Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

Querying and Browsing the Resources in Internet

Yi-Hung Wu Yen-Huan Chen Arbee L.P. Chen

Department of Computer Science

National Tsing Hua University

Hsinchu, Taiwan 300, R.O.C.

E-mail: alpchen@cs.nthu.edu.tw

Abstract

Building a system for the users on the Internet to search interested resources or to provide guided browsing services has formed an important trend as a result of the rapid accumulation of information, the booming development of information system and the increasing requirement of resource sharing. In this paper, we present the Integrated resource Query and guided Browsing System (IQBS), which has been implemented at our laboratory, to achieve this goal. The scheme we proposed, which integrates the resources in Internet with the database querying, can take advantage of the strong power of the database system to structure and query data, and greatly promote the quality and performance of searching resources in Internet. In addition, mechanisms are designed to guide the users on the Internet to conveniently browse resources with a reduced traffic cost.

1. Introduction

A few years ago, the researchers usually gathered a lot of bibliographic data or inquired other researchers about related fields before they started their research plan. Nowadays, the advanced techniques on the Internet can almost provide all these services. However, the dramatic growth of information systems over the past years has brought about the rapid accumulation of information, so the Internet resource discovery problem [ODL93] has become a very important issue. Many tools for resource discovery have recently appeared on the Internet to help users find the right positions of interested resources (see Section 2).However, many of these tools essentially keep a keyword-based index of the available resources. This approach clearly does not fit the user requests well. Our approach is to provide an integrated tool that helps users browse, search, and organize the information in Internet. As for a roving user on the Internet, s/he can pass through the complicated network smoothly by the guided browsing information on the browser, and find the expected resources quickly by the indexing mechanism. And besides, through the way to store the metadata of each resource and to organize them, the user can also save many extra costs for repeated operations on the Internet.

The *World-Wide Web* (WWW) [BCL94] combines the *Uniform Resource Locator* (URL) with the technology of *hypertext* to organize the resources in Internet into a distributed hypertext system. Each node within this environment can be regarded as a single document (resource). Under the client-server architecture, the user can make a search on a server according to the full-text index for a specific area. Moreover, the user can also roam through the information world by means of the *anchor* information provided on each node. Aside from the *HyperText Transfer Protocol* [HTTP], the communication between clients and servers of the WWW can also support many other network protocols (e.g., FTP, NNTP). As the WWW can cover a variety of applications, there are more and more browsers designed for it. Hence the number of WWW servers which have registered on the Internet has exceeded those of other information systems. Our main goal is to help the users who are browsing on the WWW to find the right locations of the related homepages. In our approach to analyzing the resources of the WWW, we apply the techniques of multidatabase resource discovery to the Internet resource discovery problem and try to solve the following issues effectively:

- How to integrate a variety of resources in Internet?
- How to precisely represent the semantics of these resources?
- How to store the metadata of the related resources?
- How to query and request some interested resources?
- How to process the user queries efficiently?

Owing to the hypertext structure, the WWW contains a lot of information about anchors. As far as a resource maintainer is concerned, the semantics of an anchor has to sufficiently express the semantics of the homepage pointed to by it. As to the viewpoint of a browsing user, the motivation to follow an anchor is based on the fact that the anchor expresses interested semantics for the user, and such semantics also sketches the pattern of the interested homepages for the user. As above, we can make a proper connection between the interested homepages for the user and those truly related homepages. Furthermore, we can cluster the anchor semantics by the *machine learning* techniques, and then generate the *classification hierarchy* for the homepages pointed to by these anchors. In the IQBS, the *resource discovery query subsystem* provides users to query their interested resources exactly by the establishment of this classification hierarchy. The metadata of the homepage itself and the linking relations among the homepages can also be taken for the conditions for querying. The construction of a classification hierarchy can be divided into two phases: At first, we adopt both the active search engine and the passive registration form to collect the resources on the WWW. Second, we classify these resources as to the

decision rules mainly derived from the semantic analysis. According to the comparison of the anchors to the decision rules, we can decide the classes for those homepages pointed to by some anchors. At last, each homepage can be placed into a data storage system as to the class that it belongs, and the contents for storage are only the metadata which can be taken as the query results.

By the WWW browsers, we can conveniently ramble over the WWW and take a look at some interested information. Unfortunately, the WWW provides such flexible services that many users have to spend a lot of time roaming about uninterested homepages. In the IQBS, the guided browsing service subsystem develops two services to prevent the users from passing meaningless nodes frequently. One is the *footmark* which records all the paths for each user and then obtains some statistics. The other is the *roadsign* which provides a service for the user on any homepage to look up which class of homepages the anchors can point to. The design about roadsign has never before, to our knowledge, been discussed in any related documents. After the analysis of user footmarks, the *hot sequences* can be derived from those with high frequency. By this mechanism, the WWW users can save time on repeated operations and quickly find their interested resources. Furthermore, the global hot sequences for all users in a specific field can be used as the guides or references for user browsing. According to the popular paths for the majority, the WWW users can find in order what they want and never get lost in the information sea. The *hot list* provided by the **Mosaic** ™ browser [Mos95] requires the user to decide whether the current homepage has to be recorded or not. There is no sequential relationship between the homepages of the hot list, thus each homepage would be recorded independently. Moreover, there is no information about which field the homepage belongs to. In our approach, the hot sequences put the locations of popular homepages in a specific field into order. And further, the hot sequences of each field are gathered, analyzed and derived by our system in an automatic way.

The paper is organized as follows. Section 3 gives an overview of the IQBS and shows the design of our prototype system. Section 4 introduces the resource discovery query subsystem, including the methodology of resource classification. The guided browsing service subsystem is described in section 5. Section 6 presents conclusions with the comparison with other related approaches.

2. Related work

Under the current environment of Internet, many tools for resource discovery have been recently developed. For example, the *Wide Area Information Servers* system (WAIS) [KM91] provides a simple interface with indexing services for each text database under the client-server architecture. The *Archie* system [ED92] is designed to enable users to search files under the distributed multi-servers architecture. This system regards the *filename-database* as the index, and stores the metadata of the file contents by the *whatis-database*. The resources on the Internet can also be organized into a hierarchical structure by the *Gopher* system [McC92]. The indexes of this system take the form of a directory to display, and enable the users to browse back and forth along the adjacent levels. In the aspect of the *text database discovery* problem, the *GLOSS* approach [GGT94, GG95] helps the users find the text databases related to their requirements. This approach keeps the statistics on the available databases to estimate which databases are potentially the most useful for a given query. Similarly, the *multidatabase resource discovery* approaches [MS95, TC96] regard the attributes of a schema as the main sources of the schema semantics so as to estimate the degree of relevance between any two schemata.

Regarding the improvement to the *hypertext management*, [JF93] proposed a technique to aid users to rove through the hypertext environment. All the paths of old users are gathered and analyzed in order to generate the *suggestion*, which can be regarded as the guiding information for new users. An algorithm presented in [AS95] can mine significant sequences of goods for the customer shopping. Such sequences named *sequential patterns* are derived from the *customer-transaction database* where all goods bought by a customer are recorded in sequence. The problem of getting the popular homepages and their sequences from the user footmarks is similar to the issue of deriving the popular goods and their sequences from the customer shopping. Hence this algorithm can also be used to derive the hot sequences, however, it would not be efficient with regard to a large scale of data. In this research, we use a new algorithm proposed in [YC95] to solve this problem with better performance.

3. System architecture

The IQBS is composed of two components: server and client (see Figure 1). The IQBS server is built on the top of a common server in Internet, and provides a user interface for querying and resource registration. Moreover, we can manage and maintain the metadata of the resources through the connection with a database system. Moreover, the IQBS client extends the functions of the existing browser to provide useful guided browsing services. As for the implementation of our prototype system, we adopt the WWW server developed by [NCSA] as the platform of the IQBS server. The **Mosaic** ™ browser is regarded as the basis of the IQBS client. The function of querying the resources can be provided at the IQBS server by way of drawing up a homepage on the WWW server and building up the connection with the database system. To add the guided browsing services to the client, we can directly trace and modify the source codes of the existing browser. To clarify the methodology of the system design, the following discussions are considered in two different aspects.

3.1. Server design

At the IQBS server, a database system is used as the data storage system to reduce the overhead of the query processing. In addition, the complexity during the data processing can be simplified by the data manipulation language and the query language of a database system. In regard to the data maintenance, such a scheme is also much easier than a file system. On the WWW, the homepages are

edited in the form of the hypertext, and the relations between any two homepages are specified by the descriptions of anchors. Therefore, the relational databases which store data in the form of tables are not suitable for representing the linkage relationships between homepages. In our approach, an object-oriented database system is adopted as the data storage system of the IQBS.

In order to overcome the difficulty on searching the resources in Internet, we undertake an approach to classify the resources in advance. After deciding which class the homepage belongs to, we can construct an object-oriented classification hierarchy (See Figure 2) which contains the metadata of the available resources. Hence the degree of relevance between a user query and the actual query target is promoted to a better level. The IQBS server is divided into the following components: (See the left-hand side of Figure 1)

- *User Interface* provides an input interface for user in the form of a homepage.
- *Collection Manager* gathers and analyzes the resources on the WWW.
- *Hierarchy Manager* constructs and maintains the classification hierarchy.
- *Query Translator* transforms the user queries into the database queries.
- *Common Gateway Interface* (CGI) connects the *user interface* and the database.
- *Database Manager* manipulates the data accesses within the database.

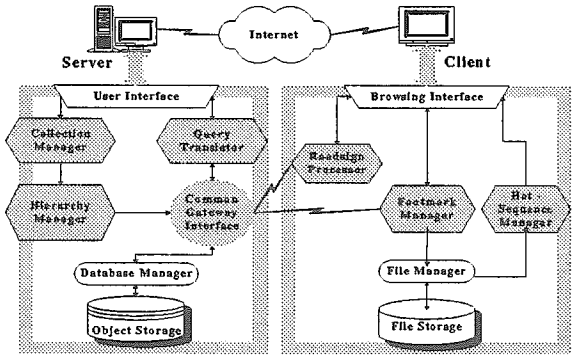

Figure 1. System architecture

3.2. Client design

A general WWW browser only provides a graphic user interface for the users to look at the homepages, but the users are often unable to choose the best anchors for quickly finding what they want. Besides, a general WWW browser only records the single path of user browsing, so there is no way to derive the most popular sequences. Moreover, there is also no information about useful sequences provided for other users. In the IQBS, we extend an existing browser to have some useful messages in order to guide the users. Such useful messages include the roadsign, the footmark and the hot sequence. A roadsign describes the related features of the homepage pointed to by an anchor. To generate the roadsign information, we make use of the *resource discovery query subsystem* directly. After a new browsing request is sent, a *recursive resource query* is sent from the IQBS client to the IQBS server. The query results obtained from the IQBS server are used as the new contents of the roadsign. As to the footmarks and the hot sequences, all the information about user browsing can be recorded at the IQBS client to guide the user. However, the IQBS server can also collect the footmarks of each field to derive the hot sequences of each field. Similarly, the IQBS client is divided into several components: (See the right-hand side of Figure 1)

- *Browsing Interface* adds a new function menu on the existing browser.
- *Roadsign Processor* sends the roadsign request and outputs the query results.
- *Footmark Manager* records and update the content of the footmarks.
- *Hot-sequence Manager* derives and displays the hot sequences.
- *File Manager* controls the data access of the footmark and the hot sequence.

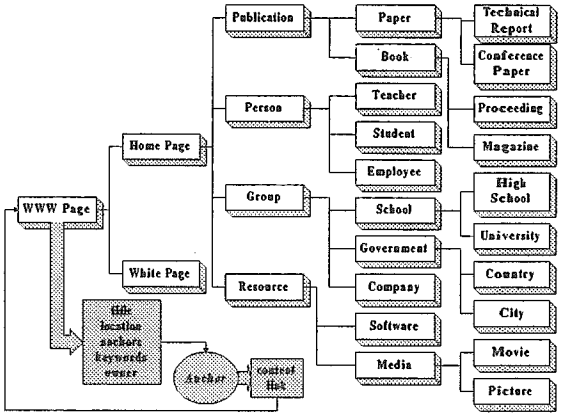

Figure 2. Classification hierarchy

4. Resource discovery query subsystem

This system is established at the IQBS server in order to enable the WWW users to search the resources by querying. It can be divided into three phases as to the different objectives: resource collection, resource classification and resource query processing. The first two phases are required for the sake of constructing the object-oriented classification hierarchy so that each resource can be put into the corresponding class in the data storage system. During the period of the resource collection, the metadata of each resource are gathered. And further, other related information is also provided for the resource classification (See Figure 3). As to the last phase, the design for querying the resources has to cooperate with the data storage system to take advantage of the query processing capability of the database system. In the IQBS, we adopt the UniSQL/X™ Object-Relational database [Uni93] as the major data storage system, so the utilities for the data processing can be directly produced through the *application program interface* (API) provided by the UniSQL/X™. These utilities can be applied respectively to the queries for insertion during the resource collection and classification, the query translation for processing the resource query, and the queries to maintain the hierarchy. In the following subsections, the design for these components are explained in detail.

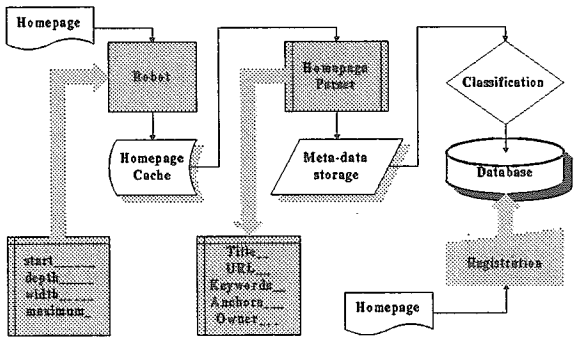

Figure 3. Resource collection

4.1. Resource collection

Before the resource query can be processed, we have to get enough data at first to construct a database system. Generally speaking, there are two ways to get the metadata of each resource in Internet. First, a *robot* program can be used to actively browse the WWW in a continuous way. The robot program usually has the basic functions to filter the documents. The other is to provide a registration service for the resource administrators to fill in the related metadata about the resources. The two ways to collect the information of the resources on the WWW are described as follows:

Robot

In the current database of the IQBS, all the homepages are gathered by the robot from the WWW. This program has two major mechanisms: One is to get the entire content of a homepage according to a given address, and the other is to analyze this homepage in order to obtain all the anchors in it. By way of running the two mechanisms in turn, the robot can continue to get other homepages. After the analysis of a homepage, the robot does not store the original content of this homepage. However, our system records the related metadata such as titles, addresses, anchors, and keywords. The former mechanism uses an existing browser *Lynx* to get the homepage back by a given address, while the latter makes use of the **Mosaic** ™ to obtain all the anchor information by the built-in function calls for processing the *HTML* documents [HTML].

Registration

Though the robot can collect a lot of homepages on the WWW automatically, still some homepages are not available. Such situations usually take place when the homepages are pointed to by very few anchors. Especially, as a new homepage is created, it is almost impossible for the robot to reach such a homepage by browsing continuously. Therefore, a mechanism for registration is provided in order to compensate for lack of the ability to get the unavailable homepages. The registration is done by a *form* which is a useful type of the HTML documents. First of all, the resource administrator fills the registration form with the metadata of a new homepage. Then our system packs up the filled data in the form of an object. At last, the packages are inserted into the database system directly by the API. In the IQBS, all the metadata of the homepages can be collected by combining the above two methods.

4.2. Resource classification

In the ways to collect the resources, the registration approach gets the information about the classes for the resources directly by filling the form, so it is not necessary to classify the registered resources any more. However, the metadata of the homepages gathered by the robot do not include the class information for the resources. Therefore, a classification procedure is needed to decide the class before storing the metadata into the database system. In the design of our classification procedure, we adopt different approaches as to the requirements for the different classes. The classification hierarchy in our system is built in the form of an object-oriented schema and preserves the properties of the *class hierarchy* concept. That is, each class is the generalization of its subclasses, and inherits all the properties of its superclass. Hence our classification procedure follows the way to filter from top to bottom and then approaches the most possible class which the homepage should belong to. Our two approaches to classifying the resources are proposed as follows:

1. semantic analysis

This approach is based on the dependency between an anchor and the homepage pointed to by it. As mentioned above, there are many anchors existing within the hypertext structure. The semantics expressed by the anchor implies the semantics of the homepage pointed to by it, and also describe the interested homepages for the users. The classification hierarchy we proposed is based on the purposes of the homepages, so the implied semantics of the anchor and the actual request for user browsing can be expressed well in the meantime. The anchors can make a correct connection between the user interests and the exactly related homepages. Therefore, to cluster the semantics of the anchors is equivalent to classify the homepages into different classes with different purposes. Under the concept of the hierarchical classification in our approach, each class contains the decision rules of its own and also inherits those of its superclass, so the whole classification procedure can be more efficient. This approach [WC96] is divided into a learning stage and a recognition stage (See Figure 4).

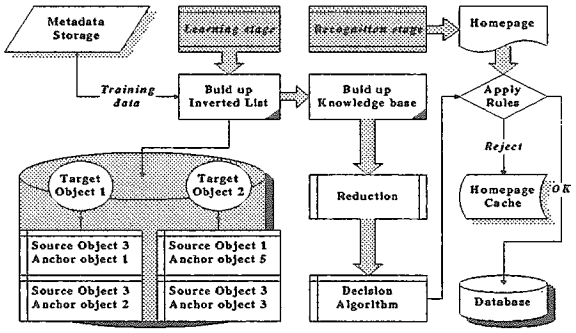

Figure 4. Semantic analysis

Supervised learning

In this stage, the decision rules of each class are derived to form the knowledge for the next recognition stage. At first,

the training data can be gathered by the robot, and further the relationships between anchors and homepages are recorded by a proper data structure. Next, the relationships between the anchors and the homepages pointed to by the anchors are stored in the form of rules after the classes for these homepages are decided by hand. Based on the concepts of the *indiscernibility relation* in the *rough set* theory [PAW91, PAW95], a large amount of rules can be reduced to several decision rules with higher reliability. Furthermore, the uncertainty among some conflicting rules can also be specified. Through the training procedures for *relevance feedback* , our system can obtain more reliable decision rules. The analysis of the anchor information can not avoid the keyword processing in the research field of *information retrieval*. In our design, an anchor is divided into several keywords which are the most significant ones in the anchor. Moreover, the appearance of the keywords are represented by the notation of zero or one (See Figure 5). Similarly, the class information is also coded by numbers so that the procedure to reduce the rules can be transform into a numerical operation and speeded up by using the XOR operations.

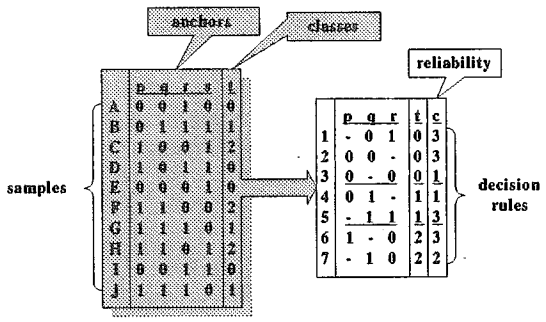

Figure 5. Rule reduction

Automatic recognition

When the robot gets a new homepage, we can apply the current decision rules to find its class directly. As the rule base is constructed step by step, new homepages may have no related rules with enough reliability. Under the circumstances, our system rejects to classify these homepages and caches them in a temporary storage space (e.g., the root of our classification hierarchy). The cached information not only reminds the system administrators of the current performance about the classification, but also provides the resources for the rule base to refine the decision rules.

2. syntactic analysis

As the previous analysis depends on the keywords of the anchors, the derived rules may be less and not reliable with respect to the classes fallen in the higher levels of the classification hierarchy. Therefore, we have to find some other features to improve the classification for these classes. Fortunately, the tendency of these classes is toward generalization so that it is easier to find common properties on their structures. For example, the *White Pages* can be distinguished from the ordinary homepages by the percentage of the number of anchors and the average deviation between every two anchors. The main goal of this approach is only to improve the precision of the semantic analysis for the classes in the higher levels. Therefore, we only have to define the features which can distinguish between those classes at the same level. Thus the whole classification procedure forms a hierarchical structure from which the homepages are classified into the related clusters. The more precise classification results can be obtained by combining the decision rules derived from the semantic analysis with the various features used on the syntactic analysis.

4.3. Resource query processing

After constructing the classification hierarchy, the last phase is to process the queries for searching the resources on the WWW. In the IQBS, we design a query language for the user to directly query the classification hierarchy. Moreover, we implement a language parser at the IQBS server to check the syntax of user queries, and a query translator is also provided to execute the queries by the database system. Therefore, our query processor is composed of the WWW server and the database system.

First of all, the users fill the form provided by the *user interface* at the IQBS server with their queries according to the classification hierarchy. Then the query strings are translated into the form of a legitimate query for the database system. Actually, most of the query execution are performed by the database system. At last, the query results are directly sent to the client in the form of a HTML document. All the messages pass between the various components, depending on the parameters assigned in the CGI programs. These mechanisms are executed as follows:

1. query translation

Before the translation, the query strings received from the *user interface* have to pass the checking by a syntax parser. Our parser is written with the existing tools in the UNIX™ system. Then, the queries are translated into the required form of the database and executed by the database management system (DBMS). The program for query translation is written with the library provided by the UniSQL/X™ API. However, the *recursive resource query* in our query language can not be directly represented by SQL. In the IQBS, we solve this problem by the *method* of the object-oriented database. For example, A user may want to find all the homepages where a homepage at 'http://www.cs.nthu.edu.tw/' can approach to them by its chains of anchors within three to five levels, and the homepages themselves contain 'WWW' and 'Database' two keywords. Thus the user can input his query as the following statements:

```
SELECT y FROM root* x, root* y
WHERE x.location='http://www.cs.nthu.edu.tw'
        and x[level 3:5.anchors.link] has y and
            y.keywords has ['WWW','Database']
```

After the query translation by the method, we have the following executable query:

```
SELECT y FROM root* x, root* y
WHERE x.location='http://www.cs.nthu.edu.tw'
        and is_recv_linked(x,y,3,5)<>0 and
            y.keywords has ['WWW', 'Database']
```

As above, the operation *is_recv_linked* is the method used by object x, so we can check whether the chains of anchors originated from a homepage x would pass the homepage y within three to five levels.

2. query execution

This component is completely controlled by the DBMS. In the IQBS, we use the UniSQL/X ™ as the database system. Therefore, the CGI programs are only responsible for sending the translated queries to the DBMS and the query results to the users.

5. Guided browsing service subsystem

This system is established at the IQBS client to provide guided browsing services for the WWW users. There are three services: roadsign, footmark, and hot sequence. The roadsign service has to cooperate with the *resource discovery query subsystem* in order to make use of the ability for resource query processing. The footmark and the hot sequence stand for the behavior of the browsing user. Therefore, the user can improve the quality of personal browsing as to the information. Furthermore, such information can be gathered at the IQBS server and clustered by the different types of users. Thus more users can benefit from the services, so the total performance of the network traffic can also be promoted further.

Most of the current browsers only provide a graphic user interface except for the product Q-Mosaic ™ developed by the *QuarterDeck Corp.*, which can display the titles of the homepages pointed to by the anchors. It is inevitable for the WWW users to spend too much time on browsing some uninterested homepages. Hence, our system provides the roadsign as a guide to show the information about the homepages pointed to by the chains of anchors through several levels while the user is browsing a homepage. Similarly, most of the browsers only store the browsed homepages into a simple data structure with the titles and their addresses. The users only know whether the nodes have ever passed before, but other users can not make use of the information. Therefore, we represent the passed nodes by a path structure to keep the order information. Moreover, the information can be provided to the IQBS server so that other users can use them, too. When the user footmarks are accumulated to a large size, we can analyze the information to find the hot sequences. We use the UNIX ™ file system as our data storage system. At both the ends of the IQBS, the footmarks and hot sequences are stored in the same well-defined format. The type of users are distinguished by the different filenames at the IQBS server. All the services are explained in detail in the following subsections.

5.1. Roadsign

In principle, this service is provided by the *roadsign processor*. However, the *browsing interface* has to provide an extra function menu to set the required number of levels, and a CGI program at the server is needed to process the *recursive resource query* to get the information about roadsign. Therefore, the roadsign service in our system are mainly composed of two related parts: the *roadsign processor* at the client, and the CGI program at the server (See Figure 6).

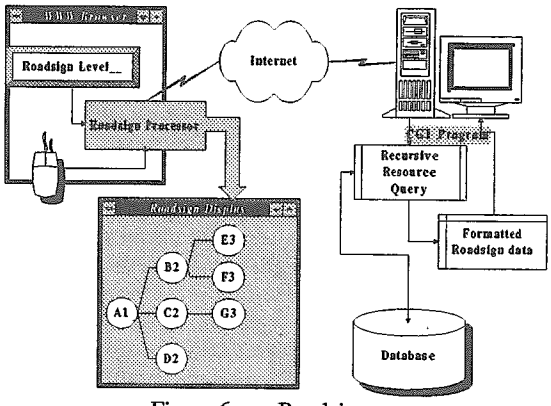

Figure 6. Roadsign

1. roadsign processor

During the period of the user browsing, this processor always gets the address of the next homepage and sends this information as a request for roadsign to the IQBS server. Then, our system shows the roadsign information in the form of a tree structure upon the *browsing interface* after the CGI program returns them. We use a display utility for tree structure to note down all the roadsign information. Thus each node of the tree exactly represents a homepage pointed to by an anchor of the current homepage within a fixed number of levels.

2. cgi program

Such a program is responsible for reacting to the requests from the *roadsign processor*. Hence this program parses the data in the requests at first, and then gets the address of the current homepage. In addition, the required number of levels for roadsign information is also obtained from the content of the requests. Afterwards, the program generates the *recursive resource queries* according to the roadsign requests and executes the queries as the *resource discovery query subsystem* does. At last, the query results are sent back to the *roadsign processor* at the IQBS client in a specific data format. The following discussions are three major components of this CGI program.

Decomposition of the environment variables

The environment variables from the WWW server are represented as follows:
"X1=xxx&X2=yyy" where X1 and X2 are the names of the variables, xxx stands for the address of the current homepage, and yyy is the required number of levels for roadsign information. Therefore, we retrieve the string between the first '=' and the first '&', and consider this string as the starting point of the roadsign. Moreover, the string after the second '=' is regarded as the required number of levels.

Recursive resource query processing

At first, the address of the homepage obtained at the previous step is found by the UniSQL/X ™ API (e.g., the starting point of the roadsign is regarded as the very beginning address). Then the program traverses all the chains of anchors based on the *depth first search* principle until the traversal within the required number of levels has

been completed. All the passed nodes form the query result in the roadsign data format.

Roadsign data format
In the roadsign processing, we transfer the roadsign information in a specific data format to reduce the data size and to simplify the interpretation. Each line in the data format is filled with the metadata of a homepage. Furthermore, we keep the tree structure by the *prefix order* traversal. The roadsign data format is illustrated as follows:

Parent node_id	Current node_id	Current URL	Current meta_information

Thus the *node_id* stands for the sequence number of the node according to the *prefix order* traversal (e.g., the sequence number of the root is equal to one).

5.2. Footmark

This mechanism is handled by the *footmark manager*. It provides the update management of the footmarks, in addition to recording the browsed paths through a specific data format. What the *footmark manager* is responsible for are as follows:

1. decision of storage unit

By the data transfer functions of the **Mosaic** ™, the address of the current homepage can be obtained during the period of browsing. Therefore, our system has only to directly call the related functions, in order to complete the task for recording the footmarks automatically. Such recorded information for user behaviors is only for personal use at the IQBS client, so this approach does not invade the user's privacy. Intuitively, all the browsed homepages can be connected one by one to make up a footmark. However, only a single path can not keep all the information of user browsing. Hence we define the starting point and the ending point of a storage unit as to the various user behaviors during the period of browsing.

- **open:** a new path is created and the next node is regarded as the starting point of it.
- **back:** store the current path and delete the last node in it.
- **forward:** add the current node to the current path.
- **follow an anchor:** add a new node to the current path.
- **click on the roadsign:** add a new node to the current path.
- **click on the footmarks or the hot sequences:** the same as the open operation.

2. footmark storage

In the IQBS, The *file manager* stores the footmarks in the form of a sequential linked list, which records the address and the title of each homepage in a single space. Moreover, the address and the title are separated by a space character. An entire footmark is composed of a series of non-space lines.

3. footmark browsing and update

The footmark information is displayed in the form of linked lists by means of an *X-Window* utility. The user can browse the personal information on line, in the meantime, s/he can make a connection to any homepage in the footmarks only to click on the corresponding node. In addition, the user can delete the nodes in the footmarks or even a whole path.. The user can also provide personal footmarks to the IQBS server in order to form the *global footmarks*. Our system uses a CGI program to manipulate the gathered data and stores them in a file system according to various types of users. For the sake of convenience, , the format for data transfer and storage of the global footmarks is the same as that of the personal footmarks. Similarly, the results of the global footmarks are presented to the user in the form of a HTML document.

5.3. Hot sequence

This mechanism is handled by the *hot-sequence manager*. It can derive several hot sequences from a large amount of user footmarks, and then store them through the *file manager*. The entire procedure is as follows:

1. data presentation

For the sake of computation, the address of each homepage in the footmarks is transformed into a number in advance. This approach sorts all the addresses and then removes the duplicates to generate a dictionary. At last, each address is given a number as to the sequence in the dictionary. After the derivation phase, the obtained hot-sequences still have to recover their address information by the same dictionary. The results are stored in the same format of the footmarks and have the similar presentation by the same *X-Window* utility.

2. derivation algorithm

This mechanism is based on the algorithm proposed in [YC95], which finds the most popular homepages by the statistics of their frequencies in the user footmarks and keeps the precedence in the meanwhile.

3. global hot sequence

In the IQBS server, we also provide a service similar to the hot sequences. This mechanism makes use of the global footmarks to produce the *global hot sequences* by the same derivation algorithm as above. However, the presentation has to be in the form of a HTML document.

6. Conclusion

In this work, we built up an integrated system for both querying and browsing the resources in Internet. In the aspect of querying, the *resource discovery query subsystem* provides a powerful facility to help users search their target resources. Moreover, this subsystem also demonstrates a complete architecture for the construction of a query interface on the WWW. Moreover, the *guided browsing service subsystem* provides three tools to speed up the browsing. The roadsign is similar to the guidepost which can guide the drivers to their destinations. On the WWW, the user can quickly find their interested resources by following the roadsign. Regarding the footmarks and the hot sequences, the behaviors of the user are recorded and analyzed. Thus the personal footmarks and hot sequences can be used to teach the user to guide efficiently. In addition, the global ones can be considered as a kind of

information exchange for the users in different fields. Therefore, the quality and performance can be substantially promoted as we make use of the IQBS to search the resources in Internet. Furthermore, the traffic cost for each user can also be reduced and hence the total network utilization is improved.

Our system implements a number of novel ideas. It integrates both the WWW server and the browser with the database system. This integration allows it to directly exploit the strengths of these mechanisms. The classification results have an important impact against the performance of resource querying. We classify the resources based on of the *rough set* theory. However, the goodness of the results needs further investigations. The classification schemes in various applications [ROM92, HC96] are also based on the *rough set* theory. Therefore, the comparisons with these schemes have to be considered. Our future research also includes the distributed server architecture, and the categorized browsing paths.

Reference

[AS95] R. Agrawal and R. Srikant. "Mining Sequential Patterns." *Proceedings of the IEEE Data Engineering*, pp. 3-14, 1995.

[BCL94] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret. "The World-Wide Web." *Communications of the ACM*, 37(8), pp. 76-82, 1994.

[ED92] A. Emtage and P. Deutsch. "Archie: An Electronic Directory Service for the Internet." *Proceedings of the Usenix Conference*, pp. 93-110, 1992.

[GGT94] L. Gravano, H. Garcia-Molina and A. Tomasic. "The Effectiveness of GLOSS for the Text Database Discovery Problem." *Proceedings of the ACM SIGMOD*, pp. 126-137, 1994.

[GG95] L. Gravano and H. Garcia-Molina. "Generalizing GLOSS to Vector-Space Databases and Broker Hierarchies." *Proceedings of the VLDB*, 1995.

[HTML]
http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html. "NCSA--A Beginner's Guide to HTML."

[HTTP]
http://www.w3.org/pub/WWW/Protocols/HTTP/HTTP2.html. "HTTP: A protocol for networked information."

[HC96] X. Hu and N. Cercone. "Mining Knowledge Rules from Databases: A Rough Set Approach." *Proceedings of the IEEE Data Engineering*, pp. 96-105, 1996.

[JF93] A. Johnson and F. Fotouhi. "Automatic Touring in Hypertext System." *Proceedings of the IEEE Phoenix Conference on Computers and Communications*, pp. 524-530, 1993.

[KM91] B. Kahle and A. Medlar. "An Information System for Corporate Users: Wide Area Information Servers." *Connexions -- The Interoperability Report*, 5(11), pp. 2-9, 1991.

[McC92] M. McCahill. "The Internet Gopher Protocol: A Distributed Server Information System." *Connexions -- The Interoperability Report*, 6(7), pp. 10-14, 1992.

[Mos95]
http://www.ncsa.uiuc.edu/SDG/Software/XMosaic/d2-intro.html "Introduction to NCSA Mosaic for X."

[MS95] D. McLeod and A. Si. "The Design and Experimental Evaluation of an Information Discovery Mechanism for Network of Autonomous Database Systems." *Proceedings of the IEEE Data Engineering*, pp. 15-24, 1995.

[NCSA]
http://www.ncsa.uiuc.edu/General/NCSAIntro.html. National Center for Supercomputing Applications.

[ODL93] K. Obraczka, P. B. Danzig, and S. H. Li. "Internet Resource Discovery Services." *IEEE Computer Magazine*, 26(9), pp. 8-22, 1993.

[PAW91] Zdzislaw Pawlak. "Rough Sets: Theoretical Aspects of Reasoning about Knowledge." *Kluwer Academic Publishers*, 1991.

[PAW95] Zdzislaw Pawlak. "Rough Set." *Communication of the ACM*, 1995.

[ROM92] Roman Slowinski(Editor). "Handbook of Applications and Advances of the Rough Sets Theory." *Kluwer Academic Publishers*, 1992.

[TC96] Pauray S. M. Tsai and Arbee L. P. Chen. "An Approach to Resource Discovery in Distributed Autonomous Database." *Proceedings of IEEE Conference on Information Networking*, pp. 236-247, 1996.

[Uni93] UniSQL Inc. *UniSQL/X™ Application Program Interface Reference Guide*, Release 2.0, 1993.

[WC96] Yi-Hung Wu and Arbee L. P. Chen. "A Rough Set based Approach to Classifying the Resources in Internet." *Tech. Report of Dep. Computer Science. NTHU*, 1996.

[YC95] Show-Jane Yen and Arbee L. P. Chen. "An Efficient Approach to Discovering Knowledge from Large Databases." *Tech. Report of Dep. Computer Science. NTHU*, 1995.