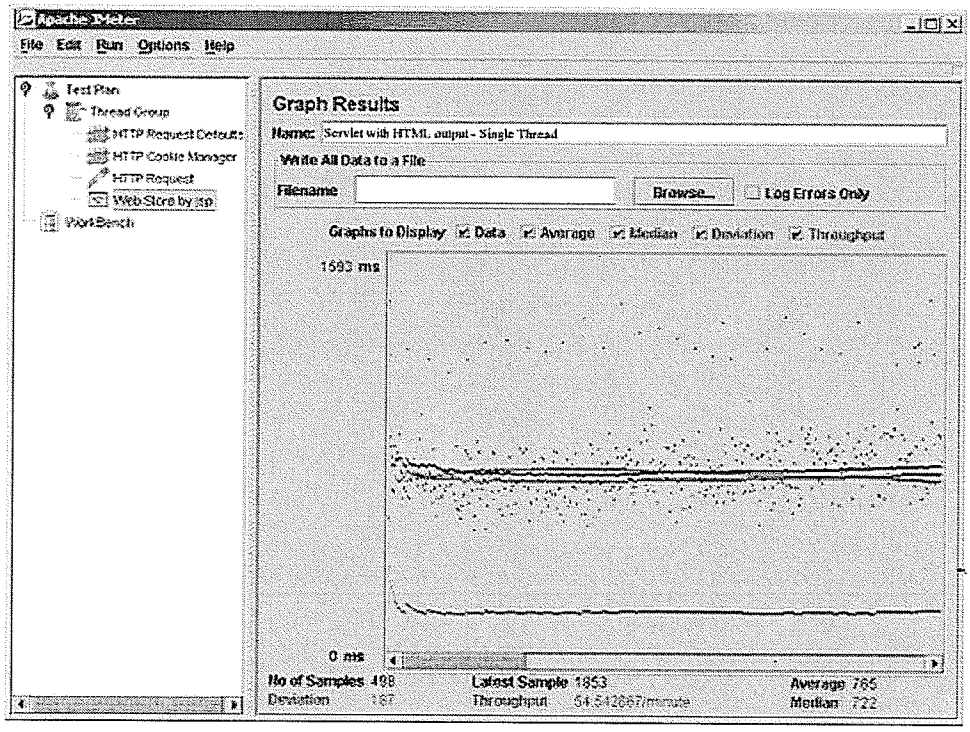

Figure 6.13 Web Request with Regular Servlet HTML Output, No Content Log (Single Thread Mode)

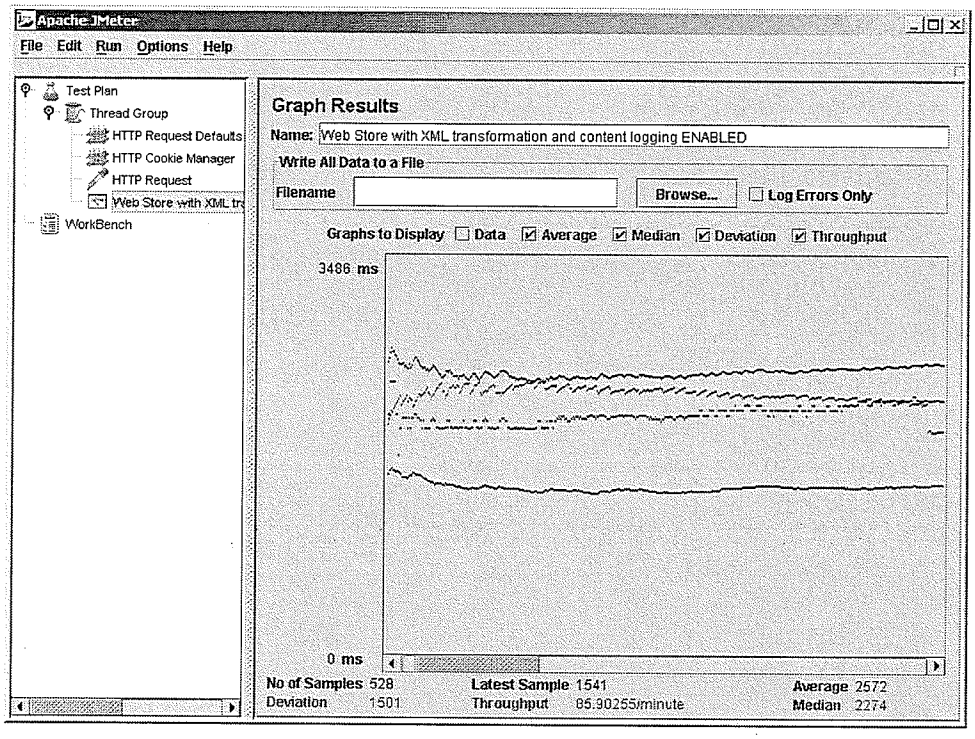

Figure 6.14 Web Request with XML & XSL Transformation Enabled, and Content Log Enabled (Five Concurrent Threads Mode)

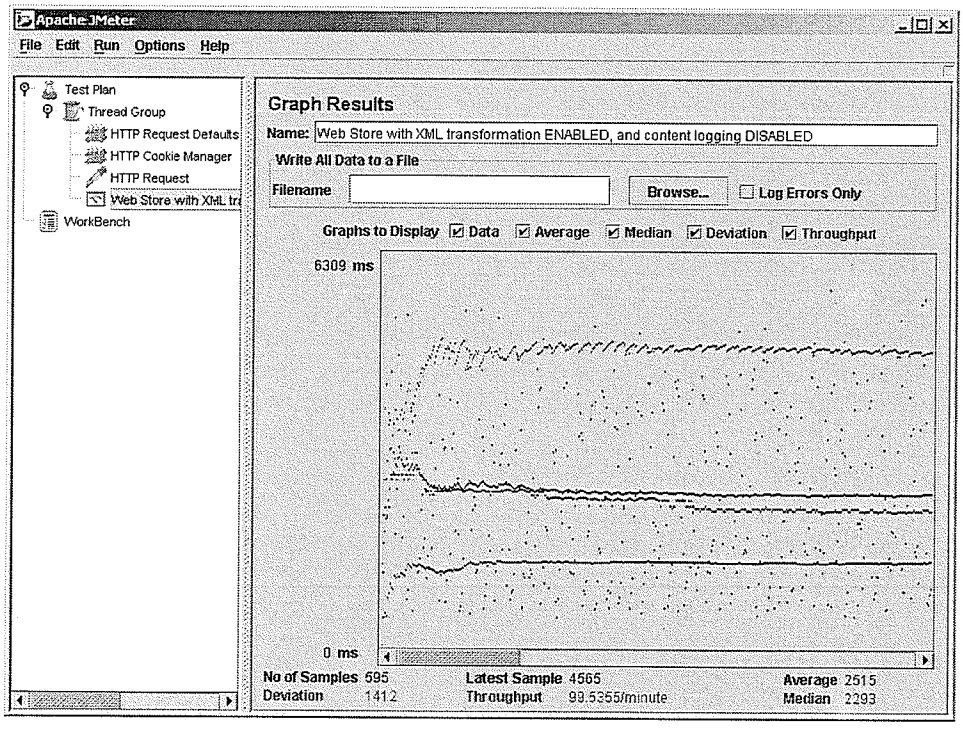

Figure 6.15 Web Request with XML & XSL Transformation Enabled, and Content Log Disabled (Five Concurrent Threads Mode)

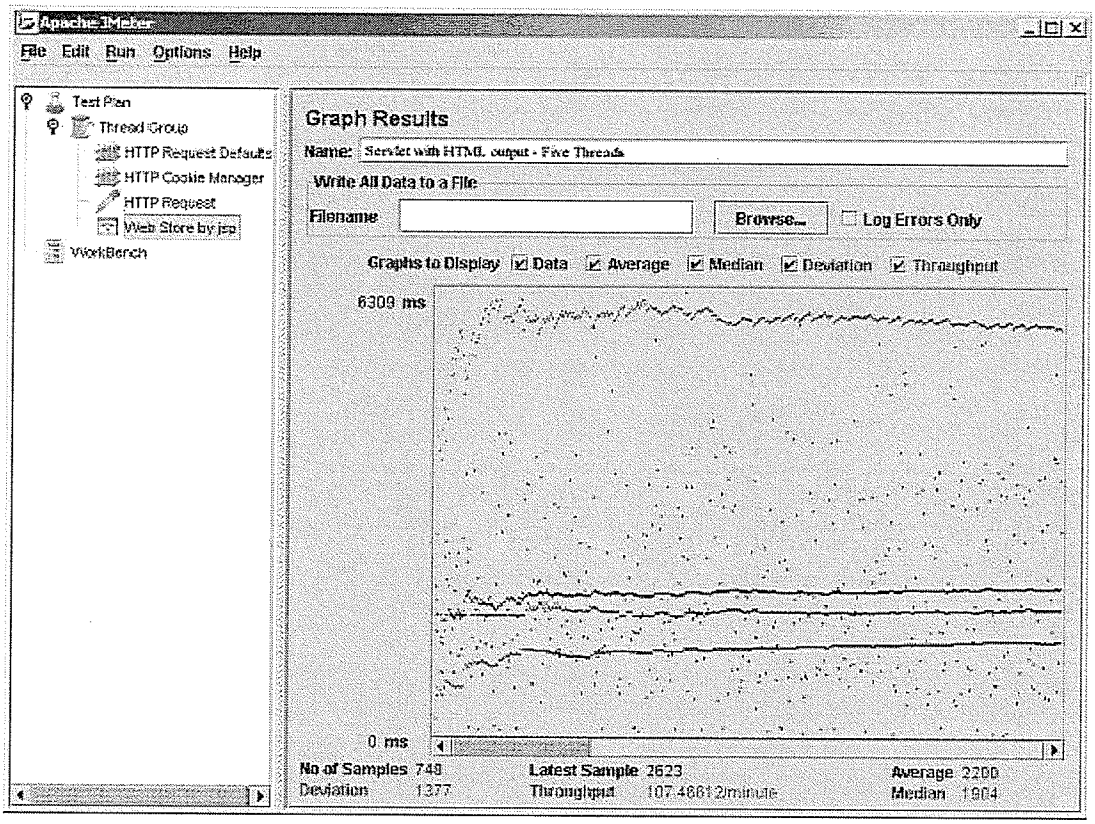

Figure 6.16 Web Request with Regular Servlet HTML Output, No content Log (Five Concurrent Threads Mode)

	Single Request (ms)	Five Concurrent Requests (ms)
XML transformation + Content logging	1137	2572
XML transformation only	1011	2515
HTML output only	765	2200

Table 6.1 Average Processing Time Comparison Chart

As shown in Table 6.1, the overhead of XML transformation and content logging in the single request test can be 48% over the performance of a simple Servlet with HTML output. But that difference quickly drops to 16.9% in a five concurrent request test.

6.3 Mining Web Content Object

Two preliminary Web content object mining experiments, a shopping cart clustering experiment and a Web object sequential pattern mining experiment, were conducted on the Web content log generated by simulated requests from the Jakarta JMeter Web traffic simulation tool. Using XML technologies, the original content can easily be extracted, transformed, and loaded for data mining.

6.3.1 Web Traffic Simulation

To generate a workload representing a real world scenario, two characteristic types of users of an e-commerce Web store are defined in the simulation. They are:

- Browsing User: A browsing user enters the Web store and browses through catalogue series. This simulates a real user who is only checking offerings.
- Order User: An order user enters the Web store, browses products, adds products to the shopping cart, logs in as a registered user, reviews the order, and checks out the products in the shopping cart.

Ten visiting thread groups for simulation are set up in the Jakarta Jmeter simulation plan. Each visit thread group uses a Gaussian Random Timer to generate Gaussian distributed random amount of time for the time intervals between requests. Table 6.2 lists the descriptions of these simulation threads.

Figure 6.16 shows sample content in the Web content log file.

Thread Group	Description	Mean (ms)	Deviation (ms)
1	Browse products across series randomly	300	300
2	Buy a product with sku: 3010-00005640	1800	500
3	Buy some products (product set 1) in 3030 series	1000	500
4	Browse 3000 series products	700	200
5	Browse 3010 series products	800	400
6	Browse 3020 series products	800	400
7	Browse 3030 series products	500	200
8	Buy some products (product set 2) in 3030 series	1200	400
9	Buy some products (product set 3) in 3030 series	1800	500
10	Buy some products in 3000 series	1600	600

Table 6.2 Web Traffic Simulation Threads

6.3.2 Shopping Cart Clustering

To demonstrate the ability of mining granular information on Web pages, the shopping cart in the checkout page is analyzed. This experiment finds direct relationship between the product unit price and purchased quantity.

The Web content log contains product number, quantity, unit price, and grand total price of each customer's purchases. With an XML enabled DB2 database, a shopping cart DB2 document access definition (DAD) file helps parse the logged shopping cart content in XML format and builds a relational database view of the shopping cart items. Each row in this database view represents one item in a shopping cart. Each row contains a session id, a product number together with purchased quantity, unit price, and grand total price as shown in Figure 6.17. Complex XML content parsing and data extraction turn into very

Figure 6.17 Sample Web Content Log

simple SQL query to minimize the data extraction, transformation, and loading (ETL) effort.

Cluster	1	2	3	4	5
Mean of Unit Price	649	263.854	219	120.729	133.5844
Std. of Unit Price	0.0001	179.028	80	82.1117	80.4477
Mean of Quantity	4	1	4	2.4063	2.4483
Std. of Quantity	0	0	0	0.4911	0.4973
Probability	0.0528	0.7009	0.1877	0.0023	0.0545
Support	5%	70%	19%	0%	6%

Table 6.3 Unit Price and Purchasing Quantity Clusters

The EM cluster mining program randomly splits the logged shopping cart data into ten folders in approximately equal size and randomly selects one of these folders as training data and the rest data as the testing data. The cluster mining result shown in Table 6.3 reveals some interesting purchasing behaviors, such as a group of customers usually purchases four units for product priced at 649. There are about 5% such transactions among all transactions.

Cluster	1	2	3	4	5
Mean of Quantity	2	4	3	1	1
Std of Quantity	0	0	0	0	0
Product ID	3010-00005640	3030-315	3010-00005640		3010-00005640
	3010-01640	3030-324	3010-01640		3030-030
	3030-00640	3030-360	3030-00640		3030-090
	3070-030				3030-315
					3030-324
					3030-360
Probability	0.0323	0.2405	0.0264	0.2312	0.4697
Support	3%	24%	3%	0%	70%

Table 6.4 Product ID and Purchasing Quantity Clusters

Table 6.4 shows discovered clusters of product id and purchased quantity. Cluster 2 is a cluster of products likely to be purchased four in quantity. Among three products in cluster 2, product 3030-360 has purchase price of 649.

6.3.3 Web Object Sequential Pattern Mining

Web object sequential pattern mining experiment demonstrates the ability of mining relationships in a sub-Web page level. The granular information tells exactly what kind of products visitors are really interested and willing to purchase.

A generic product DAD file is used to extract product object from XML formatted page content to build a product page database view as shown in Figure 6.18. To make the sequential mining program run more efficiently, a Web object statistic program uses database queries to build a Web object statistic table, which holds URI_ID, Web Object name, Web Object ID, and occurrence information, as shown in Figure 6.19 to eliminate low occurring Web objects from the sequential mining. The original content log is transformed into a frequent Web object database table.

Table 6.5 shows the top ten supported Web object sequential patterns discovered using prefixSpan algorithm with minimal sequential pattern length of three.

Sample Contents - WEBOBJLOG

J76M021 - DB2 - DM - JBZ10.WEBOBJLOG

SESSIONID	URI_ID	SEQ	WEBOBJ_ID
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	24
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	25
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	26
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	27
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	28
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	29
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	30
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	31
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	32
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	33
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	34
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	35
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	36
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	37
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	38
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	39
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	40
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	41
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	42
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	0	43
E5FE5D6C58B0C85FB12DE55E87FBBEF4	1	1	12
E5FE5D6C58B0C85FB12DE55E87FBBEF4	2	2	2
E5FE5D6C58B0C85FB12DE55E87FBBEF4	2	2	1
E5FE5D6C58B0C85FB12DE55E87FBBEF4	3	3	1
E5FE5D6C58B0C85FB12DE55E87FBBEF4	3	3	2
E5FE5D6C58B0C85FB12DE55E87FBBEF4	4	4	1
E5FE5D6C58B0C85FB12DE55E87FBBEF4	4	4	2
E9300C4B97CFABC4A67758B4B3AEB652	1	0	8
E9300C4B97CFABC4A67758B4B3AEB652	1	1	5
EAAE444E3C4803A3645938A58926EF11	1	0	68
EAAE444E3C4803A3645938A58926EF11	1	0	69
EAAE444E3C4803A3645938A58926EF11	1	0	70
EAAE444E3C4803A3645938A58926EF11	1	0	71

Next Rows in memory 50 [1801 - 1850] Filter Close Help

Figure 6.18 Transformed Web Object Log

Sample Contents - WEBOBJ_STATS

J76MD21 - DB2 - DM - JBZ10.WEBOBJ_STATS

URI_ID	WEBOBJ	WEBOBJ_ID	WEBOBJ_COUNT
1	<Product ID="3010-0000564...	5	92
1	<Product ID="3040">	72	20
1	<Product ID="3070-591-0">	103	2
1	<Product ID="3050">	73	20
1	<Product ID="3010-0000502...	1	50
2	<Product ID="3070-030">	8	2
2	<Product ID="3020-020-1">	10	2
2	<Product ID="3030-315">	2	13
2	<Product ID="3050-101">	11	1
2	<Product ID="3030-090">	6	6
2	<Product ID="3010-0000564...	4	8
2	<Product ID="3000-1-03">	7	1
2	<Product ID="3010-0000502...	9	3
2	<Product ID="3030-324">	1	20
2	<Product ID="3030-360">	3	4
2	<Product ID="3030-030">	5	11
3	<Product ID="3070-030">	8	2
3	<Product ID="3020-020-1">	11	6
3	<Product ID="3030-315">	1	9
3	<Product ID="3050-101">	10	6
3	<Product ID="3030-090">	6	6
3	<Product ID="3010-0000564...	4	10
3	<Product ID="3000-1-03">	7	1
3	<Product ID="3010-0000502...	9	6
3	<Product ID="3030-324">	2	9
3	<Product ID="3030-360">	3	4
3	<Product ID="3030-030">	5	6
4	<Product ID="3070-030">	8	1

Next Rows in memory 50 [108 - 157] Filter Close Help

Figure 6.19 Web Object Table

```
------- Pattern 1 -------
{/myshop/servlet/Product:<Product ID="3010-00005640"> } (0.2834) ->
{/myshop/servlet/Buy:<Product ID="3010-00005640"> } (0.0472) ->
{/myshop/servlet/Update:<Product ID="3010-00005640"> } (0.0472) ->
{/myshop/servlet/Bill:<Product ID="3010-00005640"> } (0.0472)

------- Pattern 2 -------
{/myshop/servlet/Product:<Product ID="3010-00005640"> } (0.2834) ->
{/myshop/servlet/Buy:<Product ID="3010-00005640"> } (0.0472) ->
{/myshop/servlet/Bill:<Product ID="3010-00005640"> } (0.0472)

------- Pattern 3 -------
{/myshop/servlet/Product:<Product ID="3010-00005640"> } (0.2834) ->
{/myshop/servlet/Update:<Product ID="3010-00005640"> } (0.0472) ->
{/myshop/servlet/Bill:<Product ID="3010-00005640"> } (0.0472)

------- Pattern 4 -------
{/myshop/servlet/Product:<Product ID="3000-1-01"> } (0.1023) ->
{/myshop/servlet/Product:<Product ID="3000-1-01"> /myshop/servlet/Product:<Product
ID="3000-9-10"> } (0.0472) ->
{/myshop/servlet/Product:<Product ID="3000-9-10"> } (0.0393)

------- Pattern 5 -------
{/myshop/servlet/Product:<Product ID="3010-0005003B">
/myshop/servlet/Product:<Product ID="3000-1-01"> /myshop/servlet/Product:<Product
ID="3000-9-10"> } (0.1338) ->
{/myshop/servlet/Product:<Product ID="3000-1-01"> } (0.0787) ->
{/myshop/servlet/Product:<Product ID="3000-1-01"> /myshop/servlet/Product:<Product
ID="3000-9-10"> } (0.0393) ->
{/myshop/servlet/Product:<Product ID="3000-9-10"> } (0.0314)

------- Pattern 6 -------
{/myshop/servlet/Product:<Product ID="3010-0005003B">
/myshop/servlet/Product:<Product ID="3000-1-01"> /myshop/servlet/Product:<Product
ID="3000-9-10"> } (0.1338) ->
{/myshop/servlet/Product:<Product ID="3000-1-01"> } (0.0787) ->
{/myshop/servlet/Product:<Product ID="3000-9-10"> } (0.0472)

------- Pattern 7 -------
{/myshop/servlet/Product:<Product ID="3010-0005003B">
/myshop/servlet/Product:<Product ID="3000-1-01"> /myshop/servlet/Product:<Product
ID="3000-9-10"> } (0.1338) ->
{/myshop/servlet/Product:<Product ID="3000-1-01"> /myshop/servlet/Product:<Product
ID="3000-9-10"> } (0.0787) ->
{/myshop/servlet/Product:<Product ID="3000-9-10"> } (0.0472)
```

------- Pattern 8 -------
{/myshop/servlet/Product:<Product ID="3010-0005003B">
/myshop/servlet/Product:<Product ID="3010-00005021">
/myshop/servlet/Product:<Product ID="3020-505-1"> /myshop/servlet/Product:<Product ID="3010-0005003B"> /myshop/servlet/Product:<Product ID="3010-00005640">
/myshop/servlet/Product:<Product ID="3010-00005630"> } (0.2992) ->
{/myshop/servlet/Product:<Product ID="3010-00005640"> } (0.1811) ->
{/myshop/servlet/Buy:<Product ID="3010-00005640"> } (0.0472) ->
{/myshop/servlet/Update:<Product ID="3010-00005640"> } (0.0472) ->
{/myshop/servlet/Bill:<Product ID="3010-00005640"> } (0.0472)

------- Pattern 9 -------
{/myshop/servlet/Product:<Product ID="3010-0005003B">
/myshop/servlet/Product:<Product ID="3010-00005021">
/myshop/servlet/Product:<Product ID="3020-505-1"> /myshop/servlet/Product:<Product ID="3010-0005003B"> /myshop/servlet/Product:<Product ID="3010-00005640">
/myshop/servlet/Product:<Product ID="3010-00005630"> } (0.2992) ->
{/myshop/servlet/Product:<Product ID="3010-00005640"> } (0.1811) ->
{/myshop/servlet/Buy:<Product ID="3010-00005640"> } (0.0472) ->
{/myshop/servlet/Bill:<Product ID="3010-00005640"> } (0.0472)

------- Pattern 10 -------
{/myshop/servlet/Product:<Product ID="3010-0005003B">
/myshop/servlet/Product:<Product ID="3010-00005021">
/myshop/servlet/Product:<Product ID="3020-505-1"> /myshop/servlet/Product:<Product ID="3010-0005003B"> /myshop/servlet/Product:<Product ID="3010-00005640">
/myshop/servlet/Product:<Product ID="3010-00005630"> } (0.2992) ->
{/myshop/servlet/Product:<Product ID="3010-00005640"> } (0.1811) ->
{/myshop/servlet/Update:<Product ID="3010-00005640"> } (0.0472) ->
{/myshop/servlet/Bill:<Product ID="3010-00005640"> } (0.0472)

Table 6.5 Sequential Web Object Mining Results

Each Web object, for example, */myshop/servlet/Product:<Product ID="3010-00005640">*, has a URI and an object name separated by column. A pair of curly brackets { }, which contains one or more Web objects, represents a Web object set in a Web page. The number in the following parentheses is the support number of this Web object set. Table 6.6 illustrates these components of a Web object.

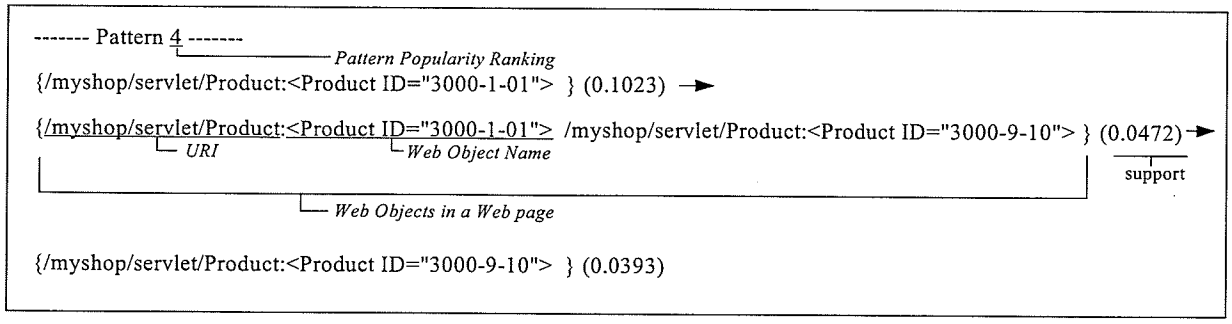

Table 6.6 A Web Object Sequential Pattern Sample

Manual inspection shows that the above frequent sequential patterns are among the product purchases and browsing threads with higher frequency in the simulation plan. Compared to Web page URI based sequential pattern mining, the result of this experiment is straightforward to business analysts and the data ETL is simple and can be implemented quickly.

The results of above experiments provide business analysts better understanding of customers' interests for potential opportunities to increase sales, lower costs, and efficiently manage supply chains. Web store owners can quickly and flexibly extract attributes of interest to them from the source for further analysis according to their business purposes.

6.4 Quality Assurance Strategies

Every software development and maintenance project requires some quality assurance activities. These activities are generally referred to as verification and validation process. "Software verification and validation is the process of ensuring that software being developed or changed will satisfy functional and other requirements (validation) and each step in the process of building the software yields the right products (verification)." [86]. Verification and validation activities cover all phases of software life cycle. A number of attributes determine the quality of a software system, however not all of these criteria are

applicable to every software developed. Some quality assurance attributes related to this application are explained blew:

In the software concept and initiation phase, the software requirements phase, and the software architectural design phase, the system uses matured object-oriented methodologies, such as UML and design patterns. Because design patterns are proven solutions, adopting appropriate design patterns guarantee the quality of the overall system design and success in later software detailed design and implementation phases.

In the detailed design phase, high level business objects were mapped to computer system objects presented by popular industry standards, such as XML and SQL. This approach ensures that the software developed incorporates underlying software infrastructures smoothly. This is extraordinarily important because this system is operating system independent and runs in a distributed heterogeneous environment. Verified UML diagrams guarantee the mapping between the specification and the implementation.

In the software implementation phase, the programming strictly follows industry standards to ensure the correctness on both client and server sides. On the client side, users can use any Web browser which supports W3C HTML standard via HTTP. On the server side, this application uses Java as the programming language following Java Standard Edition (J2SE) version 1.4 specification, and Java Servlet version 1.2 specification. This application uses JDBC as database connection and SQL 92 for database queries. This application also uses Java standard XML processing APIs, which is part of J2SE version 1.4, and strictly follows XML and XSL standards to build, parse, save, and transform XML data. Object-oriented design allows dynamic verification by comprehensive functional testing (black-box testing) from testing objects individually to testing integrated objects.

A thorough review was conducted after each stage is finished together with a careful plan for next stage. This helped discover errors and mistakes in early stages and plan

corrective actions. Together with the UML diagrams, the application self-documented source code in Javadoc standard assist the understanding of the application.

The nature of stateless Web communication, Web server managed processes and threads, and database connection pool provides redundancy system resources for the application to greatly improve the reliability of the overall system. The nature of stateless Web communication also ensures that a runtime error for a user at a given time does not affect other users. The Java Logging API used in the application provides a reliable mechanism to dynamically capture and log warnings and error messages at various levels and send to desired channels, such as a terminal, a file, a database table, or a TCP/IP port. Logged application warning and error messages are essential in debugging and monitoring during the software integration, test, delivery, and operation phases.

Finally, Jakarta Jmeter is used for comprehensive tests. The computer automated testing tool allows stress testing the Web application using a wide range of input parameters' values in random sequences. Detected defects were corrected and tests were repeated (regression testing) to account for the fixes.

Chapter 7

Conclusions and Future Work

Business intelligence solutions, such as OLAP and data mining, enable organizations to understand their customers and business management. Business intelligence applications rely on data warehouse to provide modeled data. Data warehousing reconciles data to the models and formats that OLAP and data mining applications require. The traditional Web log does not capture the content visitors request. Applying business intelligence applications on the current Web log data is unable to provide accurate in-depth analysis of visitors' interests because of lacking details in the source data.

In this thesis, a Web content logging system is designed and implemented. On top of this system, an object-oriented prototype of a Web store is developed, followed with data warehousing and data mining of the Web content usage data generated from requests using the prototype Web store. The Web content logging system provides Web administrators and business analysts the flexibility to capture Web site visitors' interests at a granular level. Any Web site designed in an object-oriented approach can benefit from this system to log interested objects' attributes. Moreover, the relationships among these attributes are also kept by using XML technology. The XML technology permits the unique designation of each attribute in the data mining phase. The data warehousing procedure is designed to reconcile logged content usage data into relational database without losing data hierarchy. Using tree structure to represent the hierarchic relationships among data gives full flexibility in choosing interesting attributes for data mining. The data mining techniques demonstrate the ability to drill down from summarized views and reports to give granularized Web content usage log analysis on

demand. Any system administrator with general knowledge of XML and XSL can manage the Web content logging system.

The performance of an object-oriented Web site has minimal difference from that developed with the traditional Web programming. The Web content logging module works in a per Web application basis (in term of Web Application Archive defined in Java Servlet specification version 2.2). Different Web applications can have different configurations for content logging even if all the Web applications are running on the same Web application server. Developers only need to have general knowledge of XML and XSL and they can benefit from this multi-tier architecture for easier development and maintenance. The XML and XSL tasks required can be accomplished easily with simplest XML tools.

In this thesis, the data warehousing and data mining procedures use an XML-enabled relational database. An XML-enabled relational database allows users to query, add, delete, and update XML documents, elements, and attributes through traditional SQL queries, while hiding the sophisticated indexing and SQL query functionality (which is not a part of the focus of this thesis) from the developers. In the real business world, the content and object XML mapping in the data warehousing procedure can be varied, and many other data mining algorithms can be chosen to satisfy different business purposes.

Running Web applications with the Web content usage module obviously has some overheads. The overheads include loading XSL files, XSL transformation for client responses, and XSL transformation for content logging. To minimize the overheads, the Web content usage logging module loads all XSL files for current Web applications into RAM to avoid lagging disk operations repeatedly. Other optimizations include adopting a faster XSL translator, and using simpler XSL with less levels in hierarchy and avoids XSL function calls in XSL document files. The overhead requires more CPU power. The five concurrent connection performance test in Chapter 6 shows there is about 17% extra processing time on a PC with single 366Mhz Intel Pentium Celeron CPU. Nowadays, on a real production server with multiple gigahertz CPU and much more L1 and L2 caches,

the overhead time will be much less noticeable. For Web applications, CPU power is unlikely to be the performance bottleneck. It is fair to say these overheads can be neglected.

This thesis presents an innovative way, the Web object content logging mechanism, for capturing interesting attributes and their relationships on the Web pages visitors requested. Saving captured attributes in structured format simplifies the data warehousing procedure for business intelligence applications while it provides great flexibility in choosing interesting attributes for different business analyse. This thesis also demonstrates how to adopt current data mining algorithms to discover business-oriented results from the granularized logged attributes.

The Web content usage system is able to capture numerous attributes together with relationships among them. However, there are some limitations of current data mining algorithms at this time:

- There are only few mature data mining algorithms that are capable to give analysis across a large number of attributes. A typical e-commerce Web site sells hundreds or even thousands of products. In this thesis, the logged attributes are first scanned to count the occurrences in the data preparation phase. Only attributes that have higher counts than a threshold are introduced to data mining algorithms. For most data mining algorithms, this approach does not affect the mining result but greatly improves the efficiency. However, for a company having many products, each product may have a number of related attributes, such as various costs, multiple inventories, and sales data, etc. The total number of interested attributes can still be very big.

- Most data mining algorithms assume attributes to be independent. So mining attributes with underlying dependency may lead to biased analysis results.

Data mining is a relatively new field in computer science. To serve business intelligence for e-commerce better, data mining algorithms need to be continuously developed. Some possible continuous works include:

- Adopt the Hypergraph partitioning data clustering algorithm [81, 82, 83] to discover clusters among large amount of attributes. Presenting items and frequent item sets by vertices and hyperedges respectively, the Hypergraph partition algorithm is able to discover frequent items sets from a hypergraph efficiently in a large number of items.
- Provide a graphic user interface (GUI) tool for business users to select interesting Web objects and attributes. This tool will save the selected Web objects and attributes in the predefined XML-based Web content logging configuration file.
- Implement a GUI presentation of data mining results. The data mining prototype programs presently run in command line mode and the results are shown in text. A graphical view will help users better understand the mining results.
- Investigate methodologies to apply additional data mining algorithms in Web object mining domain, such as decision trees, and neural networks, etc. Logged Web content usage data presented in this thesis provides a rich set of source data in XML format for easy data access by data mining algorithms. To serve for various business purposes, applying additional data mining algorithms to discover different types of patterns is always necessary in real world scenario.

In summary, the Web content usage logging system provides an extensible infrastructure to capture e-commerce Web site visitors' interests. The captured data is an excellent data source for further business intelligence analysis. The flexible structured data allows the logged data to be analyzed in diversified aspects at many granular levels.

References

[1] A. Luotonen, *The Common Logfile Format*, World Wide Web Consortium, 1995. http://www.w3.org/pub/WWW/Daemon/User/Config/Logging.html

[2] "Logging Control in W3C httpd", World Wide Web Consortium, 1995. http://www.w3.org/Daemon/User/Config/Logging.html

[3] "Extended Log File Format", World Wide Web Consortium, 1995. ,http://www.w3.org/TR/WD-logfile.html

[4] R. Cooley, B. Mobasher, and J.Srivastava, "Data Preparation for Mining World Wide Web Browsing Patterns", *Journal of Knowledge and Information Systems*, Volumn 1, No. 1, pp. 5-32, 1999.

[5] L. Shen, L. Cheng, and T. Steinberg, "Mining the Most Interesting Web Access Associations", *Proc. of the WebNet2000 World Conference on the WWW and Internet*, pp. 489-494, San Antonio, Texas, October 2000.

[6] S. Ansari, R. Kohavi, L. Mason, and Z. Zheng, "Integrating E-Commerce and Data Mining: Architecture and Challenges", *Workshop on Web Mining for E-Commerce*, pp. 49-60, Boston, MA, USA, August 2000.

[7] B. Mobasher, H. Dai, T. Luo, Y. Sun, and J. Zhu, "Integrating Web Usage and Content Mining for More Effective Personalization", *1st International Conference on Electronic Commerce and Web Technologies*, pp. 165-176, London-Greenwich, United Kingdom, September 2000.

[8] B. Meltzer and R. Glushko, "XML and Electronic Commerce: Enabling the Network Economy", *VEO Systems Technical Whitepaper*, 2000.

[9] Edd Dumbill, "The Role Played by XML in the Next-Generation Web", *XML.com,* September 2000, www.xml.com/pub/2000/09/06/distributed.html

[10] R. Hoque, *XML for Real Programmers*, Morgan Kaufmann, 2000.

[11] B. Marchal, *XML by Example*, QUE, 2000.

[12] S. Gomory, R Hoch, J. Lee, M. Podlaseck, E Schonberg, "E-Commerce Intelligence: Measuring, Analyzing, and Reporting on Merchandising Effectiveness of Online Stores", *IBM Research Report*, IBM T.J. Watson Research Center, November 2000.

[13] R. Kosala, and H. Blockeel, "Web Mining Research: A Survey", *ACMSIG on Knowledge Discovery and Data Mining Explorations,* Vol. 2, pp.1-15, July 2000.

[14] "Sample CommerceTrends Analysis", WebTrend Corporation, August 2000.

[15] "XML in HTML Meeting Report", W3C Note, May 1998. http://www.w3.org/TR/NOTE-xh

[16] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, 1999.

[17] M. Gaedke, H. Gellersen, A. Schmidt, U. Stegemuller, W. Kurr, "Object-oriented Web Engineering for Large-scale Web Service Management", *Thirty-second Annual Hawaii International Conference on System Sciences*, IEEE CD-ROM/Abstracts Proceedings, ISBN 0-7695-0001-3, Maui, Hawaii, January 1999.

[18] D. Schwabe and G. Rossi, "An Object Oriented Approach to Web-Based Application Design", *Theory and Practice of Object System*, Volumn 4, Issue 4, Wiley and Sons, New York, 1998.

[19] D. Schwabe, G. Rossi, L. Esmeraldo, F. Lyardet, "Engineering Web Applications for Reuse", *IEEE Multimedia,* pp. 2-12, Spring 2001.

[20] M. Bichler, A. Segev, and J. Zhao, "Component-based E-Commerce: Assessment of Current Practices and Future Directions", *ACM Special Interest Group on Management of Data,* Vol. 27, No. 4, pp. 7-14, December 1998.

[21] G. Rossi, F. Lyardet and D. Schwabe, "Patterns for E-Commerce Applications", *European Conference on Pattern Languages of Programs*, Irsee, Germany, July 2000.

[22] G. Menkhaus, "Architecture for Client-Independent Web Applications", *IEEE Proc. of the TOOLS-Europe Conference*, pp. 32-40, Zürich, 12-14 March 2001.

[23] J. Conallen, "Modeling Web Application Architectures with UML", *CACM Volume 42 Number 10*, October 1999.

[24] E. Althammer and W. Pree, "Design And Implementation of a MVC-Based Architecture for E-Commerce Applications", *International Journal of Computers and Applications*, Vol. 23, No.2, pp. 173-185, ACTA Press, Calgary, Canada, 2001.

[25] K. Ahmed, "Developing J2EE Applications with the UML and Rational Rose", *Rational Software White Paper*, Rational Software Corporation, 2001.

[26] P. Kruchten, "Architectural Blueprints – The 4+1 View Model of Software Architecture", *IEEE Software*, pp. 42-50, November 1995.

[27] "J2EE Design Patterns, Java BluePrints – Enterprise", *Java BluePrints*, Sun

Microsystems, Nov 2001.
http://java.sun.com/blueprints/patterns/j2ee_patterns/index.html

[28] V. Ramachandran, "Design Patterns for Building Flexible and Maintainable J2EETM Applications", *Technical Articles and Tips*, January 2002. http://developer.java.sun.com/developer/technicalArticles/J2EE/despat/

[29] B. Wampler, *The Essence of Object Oriented Programming with Java and UML*, Addison-Wesley, 2001.

[30] S. Abiteboul, "Querying Semi-Structured Data", *In Proc. of the International Conference on Data Base Theory*, pp.1-18, Springer-Varlag LNCS 1186, 1997

[31] S. A. Ehikioya, "Formal Specification of Content Logging for E-Commerce Servers", *Technical Report #99-05-04*, Sylvia Integrated Systems, Inc., Winnipeg, 1999.

[32] R. Cooley, *Web Usage Mining: Discovery and Application of Interesting Patterns from Web Data,* Ph.D. Thesis, Dept. of Computer Science, University of Minnesota, 2000.

[33] C. Shahabi, F. Banaei-Kashani and J. Faruque, "A Reliable, Efficient, and Scalable System for Web Usage Data Acquisition", *WebKDD'01 Workshop in conjunction with the ACM-SIGKDD 2001*, San Francisco, CA, August 2001.

[34] M. Zawitz, "Web Statistics — Measuring User Activity, An Analysis of Bureau of Justice Statistics (BJS) Website Usage Statistics", *U.S. Department of Justice Office of Justice Programs, Bureau of Justice Statistics*, May 1998.

[35] M. Koster, "A Standard for Robot Exclusion", *The Web Robots Pages*, 1994, http://www.robotstxt.org/wc/norobots.html

[36] J. Pitkow, H.Nielsen, "W3C Web Characterization Activity Terminology Sheet", *World Wide Web Consortium*, December 1998.

[37] B. Lavoie and H. Nielsen, "Web Characterization Terminology & Definitions Sheet W3C Working Draft", *World Wide Web Consortium*, May 1999, http://www.w3.org/1999/05/WCA-terms/

[38] L. Catledge and J. Pitkow, "Characterizing Browsing Behaviors on the World Wide Web", *Computer Networks and ISDN System,* Vol 27, No 6, pp. 1065-1073, 1996.

[39] A. Mann and K. Hecht, "DevEdge Newsgroup FAQ: Client Technical", *DevEdge Champions for the Client Technical Newsgroup*, December 1998, http://developer.netscape.com/support/faqs/champions/client_tech.html

[40] "HOWTO: Prevent Caching in Internet Explorer (Q234067)", *Microsoft Knowledge Base Article*, 2003, http://support.microsoft.com/support/kb/articles/Q234/0/67.asp

[41] "Apache Tomcat Homepage", Apache Software Foundation, 2001, http://jakarta.apache.org/tomcat/index.html

[42] "Java™ Servlet Technology – The Power Behind the Server", *Java Technology*, Sun Microsystem, 2001, http://java.sun.com/products/servlet/index.html

[43] "JavaServer Pages™ - Dynamically Generated Web Content", *Java Technology*, Sun Microsystem, 2001, http://java.sun.com/products/jsp/

[44] "Tomcat - A Minimalistic User's Guide", Apache Software Foundation, 2000, http://jakarta.apache.org/tomcat/tomcat-3.2-doc/uguide/tomcat_ug.html

[45] G. Shachor , "Tomcat Workers How-to", *Tomcat Documentation*, 2000, http://jakarta.apache.org/tomcat/tomcat-3.2-doc/Tomcat-Workers-HowTo.html

[46] K. Wu and P. Yu, "Report on Second International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems", *IEEE Second International Workshop on Advanced issues of E-Commerce and Web-Based Information Systems*, pp. 19-23, Milpitas, California, June 2000.

[47] W. Chang and S. Yuan, "A Synthesized Learning Approach for Web-Based CRM", *WEBKDD Workshop on Web Mining for E-Commerce*, pp. 43-59, Boston, MA, August 2000.

[48] T. Flanagan and E. Safdie, "A Practical Guide to Implementing Enterprise Customer Management", *TechGuide White Papers,* techguide.com, 1998.

[49] J. Ryan, "Achieving Business Success Through Customer Relationship Management", *TechGuide White Papers,* techguide.com, 1999.

[50] S. Adelman and L. Moss, *Data Warehousing Project Management*, Addison-Wesley, 2000.

[51] B. Devlin, *Data Warehouse: From Architecture to Implementation*, Addison Wesley Longman, Inc., 1997.

[52] J. Srivastava, R. Cooley, M. Deshpande, P. Tan, "Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data", *ACM Special Interest Group on Knowledge Discovery in Data and Data Mining Explorations*, Vol 1, Issue 2, pp.12-23, January 2000.

[53] A. Siepel, A.Tolopko, A. Farmer, P. Steadman, F. Schilkey, B. Perry, and W. Beavis, "An integration platform for heterogeneous bioinformatics software components", *IBM Systems Journal*, Vol. 40, No. 2, pp. 570-591, 2001.

[54] C. Thusinger, and K. Huber, "Analyzing the footsteps of your customers", *WEBKDD Workshop on Web Mining for E-Commerce*, Boston, MA, USA, August 2000.

[55] "E-Commerce Business Data Analysis", *Microsoft TechNet*, July 2000.

[56] J. Pei, J.Han, B.Mortazavi-asl and H. Zhu, "Mining Access Patterns Efficiently from Web Logs", *Proc. 2000 Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'00)*, pp. 396-407, Kyoto, Japan, April 2000.

[57] "ScienceDaily.com Web Traffic Report by Webtrends", *ScienceDaily Magazine*, March 2001, http://www.sciencedaily.com/traffic/2001/March/Complete.htm

[58] J. Friedman, "Data Mining and Statistics: What's the Connection", *29th Symposium on the Interface: Computing Science and Statistics*, Houston, Texas, 1997.

[59] *IBM DB2 Universal Database XML Extender Administration and Programming Version 7*, IBM Corp., 2000.

[60] L. Ennser, C. Delporte, M. Oba and K. Sunil, *Integrating XML with DB2 XML extender and DB2 Text Extender First Edition*, IBM Redbooks, December 2000.

[61] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules", *Proc. of the VLDB Conference*, pp. 487-499, Santiago, Chile, 1994.

[62] R. Agrawal and R. Srikant, "Mining Sequential Patterns", *Proc. of the 11th International Conference on Data Engineering,* pp. 3-14, IEEE Computer Society Press, Taipei, Taiwan, 1995.

[63] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules Between Sets of Items in Large Database", *Proc. of the ACM SIGMOD Conference on Management of Data,* pp. *207-216*, Washington, D.C., May 1993.

[64] R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalization and Performance Improvements", *Proc. 5th International Conference in Extending Database Technology,* vol 1057, pp. 3-17, Springer, Avignon, France, 1996.

[65] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U Dayal, and M. Hsu, "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth", *Proc. of the International Conference on Data Engineering (ICDE'01)*, pp. 215-224, Heidelberg, Germany, April 2001.

[66] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal, "Multi-dimensional Sequential Pattern Mining", *Proc. of the 10th ACM International Conference on Information and Knowledge Management (CIKM'01)*, pp. 81-88, Atlanta, Georgia, November 2001.

[67] B. Mobasher, N. Jain, E. Han and J. Srivastava, "WEBMINER: Pattern Discovery from World Wide Web Transactions", *Technical Report TR96-050*, Department of Computer Science, University of Minnesota, USA, February 1996.

[68] B. Mobasher, R. Cooley and J. Srivastava, "Grouping Web Page References into Transactions for Mining World Wide Web Browsing Patterns", *Proc. of the 1997 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX-97),* Newport Beach, CA, November 1997.

[69] B. Mobasher, R. Cooley and J. Srivastava, "Web Mining: Information and Pattern Discovery on the World Wide Web", *Proc. of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97),* pp. 558-567, Newport Beach, CA, November 1997.

[70] F. Bonchi, F. Giannotti, C. Gozzi, G. Manco, M. Nanni, D. Pedreschi, C. Renso and S. Ruggieri, "Web Log Data Warehousing and Mining for Intelligent Web Caching", *Data & Knowledge Engineering,* Volume 39, Issue 2, pp. 165-189, Elsevier Science B.V., November 2001.

[71] J. Moore, E. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, and B. Mobasher, "Web Page Categorization and Feature Selection Using Association Rule and Principal Component Clustering", *7th Workshop on Information Technologies and Systems*, Vol. 27, No. 3, pp.329-341, Atlanta, GA, December 1997.

[72] M. Hernandez and S. Stolfo, "Real-World Data is Dirty: Data Cleansing and The Merge/Purge Problem", *Journal of Data Mining and Knowledge Discovery*, Vol. 2, Issue 1, pp. 9-37, 1998.

[73] C. Kamath, E. Cantu-Paz, "On the Design of a Parallel Object-Oriented Data Mining Toolkit", *SIGKDD Workshop on Distributed and Parallel Knowledge Discovery*, Boston, MA, 2000.

[74] D. Hand, "Statistics and Data Mining: Intersecting Disciplines", *SIGKDD Explorations*, Volume 1, Issue 1, pp. 16-19, June 1999.

[75] D. Hand, "Data Mining: Statistics and More?" *The American Statistician*, vol 52, pp.112-118, 1998.

[76] A. Dempster, N. Larid, and D. Rubin, "Maximum-Likelihood from Incomplete

Data via the EM Algorithm", *J. Royal Statist. Soc. Ser. B.*, 39, pp.1-38, 1977.

[77] J. Bilmes, "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models", *Technical Report ICSI-TR-97-021,* U.C. Berkeley, 1997.

[78] N. Vlassis, and A. Likas, "A Kurtosis-Based Dynamic Approach to Gaussian Mixture Modeling", *IEEE Transactions on Systems, Man and Cybernetics: Part A*, vol. 29, no. 4, pp. 393-399, July 1999.

[79] Q. Jackson and D. Landgrebe, "An Adaptive Classifier Design for High-Dimensional Data Analysis with a Limited Training Data Set," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 39, No. 12, pp. 2664-2679, December 2001.

[80] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, 2000.

[81] G. Karypis, "Multilevel Hypergraph Partition", *IEEE Transactions on VLSI Systems*, Vol. 7, No. 1, pp.526-529, 1999.

[82] M. Ozdal and C. Aykanat, "Hypergraph Models and Algorithms for Data-Pattern Based Clustering", *The Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, California, August 2001.

[83] E. Han, and G. Karypis, V. Kumar, and B. Mobasher, "Clustering Based On Association Rule Hypergraphs", *Proc. of the Workshop on Research Issues on Data Mining and Knowledge Discovery*, Vol. 21, Issue 1, pp. 15-22, Tucson, Arizona, 1997.

[84] Sing Li, "Filtering Tricks for your Tomcat: The Addition of Filtering to the Servlet 2.3 Spec Offers Enhanced Performance for your J2EE Apps", *IBM DeveloperWorks Technical Article*, June 2001.

[85] Tomcat Internals - The Design of Tomcat 3.x and the Reasons Behind It, http://jakarta.apache.org/tomcat/tomcat-3.3-doc/internal.html, 2000.

[86] *Software Assurance Guidebook*, NASA Software Assurance Technology Center, 1989, http://satc.gsfc.nasa.gov/assure/agb.txt

[87] *Software Assurance Standard*, NASA Software Assurance Technology Center, 1989, http://satc.gsfc.nasa.gov/assure/astd.txt

[88] A. O. Allen, *Probability, Statistics, and Queuing Theory with Computer Science Applications 2nd Edition*, Academic Press, New York, NY, 1990.