Finding and Evaluating Patterns in Web Repository Using Database Technology and Data Mining Algorithms

121008

By

Belgin ÖZAKAR

A Dissertation Submitted to the
Graduate School in Partial Fulfillment of the
Requirement for the Degree of

MASTER OF SCIENCE

Department : Computer Engineering
Major : Computer Software

İzmir Institute of Technology
İzmir, Turkey

121008

June 2002

Date of Signature

[illegible]

.... **25.06.2002**

Prof. Dr. Halis PÜSKÜLCÜ

Supervisor

Department of Computer Engineering

[illegible]

.... **25.06.2002**

Prof. Dr. Sıtkı AYTAÇ

Department of Computer Engineering

[illegible]

.... **25.06.2002**

Prof. Dr. Şaban EREN

Department of Computer Engineering

Egean University

[illegible]

.... **25.06.2002**

Prof. Dr. Sıtkı AYTAÇ

Head of Department

ACKNOWLEDGEMENTS

I would like to express my thanks to my advisor, Prof. Dr. Halis PÜSKÜLCÜ for the suggestions and support he provided during this study.

I would also like to thank the people of the Department of Computer Engineering of İzmir Institute of Technology whose friendship and support made my return to academic world after all those years, possible.

Finally, I thank my family for their love and patience.

ABSTRACT

Web mining is a very hot research topic, which combines two of the active research areas: Data Mining and World Wide Web. The Web mining research relates to several research communities such as Database, Statistics, Artificial Intelligence and Visualization. Although there exists some confusion about the Web mining, the most recognized approach is to categorize Web mining into three areas: Web content mining, Web structure mining, and Web usage mining. Web content mining focuses on the discovery/retrieval of the useful information from the Web contents/data/documents, while the Web structure mining emphasizes to the discovery of how to model the underlying link structures of the Web. Sometimes the distinction between these two categories is not very clear. Web usage mining is relatively independent, but not isolated category, in which the following studies continue; General Web Usage Mining, Site Modification, Systems Improvement and Personalization. General Web Usage Mining systems aim to discover general trends and patterns from the log files by adapting data mining techniques. The objective of the Site Modification systems is to improve the design of a web site by suggesting modifications in its content and structure. The research on System Improvement focuses on using the web usage mining for improving the web traffic. Finally, personalization systems aim to understand individual trends used for personalizing the web sites.

The study subject to this thesis, IYTE Web Usage Mining (WUM) System was an example of system development in the field of General Web Usage Mining with a database approach where the flexible query capability of SQL (Structured Query Language) was explored. The data mining and database techniques were applied on the access/error/user logs of the web server of Izmir Institute of Technology.

The main objective was to create a site improvement tool for the web administrator by reporting the distribution of the hits received by the web server according to the time stamp, users, service and URL types and at the same time revealing the nature of the errors generated by the web server. All data cleaning and transaction identification processes were handled by the software routines coded in Java. Clean transactions were imported into

IYTE Web Usage Mining (IYTE WUM) relational database. Flexible features of SQL were utilized for application of algorithm Apriori to discover most frequent pair of URL's visited, in addition to extraction of general knowledge from data.

ÖZ

Web madenciliği son zamanlarda çok yaygın olarak kullanılan Veri Madenciliği ve World Wide Web'i birleştiren bir araştırma alanıdır. Web madenciliği konusundaki araştırmalar Veritabanı, İstatistik, Yapay Zeka ve Görsellik gibi araştırma ekiplerinin ilgisini çekmektedir. Web madenciliği konusu henüz çok yeni olduğundan bazı kavramlar netlik kazanamamıştır. Ancak kabul gören bir yaklaşım bu konudaki çalışmaları üç ana başlık altında toplar: Web içerik madenciliği, Web yapı madenciliği ve Web kullanım madenciliği.Web içerik madenciliği web içinde bulunan veri ve dökümanlardan faydalı bilgi bulmaya/çekmeye odaklanırken, Web yapı madenciliği bağlantı yapılarını bulmaya ve bunları modellemeye çalışır. Bu iki kategori arasındaki ayrım bazen çok net değildir. Web kullanım madenciliği göreceli olarak daha bağımsız bir alan olup bu konuda şu çalışmalar sürdürülmektedir: Genel Web Kullanım Madenciliği, Site Modifikasyonu, Sistem İyileştirme ve Kişiselleştirme. Genel Web Kullanım Madenciliği log dosyalarına veri madenciliği tekniklerini uygulayarak genel akımları ve paternleri bulmayı hedefler. Site Modifikasyonu sistemleri sitelerin içerik ve yapısında iyileştirmeler önererek site tasarımını iyileştirmeye çalışır. Sistem İyileştirme, web trafiğini iyileştirmek için web madenciliği yapar. Son olarak, kişiselleştirme sistemleri kullanıcılarının tercihlerini anlamayı ve web sitelerini kişisel hale getirmeyi amaçlar.

Bu teze konu çalışma, İYTE Web Kullanım Madenciliği Sistemi; Genel Web Kullanım Madenciliği alanında esnek SQL (Yapısal Sorgulama Dili) sorgulama yeteneklerini kullanan veritabanı yaklaşımına bir örnek teşkil etmektedir. İzmir Yüksek Teknoloji Enstitüsü web sitesinin erişim/hata/kullanıcı log dosyaları üzerinde veri madenciliği ve veritabanı teknikleri uygulanmıştır.

Web yöneticisi için, web sunucusuna gelen taleplerin zamana, kullanıcılara, servis ve URL tiplerine gore dağılımlarını, web server tarafından oluşturulan hataların dağılımları ile birlikte bildiren, web sitesi iyileştirmesinde kullanabileceği bir araç oluşturmak hedeflenmiştir. Tüm veri temizleme ve kayıt tanıma işlemleri Java'da kodlanmış yazılımlar ile yapılmıştır. Temiz kayıtlar İYTE Web Kullanım Madenciliği ilişkisel veri tabanına aktarılmıştır. Veriden bilgi elde etmeye dönük sorgulamalara ek olarak SQL'in esnek özelliklerini kullanarak Apriori algoritması uygulanmış ve en sık ziyaret edilen URL çiftleri bulunmuştur.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1. Motivation

It is not an exaggeration to say the World Wide Web has the most profound impact on the human society in the last 10 years. It has changed the way of doing business, providing and receiving education, managing the organization etc. The most direct effect was the completed change of information collection, conveying, and exchange. Today, Web has turned to be the largest information source available in this planet.

The Web is a huge, explosive, diverse, dynamic and mostly unstructured data repository, which supplies incredible amount of information, and also raises the complexity of how to deal with the information from the different perspectives of view -- users, Web service providers, business analysts. The users want to have the effective search tools to find relevant information easily and precisely. In Figure 1.1. there are some statistics taken from http://www.searchengineshowdown.com/stats/ about popular search engines, which can give idea about the size of the links reachable through search engines. The second and third columns of the table show the freshness range of the data in the search engines. The fourth column indicates the number of URLs (in millions) in the database of the search engines.

The Web service providers want to find the way to predict the users' behaviors and personalize information to reduce the traffic load and design the Web site suited for the different group of users. The business analysts want to have tools to learn the users/consumers' needs. All of them are expecting tools or techniques to help them satisfy their demands and/or solve the problems encountered on the Web. Therefore, Web mining becomes an active and popular research field.

	Newest Page Found	Oldest Page Found	Showdown Estimate (millions)	Claim (millions)
Google	1 day	68 days	968	1,500
MSN (Inktomi)	1 day	80 days	292	500
HotBot (Ink.)	1 day	136 days	332	500
AltaVista	12 days	51 days	397	500
AllTheWeb	16 days	191 days	580	507
WiseNut	247 days	286 days	579	1,500

Figure 1.1. Statistics About Popular Search Engines (as of May, 2002)

1.2. Objective

By the application developed in this study we aim to have an example of general web usage mining exploring the capabilities of Standard Query Language (SQL). The main objective of the IYTE Web Usage Mining System is being a non-trivial application, which can be used by the web administrator to clarify the distribution of the hits according to the time stamp, users, service and URL types. The system can also reveal the nature of the errors generated by the web server. By the application of an association rule algorithm on the database the value of the knowledge is planned to be carried one step further since the relation between the hits can be analyzed. Web administrator can use the system developed in this study to understand general trends about his/her site and to modify the site to meet the changing requirements of the users. Some examples of query results are also presented in the study but the meaning of information extracted has not been discussed since the primary goal of the study was to show the possibility of importing database (mainly SQL) flexibility into the web usage mining systems.

Two data logs; access log and error log of IYTE web server together with user data are the inputs of this system. All data cleaning and transaction identification processes were handled by the software routines coded in Java. Clean transactions were imported into IYTE Web Usage Mining (IYTE WUM) relational database generated on Mysql. Both Java and Mysql are chosen for their freeware and portable nature. It was then possible to design and submit queries to describe the data and further create an environment favorable for application of predictive data mining algorithms. One of such algorithms for finding associative rules will be applied on a small portion of the database in order to demonstrate the outcomes.

1.3. Scope and Structure of the Study

According to Etzioni [1] ***web mining is the term of applying data mining techniques to automatically discover and extract useful information from the World Wide Web documents and services***. In Chapter 2 all the attributes of web mining together with the taxonomy is discussed. Although Web mining puts down the roots deeply in data mining, it is not equivalent to data mining. In Chapter 3 a brief description of data mining techniques are given. The unstructured feature of Web data triggers more complexity of Web mining. According to Kosala and Blockeel [2] web mining research is actually a converging area from several research communities, such as Database, Information Retrieval, Artificial Intelligence, and also Psychology and Statistics as well.

The application developed and explained in detail in Chapter 4 of this thesis is an example of web usage mining exploring the capabilities of standard query language (SQL). In this chapter system architecture, process and data flow, SQL DDL commands, SQL DML commands, query results, fragments of log files and tables can be found in order to make all the system easy to understand.

In Chapter 5 the experience gained from the development of IYTE Web Usage Mining (WUM) System is discussed. The main findings and benefits of the system together with suggestions for future work to enhance the system will take place in this last chapter.

Chapter 2

BACKGROUND OF WEB MINING

2.1. Overview

As many believe, it is Etzioni first proposed the term of Web mining in his paper [1] in 1996. In this paper, he claimed the ***Web mining is the use of data mining techniques to automatically discover and extract information from World Wide Web documents and services***. Many of the following researchers cited this explanation in their works. In the same paper, Etzioni came up with the question: Whether effective Web mining is feasible in practice? Today, with the tremendous growth of the data sources available on the Web and the dramatic popularity of e-commerce in the business community, Web mining has become the focus of quite a few research projects, papers and some commercial products.

In Etzioni [1] and Kosala and Blockeel [2], they suggested a similar way to decompose Web mining into the following subtasks:

a. Resource Discovery: the task of retrieving the intended information from Web.

b. Information Extraction: automatically selecting and preprocessing specific information from the retrieved Web resources.

c. Generalization: automatically discovers general patters at the both individual Web sites and across multiple sites.

d. Analysis: analyzing the mined pattern.

In brief, Web mining is a technique to discover and analyze the useful information from the Web data. Madria et al. [3] claim the Web involves three types of data: data on the Web (content), Web log data (usage) and Web structure data. Cooley [4] classified the data type as content data, structure data, usage data, and user profile data. Spiliopoulou [5] categorized the Web mining into Web usage mining, Web text mining and user modeling mining; while today the most recognized categories of the Web data mining are Web content mining, Web structure mining, and Web usage mining according to Borges and

Levene [6] and Kosala and Blockeel [2]. It is clear that the classification as seen in Figure 2.1. is based on what type of Web data to mine.

2.2. Web Content Mining

Web content mining describes the automatic search of information resource available online, and involves mining web data contents. In the Web mining domain, Web content mining essentially is an analog of data mining techniques for relational databases, since it is possible to find similar types of knowledge from the unstructured data residing in Web documents. The Web document usually contains several types of data, such as text, image, audio, video, metadata and hyperlinks. Some of them are semi structured such as HTML documents, or a more structured data like the data in the tables or database generated HTML pages, but most of the data is unstructured text data. The unstructured characteristic of Web data force the Web content mining towards a more complicated approach.

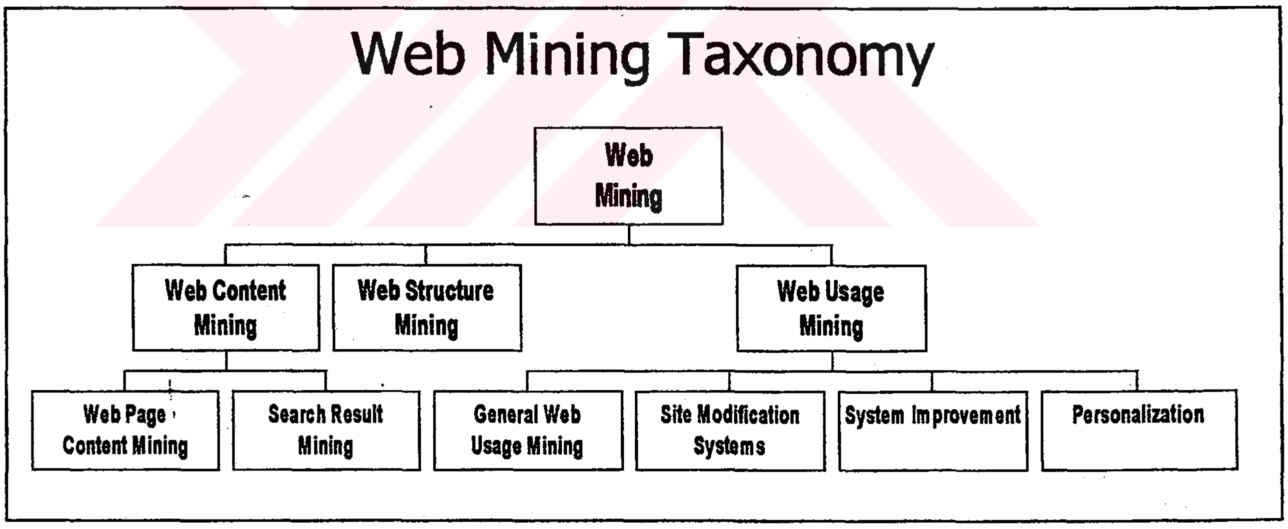


Figure 2.1. Web Mining Taxonomy

The Web content mining is differentiated from two different points of view, Cooley et al. [7]: Information Retrieval View and Database View. Kosala and Blockeel [2] summarized the research works done for unstructured data and semi structured data from information retrieval view. It shows that most of the researches use bag of words, which is based on the statistics about single words in isolation, to represent unstructured text and take single word found in the training corpus as features. For the semi-structured data, all the works use the HTML structures inside the documents and some use the hyperlink structure between the documents for document representation. As for the database view, in order to have the better information management and querying on the Web, the mining always tries to infer the structure of the Web site of to transform a Web site to become a database.

Chakrabarti [8] provides an in-depth survey of the research on the application of the techniques from machine learning, statistical pattern recognition, and data mining to analyzing hypertext. It is a good resource to be aware of the recent advances in content mining research.

Multimedia data mining is part of the content mining, which is engaged to mine the high-level information and knowledge from large online multimedia sources. Multimedia data mining on the Web has gained many researchers' attention recently. Working towards a unifying framework for representation, problem solving, and learning from multimedia is really a challenge, this research area is still in its infancy indeed, and many works are waiting to be done. The details about multimedia mining are given in Zaiane et al. [9].

2.3. Web Structure Mining

Most of the Web information retrieval tools only use the textual information, while ignoring the link information that could be very valuable. The goal of Web structure mining is to generate structural summary about the Web site and Web page. Technically, Web content mining mainly focuses on the structure of inner document, while Web structure mining tries to discover the link structure of the hyperlinks at the interdocument level. Based on the topology of the hyperlinks, Web structure mining will categorize the Web pages and generate the information, such as the similarity and relationship between different Web sites.

Web structure mining can also have another direction – discovering the structure of Web document itself. This type of structure mining can be used to reveal the structure (schema) of Web pages; this would be good for navigation purpose and make it possible to compare/integrate Web page schemes. This type of structure mining will facilitate introducing database techniques for accessing information in Web pages by providing a reference schema.

What is the structural information, and how to discover it? Madria et al. [3] gave a detailed description about how to discover interesting and informative facts describing the connectivity in the Web subset, based on the given collection of interconnected web documents. The structural information generated from the Web structure mining includes the follows: the information measuring the frequency of the local links in the Web tuples in a Web table; the information measuring the frequency of Web tuples in a Web table containing links that are interior and the links that are within the same document; the information measuring the frequency of Web tuples in a Web table that contains links that are global and the links that span different Web sites; the information measuring the frequency of identical Web tuples that appear in a Web table or among the Web tables.

In general, if a Web page is linked to another Web page directly, or the Web pages are neighbors, we would like to discover the relationships among those Web pages. The relations maybe fall in one of the types, such as they related by synonyms or ontology, they may have similar contents; both of them may sit in the same Web server therefore created by the same person. Another task of Web structure mining is to discover the nature of the hierarchy or network of hyperlinks in the Web sites of a particular domain. This may help to generalize the flow of information in Web sites that may represent some particular domain; therefore, the query processing will be easier and more efficient.

Web structure mining has a nature relation with the Web content mining, since it is very likely that the Web documents contain links, and they both use the real or primary data on the Web. It is quite often to combine these two mining tasks in an application.

2.4. Web Usage Mining

Web usage mining tries to discover the useful information from the secondary data derived from the interactions of the users while surfing on the Web. Web Usage mining studies can be classified under the headlines of. General Web Usage Mining, Site Modification, System Improvement and Personalization. General Web Usage Mining systems aim to discover general trends and patterns from the log files either by adapting well known data mining techniques or by proposing new data mining techniques. The objective of the Site Modification systems is to improve the design of a web site by suggesting modifications in its content and structure. The research on System Improvement focuses on using the web usage mining for improving the web traffic. Lastly, personalization systems aim to understand individual trends used for personalizing the web sites. Throughout the following sections, detailed description of the projects belonging to each category will is given.

2.4.1. General Web Usage Mining Systems

General Web Usage Mining systems focus on the analysis of log files using data mining techniques for discovering general access patterns and trends of users. Majority of the studies under this category aim to discover user navigation paths from the log files. In general, navigation path of a visitor is mainly the path followed by him/her through out his/her visit to the web site. It should be noted that each different study enlarges this definition by determining some specifications on what can be accepted as a path. Systems also differentiate on how they use the paths found. Some of the general usage mining systems present the user navigation paths without any further processing. Some others provide a query language for analyzing the paths better. In addition, there are studies on clustering paths or just user sessions with the aim of finding similar interest groups among visitors. The other types of studies under this category proposes to adapt well known data mining techniques such as association rule mining to the problem of web usage mining. In this section, we will explain each of these studies in detail.

2.4.1.1. Mining Traversal Paths

One of the existing approaches for mining navigation patterns from log files is to make use of well-known techniques from data mining. An example of such a study aims to find frequently occurring paths, which are named as maximal reference sequences from the log file by using a methodology similar to the association rule mining, M.-S. Chen et al. [10]. The first step of the proposed solution procedure is to traverse the whole log file for finding maximal forward references for each user. To be able to do that, the log file is divided into user paths where each path contains the accesses belonging to a specific user. Then, each user path is processed to find maximal forward references contained it. A maximal forward reference is defined as a sequence of pages that are visited consecutively by the visitor in which each page is seen only once. Whenever a backward reference to a page previously visited is seen, the current maximum forward reference path is terminated, added into the database and a new one starts. While traveling through the pages, visitors generally turn back to the previously visited pages and choose other links from them. The pages seen on the way back to the previous pages are visited only because of their location, but not their content. In the light of this observation, the study concentrates only on forward references. As an example, assume that the traversal sequence of the Visitor A is as follows: (P1, P2, P3, P4, P3, P2, P7). In this example, Visitor A turns back to the page P3 and P2 consecutively after retrieving page P4. Here, the page P3 is retrieved only for being able to retrieve page P7 from page P2. The algorithm forming maximal forward references solves that problem by removing the backward references from the paths. The algorithm produces the following maximum forward references for Visitor A: (P1 P2 P3 P4, P1 P2 P7) Once the maximal forward references for each user are formed, the next step of the solution for mining path traversal patterns is ready for the execution. In this step, the database containing maximal forward references for all users is processed to be able to form large reference sequences, which are frequently occurring consecutive subsequences among all maximal forward references. Full Scan (FS) and selective scan (SC) are two different algorithms for finding the large reference sequences. FS algorithm is indicated to be similar to the well-known algorithm called Direct Hashing with Pruning (DHP) for mining association rules with adaptations to the current problem. Because, the problem of finding large reference sequences from the database of maximal forward references has common points with finding large item sets from the database of transactions in association rule mining. The main difference between these two problems is that the order of the items in an item set is not important in association rule mining while it is crucial in mining traversal patterns. So, Full Scan algorithm changes the joining strategy used in the candidate generation phase of the DHP algorithm. Selective Scan algorithm is similar to Full Scan algorithm with optimizations to reduce I/O cost. Maximal reference sequences are the subset of large reference sequences so that no maximal reference sequence is contained in the other one. For example, if the large reference sequences are AB, AE, AGH, and ABD then maximal reference sequences become AE, AGH, and ABD. The sequences obtained through this way can then be used by web masters in redesigning the links between the pages that are accessed together in making marketing decisions.

2.4.1.2. Mining Navigation Patterns with Hypertext Probabilistic Grammars

Another study towards the problem of mining access patterns of visitors proposes to model user navigation sessions as a hypertext probabilistic grammar (HPG), Borges and Levene [6]. A user session is defined as a sequence of page requests coming from the same machine where the time passing between each request is less then a certain time limit. After the HPG is formed, the paths followed frequently by the visitors are discovered by applying a special case of depth first search algorithm on it. Each terminal and no terminal symbol of HPG built from user sessions correspond to a web page and there is a one-to-one correspondence between terminal and no terminal symbols. The links between web pages are represented by the production rules of the grammar. Two additional states, S and F are added into the grammar to represent the start and finish of the paths. In the corresponding automata, states represent no terminal symbols where transitions between states are formed by productions. Each production originating from a state is attached with a value, which is the probability that the link corresponding to a production was chosen from the links on a page represented by that state. In case of a start state, the probabilities of the productions are derived from the rate of the number of times that the page is visited to the overall number of hits. When navigating through a web site, visitors may concentrate on unrelated topics in a single session. Accordingly, the concept of N-Grammar is suitable to be employed in building HPG. N-Grammar dictates that the link that will be chosen by a visitor on any page is effected only by the last N pages retrieved by him/her. In HPG that makes use of the concept of N-Grammar, the number of states may increase too much if N is chosen to be very large. This is because each distinct consecutive sequence of N pages visited by any user should have a corresponding state in HPG. After the construction process, user preferred paths are discovered from HPG by applying depth first search like algorithm. Before mining, the mining expert should specify support and confidence thresholds, which will affect the quality of the paths discovered. Support threshold ensures that the path is frequently visited while confidence threshold ensures that the derivation probability of the corresponding string is high enough. The support value for a particular path is obtained by looking at the probability of the derivation of the first state of this path from the start state. In addition, the confidence value is obtained from the derivation probabilities of other the pages on the path. By the help of support and confidence thresholds, it becomes possible to discover the paths that describe the common visitor behavior best.

2.4.1.3. Analysis of Web Logs through OLAP Mining

A totally different approach to web usage mining is to make use of OLAP (Online Analytical Processing) technology on mining process. WebLogMiner, O. Zaiane et al. [11], is one of the tools, which aim to incorporate the OLAP technology and the data mining techniques. OLAP techniques are being used to obtain a portion of the data that is interesting to the analyst who can also determine the abstraction level on which the data will be presented. Then, this data can be used as an input to the data mining algorithms. Results obtained through mining the data can also be presented in different ways by using OLAP techniques. So, the mining process becomes more interactive and flexible. OLAP technology firstly places the data into a data cube, which is stored in multidimensional array structures or relational databases. Each dimension of the data cube represents a distinct field of the data, such as URL or domain name. If the data cube has n dimensions, each cell is characterized by having distinct values for the fields represented by these dimensions. Each cell in the data cube stores the number of visitors, which have the same values with the values characterizing the cell. The advantage of the data cube representation is to make it possible to view from different perspectives and abstraction levels, which are performed by the OLAP operations such as, drill down, roll up and slice.

The following statements are examples for the simple queries that OLAP can answer quickly:

- Hits coming from Turkey, between March 2001 and May 2001
- Hits coming from edu domain on 23/03/2001 with agent Mozilla

In addition to the analysis performed by OLAP technology, WebLogMiner makes use of data mining techniques to analyze the data to answer questions that OLAP cannot. For this, it applies well-known data mining techniques such as association rule mining, clustering or time series analysis on the data stored in the data cube. For example, by performing time series analysis, it answers the following questions: - what are the typical page request sequences performed by the visitors? Namely, are there request sequences that are common to most of the visitors?

- What are the event trees belonging to specific time intervals? Here, event trees contain the traversal patterns of the visitors in an aggregated form.

- How the traffic on a web site changes depending on time? Are there particular trends on particular times of a day, month etc.?

2.4.1.4. Web Utilization Miner

Web Utilization Miner (WUM) is another data mining tool designed for mining user navigation patterns from web logs. The distinguishing facility of this tool is to provide a mining language by which users can dynamically specify constraints on the mining result. WUM is composed of two major modules: Aggregation service and the query processor, Spiliopoulou [5]. Aggregation service firstly processes the log file and divides it into the visits that are used in constructing the aggregate tree on which the mining will be performed. Adding each path seen on the log file extends the tree. While forming the tree, paths that have a common prefix are merged. So, all paths are represented in the tree at the end. Each node taken together tree contains a URL, occurrence count and the number of visitors reaching that node by following the path starting from the root node. Because there exists visits that contain the same URL more than once, each node is associated with an occurrence count to show which occurrence of the URL this is. The way of storing paths in an aggregate tree was chosen for reducing the space requirements and speed up the mining process. The aggregate tree is built for once and used as input for the other module. Query processor is the module that performs the interactive mining on the aggregate tree constructed by the Aggregation service. The user can specify structural, textual and statistical constraints on the mining result. For example, the following query, which is expressed in MINT syntax, will result in a graph showing navigation patterns between B.HTML and any page whose support is larger than 1. The wildcard between the nodes means that there may be any number of nodes between X and Y. But the order of the template variables should stay the same as given in the query.

select T nodes as X Y, template X*Y as T and X.name= B.HTML and Y.support!1

The algorithm that is used for mining according to the given template and other constraints firstly finds all possible bindings for all template variables. At first, all possible bindings for the first template variable are found by checking all nodes taken together tree. This is the first and the last time that the query processor processes the whole tree. After that, only the trees rooted at the nodes that contain the URLs bound to the first template variable are processed for finding the possible bindings for the second template variable. Each different binding obtained for the template variables is named as pattern descriptor. That is, pattern descriptors contain identifiers that match to the template variables in given queries and wildcards. Namely, in this step of the algorithm, the structural and textual constraints specified by the user are taken into consideration. Next, the algorithm obtains navigation pattern corresponding to each pattern descriptor found in the first step. The designers of the system define a navigation pattern as a graph formed according to the pattern descriptor. For each pattern descriptor, the algorithm firstly finds all branches of the Aggregate Tree that contains the pattern represented in that descriptor. Then, these branches are merged at their common prefixes and on the identifiers existing in the pattern descriptor. While merging the branches, the counts of the nodes that are merged together are added. After merging, the statistical constraints on the mining result are checked and a descriptor is ignored if these constraints are not satisfied. At the end, remaining navigation patterns are shown to the user.

2.4.1.5. WebSift

WebSift is a web usage mining system, which aims to apply well-known data mining techniques on the usage data obtained through web logs, Cooley et al. [7]. It divides the mining process into three main phases: Preprocessing, Pattern Discovery and Pattern Analysis. The aim of the Preprocessing step is to turn the raw data in the log file into a form that is suitable for mining. Then, in the second phase of the usage mining process, well-known data mining techniques such as association rule mining, sequential pattern mining or clustering is applied on the transactions obtained in the previous phase. In the third phase, creators of the WebSift system propose to provide a query language and visualization facilities. In addition, in this phase of the mining process, using the Information Filter filters uninteresting results. Preprocessing step includes cleaning the data, identifying users and sessions belonging to them, completing the missing references in paths and formatting the data to obtain the appropriate transaction type for the type of the mining operation that will be performed. Data cleaning is the removal of irrelevant and redundant data in the log file such as requests for graphics. Besides, user identification tries to identify the requests belonging to each user. Authors indicate that an IP address may not be suitable to differentiate between the users, because two visitors may be using the same IP at the same time. For the solution of this problem, they propose to make use of some heuristics. For example, if two requests come from different types of browsers from the same machine, these requests are accepted to be performed different users. After the users are determined, the accesses belonging to them are divided into sessions. Then, the pages that are not recorded but accessed by the visitor are determined and added to the sessions. In the Information Filter, the interestingness level of the rules is determined by looking at the site structure. Currently, the system is capable of determining the interestingness level of frequent item sets and association rules with two different techniques: BME (Belief Mined Evidence) and BCE (Beliefs with conflicting Evidence). BME finds the frequent item sets, which contain pages that are not directly linked. Frequent item sets that contain linked pages are not that interesting because it is already guessed by the site designer who put a link between them. On the other hand, if many visitors retrieve pages that have no link in between together, this may be an interesting result for the site designer who may notice a deficiency in the site design. The pages that are linked, but not in the same frequent item set may also be interesting. BCE finds that kind of pages. The result shows that the link between these pages is rarely used by the visitors, which may give a clue to the site designer for the removal of this redundant link.

2.4.1.6. WAP Mine

Wap (Web Access Pattern) Mine is an efficient data-mining algorithm for discovering web access patterns from the Wap Tree (Web Access Pattern Tree), which is a compact data structure, designed for storing the data obtained from the logs, Perkowitz and Etzioni [12]. The result of the algorithm is the set of frequent access patterns, which contain pages requested sequentially by enough number of visitors. Wap Tree is formed by the addition of the frequent access subsequences that take part in the log in hand. Before the construction of the Wap tree, the log is traversed once for finding frequent 1-sequences, URLs that are seen in efficient number of user sessions. Then, the URLs that are not frequent are filtered from the sessions resulting in frequent access subsequences. Wap Tree is constructed by merging the frequent access subsequences on their common prefixes. Another feature of the Wap Tree is that all nodes that contain the same URL are linked into a queue and another data structure, header table, contains a pointer to the head of all queues. The foundation of the Wap Mine algorithm is based on a heuristic called Suffix Heuristic. Suffix Heuristic says that if an event (page reference) e is frequent in the prefixes of sequences that have a suffix, which contains pattern P as a subsequence, and then eP is a pattern. The algorithm for finding frequent sequences based on that heuristic is named as conditional search, which is employed in Wap Mine for mining sequences. Wap Mine algorithm processes each event one by one. For each event (page reference) e i, it firstly forms the conditional tree for that event. Conditional Wap tree for e i contain the set of prefixes of the subsequences that contain e i as a suffix. After this, the algorithm continues to mine Conditional Wap Tree recursively. Finally, the results obtained from mining conditional Wap Tree are concatenated with e i. Page sequences obtained through this algorithm correspond to the frequent access patterns.

2.4.1.7. Clustering User Sessions

Another study towards Web Usage Mining proposes to cluster visitors of a website based on the page requests taking place on the sessions belonging to them. The aim of this study presented in Fu et al. [13] is to discover the groups of pages that are visited together by many visitors. This information can then be used by the Web master in redesigning the Web Site or updating it with extra links between these pages. In this study, a log file of the web site is initially divided into user sessions. For clustering, each user session should be represented with a vector of pages in which each entry corresponds to the time spent on that page. However, in a web site that has many pages, the size of the vectors will increase dramatically. Because, the vectors should have an entry for each page in the web site. To overcome this, session vectors are generalized by using Attribute Oriented Induction method. Entries in the session vector are generalized by looking at the page hierarchy of the web site. A page hierarchy can be derived from the directory structure of the server and represented in the form of tree. The leaves of the tree correspond to the URLs. The parents of the leaf nodes keep the names of the web pages corresponding to innermost directories containing the URLs represented by their children. The parents of the no leaf nodes are the web pages representing outer directories containing them. The directories are derived directly from the names of the URLs. For example, the parent of the URL http://www.umr.edu/,regwww/ugcrc97/ee.html is http://www.umr.edu/,regwww/ugcrc97. By using the page hierarchy obtained by this way, the pages in the sessions are generalized as much as specified by the mining expert by using the tree climbing method from attribute oriented induction. During generalization, each page is replaced with one of the parents depending on the level to which pages are generalized. After generalization of the pages in the session vector, the duplicate general pages are merged into one by adding the times spent on them. By this way, the size of the session vectors is decreased dramatically, because the number of general pages is smaller than the number of URLs. Sessions obtained by this way are then clustered by using BIRCH hierarchical clustering algorithm. In BIRCH, using a tree structure, which it calls as CF tree, performs clustering. In CF tree, leaf nodes contain the current clusters, which are the collection of session vectors while the no leaf node store CF vectors, which characterize the clusters below them. When a new session vector should be placed into the tree, it goes until a leaf node by choosing the branches that are the closest to him/her. The CF vectors of the parent nodes are updated accordingly. In case there is no matching entry in leaf with given thresholds, the session vector is put into a new entry in the leaf node. If there is no empty entry, the leaf is split into two, which may cause additional splits in the parent nodes.

2.4.1.8. Clustering of the Paths Followed by the Visitors

Different from the most of the other usage mining systems, the system provides a profiler for obtaining more accurate, reliable and detailed information about the behavior of the visitors of the web site, Shahabi et al. [14]. The data obtained by this profiler is then used for obtaining the paths followed by the visitor and times spent on each page of the paths. Then, these paths are clustered by using Path-Mining method. The profiler provided by the system works on the client side. It is a Java applet loaded into the client side with the first page request and staying in the client cache afterwards. A call to this applet is added to each page in the site. The aim of the Java applet is to determine exact viewing time of the pages and catching the page views missing due to the retrieval of them from the client cache instead of a server. In addition to the profiler, each link on each page is updated to make it transfer more information to the server side when clicked. This information will be used for determining which links the visitor selects. Then, the link names will be added into the paths so that each pair of page requests are separated by the link, which is selected for retrieving the second page. The authors indicate that link information may provide additional clues on user behavior especially if two links from the same page are pointing to the same page. After the page requests and times spent on them are determined for each visitor, paths found are clustered for obtaining the groups of visitors with similar interests by using the Path-Mining methodology. Via the usage of this methodology, the order of the requests in the path is also taken into consideration on the contrary to the work explained in the previous section. To be able to this, the system needs a way of measuring the similarity between two paths. The similarity between two paths is measured by finding the angle between them. Briefly, the angle between two paths are calculated by using the inner product over the feature space where feature space contains all sub paths of these two paths. After the angles between each pair of paths are calculated, the results are fed into k-means algorithm for finding clusters of paths. The resulting clusters are considered to be containing groups of visitors with similar interests.

2.4.2. System Improvement

Research on System Improvement aims to use web usage mining for improving the web traffic and increasing the speed at which the visitors are responded. One way to do this is to provide the web server with the capability of guessing the pages that may be retrieved by the visitors next and generate the dynamic content of these pages before user retrieves them. For guessing the pages, the system proposed by Schecter et al. makes use of the concept of path profile, which is constructed from the data contained in the web logs, Schecter and Smith [15]. Path profile is the set of paths followed by the visitors of a site and the number of people following them. This system provides an efficient technique for generating and storing the path profiles. The paths are stored in the form of a tree in which the paths are merged on their common prefixes. While constructing the tree, only the paths whose maximal prefix is seen in at least T of the paths are added into the tree for reducing the memory cost. By following this rule, the algorithm for forming the tree is run on the tree more than once to be able to obtain all paths suiting to the threshold value. A path profile is then used by online working part of the system for guessing the next access of the user. The system starts with the shortest suffix of the current user path and tries to find a path whose maximal prefix matches with it. As long as a matching path whose maximal prefix equals to the suffix in hand is found, suffix size is increased. Assume that the pages retrieved by the visitor An are as follows: [P1, P2, P3]. In that case, the system initially checks the tree for finding the paths that have a maximal prefix [P3], the smallest suffix of the user path. Then, the suffix size is increased by one and the paths that have [P2 P3] as maximal prefix are found if there exists any. Assume that [P2 P3 PY] is such a path. Increase in the suffix size continues as long as the corresponding paths are found. If there is a path [P1 P2 P3 PX] in the tree, the system creates the dynamic content for the page PX automatically. If there exists no path having [P1 P2 P3] as maximal prefix, then the system will create the dynamic content for the page PY.

Another prediction technique is named as point based prediction in which the next page is guessed only by considering the last page retrieved not the whole path. Experiments with the system show that agreement prediction technique gives the most accurate results. In this technique, dynamic content creation of a particular is done only if both point and path based prediction techniques agree on that page.

2.4.3. Personalization

Because of the increasing demand to the e-commerce, many companies are eager to make their sites that exhibit their products more serviceable and effective for their visitors to be able to turn them into customers. The number of people visiting the web site of a company may be too high whereas only small percentage of the visitors may be turning into a customer. The number of customers gained through web site heavily depends on the success of the site and personalization is critical aspect of this success. Web Personalization simply means to understand the needs and interests of the visitors of the site and respond accordingly. Such a web site recognizes each visitor and customizes itself by various ways such as determining the information that should be shown to the visitor or automatically changing the site structure in a way that will be useful and attractive for the current user. Personalization is attractive research topic, because it is critically important for the success of e-commerce companies. Some of the different techniques for personalization will be explained in the subsequent sections.

2.4.3.1. Content Based Filtering

The main idea of Content Based Filtering is to make use of content similarity between stated user interests and web pages for personalization, Satıroğlu [16]. WebWatcher is an agent that trusts on content based filtering for personalization of the web sites. It guides visitors during their navigation through the web site according to their interests. At the beginning of a visit, WebWatcher asks user to enter his interest or the thing that he is looking for in the form of keywords. By using this information, WebWatcher highlights the links that are best suited to the needs of the visitor on each page retrieved by him/her through out his visit. WebWatcher accomplishes the task of choosing the best links for that user by using the information learned from the past users. In addition, the actions performed by each visitor are continuously used as training samples for improving the performance of the tool in future recommendations. Three different learning techniques are tried in WebWatcher: Learning from previous tours, Learning from Hypertext Structure and the combination of first two. First method proposes to store a description for each link in each page in the form of a high dimensional feature vector whose elements are English words. Interests of the users are also represented by a feature vector. Whenever a visitor follows a link in a page, the interests of the user, which consists of some number of keywords, is added to the description of that link. What are used in choosing the links to highlight in each page is the descriptions of the links determined as a result of this learning mechanism. While choosing the links that will be recommended to the visitor, WebWatcher calculates the similarity of each link in the page to the interest of the user. The links that will be highlighted are the ones with the highest similarity values with the user interest. To learn from the hypertext structure, WebWatcher makes use of Reinforcement Learning. If we take an agent moving across states as an example case, the aim of the Reinforcement Learning is to train the agent so that it will reach the final state from the initial state by choosing the best action to take in each state it encounters. Here, the action means choosing a next state to go. Goodness of choosing an action a in state s is represented as Q(s; a). The optimal strategy is to choose an action that will maximize the Q value for the current state. Turning back to hypertext environment, pages are the states and links are the actions. The system learns Q(s; a) function for each page and word pair, which means that the best action to take is different for different words in the same page. So, in each page the system recommends the hyperlinks, which maximize the total of Q values, belong the current page and words given as an interest of the user. The third method, which is detected to be giving the best results, combines the results obtained from first two methods plus two additional methods. The first additional method chooses the links that are mostly preferred while the second method chooses the links whose textual content is most similar to the interest words of the current user.

2.4.3.2. Usage Based Web Personalization

Most of the recent research on personalization aims to incorporate pattern discovery with personalization, resulting in a usage based Web personalization or customized usage tracking, Cooley [7]. In that case, profiles of the visitors are dynamically created according their access patterns. Dynamic creation of profiles is advantageous when compared to the profiles specified by the visitors themselves. Older personalization tools and techniques rely on that kind of profiles, which are static and most probably biased. As the time passes, user preferences may change although static profiles remain unchanged which decreases the performance of the personalization system. On the other hand, dynamically created profiles capture the current interests of users. Dynamically created profiles cannot be hundred percent faultless, but they achieve considerable amount of success in helping users without waiting for the user asking for it. Usage based Web Personalization systems generally comprise two major components: Offline component and online recommendation engine. Offline component of the system analyzes the log files, which contain the footprints of all visitors visiting the site. First it puts the data in the logs into a form that is amenable for applying data mining techniques. Analysis of log files by various data mining techniques result in aggregate usage profiles, which are common profiles of visitors of the web site. Then, the online component of the system matches the current user to these profiles based on his navigation pattern up to that point and customizes the current page accordingly. Customization can be done through recommending some links or putting advertisements or product news that may interest the customer. Through out the following sections, we will explain current Web Personalization systems and tools in more detail.

2.4.3.3. Analog

Analog, Yan et al. [17] is one of the first usage based Web Personalization systems. Its offline module clusters the users of a web site according to their access patterns. Offline module first processes the log file of the target site to find out user sessions, which are represented as n dimensional vectors where n is the number of distinct pages in the site. The weights of the entries corresponding to the pages visited by the visitor are larger than 0, while the weights for no visited pages are zero in the session vector. Therefore, the system does not take into account the order in which pages are retrieved. After all session vectors are obtained in this way, LEADER algorithm is applied to find clusters. LEADER is a simple clustering algorithm, which has some drawbacks. After clustering is completed, median vector of each cluster is computed as a representative of the cluster. The online module of the system recommends some links to the active visitors by looking at the pages that they retrieve before. Active user sessions are represented as n dimensional vectors as in the offline module. Whenever user retrieves a new page, the session vector belonging to that user is updated accordingly. Online module of the system tries to match the active user session to existing clusters. User session is accepted to be matching to a particular cluster if the number of common pages between user session vector and cluster median is larger than some threshold value. The pages in the median vectors of matching clusters are then recommended to the user if they are not already retrieved by him/her.

2.4.3.4. Web Personalizer

Web Personalizer, Cooley [7], is one of the other systems that make use of the explained framework for usage based Web Personalization. The main aim of the offline module of the system is to obtain aggregate usage profiles, which are represented as weighted collection of URLs. The reason for preferring this representation style is to be able to make use of classical vector operations that are used in clustering. Two different methods for forming the aggregate usage profiles are presented in. The first method is to cluster the user sessions by using standard clustering algorithms for grouping the visitors that have similar interests together as in Analog. This method proposes to represent each user session, as an n-dimensional vector where n is the number of distinct URLs that exist in the user sessions. The values kept in each entry of the user session vector can be chosen to be binary to indicate the existence or nonexistence of that URL in that session. After putting them into the vector form, user sessions are clustered by using classical clustering techniques from data mining to obtain session clusters. The next step to form Aggregate Usage Profiles is to find the mean vector of each cluster. Entries in the mean vector of a cluster are calculated by finding the ratio of the number of user sessions that contain the URL that is represented by that entry to the total number of sessions in that cluster. Because of this calculation, some of the URLs are filtered out because of having very low support, which means that only minority of the user sessions in the cluster contain them. The resulting mean vectors are representative aggregate usage profiles for the log data processed. The other method proposed in clusters URLs instead of sessions. It is indicated that users that have very different sessions may have common interest to a group of URLs. This information will remain undiscovered with the previous method. At the end of this method, each cluster will contain a set of URLs which tend be together in majority of the sessions. Standard clustering algorithms are difficult to be applied in that case because of the nature and the size of the feature space, which consists of the sessions. Therefore, another clustering technique, which is named as Association Rule Hypergraph Partitioning (ARHP), is employed by this method. The hyper graph to be clustered by this technique composed of URLs as vertices and the frequent item sets as the hyper edges, which connect the vertices representing the URLs in that item set. As known, frequent item sets are formed by a well-known technique from association rule mining. Application of the ARHP technique on the hyper graph obtained by this way results in a set of clusters which contain a set of URLs that are frequently accessed together. Usage profile for each session is obtained by associating a connectivity value of the vertex as a weight for the corresponding URL. As it is in the other usage based personalization systems, online component of the system keeps track of the active user sessions to recommend some links attached with significance scores to the users. The way to do this is to find aggregate usage profiles that match to the current user session best. The matching scores are calculated by standard distance and similarity measures between vectors. Besides, site structure becomes effective in calculating the matching scores by increasing the score of the pages that are farther away from the current page. History depth is an important concept employed by the online component of the system for obtaining more successful recommendations. It determines the number of previous pages that will be effective on the recommendation. It is indicated that user sessions are mostly composed of some number of episodes, which are paths, followed for reaching different kinds of information. The length of episodes is indicated to be 2 or 3 in general. Therefore, by the help of the concept of history depth, it is aimed to make recommendations based on the pages retrieved only in the current episode.

2.4.4. Site Modification

Another way of benefiting from the usage data discovered from the logs is to use it to improve the design of the web site. In personalization, web sites are dynamically customizing themselves differently for each visitor. On the other hand, site modification systems offer static changes in the structure and content of the web sites to meet the needs of all visitors, Perkowitz and Etzioni [12]. IndexFinder is one example for that kind of tools. It aims to discover index pages whose addition is very likely to improve the site design. These pages, which are created offline, consist of links to the conceptually related, but currently unlinked pages, which coexist in most of the user sessions. The addition of automatically created index pages to the site is performed with the authorization of the Web Master. Index page creation in IndexFinder is performed in three phases: processing logs, cluster mining and conceptual clustering. In the first stage of the algorithm, a log is processed to be divided into visits. What comes next is the calculation of the co-occurrence frequencies between each pair of pages to determine to what extend these pages are related. Cooccurrence frequency between two pages is simply calculated by taking the minimum of the two probabilities, probability of the existence of first page in the visit given the fact that second page is in the visit and visa versa. The concurrence frequency between linked pages is taken as zero to avoid uninteresting clusters. After the concurrence frequencies are calculated, a similarity matrix is constructed which is then converted into a graph form by taking the pages as nodes and concurrence values as edges between these nodes. Naturally, the nodes will be unlinked if the concurrence frequency between the pages denoted by them found to be 0 from the similarity matrix. The connected components in the graph built in this way are accepted as clusters. The pages corresponding to the nodes of a connected component found by this way are put into one cluster. The Cluster Mining algorithm is PageGather, which differs from the other clustering algorithms because of not insisting on putting every instance in one and only one cluster. Instead, the algorithm discovers small number of high quality clusters. The clusters obtained by this way may contain pages that are conceptually unrelated. Yet, the aim of the IndexFinder system is to produce index pages that contain links which are conceptually related in addition to be visited together by the majority of the visitors. This constraint is satisfied by applying a concept-learning algorithm on the clusters found in the previous step. To be able to apply that algorithm, first each page should be tagged manually with the correct values for predefined enumerated concepts. Concept learning algorithm finds the most common and basic concept that summarizes the pages in the cluster. Then, the noisy pages that conflict with the concept found are removed from the cluster while the nonexistent pages that conform to the given concept are joined to it. Each cluster obtained by this way is used to form one index page, which is composed of the links to the pages that are in that cluster. The candidate pages are presented to the web master who will give the final decision on the addition of these pages to the site and the location and the title of them.

Chapter 3

PATTERN DISCOVERY TECHNIQUES

Each Web mining process requires a pattern discovery phase in which the algorithms and techniques from several research areas, such as data mining, machine learning, statistics, and pattern recognition techniques can be adopted. Throughout this section, some of the techniques used for pattern discovery are explained.

3.1. Statistical Analysis

Statistical techniques are the most powerful tools in extracting knowledge about visitors to a Web site. The analysts may perform different kinds of descriptive statistical analyses based on different variables when analyzing the session file. By analyzing the statistical information contained in the periodic Web system report, the extracted report can be potentially useful for improving the system performance, enhancing the security of the system, facilitation the site modification task, and providing support for marketing decisions, Cooley [4].

3.2. Association Rules

In the Web domain, the pages, which are most often referenced together, can be put in one single server session by applying the association rule generation. Association rule mining techniques can be used to discover unordered correlation between items found in a database of transactions. Cooley [4] pointed that in the term of the Web usage mining, the association rules refer to sets of pages that are accessed together with a support value exceeding some specified threshold. The support is the percentage of the transactions that contain a given pattern. The Web designers can restructure their Web sites efficiently with the help of the presence or absence of the association rules. When loading a page from a remote site, association rules can be used as a trigger for prefetching documents to reduce user perceived latency.

3.3. Clustering

Clustering analysis is a technique to group together users or data items (pages) with the similar characteristics. Clustering of user information or pages can facilitate the development and execution of future marketing strategies, Cooley [4]. Clustering of users will help to discover the group of users, who have similar navigation pattern. It is very useful for inferring user demographics to perform market segmentation in E-commerce applications or provide personalized Web content to the individual users. The clustering of pages is useful for Internet search engines and Web service providers, since it can be used to discover the groups of pages having related content.

3.4. Classification

Classification is the technique to map a data item into one of several predefined classes. In the Web domain, Web master or marketer will have to use this technique if he/she wants to establish a profile of users belonging to a particular class or category. This requires extraction and selection of features that best describe the properties of a given class or category. Cooley [4] indicates that the classification can be done by using supervised inductive learning algorithms such as decision tree classifiers, naïve Bayesian classifiers, k-nearest neighbor classifier, Support Vector Machines etc.

3.5. Sequential Pattern

This technique intends to find the inter-session pattern, such that a set of the items follows the presence of another in a time ordered set of sessions or episodes. It is very meaningful for the Web marketer to predict the future trend, which help to place advertisements aimed at certain user groups. Sequential patterns also include some other types of temporal analysis such as trend analysis, change point detection, or similarity analysis, Cooley [4].

3.6. Dependency Modeling

The goal of this technique is to establish a model that is able to represent significant dependencies among the various variables in the Web domain. The modeling technique provides a theoretical framework for analyzing the behavior of users, and is potentially useful for predicting future Web resource consumption.

Chapter 4

IYTE WEB USAGE MINING SYSTEM

The aim of the IYTE Web Usage Mining(IYTE WUM) System developed in this study is to discover the usage patterns of the IYTE web site by analyzing the access log, error log and user data of the web server. The architecture of the system is given in Figure 4.1. The IYTE WUM System has two independent parts: DataPreparation and QueryMechanism.

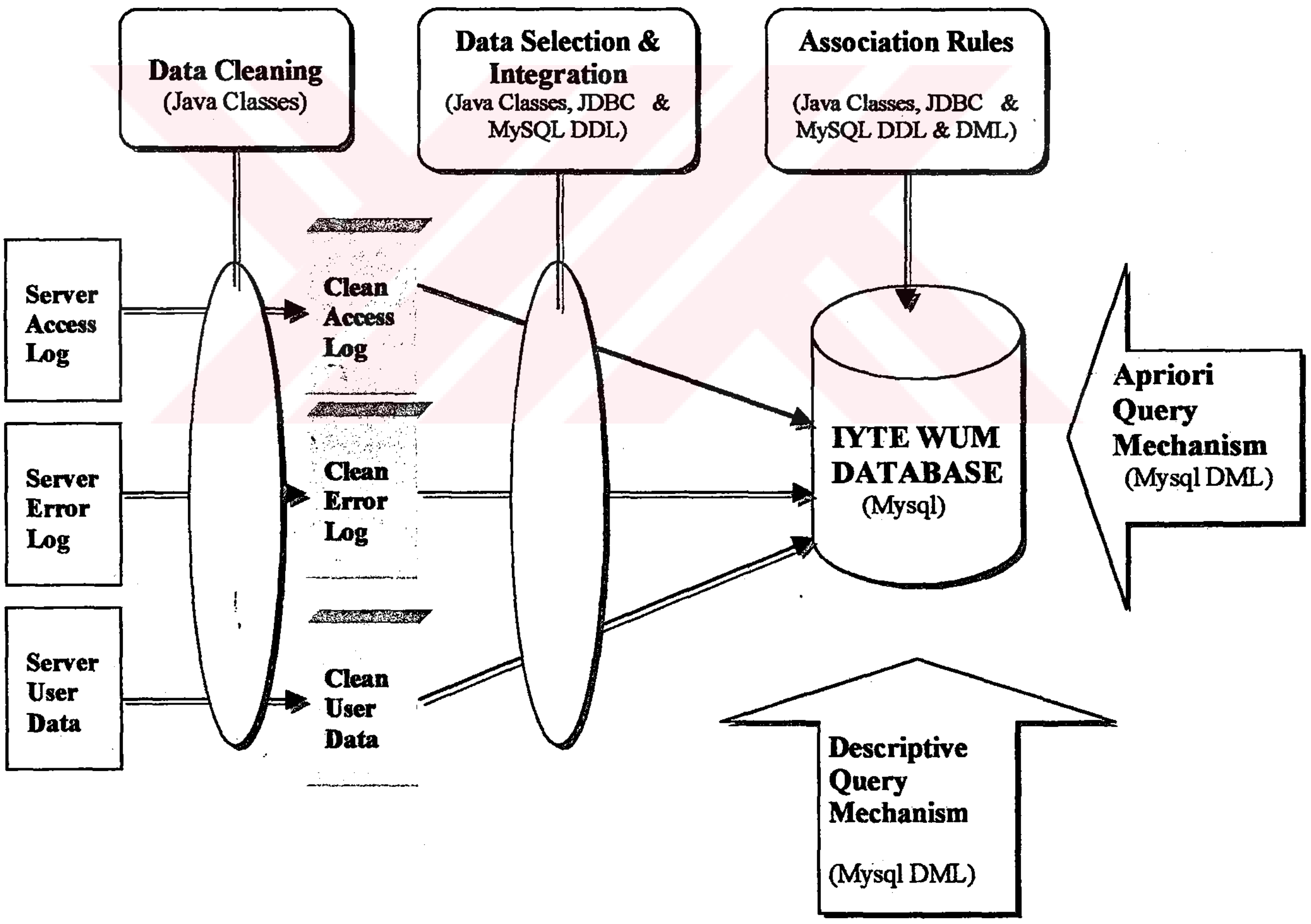


Figure 4.1. IYTE WUM System Architecture

The DataPreparation part was composed of different classes, which make the raw logs ready to be imported into the IYTE WUM relational database. The Query Mechanisms function on the IYTE WUM relational database to upload data from clean logs, to submit different queries and to run a Data Mining algorithm. As overall system it is an example of Web Usage Mining System with data base approach helping the web administrator to enhance the web site according to the changing requirements of the users. Basic concepts of the system and the detailed description of the system components are as follows.

4.1. Basic Concepts

In this section, we define the basic concepts such as visitor, file request, page request or daily use, which we will use many times throughout the rest of the thesis.

. ***Visitor:*** A person accessing the files on a web site from a particular machine. We tried to identify and distinguish visitors by using their IP addresses or host names. We assumed that two requests coming from the same IP address or host name were performed by the same visitor. Actually, IP addresses and host names may not be enough for identifying the visitors in some cases where the visitors are behind a proxy server or corporate firewalls because, the requests coming from the machines behind a proxy server contains the IP address of the proxy server instead of the real IP addresses. Therefore it is not possible to differentiate between these kinds of machines. According to our experience the most accurate solution for this problem seemed to be relying on the user cooperation although some heuristics may be used for differentiating between visitors. For example, the requests coming from same IP Address with different agents may be the indication of two different visitors.

. ***Valid File Request:*** Any type of data including graphics, scripts or html pages requested by the visitor and submitted to him by the corresponding web server.

. ***Valid Page Request:*** Any successfully answered request for one of the actual web pages taking place in the web site in process. We needed to differentiate between requests for actual web pages and the other types of files. Whenever a page containing images, sound facilities, etc. was requested by a particular visitor, all of the files utilized by that page were retrieved automatically by the web server, which added a new record to the log file for each of these file transfers. So, we needed to filter the log file for obtaining actual page requests. This could be performed easily by checking the filename suffixes of the requested files. The entries which contains request for certain types of files with suffixes such as WAV, CLASS, TXT, etc. were filtered before running the mining algorithms. In addition to these, we needed to eliminate unsuccessful requests that take place in the log file. A request was unsuccessful if it was not answered by the corresponding web server. These kinds of situations could easily be detected by looking at the status field of the log entries. For example, if the status code belonging to a particular log entry were 404 then we should ignore that entry, because the requested page of that entry was not found by the web server.

. ***Daily Use:*** Ordered set of page requests performed by a particular visitor on a given date. All paths start from a particular page in the site and expanded by the addition of the new pages retrieved by following the links on the previously retrieved pages or typing their addresses. In the light of these observations, path of a particular visitor is the ordered list of pages, which are requested by him/her consecutively. Inferring user paths is not an easy task, as it seems because of the nature of the web environment. The difficulty comes with the usage of BACK and FORWARD buttons provided by most of the web browsers. As known, recently requested pages are cached by the web browsers who display these cached copies of pages if visitors backtrack to them by using BACK and FORWARD buttons. Since no new page is being requested in such situations, web server does not become aware of the user behavior, hence this kind of situations are not reflected into the access log files. Although detecting user access paths was difficult, this was ignored in our case since our system aimed to retrieve general usage characteristics of the site.

4.2. Data Preparation

Main data source for our usage mining system was the server log files of the web site considered. Log files contained a huge amount of data, some of which was irrelevant to the usage mining process. In data preparation phase, raw data contained in the log file was filtered out to eliminate these irrelevant entries. The second important data preparation task was to put the relevant data into a form that is amenable for mining.

4.2.1. Access Log

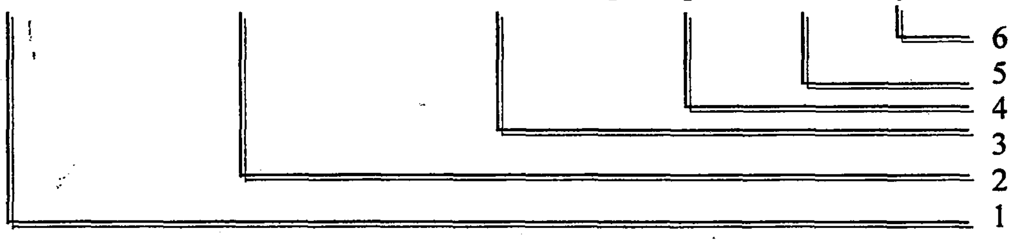
Server access log files store an entry for every single request the server gets. The format of the log file produced by the web server depends on the configuration of the server. Two of the possible log file formats are ***Common Log Format*** and ***Combined Log Format***, which differs in the amount of information that they store, related to each request. The log files of the IYTE web server (www.iyte.edu.edu.tr) are created in the Common Log Format. A fragment of IYTE web server access log is given in Figure 4.2.

```
pergamon.iyte.edu.tr  -- [20/Jun/2000:15:13:05  +0300]"GET /courses.html HTTP/1.1       " 304 -
pergamon.iyte.edu.tr  -- [20/Jun/2000:15:13:05 +0300]"GET / will/courses/CS101/ HTTP/1.1" 304 -
pergamon.iyte.edu.tr --  [20/Jun/2000:15:13:05  +0300]"GET / gif/geney.jpg HTTP/1.0   " 304 -
pergamon.iyte.edu.tr --  [20/Jun/2000: 15:13:05  +0300]" GET / gif/acad.gif HTTP/1.0     " 304 -
pergamon.iyte.edu.tr --  [20/Jun/2000:15:13:05| +0300]" GET / gif/ciz7.gif HTTP/1.0     "304 -
```

Figure 4.2. Fragment of Access Log

Record structure of Common Log Format is as follows;

Remote Host – Ident and Authuser – [Date and Time] "Request" Status – Bytes



The following lines explain what each entry stands for;

1-Remote Host: This field contains the hostname of the connecting machine. If the machine does not have DNS hostname, IP Address is used instead.

pergamon.iyte.edu.tr -- [20/Jun/2000:15:13:05 +0300]"GET /courses.html HTTP/1.1 " 304 -

2-Ident and Authuser: Ident is the remote login name of the user. If the requested document is password protected, Authuser field contains the user name. Since usually web browsers do not send this information, this field is generally empty.

pergamon.iyte.edu.tr **--** [20/Jun/2000:15:13:05 +0300]"GET /courses.html HTTP/1.1 " 304 -

3-Date and Time: This field contains the date and time of the request.

pergamon.iyte.edu.tr -- **[20/Jun/2000:15:13:05 +0300]**"GET /courses.html HTTP/1.1 " 304 -

4-Request [Method URL Protocol]: Request coming from the visitor is exactly kept in this field. Method field is set to GET for page requests and POST for form submissions. URL part is reserved for the related URL. Protocol specifies the HyperText Transfer Protocol used.

pergamon.iyte.edu.tr -- [20/Jun/2000:15:13:05 +0300]**"GET /courses.html HTTP/1.1** " 304 -

5-Status: Status field contains the return status of the request, which shows whether the transfer is successful or not.

pergamon.iyte.edu.tr -- [20/Jun/2000:15:13:05 +0300]"GET /courses.html HTTP/1.1 " **304** -

6-Bytes: The number of bytes sent by the server is stored in bytes field following the Status field. The number of bytes is set to null in the following record since bytes sent by the server is zero.

pergamon.iyte.edu.tr -- [20/Jun/2000:15:13:05 +0300]"GET /courses.html HTTP/1.1 " 304 -

Removing the requests for irrelevant types of files and unsuccessful requests was the first operation performed in data preparation phase. AccessLogRead( ) class coded in Java handled these functions together with date field conversion, extraction of the fields mentioned above, and addition of field separators between the fields. The source code of the AccessLogRead( ) is given in Appendix A1. Figure 4.3. shows a short fragment of access log after the clean up operation.

```
+--------------------+------------+----------+--------+--------+--------+----------------------+-------------+------+
| user               | ldate      | ltime    | rtime  | demand | url_main | url                | status_main | status |
+--------------------+------------+----------+--------+--------+--------+----------------------+-------------+------+
| pergamon.iyte.edu.tr | 2000-06-20 | 15:13:05 | +0300 | GET | gif | gif/yazi.gif HTTP/1.0    | 304 | 304 -|
| pergamon.iyte.edu.tr | 2000-06-20 | 15:13:05 | +0300 | GET | gif | gif/zemin1.gif HTTP/1.0  | 304 | 304 -|
| pergamon.iyte.edu.tr | 2000-06-20 | 15:13:05 | +0300 | GET | gif | gif/geney.jpgHTTP/1.0    | 304 | 304 -|
| pergamon.iyte.edu.tr | 2000-06-20 | 15:13:05 | +0300 | GET | gif | gif/acad.gif HTTP/1.0    | 304 | 304 -|
| pergamon.iyte.edu.tr | 2000-06-20 | 15:13:05 | +0300 | GET | gif | gif/living1.gif HTTP/1.0 | 304 | 304 -|
+--------------------+------------+----------+--------+--------+--------+----------------------+-------------+------+
```

Figure 4.3. Fragment of Clean Access Log

4.2.2. Error Log

Server error log files stores one entry for every single request the server could not respond. The format of the log file produced by the web server can be seen in Figure 4.4.

```
[Thu Aug 19 14:02:34 1999] Server configured -- resuming normal operations
[Thu Aug 19 14:12:27 1999] accept: (client socket): Connection timed out
[Thu Aug 19 14:12:27 1999] accept: (client socket): Connection timed out
[Thu Aug 19 14:13:01 1999] accept: (client socket): Connection reset by peer
[Thu Aug 19 14:17:05 1999] accept: (client socket): Connection timed out
[Thu Aug 19 14:17:59 1999] accept: (client socket): Connection timed out
[Thu Aug 19 14:18:52 1999] accept: (client socket): Connection timed out
[Thu Aug 19 14:44:40 1999] accept: (client socket): No route to host
[Thu Aug 19 14:44:43 1999] accept: (client socket): No route to host
[Thu Aug 19 14:44:47 1999] accept: (client socket): No route to host
```

Figure 4.4. Fragment of Error Log

A class coded in Java ErrorLogRead( ), Appendix A2, reads the error log, extracts date, time and the message text of each record in the log and adds field separator characters and generates a text file. Clean log becomes ready to be imported into the IYTE WUM relational database as seen in Figure 4.5. The details of loading are explained in the next sections.

```
+------------+----------+-----------------------------------------------+
| date       | time     | message                                       |
+------------+----------+-----------------------------------------------+
| 1999-08-19 | 14:02:34 | created shared memory segment #0              |
| 1999-08-19 | 14:02:34 | Server configured -- resuming normal operations |
| 1999-08-19 | 14:12:27 | accept: (client socket): Connection timed out |
| 1999-08-19 | 14:12:27 | accept: (client socket): Connection timed out |
| 1999-08-19 | 14:13:01 | accept: (client socket): Connection reset by peer |
| 1999-08-19 | 14:17:05 | accept: (client socket): Connection timed out |
| 1999-08-19 | 14:17:59 | accept: (client socket): Connection timed out |
| 1999-08-19 | 14:18:52 | accept: (client socket): Connection timed out |
| 1999-08-19 | 14:44:40 | accept: (client socket): No route to host     |
| 1999-08-19 | 14:44:43 | accept: (client socket): No route to host     |
+------------+----------+-----------------------------------------------+
```

Figure 4.5. Fragment of Clean Error Log

4.2.3. Users Data

User definitions of IYTE Urla campus are kept in a text file on the server. Each record in that file indicates the IP address of the user and the hub it is connected. Since the number of records is not large in this file data, the table was arranged and cleaned manually. Figure 4.6 shows a fragment of clean users text file, which is ready to be uploaded into the WUM relational database.

```
+--------------------+-----------+----------------------+--------+
| user_name          | type      | user_ip              | depcode |
+--------------------+-----------+----------------------+--------+
| busra              | MX 5      | busra.iyte.edu.tr.   |       0 |
| pergamon           | MX 5      | pergamon.iyte.edu.tr. |      0 |
| buamtest           | MX 5      | buamtest.iyte.edu.tr. |      0 |
| radyo-bahattin     | CNAME     | troya                |       0 |
| radyo-tolga        | CNAME     | troya                |       0 |
| sevgi-canlier      | A         | 193.140.248.37       |       0 |
| edibe-ciftci       | A         | 193.140.248.38       |       0 |
| bulent-kusev       | A         | 193.140.248.39       |       0 |
| yasar-olmez        | A         | 193.140.248.40       |       0 |
| haluk-yaren        | A         | 193.140.248.44       |       0 |
+--------------------+-----------+----------------------+--------+
```

Figure 4.6. Fragment of Users Data

4.3. Query Mechanism

Query mechanism was the heart of our study, which in fact contained four layers (Figure 4.8). In the first layer main tables of IYTE WUM relational database were loaded from clean log files. In the second layer first pass of algorithm Apriori (finding frequent itemset) was done with SQL and two intermediary tables were formed. In the third layer two more intermediary tables were formed and in the fourth layer final table of the algorithm was extracted which held frequent pairs of URLs visited during each daily use of users.

4.3.1. Filling up the Main Tables

Figure 4.8. explains the data and process flow in IYTE WUM System. Clean access and error log files and user data, which were prepared by the Data Preparation part, were stored under directory \data directory of SQL engine on the server with the consecutive names accesslogfile.txt, errorlogfile.txt and users.txt. A SQL session whose SQL commands are seen in Figure 4.7, loaded the main tables of the database logfile, errors and users.

```
use wum;
load data infile 'accesslogfile.txt' into table logfile
        fields terminated by '|'
        lines terminated by  '&';
load data infile "errorlogfile.txt" into table errors
        fields terminated by "|"
        lines terminated by "&";
load data infile "users.txt" into table users
        fields terminated by "|"
                                lines terminated by "&";
```

Figure 4.7. Filling the Main Tables

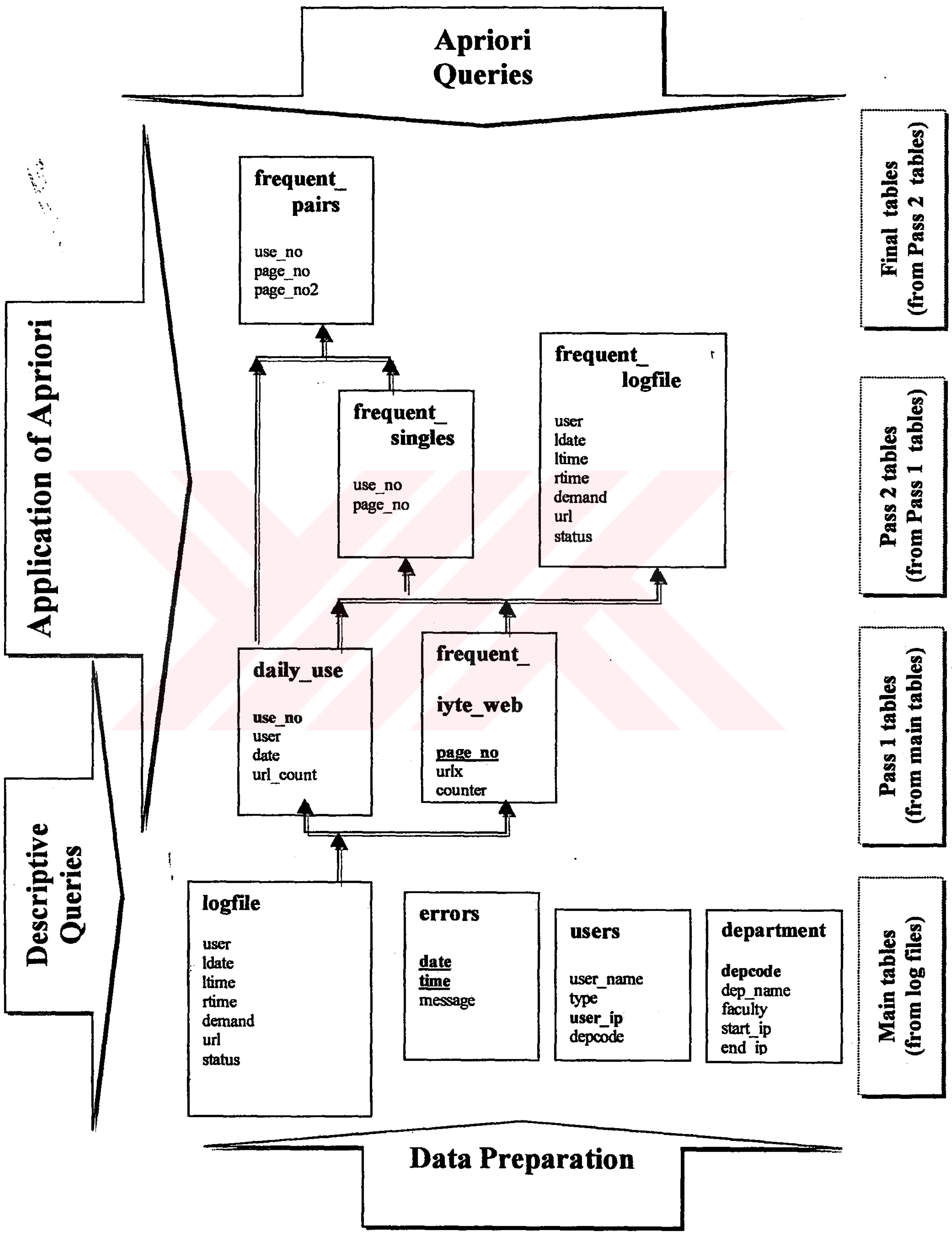


Figure 4.8. Process and Data Flow

4.3.2. Descriptive Queries

In this study special emphasis is placed on the architecture and development of web usage mining system by exploring database flexibility. In this section a few of the possible query examples are given.

In Figure 4.9. a query is submitted to retrieve the number of total hits and the result is ***10.740.138***. As it can be seeen in Figure 4.10. ***85.353*** hits are from internal users whose IP's are in users table. The query in Figure 4.11. returns the minimum, maximum and average number of hits per day respectively ***74, 46.552, 18.296***.

```
mysql> select "Number of total hits :", count(*) from logfile;
+----------------------+----------+
| Number of total hits : | count(*)   |
+----------------------+----------+
| Number of total hits : | 10740138 |
+----------------------+----------+
1 row in set (0.33 sec)
```

Figure 4.9. Total Hits

```
mysql> select "Number of hits received from known users :", count(*)
   -> from logfile, users use index(user_ip)
   -> where SUBSTRING(logfile.user,1,25) = users.user_ip;
+------------------------------------------+----------+
| Number of hits received from known users : | count(*) |
+------------------------------------------+----------+
| Number of hits received from known users : |    85353 |
+------------------------------------------+----------+
1 row in set (11 min 5.36 sec)
```

Figure 4.10. Hits of Internal Users

```
mysql> select "Min., Max., Avg. number of daily hits : ", min(daily_hit), max(daily_hit),
avg(daily_hit) from ldate_group;
+---------------------------------------+----------------+----------------+----------------+
| Min., Max., Avg.number of daily hits : | min(daily_hit) | max(daily_hit) | avg(daily_hit) |
+---------------------------------------+----------------+----------------+----------------+
|                                       |             74 |          46552 |     18296.6576 |
+---------------------------------------+----------------+----------------+----------------+
1 row in set (0.00 sec)
```

Figure 4.11. Aggregates of Daily Use Hits

The graphics seen in Figure 4.12. which is prepared using MS Excel and a text file containing the data in the table ldate_group shows the distribution of daily hits received. The plotted data seem to accumulate in two different zones. The lower zone indicates the hits received by the web server during holidays whereas the upper one correspond to the days the faculties are open. X-axis shows the days.

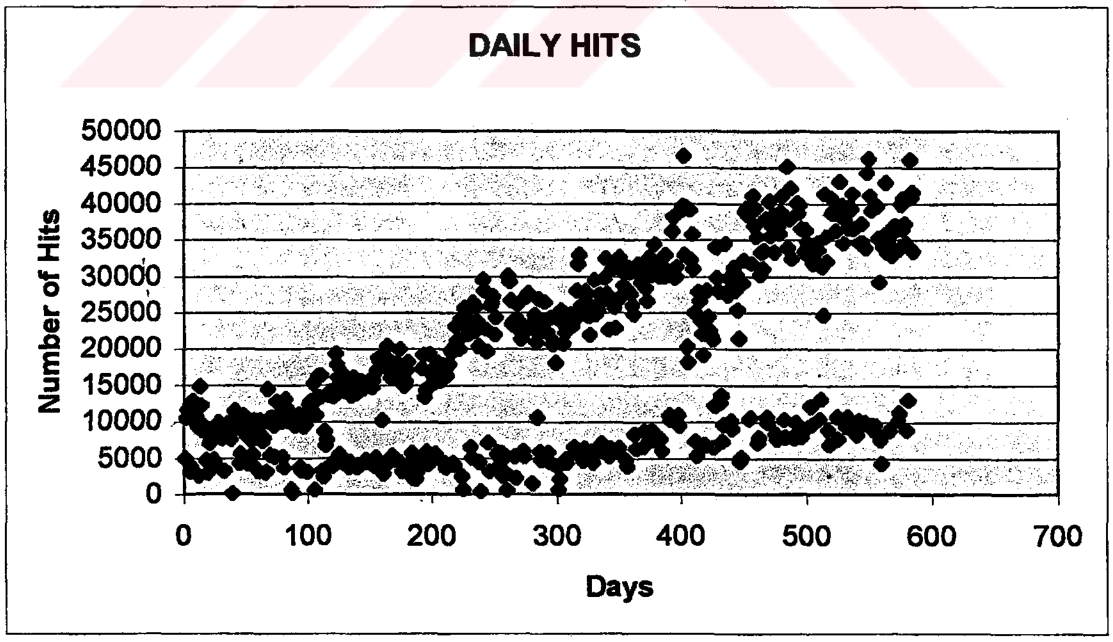


Figure 4.12. Graphics of Daily Hits

First day of the access log is 2000-06-20. There might be records with date 0000-00-00 indicating an error in time settings of the web server. So the first query in 4.13.shows the first day of the access log filtering out any possible erroneous date and the second query indicates the last date, which is 2002-01-31.

```
mysql> select "Start date of the log :", min(ldate) from ldate_group where ldate <> '0000-00-00';
+-----------------------+--------------+
| Start date of the log : |    min(ldate) |
+-----------------------+--------------+
| Start date of the log : | 2000-06-20 |
+-----------------------+--------------+
1 row in set (0.00 sec)
mysql> select "End date of the log :", max(ldate) from ldate_group;
+----------------------+--------------+
| End date of the log : |   max(ldate) |
+----------------------+--------------+
| End date of the log : | 2002-01-31 |
+----------------------+--------------+
1 row in set (0.06 sec)
```

Figure 4.13. **Time Interval of the Hits in Access Log**

The queries seen in Figure 4.14. try to describe the characteristic of the daily use of users. Minimum number of hits done by a user on any given day is ***2*** whereas the maximum hits done by a user on any given day is ***7.083***. Average number of hits realized by a user on any given day is ***54***.

```
mysql> select "Min number of hits of a user : ", min(url_count)
from daily_use;
+----------------------------------+-----------------+
| Min number of hits of a user :   | min(url_count)  |
+----------------------------------+-----------------+
| Min number of hits of a user :   |               2 |
+----------------------------------+-----------------+
1 row in set (1.49 sec)

mysql> select "Max number of hits of a user : ", max(url_count)
from daily_use;
+----------------------------------+-----------------+
| Max number of hits of a user :   | max(url_count)  |
+----------------------------------+-----------------+
| Max number of hits of a user :   |            7083 |
+----------------------------------+-----------------+
1 row in set (0.38 sec)

mysql> select "Average number of hits of a user : ",
avg(url_count) from daily_use;
+--------------------------------------+-----------------+
| Average number of hits of a user :   | avg(url_count)  |
+--------------------------------------+-----------------+
| Average number of hits of a user :   |         54.2074 |
+--------------------------------------+-----------------+
1 row in set (0.32 sec)
```

Figure 4.14. Aggregates of User Daily Hits

As the query in Figure 4.15. indicates the start date of the error log in the study is 1999-08-19. Again any possible erronous date of '0000-00-00' is filtered out by the SQL command. The last date is 2002-01-31.

```
mysql> select "First day of the error log : "    , min(date) from
errors where date <> '0000-00-00';
+-------------------------------+------------+
| First day of the error log :  | min(date)  |
+-------------------------------+------------+
| First day of the error log :  | 1999-08-19 |
+-------------------------------+------------+
1 row in set (1.76 sec)
mysql> select "Last day of the error log : ", max(date) from
errors;
+------------------------------+------------+
| Last day of the error log :  | max(date)  |
+------------------------------+------------+
| Last day of the error log :  | 2002-01-31 |
+------------------------------+------------+
1 row in set (1.76 sec)
```

Figure 4.15. Time Interval of Error Log

Figure 4.16 shows the queries extracting the minimum number of daily errors as ***1***, maximum number of daily errors as ***223.210*** and average number of daily errors as ***1.109***.

```
mysql> select "Minimum number of daily errors : ",
min(error_count) from daily_errors;
+------------------------------------+------------------+
| Minimum number of daily errors :   | min(error_count) |
+------------------------------------+------------------+
| Minimum number of daily errors :   |                1 |
+------------------------------------+------------------+
1 row in set (0.00 sec)

mysql> select "Maximum number of daily errors : ",
max(error_count) from daily_errors;
+------------------------------------+------------------+
| Maximum number of daily errors :   | max(error_count) |
+------------------------------------+------------------+
| Maximum number of daily errors :   |           223210 |
+------------------------------------+------------------+
1 row in set (0.00 sec)

mysql> select "Average number of daily errors :", avg(error_count)
from daily_errors;
+------------------------------------+------------------+
| Average number of daily errpors :  | avg(error_count) |
+------------------------------------+------------------+
| Average number of daily errpors :  |        1109.4031 |
+------------------------------------+------------------+
1 row in set (0.00 sec)
```

Figure 4.16. Aggregates of Daily Hit Counts

The graphics seen in Figure 4.17. is prepared on MS Excel by transferring the data in the table daily_errors by a temporary text file and shows the distribution of the number of daily errors realized by the IYTE web server.

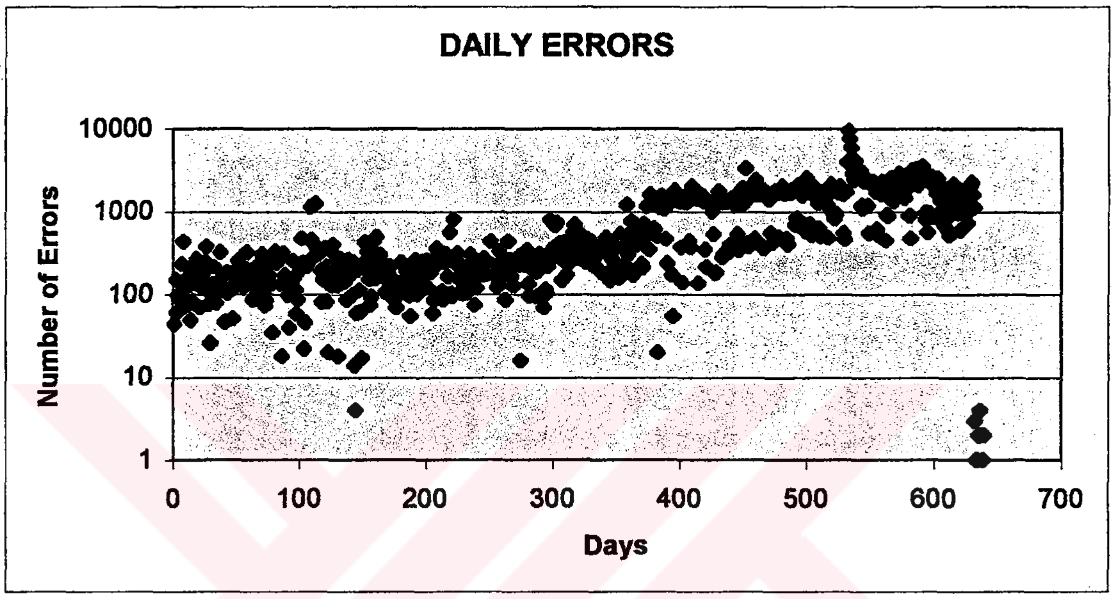


Figure 4.17. Graphics of Daily Errors

4.3.3. Association Rule Mining Algorithms

The most famous problem related with mining association rules is the one applied over basket data. An example of such a rule might be that 98% of customers that purchase tires and auto accessories also get automotive services done. Finding all such rules is valuable for cross-marketing and attached mailing applications. Other applications include catalog design, add-on sales, store layout, user logs and customer segmentation based on buying patterns. The databases involved in these applications are very large. It is imperative, therefore, to have fast algorithms for this task.

The following is a formal statement of the problem : Let $I = \{i_1 ; i_2 ; . . . ; i_m\}$be a set of literals, called items. Let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. Associated with each transaction is a unique identifier, called its TID. We say that a transaction T contains X, a set of some items in I, if $X \subseteq T$. An association rule is an implication of the form X => Y, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. The rule X => Y holds in the transaction set D with confidence c if c% of transactions in D that contain X also contains Y. The rule X => Y has support s in the transaction set D if s% of transactions in D contains X U Y.

Given a set of transactions D, the problem of mining association rules is to generate all association rules that have support and confidence greater than the user specified minimum support and minimum confidence respectively. Our discussion is independent of the representation of D. For example, D could be a data file, a relational table, or the result of a relational expression, Agrawal and Srikant [18].

In IYTE WUM System we used the algorithm Apriori since it always outperformed the earlier algorithms. Experiments have shown that the Apriori has excellent scale up properties, opening up the feasibility of mining association rules over very large databases, Agrawal and Srikant [18].

The problem of finding association rules falls within the scope of database mining which is also called knowledge discovery in databases. Related work in the database literature is the work on inferring functional dependencies from data. Functional dependencies are rules requiring strict satisfaction. Consequently, having determined a dependency $X \rightarrow A$ (A is functionally dependent on X; meaning A cannot be defined without X), the algorithms consider any other dependency of the form $X + Y \rightarrow A$ redundant and do not generate it. The association rules we consider are probabilistic in nature. The presence of a rule $X \rightarrow A$ does not necessarily mean that $X + Y \rightarrow A$ also holds because the latter may not have minimum support. Similarly, the presence of rules $X \rightarrow Y$ and $Y \rightarrow Z$ does not necessarily mean that $X \rightarrow Z$ holds because the latter may not have minimum confidence.

There has been work on quantifying the "usefulness" or "interestingness" of a rule. What is useful or interesting is often application dependent. The need for a human in the loop and providing tools to allow human guidance of the rule discovery process has been articulated. We do not discuss these issues in this thesis, except to point out that these are necessary features of a rule discovery system that may use our algorithms as the engine of the discovery process.

The problem of discovering all association rules can be decomposed into two sub problems:

1. Find all sets of items (item sets) that have transaction support above minimum support. The support for an item set is the number of transactions that contain the item set. Item sets with minimum support are called large item sets, and all others small item sets.

2. Use the large item sets to generate the desired rules. A straightforward algorithm for this task is as follows: For every large item set l, find all non-empty subsets of l. For every such subset a, output a rule of the form a => (l – a) if the ratio of support(l) to support(a) is at least minimum confidence. We need to consider all subsets of l to generate rules with multiple consequents.

Algorithms for discovering large item sets make multiple passes over the data. In the first pass, we count the support of individual items and determine which of them is large, i.e. have minimum support. In each subsequent pass, we start with a seed set of item sets found to be large in the previous pass. We use this seed set for generating new potentially large item sets, called candidate item sets, and count the actual support for these candidate item sets during the pass over the data. At the end of the pass, we determine which of the candidate item sets are actually large, and they become the seed for the next pass. This process continues until no new large item sets are found.

4.3.3.1. Algorithm Apriori

Agrawal and Srikant [18] express that the Apriori algorithm generate the candidate item sets to be counted in a pass by using only the item sets found large in the previous pass – without considering the transactions in the database. The basic intuition is that any subset of a large item set must be large. Therefore, the candidate item sets having k items can be generated by joining large item sets having k – 1 items, and deleting those that contain any subset that is not large. The procedure results in generation of a much smaller number of candidate item sets.

Figure 4.18. gives the notation used Apriori algorithm and Figure 4.19. gives the algorithm. The first pass of the algorithm simply counts item occurrences to determine the large 1-itemsets. A subsequent pass, say pass k, consists of two phases. First, the large itemsets L k apriori-gen function. Next, the database is scanned and the support of candidates in C k is counted. For fast counting, we need to efficiently determine the candidates in C k that are contained in a given transaction t.

k-itemset	An itemset having k items
L_k	Set of large k-itemsets (those with minimum support) Each member of this set has two fields: i) item set and ii) support count
C_k	Set of candidate k-itemsets (potentially large item sets) Each member of this set has two fields: i) item set and ii) support count
$\bar{C}_k$	Set of candidate k-item sets when TIDs of the generating transactions are kept associated with the candidates.

Figure 4.18. Notation

Apriori algorithm is founded on the observation that if any given set of attributes S is not adequately supported, any superset of S will also not be adequately supported. Consequently any effort to calculate the support for such supersets is wasted. For example if we know that {A,B} is not supported it follows that {A,B,C}, {A,B,D}, etc. will also not be supported. The algorithm proceeds as follows:

1. Determine the support for all single attributes (sets of cardinality 1) in the data set.
2. Delete all the single attributes that are not adequately supported.
3. For all supported single attributes construct pairs of attributes (sets of cardinality 2). If no pairs end, otherwise determine the support for the constructed pairs.
4. For all supported pairs of attributes construct "candidate" sets of cardinality 3 (triples).
5. If no triples end, otherwise determine the support for the constructed triples.

Continue until no more candidate sets can be produced

Figure 4.19. Algorithm Apriori

Han et al. [19] express critical remarks on the Algorithm Apriori. According to them apriori technique works well in terms of reducing the candidate set. However, where there are many patterns, long patterns or low support thresholds:

1. Many candidate items sets must still be generated.

2. Requires repeated scans of the databases (particularly when considering large candidate sets.

After application of algorithm Apriori to IYTE WUM database we experienced similar findings such as the need for generating many candidate item sets for which the time and space requirements of the SQL commands were extremely high.

4.3.3.2. Application of Algorithm Apriori with SQL

In the following sections the application steps of the algorithm Apriori into IYTE WUM relational database with the aid of the features of SQL will be explained.

1. A table containing summary data was created from logfile in which each row corresponded to all the hits of a specific user on a specific date. The attribute url_count contained the number of hits. The attribute use_no was added to this table in order to give unique number to each row, Figure 4.20.

```
create table daily_use
        select user, date, count(*) AS url_count
        from logfile
        group by user, ldate
        order by user, ldate;

alter table daily_use ADD (use_no INTEGER);
```

Figure 4.20. Creation of table daily_use

Figure 4.21. shows a fragment of table daily_use where 17 records of users containing more than 10 hits are shown.

```
mysql> select * from daily_use where url_count > 10 order by date desc
limit 17;
+--------+------------------+-----------+-----------+
| use_no | user             | date      | url_count |
+--------+------------------+-----------+-----------+
|   2480 | 144.122.148.197 | 2002-01-31 |       26 |
|   3671 | 155.223.136.244 | 2002-01-31 |       55 |
|   3794 | 155.223.2.100   | 2002-01-31 |       80 |
|   4272 | 155.223.37.23   | 2002-01-31 |       36 |
|   4758 | 155.223.85.170  | 2002-01-31 |       23 |
|   6308 | 170.12.117.67   | 2002-01-31 |       18 |
|   7482 | 193.140.134.25  | 2002-01-31 |       14 |
|   8391 | 193.140.168.3   | 2002-01-31 |       21 |
|   8758 | 193.140.181.20  | 2002-01-31 |       34 |
|   8775 | 193.140.182.171 | 2002-01-31 |       28 |
|  11809 | 193.140.249.14  | 2002-01-31 |       30 |
|  14485 | 193.140.249.63  | 2002-01-31 |       18 |
|  14529 | 193.140.249.72  | 2002-01-31 |       65 |
|  14883 | 193.140.249.89  | 2002-01-31 |       68 |
|  18907 | 193.140.250.78  | 2002-01-31 |       18 |
|  19141 | 193.140.251.106 | 2002-01-31 |      136 |
|  19257 | 193.140.251.121 | 2002-01-31 |       16 |
+--------+------------------+-----------+-----------+
17 rows in set (0.55 sec)
```

Figure 4.21. Fragment of Daily_use Table

2. Another table named frequent_iyte_web was created from logfile. This table contained records of URL's, which received more than 1000 hits. The attribute page_no was added to this table in order to give unique number to each row. Figure 4.22. This step corresponds to the preparation of L_k (large k-itemsets) mentioned in Figure 4.18.

```
create table frequent_iyte _web
        select url AS urlx, count(*) AS counter
        from logfile
        where counter > 1000
        group by url
        order by url;
alter table frequent_iyte_web ADD (page_no INTEGER);
```

Figure 4.22. Creation of Frequent_iyte_web Table

```
Logging to file 'frequent_iyte_web;'
mysql> select * from frequent_iyte_web order by counter desc limit 15;
+---------+-------------------------------------------+---------+
| page_no | urlx                                      | counter |
+---------+-------------------------------------------+---------+
|     175 |   HTTP/1.0                                |  149628 |
|     176 |   HTTP/1.1                                |  130631 |
|   22456 |  gif/gif/lib2.gif HTTP/1.0                |  117222 |
|   22458 |  gif/gn1.gif HTTP/1.0                     |  104840 |
|   22230 |  gif/ac1.gif HTTP/1.0                     |  103993 |
|   22535 |  gif/li1.gif HTTP/1.0                     |  103780 |
|   22656 |  gif/rs1.gif HTTP/1.0                     |  103650 |
|   22457 |  gif/gif/lib2.gif HTTP/1.1                |  103572 |
|   22624 |  gif/pr1.gif HTTP/1.0                     |  103439 |
|   22620 |  gif/ph1.gif HTTP/1.0                     |  103242 |
|   22707 |  gif/sr1.gif HTTP/1.0                     |  103134 |
|   22569 |  gif/ln1.gif HTTP/1.0                     |  103056 |
|   22263 |  gif/ano1.gif HTTP/1.0                    |  102831 |
|   22488 |  gif/ilogo.gif HTTP/1.0                   |  102453 |
|   22284 |  gif/baslik1.gif HTTP/1.0                 |  101809 |
|   22232 |  gif/ac2.gif HTTP/1.0                     |   97367 |
|   22537 |  gif/li2.gif HTTP/1.0                     |   97105 |
+---------+-------------------------------------------+---------+
15 rows in set (0.14 sec)
```

Figure 4.23. Fragment of Frequent_iyte_web Table

Figure 4.23. shows a fragment of table frequent_iyte_web where 15 records of URL records in descending order of hits received.

3. Table frequent_logfile was created from logfile where only the records of hits found on iyte_frequent_web table found were recorded, Figure 4.24. In Figure 4.25 fragment of records from that table is seen. This step corresponds to the generation of C_k (candidate k-itemset) mentioned in Figure 4.18.

```
create table frequent_logfile
        select *
        from frequent_logfile L, frequent_iyte_web F
        where  L.url = F.urlx;
```

Figure 4.24. Creation of Frequent_logfile Table

```
Logging to file 'frequent_logfile;'
mysql> select *from frequent_logfile order by ldate desc limit 50;
+-----------------+------------+----------+-------+-----+------------------------+---------+
| user            | ldate      | ltime    | rtime | dem | url                    | status  |
+-----------------+------------+----------+-------+-----+------------------------+---------+
| 195.175.212.118 | 2002-01-31 | 00:01:24 | +0200 | GET | gif/new_a.gif HTTP/1.1 | 200 416 |
| 195.175.212.118 | 2002-01-31 | 00:01:27 | +0200 | GET | gif/as20.gif HTTP/1.1  | 200 1653|
| 195.175.212.118 | 2002-01-31 | 00:02:14 | +0200 | GET | academic.htm HTTP/1.1| |200 20599|
| 195.175.212.118 | 2002-01-31 | 00:02:26 | +0200 | GET | gif/zemin.gif HTTP/1.1 |200 35011|
| ppp-ankara-nas1-| 2002-01-31 | 00:04:42 | +0200 | GET | HTTP/1.1               |  304 -  |
| ppp-ankara-nas1-| 2002-01-31 | 00:04:43 | +0200 | GET | gif/gn1.gif HTTP/1.1   |  304 -  |
| ppp-ankara-nas1-| 2002-01-31 | 00:04:45 | +0200 | GET | gif/gn2.gif HTTP/1.1   |  304 -  |
| ppp-ankara-nas1-| 2002-01-31 | 00:04:46 | +0200 | GET | gif/office1.gif HTTP/1.|  304 -  |
| ppp-ankara-nas1-| 2002-01-31 | 00:04:46 | +0200 | GET | gif/office2.gif HTTP/1.|  304 -  |
+-----------------+------------+----------+-------+-----+------------------------+---------+
50 rows in set (2 min 53.29 sec)
```

Figure 4.25. Fragment of Frequent_logfile

A table containing only use_no and page_no was created where the hits were frequent ones, Figure 4.26. Figure 4.27 show first 25 rows of this table. The work done in this step corresponds to the generation of $\overline{C}k$ (candidate k-itemset with TID's) mentioned in Figure 4.18. where the TID's are use_no in our case.

```
create table frequent_singles
        select D.use_no, F.page_no
        from daily_use D, frequent_iyte_web F, frequent_logfile L
        where D.user = L.user AND
                D.date = L.ldate AND
                L.url = F.url;
```

Figure 4.26. Creation of Frequent_singles Table

```
Logging to file 'frequent_singles;'
mysql> select * from frequent_singles limit 25;
+--------+---------+
| use_no | page_no |
+--------+---------+
|     81 |     105 |
|     81 |     108 |
|     81 |      35 |
|     81 |      19 |
|     81 |      59 |
|     81 |     100 |
|     81 |      72 |
|     81 |      46 |
|     81 |      58 |
|     81 |      14 |
|     81 |      32 |
|     81 |      34 |
|     81 |      33 |
|     81 |      59 |
|     81 |      72 |
|     81 |      46 |
|     81 |      87 |
|     81 |      21 |
|     81 |      14 |
|     81 |      58 |
|     81 |      87 |
|     81 |      49 |
|     81 |       1 |
|     81 |     108 |
|     81 |     105 |
+--------+---------+
25 rows in set (0.00 sec)
```

Figure 4.27. Fragment of Frequent_singles

4. Pairs of page_no's of the same use_no were formed in table frequent_pairs. This indicated each pair of URL visits done by the same user on the same day, Figure 4.28. As seen in Figure 4.29. daily use of number 110001, 110002 and 110003 contained pairs of pages 2, 2 and 46 consecutively.

```
create table frequent_pairs
        select E.use_no, E.page_no AS page_no1, F.page_no AS page_no2
        from frequent_singles E, frequent_singles F
        where  E.use_no = F.use_no AND
                E.page_no <> F.page_no;
```

Figure 4.28. Creation of Frequent_pairs Table

```
Logging to file 'frequent_pairs;'
mysql> select * from frequent_pairs limit 50;
+--------+---------+----------+
| use_no | page_no | page_no2 |
+--------+---------+----------+
| 110001 |       2 |       67 |
| 110001 |      67 |        2 |
| 110002 |      16 |       55 |
| 110002 |      55 |       16 |
| 110003 |      15 |       50 |
| 110003 |      15 |       54 |
| 110003 |      15 |       81 |
| 110003 |      15 |        5 |
| 110003 |      15 |       38 |
| 110003 |      15 |       24 |
| 110003 |      15 |       40 |
| 110003 |      15 |       90 |
| 110003 |      15 |       70 |
| 110003 |      15 |        5 |
| 110003 |      15 |       22 |
| 110003 |      15 |       50 |
| 110003 |      15 |       54 |
| 110003 |      15 |       81 |
| 110003 |      15 |       40 |
| 110003 |      15 |       73 |
| 110003 |      15 |       60 |
| 110003 |      15 |      101 |
| 110003 |      15 |       73 |
| 110003 |      15 |       22 |
| 110003 |      15 |      101 |
| 110003 |      15 |       73 |
| 110003 |      15 |       36 |
| 110003 |      15 |       50 |
| 110003 |      15 |       54 |
+--------+---------+----------+
50 rows in set (0.00 sec)
```

Figure 4.29. Fragment of Frequent_pairs Table

4.3.4. Apriori Queries

This is the last step in our study. We've found out the pair of pages frequently visited at the same daily use.

```
Logging to file 'sonki'
mysql> select A.page_no, B.url, A.page_no2, C.url, A.counter
    -> from frequent_pair_counts A, frequent_urls B, frequent_urls C
    -> where A.page_no = B.page_no AND
    ->           A.page_no2 = C.page_no AND
    ->                A.page_no <> 1 AND
    ->                     A.page_no2 <> 1 AND
    ->                          B.url NOT LIKE '%.gif%' AND
    ->                               C.url NOT LIKE '%.gif%' AND
    ->                                    B.url NOT LIKE '%.jpg%'
AND
    ->                C.url NOT LIKE '%.jpg%' AND
    ->                B.url NOT LIKE '%.cgi%' AND
    ->                C.url NOT LIKE '%.cgi%'
    -> order by counter desc
    -> limit 30;
+---------+------------------------------------------+----------+------------------------------------------+---------+
| page_no | url                                      | page_no2 | url                                      | counter |
+---------+------------------------------------------+----------+------------------------------------------+---------+
|     114 | mechweb/facultyAndStaff/personnelTinyPic |      115 | mechweb/mainPagePictures/randomizedPictu |   51246 |
|     115 | mechweb/mainPagePictures/randomizedPictu |      114 | mechweb/facultyAndStaff/personnelTinyPic |   51246 |
|       2 | HTTP/1.1                                 |      114 | mechweb/facultyAndStaff/personnelTinyPic |   30678 |
|     114 | mechweb/facultyAndStaff/personnelTinyPic |        2 | HTTP/1.1                                 |   30678 |
|       2 | HTTP/1.1                                 |        6 | academic.htm HTTP/1.1                    |   12751 |
|       6 | academic.htm HTTP/1.1                    |        2 | HTTP/1.1                                 |   12751 |
|       2 | HTTP/1.1                                 |      113 | iyte-services.htm HTTP/1.1               |   12131 |
|     113 | iyte-services.htm HTTP/1.1               |        2 | HTTP/1.1                                 |   12131 |
|       6 | academic.htm HTTP/1.1                    |      114 | mechweb/facultyAndStaff/personnelTinyPic |   10707 |
|     114 | mechweb/facultyAndStaff/personnelTinyPic |        6 | academic.htm HTTP/1.1                    |   10707 |
|       2 | HTTP/1.1                                 |      115 | mechweb/mainPagePictures/randomizedPictu |    9264 |
|     115 | mechweb/mainPagePictures/randomizedPictu |        2 | HTTP/1.1                                 |    9264 |
|       5 | academic.htm HTTP/1.0                    |      114 | mechweb/facultyAndStaff/personnelTinyPic |    3890 |
|     114 | mechweb/facultyAndStaff/personnelTinyPic |        5 | academic.htm HTTP/1.0                    |    3890 |
|     113 | iyte-services.htm HTTP/1.1               |      114 | mechweb/facultyAndStaff/personnelTinyPic |    3589 |
|     114 | mechweb/facultyAndStaff/personnelTinyPic |      113 | iyte-services.htm HTTP/1.1               |    3589 |
|       6 | academic.htm HTTP/1.1                    |      115 | mechweb/mainPagePictures/randomizedPictu |    2562 |
|     115 | mechweb/mainPagePictures/randomizedPictu |        6 | academic.htm HTTP/1.1                    |    2562 |
|       2 | HTTP/1.1                                 |      112 | iyte-services.htm HTTP/1.0               |    2360 |
|     112 | iyte-services.htm HTTP/1.0               |        2 | HTTP/1.1                                 |    2360 |
|       2 | HTTP/1.1                                 |        5 | academic.htm HTTP/1.0                    |    1514 |
|       5 | academic.htm HTTP/1.0                    |        2 | HTTP/1.1                                 |    1514 |
|       5 | academic.htm HTTP/1.0                    |      115 | mechweb/mainPagePictures/randomizedPictu |    1476 |
|     115 | mechweb/mainPagePictures/randomizedPictu |        5 | academic.htm HTTP/1.0                    |    1476 |
|       6 | academic.htm HTTP/1.1                    |      113 | iyte-services.htm HTTP/1.1               |    1176 |
|     113 | iyte-services.htm HTTP/1.1               |        6 | academic.htm HTTP/1.1                    |    1176 |
|     112 | iyte-services.htm HTTP/1.0               |      114 | mechweb/facultyAndStaff/personnelTinyPic |    1154 |
|     114 | mechweb/facultyAndStaff/personnelTinyPic |      112 | iyte-services.htm HTTP/1.0               |    1154 |
|     113 | iyte-services.htm HTTP/1.1               |      115 | mechweb/mainPagePictures/randomizedPictu |     929 |
|     115 | mechweb/mainPagePictures/randomizedPictu |      113 | iyte-services.htm HTTP/1.1               |     929 |
+---------+------------------------------------------+----------+------------------------------------------+---------+
30 rows in set (0.44 sec)
```

Figure 4.30. Fragment of frequent pairs query

The query shown with its SQL command and the result in Figure 4.30. retrieved the most frequent pairs ignoring the ones with cgi, jpg and gif extensions since they apparently were only the components of the page view and also the first page because the first page was the entry point to the site and it was misleading to assume it to be most frequent. The pair of pages ***(114,115) (2,114) (2,6)*** seems to be frequently visited.

```
mysql> select A.page_no, B.url, A.page_no2, C.url, A.support, A.confidence
   -> from final A, frequent_urls B, frequent_urls C
   -> where A.page_no = B.page_no AND
   ->               A.page_no2 = C.page_no AND order by confidence desc limit 150;
+------+----------------------------------------+------+----------------------------------------+-------------+---------+
|hit_no| url                                    |hit_no| url                                    | support     |confiden.|
+------+----------------------------------------+------+----------------------------------------+-------------+---------+
|  116 | robots.txt HTTP/1.0                    |    2 | HTTP/1.1                               | 1.61845e-005|  1.6129 |
|  116 | robots.txt HTTP/1.0                    |    5 | academic.htm HTTP/1.0                  | 0.000166469 |  1.6129 |
|  116 | robots.txt HTTP/1.0                    |  112 | iyte-services.htm HTTP/1.0             | 0.00018381  |  1.6129 |
|  116 | robots.txt HTTP/1.0                    |  113 | iyte-services.htm HTTP/1.1             | 6.93621e-006|  1.6129 |
|    6 | academic.htm HTTP/1.1                  |    2 | HTTP/1.1                               | 0.0147406   | 1.04167 |
|    6 | academic.htm HTTP/1.1                  |    5 | academic.htm HTTP/1.0                  | 0.000224271 | 1.04167 |
|    6 | academic.htm HTTP/1.1                  |  112 | iyte-services.htm HTTP/1.0             | 0.000109823 | 1.04167 |
|    6 | academic.htm HTTP/1.1                  |  113 | iyte-services.htm HTTP/1.1             | 0.0013595   | 1.04167 |
|    6 | academic.htm HTTP/1.1                  |  114 | mechweb/facultyAndStaff/personnelTinyPic | 0.0123777 | 1.04167 |
|    6 | academic.htm HTTP/1.1                  |  115 | mechweb/mainPagePictures/randomizedPictu | 0.00296176 | 1.04167 |
|  114 | mechweb/facultyAndStaff/personnelTinyPic |  2 | HTTP/1.1                               | 0.0354648   | 1.04167 |
|  114 | mechweb/facultyAndStaff/personnelTinyPic |  5 | academic.htm HTTP/1.0                  | 0.00449698  | 1.04167 |
|  114 | mechweb/facultyAndStaff/personnelTinyPic |  6 | academic.htm HTTP/1.1                  | 0.0123777   | 1.04167 |
|  114 | mechweb/facultyAndStaff/personnelTinyPic | 112 | iyte-services.htm HTTP/1.0             | 0.00133406  | 1.04167 |
|  114 | mechweb/facultyAndStaff/personnelTinyPic | 113 | iyte-services.htm HTTP/1.1             | 0.00414901  | 1.04167 |
|  114 | mechweb/facultyAndStaff/personnelTinyPic | 115 | mechweb/mainPagePictures/randomizedPictu | 0.0592422 | 1.04167 |
|  115 | mechweb/mainPagePictures/randomizedPictu |  2 | HTTP/1.1                               | 0.0107095   | 1.04167 |
|  115 | mechweb/mainPagePictures/randomizedPictu |  5 | academic.htm HTTP/1.0                  | 0.00170631  | 1.04167 |
|  115 | mechweb/mainPagePictures/randomizedPictu |  6 | academic.htm HTTP/1.1                  | 0.00296176  | 1.04167 |
|  115 | mechweb/mainPagePictures/randomizedPictu | 112 | iyte-services.htm HTTP/1.0             | 0.000786104 | 1.04167 |
|  115 | mechweb/mainPagePictures/randomizedPictu | 113 | iyte-services.htm HTTP/1.1             | 0.00107396  | 1.04167 |
|  115 | mechweb/mainPagePictures/randomizedPictu | 114 | mechweb/facultyAndStaff/personnelTinyPic | 0.0592422 | 1.04167 |
+------+----------------------------------------+------+----------------------------------------+-------------+---------+
```

Figure 4.31. Fragment of the Query Pairs with Support & Confidence

The query of Figure 4.31. worked on a temporary table final and showed the support and confidence values of some frequent_pairs. The pair of page 19 and 2 has be seen 1.6 % of the total logfile records. In all the pairs having page 19 only 2.43 % of records had also page 2.

Chapter 5

CONCLUSION

5.1. Remarks on the Study

This study is first investigation conducted in IYTE Computer Engineering Department over the topic of web mining. As a start a long site survey has been done in order to get familiar with the terminology and the research issues related with the fields of data mining, data warehousing, text mining and web mining. Benchmarking of the freeware or available software of data mining and log analyzers was the following step. Access log of the IYTE web server seemed suitable for mining since it was reachable, raw and huge. Preliminary analysis, requirement analysis, data analysis and process analysis of the project IYTE WUM System were some steps of the study before starting coding. Java was chosen to be the environment for data cleaning and preparation. Mysql was chosen to be data base management system. Both of the tools were shareware and portable in nature. During coding and testing the speed and storage capacity of the development environment was a constraint. This study outlined the borders of the future work in terms of context and the infrastructure.

5.2. Evaluation of the IYTE WUM System

The system developed is a ***prototype*** to demonstrate an application example for the topic of general web usage mining together with an application of a data mining algorithm with a database approach. Through the development of this system it is clearly seen the steps of a web mining system architecture; data preparation, data description, application of data mining techniques, knowledge extraction, evaluation of the results. This experience also revealed all the difficulties mentioned in the previous research studies such as the data being abundant, complex and unstructured. Requirements of the development process as manpower, infrastructure, time and development tools can be made much more clear for future work.

5.3. Future Work

Although some of the features subject to this section were mentioned in the initial system analysis studies of the IYTE WUM System, they had to be skipped and left for future work since the infrastructure requirements were more than initially anticipated. The following are our suggestions for future work, which we believe, will make the system easier to use and bring more benefits to the web administrator.

- ***Graphical user interface***: User interface can be added. This interface can be composed of two main functions;
  1. Adjustment of log cleaning parameters: User can indicate the time interval, user(s), service or URL types to be analyzed. That shortens the time for loading the database and the size of the query results. In short special focus research can be done on the logs.
  2. Design of queries: User can chose the attributes, conditions to be met, order, grouping and aggregates in order to use the full power of SQL.
- ***Site crawler:*** A mechanism, which parses the site periodically and maintains a site-URL table in the IYTE WUM database can also be an addition to the existing system. This will allow the system to report the links never visited or the dead-ends which cannot be reached from any existing links. The web administrator also becomes aware of all the changes done in the site.
- ***Data mining algorithms***: Data mining rules other than association rules can be applied to the data present in IYTE WUM relational database to increase the amount of knowledge extracted. A time series analysis might reveal the changes in user behaviors, a personalization approach can be studied in order to offer users the pages or services, which they would be interested in.

- ***Integration to IYTE web server***: This system can be made integrated part of the web server in the campus network. All additions to the log files each day can be processed as they occur or periodically. Dynamic and up-to-date information can be supplied to the web administrator being an example of automatic warning mechanism.
- ***Other DBMS's***: The tool used in the subject application was Mysql, which did not have features for the execution of nested SQL statements. This limitation brought the necessity of creating temporary tables and the need for housekeeping in the database. So for new versions of the system, other DBMS's can be chosen.
- ***Performance evaluation and multi threading approach:*** A benchmark study can be done for each query, indexes and tuning performance parameters of the DBMS. Multi threading approach may increase the efficiency.

[9] **O. Zaiane, J. Han, Z. Li, S.H. Chee, J.Y. Chiang,** MultiMediaMiner: A System Prototype for MultiMedia Data Mining. Intelligent Database Systems Research Laboratory and Vision and Media Laboratory, School of Computing Science, Simon Fraser University, (1998)

[10] **M.-S. Chen, J.S. Park, and P.S. Yu,** Efficient data mining for path traversal patterns. Knowledge and Data Engineering, 10(2):209-221, (1998)

[11] **O. Zaiane, M. Xin, and J. Han,** Discovering web access patterns and trends by applying olap and data mining technology on web logs. In Advances in Digital Libraries, pages 19-29, (April, 1998)

[12] **M. Perkowitz and O. Etzioni,** Adaptive Sites: Automatically synthesizing web pages. In Proc. Of the Fifteenth National Conference on Artificial Intelligence, pages 727-732, (July, 1998)

[13] **Y. Fu, K. Sandhu, and M. Shih,** Clustering of web users based on access patterns. In Proceedings of the 1999 KDD Workshop on Web Mining, (1999)

[14] **C. Shahabi, A.Zarkesh, J. Adibi, and V. Shah,** Knowledge discovery from users web-page navigation. In Proceedings of IEEE RIDE '97 Workshop, (April, 1997)

[15] **M. K. S. Schecter and M. Smith,** Using path profiles to predict HTTP request. In Proceedings of 7th International WWW Conference, (1998)

[16] **E. Satiroglu,** Mining User Access Patterns and Identity Information from Web Logs for Effective Personalization, Msc. Thesis, Bilkent University, (September, 2001)

[17] **T.Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal,** From user access patterns to dynamic hypertext linking. Computer Networks, 28(7):1007-10014, (May, 1996)

[18] **R. Agrawal and A. Srikant,** Fast algorithms for mining association rules. Proc. VLDB'94, pp487-499, (1994)

[19] **Han, J., Pei, J. and Yin, Y.,** Mining Frequent Patterns without Candidate Generation. Proc. of the ACM SIGMOD, int. conf. on Management of Data, pp1-12, (2000)

APPENDIX

A1. AccessLogRead( ) Class

```
import java.io.*;

public class AccessLogRead {
  public static void main(String[] args) {

  String newLine;
  String user;
  String date;
  String time;
  String timeRemaining;
  String demand;
  String url;
  String urlMain;
  String status;
  String statusMain;
  String oldMonth = "   ";
  String newMonth = "  ";

  int logRecordNo  = 0;
  int lengthOfLine = 0;

  System.setProperty("lineSeparator", "&");

       try {
      DataInputStream in =
        new DataInputStream(
         new BufferedInputStream(
          new FileInputStream("accesslog")));

        DataOutputStream out =
                   new DataOutputStream(
                    new BufferedOutputStream(
                     new FileOutputStream("accesslogfile")));

       while (in.available() != 0)
            try {
                   newLine             = in.readLine();
                   lengthOfLine        = newLine.length();
                   logRecordNo         = logRecordNo + 1;
                   int normalLogLine   = 0;
                   try {
                          int ptr1     = newLine.indexOf("[");
                          int ptr2     = newLine.indexOf(":");
                          int ptr3     = newLine.indexOf("+");
                          int ptr4     = newLine.indexOf("]");
                          int ptr5     = newLine.indexOf('"');
                          int ptr6     = newLine.lastIndexOf('"');

                          user         =
                                       newLine.substring(0,newLine.indexOf("[")
                                       - 5);
                          date         = newLine.substring(ptr1 + 1,  ptr1 + 12);
            // Date format conversion from dd/mmm/yyyy to yyyy-mm-dd
                  oldMonth = date.substring(3,6);
                  newMonth = "??";
                  if(oldMonth.compareTo("Jan") == 0) newMonth = "01";
                  if(oldMonth.compareTo("Feb") == 0) newMonth = "02";
                  if(oldMonth.compareTo("Mar") == 0) newMonth = "03";
                  if(oldMonth.compareTo("Apr") == 0) newMonth = "04";
                  if(oldMonth.compareTo("May") == 0) newMonth = "05";
                  if(oldMonth.compareTo("Jun") == 0) newMonth = "06";
                  if(oldMonth.compareTo("Jul") == 0) newMonth = "07";
                  if(oldMonth.compareTo("Aug") == 0) newMonth = "08";
                  if(oldMonth.compareTo("Sep") == 0) newMonth = "09";
                  if(oldMonth.compareTo("Oct") == 0) newMonth = "10";
                  if(oldMonth.compareTo("Nov") == 0) newMonth = "11";
                  if(oldMonth.compareTo("Dec") == 0) newMonth = "12";

                  date         = date.substring(7, 11) + "-" + newMonth
                  + "-" +
                                     date.substring(0,2);
                  time         = newLine.substring(ptr2 + 1,  ptr2 + 9);

                  timeRemaining= newLine.substring(ptr3,    ptr4)
                  demand        = newLine.substring(ptr5 + 1,  ptr5 + 4);
                  url           = newLine.substring(ptr5 + 6,  ptr6) ;
                  urlMain       = url.substring(0, url.indexOf("/"));
                  if (urlMain.indexOf(" ") != -1) urlMain =
                  urlMain.substring(0,
                        urlMain.indexOf(" "));
                  if (urlMain.indexOf(".") != -1) urlMain =
                  urlMain.substring(0, urlMain.indexOf("."));
                  status              = newLine.substring(ptr6 + 1,
                        lengthOfLine) ;
                  statusMain    = status.substring(1, 4);

                  String x = System.getProperty("lineSeparator");
                  out.writeBytes(user + '|' + date + '|' + time + '|' +
                  timeRemaining +'|' + demand + '|' + urlMain + '|' + url
                  + '|' + statusMain +  '|'+ status + '|');


            } catch (IndexOutOfBoundsException s)
            {
            System.err.println("String Exception");
            }
    } catch (IOException e)
    {
      System.err.println("New IO Exception");
    }
    } catch (IOException e) {
  System.err.println("IOException");
  }
 }
}
```

A2. ErrorLogRead( ) Class

```
import java.io.*;


public class ErrorLogRead {
  public static void main(String[] args) {

    String newLine;
    String date;
    String time;
    String messageDesc;

    String dateAndTime;
    String oldMonth = "   ";
    String newMonth = "  ";

    int logRecordNo  = 0;
    int lengthOfLine = 0;

    System.setProperty("lineSeparator", "&");

         try {
         DataInputStream in =
           new DataInputStream(
            new BufferedInputStream(
             new FileInputStream("errorlog.TR-ERROR_LOG")));

           DataOutputStream out =
                      new DataOutputStream(
                       new BufferedOutputStream(
                        new FileOutputStream("errorlogfile")));

         while (in.available() != 0)
                try {
                      newLine               = in.readLine();
                      lengthOfLine          = newLine.length();
                      logRecordNo           = logRecordNo + 1;
                      int normalLogLine     = 0;
                      try {
                            int ptr1        = newLine.indexOf("[");
                            int ptr4        = newLine.indexOf("]");

                            // format in the log mmm dd tt:mm:ss yyyy

                            dateAndTime     = newLine.substring(ptr1 + 4, ptr4);
                            oldMonth        = dateAndTime.substring(0,4);
                            oldMonth        = oldMonth.trim();

                            newMonth = "??";
                            if(oldMonth.compareTo("Jan") == 0) newMonth = "01";
                            if(oldMonth.compareTo("Feb") == 0) newMonth = "02";
                            if(oldMonth.compareTo("Mar") == 0) newMonth = "03";
                            if(oldMonth.compareTo("Apr") == 0) newMonth = "04";
                            if(oldMonth.compareTo("May") == 0) newMonth = "05";
                            if(oldMonth.compareTo("Jun") == 0) newMonth = "06";
                            if(oldMonth.compareTo("Jul") == 0) newMonth = "07";
                            if(oldMonth.compareTo("Aug") == 0) newMonth = "08";
                            if(oldMonth.compareTo("Sep") == 0) newMonth = "09";
                            if(oldMonth.compareTo("Oct") == 0) newMonth = "10";
                            if(oldMonth.compareTo("Nov") == 0) newMonth = "11";
                            if(oldMonth.compareTo("Dec") == 0) newMonth = "12";
```

```
                                // new date format yyyy-mm-dd
                                date   = dateAndTime.substring(dateAndTime.length() -
                                5, dateAndTime.length()) + "-" + newMonth + "-" +
                                                    dateAndTime.substring(5,7);
                                time          = dateAndTime.substring(7,  16);
                                messageDesc  = (newLine.substring(ptr4 + 1,
                                lengthOfLine)).trim();

                                String x = System.getProperty("lineSeparator");
                                      out.writeBytes(date + '|' + time + '|' +
                                messageDesc +
                                '|' + System.getProperty("lineSeparator"));

                          } catch (IndexOutOfBoundsException s)
                          {
                          System.err.println("String Exception");
                          }
            } catch (IOException e)
            {
              System.err.println("New IO Exception");
            }
            } catch (IOException e) {
          System.err.println("IOException");
          }
        }
      }
```

A3. IYTE WUM Relational Database Schema

```
mysql> show tables;
+----------------------+
| Tables_in_wum        |
+----------------------+
| daily_use            |
| department           |
| errors               |
| final                |
| frequent_iyte_web    |
| frequent_logfile     |
| frequent_pair_counts |
| frequent_pairs       |
| frequent_singles     |
| frequent_urls        |
| iyte_web             |
| logfile              |
| top_daily_use        |
| top_frequent         |
| ttx                  |
| tty                  |
| users                |
+----------------------+
17 rows in set (0.00 sec)
```

```
mysql> describe department;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| depcode  | int(11)     | YES  |     | NULL    |       |
| depname  | varchar(40) | YES  |     | NULL    |       |
| fakulte  | varchar(30) | YES  |     | NULL    |       |
| start_ip | varchar(25) | YES  |     | NULL    |       |
| end_ip   | varchar(25) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

```
mysql> describe logfile;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| user        | varchar(40) | YES  |     | NULL    |       |
| ldate       | date        | YES  |     | NULL    |       |
| ltime       | time        | YES  |     | NULL    |       |
| rtime       | varchar(15) | YES  |     | NULL    |       |
| demand      | varchar(5)  | YES  |     | NULL    |       |
| url_main    | varchar(15) | YES  |     | NULL    |       |
| url         | varchar(40) | YES  |     | NULL    |       |
| status_main | char(3)     | YES  |     | NULL    |       |
| status      | varchar(15) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
9 rows in set (0.00 sec)
```

```
mysql> describe errors;
+---------+------------+------+-----+---------+-------+
| Field   | Type       | Null | Key | Default | Extra |
+---------+------------+------+-----+---------+-------+
| date    | date       | YES  |     | NULL    |       |
| time    | time       | YES  |     | NULL    |       |
| message | varchar(80) | YES  |     | NULL    |       |
+---------+------------+------+-----+---------+-------+
3 rows in set (0.00 sec)


mysql> describe users;
+-----------+------------+------+-----+---------+-------+
| Field     | Type       | Null | Key | Default | Extra |
+-----------+------------+------+-----+---------+-------+
| user_name | varchar(30) | YES  |     | NULL    |       |
| type      | varchar(10) | YES  |     | NULL    |       |
| user_ip   | varchar(25) | YES  | MUL | NULL    |       |
| depcode   | int(11)    | YES  |     | NULL    |       |
+-----------+------------+------+-----+---------+-------+
4 rows in set (0.00 sec)


mysql> describe daily_use;
+-----------+------------+------+-----+---------+----------------+
| Field     | Type       | Null | Key | Default | Extra          |
+-----------+------------+------+-----+---------+----------------+
| use_no    | int(11)    |      | PRI | NULL    | auto_increment |
| user      | varchar(40) | YES  | MUL | NULL    |                |
| date      | date       | YES  |     | NULL    |                |
| url_count | int(11)    | YES  |     | NULL    |                |
+-----------+------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)


mysql> describe iyte_web;
+---------+------------+------+-----+---------+----------------+
| Field   | Type       | Null | Key | Default | Extra          |
+---------+------------+------+-----+---------+----------------+
| page_no | int(11)    |      | PRI | NULL    | auto_increment |
| url     | varchar(40) | YES  |     | NULL    |                |
| counter | int(11)    | YES  |     | NULL    |                |
+---------+------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

```
mysql> describe frequent_logfile;
+-------------+------------+------+-----+---------+-------+
| Field       | Type       | Null | Key | Default | Extra |
+-------------+------------+------+-----+---------+-------+
| user        | char(40)   | YES  |     | NULL    |       |
| ldate       | date       | YES  |     | NULL    |       |
| ltime       | time       | YES  |     | NULL    |       |
| rtime       | char(15)   | YES  |     | NULL    |       |
| demand      | char(5)    | YES  |     | NULL    |       |
| url_main    | char(15)   | YES  |     | NULL    |       |
| url         | char(40)   | YES  |     | NULL    |       |
| status_main | char(3)    | YES  |     | NULL    |       |
| status      | char(15)   | YES  |     | NULL    |       |
| urlx        | char(40)   | YES  |     | NULL    |       |
| counter     | bigint(21) |      |     | 0       |       |
+-------------+------------+------+-----+---------+-------+
11 rows in set (0.00 sec)

mysql> describe frequent_iyte_web;
+---------+------------+------+-----+---------+----------------+
| Field   | Type       | Null | Key | Default | Extra          |
+---------+------------+------+-----+---------+----------------+
| urlx    | char(40)   | YES  | MUL | NULL    |                |
| counter | bigint(21) |      |     | 0       |                |
| page_no | int(11)    |      | PRI | NULL    | auto_increment |
+---------+------------+------+-----+---------+----------------+
3 rows in set (0.05 sec)

mysql> describe frequent_singles;
+---------+---------+------+-----+---------+-------+
| Field   | Type    | Null | Key | Default | Extra |
+---------+---------+------+-----+---------+-------+
| use_no  | int(11) |      | MUL | 0       |       |
| page_no | int(11) |      |     | 0       |       |
+---------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> describe frequent_pairs;
+----------+---------+------+-----+---------+-------+
| Field    | Type    | Null | Key | Default | Extra |
+----------+---------+------+-----+---------+-------+
| use_no   | int(11) |      |     | 0       |       |
| page_no  | int(11) |      |     | 0       |       |
| page_no2 | int(11) |      |     | 0       |       |
+----------+---------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

```
mysql> describe final;
+------------+------------+------+-----+---------+-------+
| Field      | Type       | Null | Key | Default | Extra |
+------------+------------+------+-----+---------+-------+
| page_no    | int(11)    |      |     | 0       |       |
| page_no2   | int(11)    |      |     | 0       |       |
| counter    | bigint(21) |      |     | 0       |       |
| support    | float      | YES  |     | NULL    |       |
| confidence | float      | YES  |     | NULL    |       |
| total      | int(11)    | YES  |     | NULL    |       |
| sayac      | bigint(21) |      |     | 0       |       |
| pair_count | bigint(21) |      |     | 0       |       |
+------------+------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

// Table descriptions which are not shown in this document are temporary tables.