Enhancement of the CINDI System

YANHONG LI

A THESIS

IN

THE DEPARTMENT

OF

COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE AT
CONCORDIA UNIVERSITY
MONTREAL, QUEBEC, CANADA

AUGUST 2003

Your file Votre référence
ISBN: 0-612-83910-9
Our file Notre référence
ISBN: 0-612-83910-9

ABSTRACT

Enhancement of the CINDI System

Li, Yanhong

Development of computer and Internet technology provides the necessary tools to develop digital libraries that could provide a cost-effective access to information for users on a global area. To address the problems, the CINDI (Concordia INdexing and Discovery) system proposes the semantic header for bibliographic index and the graphic interface for cataloging, searching as well as access, and provides feedback and annotation. In this project, our aim is to provide a high availability of the CINDI system, and improve the functionality of the system. A high availability solution presented herein is to use the file-based and database replication methods. In addition, we propose a scheme to integrate the CONFSYS (conference support system) with the CINDI system. We also address the issues of optimizing the database design and query processing.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to Prof. Dr. Bipin C. Desai for his invaluable help, patience, guidance and support. Special thanks go to my friends for their help, companionship, and support, especially to Mr. Zhan Zhang, who was a member of the CINDI project and helped me solve the problems with the ASHG.

I would like to express my deepest gratitude to my parents, my husband as well as my brother. Without their love, encouragement and support, I would never have accomplished my study and reached this level.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1. Problem statement

With the tremendous growth of World Wide Web (WWW), the Internet with enormous amounts of information has become the principal repository of knowledge. Billions of people have access via web browsers to this wealth of information interconnected by millions of hyper-links. In order to make efficient and effective use of the large amount of digital information on the Internet, many institutes and business organizations have made substantial progress in Information Retrieval (IR) over the last decade and produced a number of popular search engines, such as Google, AltaVista, Yahoo, as well as Lycos.

A series of tests [DESAI02] on the effectiveness of the contemporary search systems were conducted in 1995, 1997, and 2001, respectively, in which the same search item **Bipin (AND) Desai** was used. Here, the Boolean **(AND)** was used only when it was required by the search engine to represent conjunction. For the tests, the following terms are defined:

- **Number of Hits**: A URL is a hit if the corresponding document contains the search string, and it pertains to the intent of the search, namely a document about or by the search items.
- **Number of Duplicates**: If the same document is served from more than one server, then it is considered as a duplicate. Here, the URLs are different, but the contents are the same. This is one of the problems that have plagued search engines from the start; some search engines have addressed this problem better as the experiments illustrate.

- **Number of Miss-hits**: A URL is considered as a miss-hit if the document is not relevant for the search. Here, even though the search terms may exist in the document they occur out of context. Search engines again have difficulty with the context of words in documents.
- **Number Missed**: The number of relevant documents not found even though they existed on the Web. Since we started with a list known URLs being served long before the tests, this was easy to determine for these experiments.

Based on the test results (Table 1, Table 2, and Table3), precision (*P*) and recall (*R*), which are two of the measures used to express the effectiveness of an information search operation, were calculated as follows [DESAI02]:

$$P = \frac{number\ of\ relevant\ retrieved}{number\ of\ retrived}$$

$$R = \frac{number\ of\ relevant\ retrieved}{number\ of\ existing\ retrieved}$$

Table 1. Test results [DESAI02]

Search System	Number of Hits	Number of Duplicates	Number of Miss-hits	Number Missed	Precision (P) %	Recall (R) %
Aliweb	0	0	0	24	0	0
DA-CLOD	0	0	0	24	0	0
EINET	6	0	4	22	7	8
GNA Meta Lib.	0	0	0	24	0	0
Harvest	0	0	0	24	0	0
InfoSeek	7	0	0	24	29	29
Lycos	7	2	222	17	3	29
Nikos	0	0	0	24	0	0
RBSE	0	0	8	24	0	0
W3 Catalog	0	0	0	24	0	0
WebCrawler	4	3	0	24	15	17
WWWW	2	0	0	22	8	8
Yahoo	None	0	0	24	0	0
NOTE: In 1995, the number of known URLs with the search string was 24. Result average: $P_{average}$= 4.8%, $R_{average}$= 7%						

Table 2. Test results [DESAI02]

Search System	Number of Hits	Number of Duplicates	Number of Miss-hits	Number Missed	Precision (P) %	Recall (R) %
Alta Vista/Yahoo	97	9	23	264	25	27
Excite	114	10	29	247	29	32
Infoseek	8	2	1	319	2	2
Lycos	57	7	15	297	15	16
Hotbot	247	28	58	155	51	61
OpenText	19	-	7	318	6	6
NOTE: In 1997, the number of known URLs with the search string was 325. Result average: $P_{average}$= 21.3%, $R_{average}$= 24%						

Table 3. Test results [DESAI02]

Search System	Number of Hits	Number of Duplicates	Number of Miss-hits	Number Missed	Precision (P) %	Recall (R) %
Alta Vista/Yahoo	99	24	67	230	24	30
Google	155	10	403	174	21	47
Hotbot	62	21	121	267	13	19
Lycos	239	27	711	90	22	73
NOTE: In 2001, the number of known URLs with the search string was 329. Result average: $P_{average}$= 20%, $R_{average}$= 42.3%						

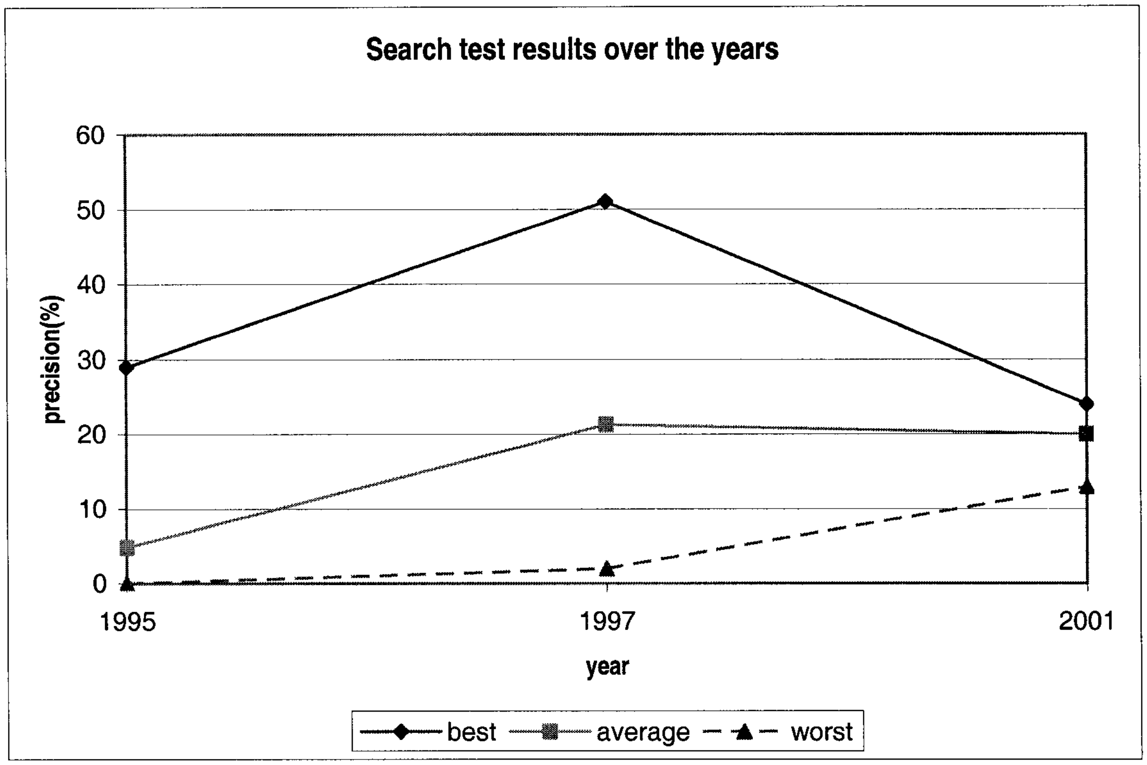


Figure 1.1(a) 1995-2001 Test trends

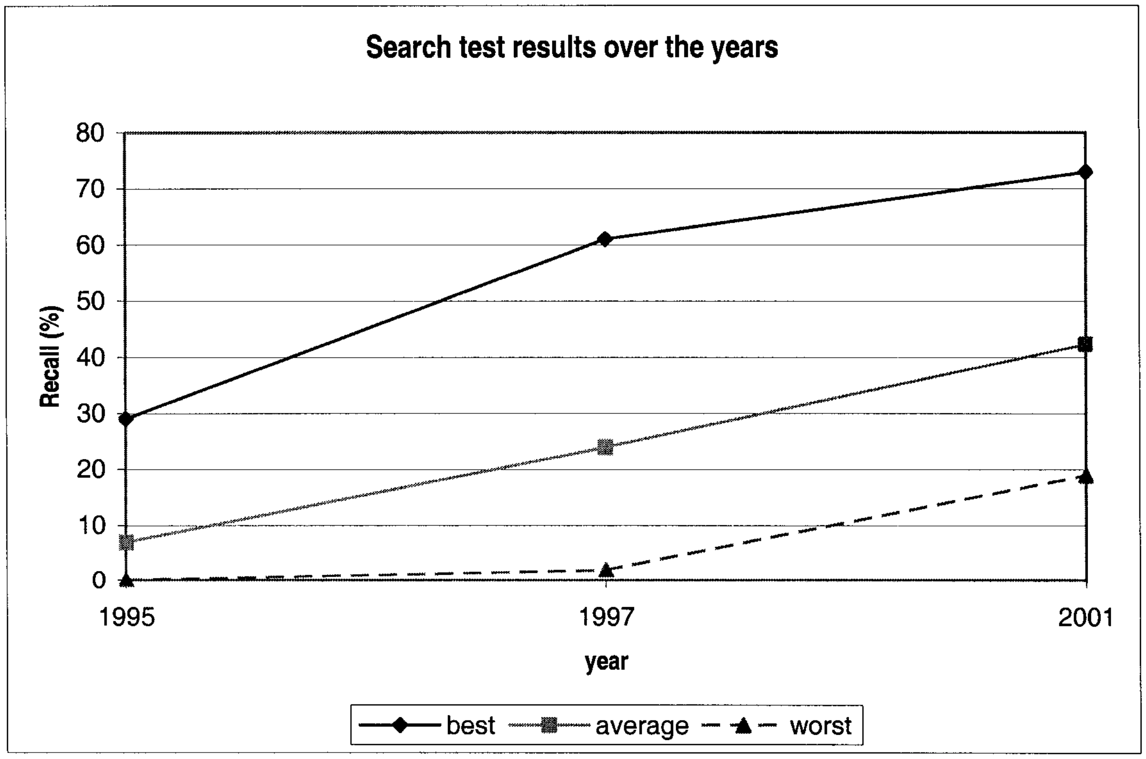


Figure 1.1(b) 1995-2001 Test trends

Figure 1.1 (a)-(b) show that while the recall has gone up to a very respectable percent, the precision has actually been reduced from the tests in 1997 and the best value for the tests in 2001 is less than 25%. Although search engines have improved over the years, there are still some problems faced by users that are not solved. For example, by using the links provided by the WWW, users can easily access the pages or sites that they are familiar with. However, if users look for a specific page or a particular subject with these links, it is neither precise nor efficient. The problem is that they may get information that may be useless or lose sight of their original purposes in the web.

In order to make effective use of the information and services of the Internet, some popular search engines are classifying the information in hierarchal catalogs or indexes to help users in their search, such as Google and Yahoo. Although cataloguing and indexing information is good for general topics, it may be time-consuming for users who are interested in a particular topic, which may not fit in one of the categories offered by the public catalog. Since most people, in fact, just need one or at most a few relevant items, they do not need to learn the technology to get the appropriate search results. That is the developing trend of search engines. [DESAI02]

1.2. Proposed solution

In practice, the books and other materials a library acquires are well organized by using indexing, classifying, and descriptive as well as subjective cataloging. Users, who are interested in a particular topic or a special subject, without the specialized knowledge of library cataloging, may acquire the appropriate search results easily. With the help of an experienced librarian, users are able to find the required material quickly as well.

In order to enable individuals to make efficient and effective use of the wealth of electronic resources of various types on the Internet, an indexing system is to be built to provide a system that will have users to retrieve electronic resources distributed geographically over nodes interconnected by wide area networks (WAN). The system should be an open library system that allows an integration of data from multiple information sources. Meanwhile, with the purpose of avoiding the above problems, a standard index structure should be constructed, and a bibliographic system must be built by using standardized control definitions. Furthermore, with the aim of allowing users to have easy access to the original materials, indexing and cataloging must be accurate, easy to use, properly classified, up-to-date and complete for its area of coverage. Moreover, the system should provide an ease-to-use graphical interface for users.

1.3. Organization of the thesis

The rest of the thesis is organized as follows:

Chapter 2 introduces the concept and development of a digital library, and highlights the solution of the CINDI to the current problems of information retrieval and discovery. It attempts to give an overview of the earlier work this thesis is based upon as well as emphasis of the direct relationship between the CONFSYS system and the CINDI system, from the physical architecture to the functional structure.

Chapter 3 describes the general approaches for high availability, including file-based methods and database replication provided by database management systems. In the current version of the CINDI system, we provide a high availability solution by combining a file-based approach with database replication.

Chapter 4 presents the query processing methods applied to the CINDI system. Here, we report on how to protect the ACID properties of the transactions, and how to speed up the searching. The fuzzy search is introduced to provide wider latitude in searching. In addition, we provide the details of the integration of the CONFSYS with the CINDI system.

Chapter 5 gives the methods used to cope with the security problems in the CINDI system. Several approaches for security are introduced, such as cookies and sessions, HTTP authentication, as well as cryptography. By using a judicious integration of these approaches, the CINDI system provides adequate security.

Chapter 6 gives the results of some tests used and the feedback of the uses of the CINDI system.

Chapter 7 concludes the thesis by highlighting the significant contributions of the current work, discussing various possibilities of improving the CINDI system, and pointing out the directions for future development.

Chapter 2 The CINDI System

A digital library [ARMS02] is informationally defined as a managed collection of information in digital formats that is accessible over a network with associated services. The information stored in the library should be partitioned based on the topics that are common to the relevant data. For example, a digital library can be designed and built for computer science; another one can be for mathematics. These separate libraries although distributed could be integrated with a common interface, while the information contained within each library remains separate. The purpose of a digital library is to provide a central location for accessing information on a particular topic. A digital library should also have an easy-to-use user interface for users to search and browse the contents.

To use a traditional library, a reader must take time to go there. Since many people do not have a library nearby, they have poor access to the relevant information. In contrast to a traditional library, a digital library brings the information to the users' desks by using computer and network. Unlike the paper documents contained in a library uniquely, the digital information is always available on a network although it is maintained at a single site. Moreover, to keep the information up-to-date is much easier in a digital library than in a traditional library. With the technical developments, more and more universities and organizations are working on digital libraries for information retrieval and discovery [UCR00].

Catalogs are used by almost all libraries as the core technology to help users search the materials in a library. Most such catalogs can provide comprehensive bibliographic information, such as title, authors, as well as administrative information (e.g. where items are stored). Currently, some cataloging rules adopted by some digital libraries are created by professionals for some specific subjects. For instance, the Machine-Readable Cataloging (MARC), initially developed and used for conventional materials on magnetic tape in the Library of Congress, was extended and utilized by the Online Computer Library Center (OCLC). The MARC uses numbers and special characters to present the items of a bibliographic index (Table 4), and is hard to be understood. Another example is the Medical Subject Headings (MeSH) developed by the National Library of Medicine (NLM). Because it adopts the controlled-vocabulary approach [ARMS02] that requires professional training to apply effectively, its users and reference librarians must understand medicines and are familiar with the cataloging indexes.

A wide variety of materials in digital forms are stored in a digital library, including textual and non-textual; however, the library catalogs developed for the traditional materials are not suitable for electronic information. The Dublin Core [ARMS02] with only fifteen descriptive items, developed by a workgroup in 1995, was initially created as a simple and single set of metadata for the untrained people who would like to publish their electronic works. The following table shows that the Dublin Core (DC) is simpler for maintenance and easier to be understood than the MARC, which is more precise due to its complexity in description. Although simplicity results in imprecision in information retrieval, it is so extensible as to be internationally [NP02] [MILLER96] [AD02] [OKK01] applied for resource description.

Table 4. MARC to Dublin Core Crosswalk [MARC01]

DC Elements	MARC fields		Notes
Title	245		
Creator	100, 110, 111, 700, 710, 711		Name fields in MARC are mapped to DC Creator rather than Contributor.
	720		
Subject	600, 610, 611, 630, 650, 653		
Description	500-599, except 506, 530, 540, 546		
Contributor			Contributor is not used in conversion.
Publisher	260ab		"$" is used to represent the control character subfield delimiter.
Date	260$c		
Type	Leader06	a,c,d,t	text
		e,f,g,k	image
		i,j	sound
		m,o,p,r	no type provided
	Leader07	c,s,p	collection
	655		Field 655 may be used more specific type information.
Format	856$q		
Identifier	856$u		
Source	786ot		
Language	008/35-37 546		
Relation	530, 760-787ot		
Coverage	651		
	752		
Rights	506, 540		

2.1. The proposal of CINDI

The Concordia INdexing and DIscovery System (CINDI), which was proposed by Desai *et al.* in 1994 [DESAI94], is to build a digital library system with user-friendly interface with which resource providers or contributors can "publish" their resources. Combined with the basic human knowledgebase of cataloging, the system will provide a number of features of expert system for subject classification that can be used to scan the information resources to be submitted to the system and assign appropriate subjects for each resource. Furthermore, an alternate interface system allows the contributors to prepare and enter the bibliographic information about each resource manually using the standardized index scheme. By using this system, any users of the system can search for a special topic or subject with typical search items, such as author, title, subjects, and so on.

2.2. The solution of CINDI

The heart of any bibliography or indexing system is the record that is kept for each item that is being indexed. Standardization of a bibliographic entry allows libraries to exchange information about their collections. A number of projects in the Library domain have addressed the problem of cataloging and in particular cataloging of information in electronic and multi-media format, such as the MeSH and MARC systems. However, such systems are designed for professional catalogers, and many of the items include in them, though useful, are beyond the comprehension of most ordinary users. In addition, the sciences and other technical fields rely on abstracting and indexing services to determine the relevance of a document rather than catalogs. As a result, the Dublin Core with the fifteen elements is too simple to provide such helpful information about a resource.

In order to address these problems, the CINDI uses index entry called Semantic Header [DESAI95] and provides a semi-automatic mechanism to register, manage as well as search the bibliography. Although currently the system requires the provider or contributor of information to register the resource by entering an index for the resource, there are plans to automate this operation. Considering the function requirement of users of the CINDI system, there are three general categories of users defined as follows:

- **Provider** or **contributor**: who adds an information resource to the CINDI collection.
- **User**: who searches the CINDI index for a resource and downloads it.

- **Reader**: who, registering the CINDI system as a user, has downloaded a resource and wants to register some comments or annotations on the corresponding semantic header.

The overall system uses knowledge bases and expert subsystems to help the users during the register and search processes. One such need for an expert system is in avoiding chaos introduced by differences in perception of different indexers.

2.3. The architecture of CINDI

Most contemporary software systems are organized as described in Figure 2.1, with three main components: information presentation (i.e. graphic user interface or GUI), information processing and information storage. For database application, based on the general software architecture, the implementation structure is made up of a two-tier client/server, a three-tier or an n-tier system that could, in addition, be a distributed system. The structures define a relationship between information presentation, processing and storage within a database system.

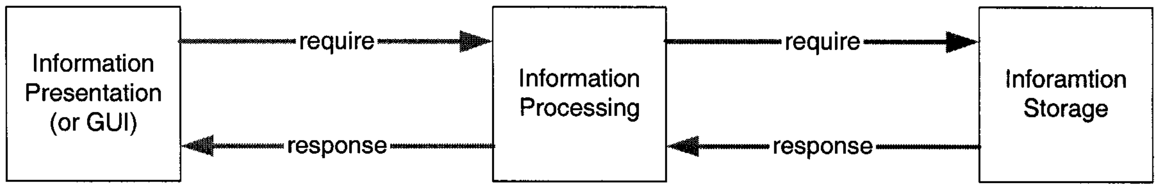


Figure 2.1 The structure of software

Two-tier systems are the traditional approach to database architectures. This architecture creates two layers: the client layer, which performs information presentation, and the server layer, which takes care of information storage. Processing management is split between the client and the server. (Figure 2.2)

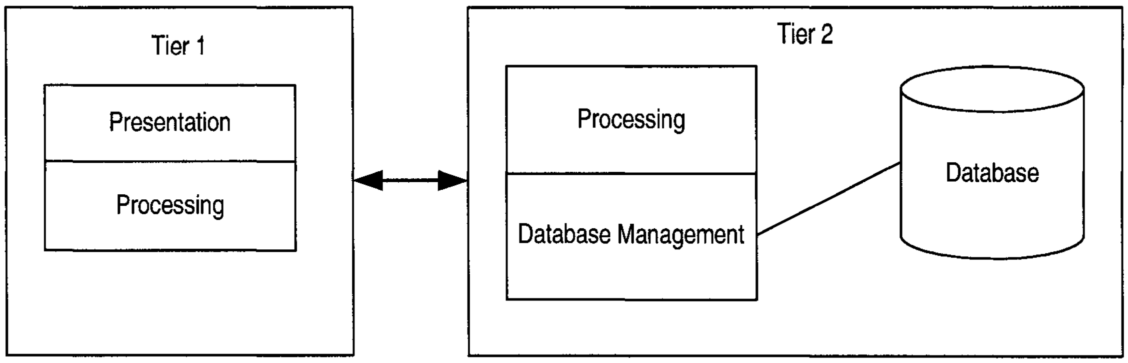


Figure 2.2 The logical view of the 2-tier systems

Because all technologies can be provided by one vendor with the same formats and protocols in the two-tier system, it is feasible for the small-scale application on a local area network (LAN). However, its scalability problem limits its application as a web application because of the session management. Furthermore, a server must maintain a connection via "Keep-alive" messages with each client even when no work is being done. This results in less scalability in the two-tier system. To solve the above problem, a middle tier is introduced to reduce the number of sessions.

The three-tier systems separate the processing management from the user system interface client environment and the database management server environment to be a middle tier. That is, the client layer performs presentation only; the middle tier provides processing management, which is implemented in a variety of ways, such as transaction processing monitors, message servers, or application servers. For the web application, the middle tier is viewed as the web server. (Figure 2.3)

Although the three-tier system development environment is reportedly more difficult to be used than the visually oriented development of two-tier systems, the advantages of a three-tier approach outweigh its complexity. First, as mentioned previously, the middle tier can help reduce the number of sessions by providing functions such as queuing, application execution, and database staging so as to enlarge the capacity of the three-tier systems. Furthermore, with the low cost of ownership for the client, it hides the complexity of distributed processing from the users and improves its performance and flexibility. Moreover, easier upgradeability and maintenance can be achieved. These benefits make the three-tier more flexible, and it has become more prevalent in web-based applications.

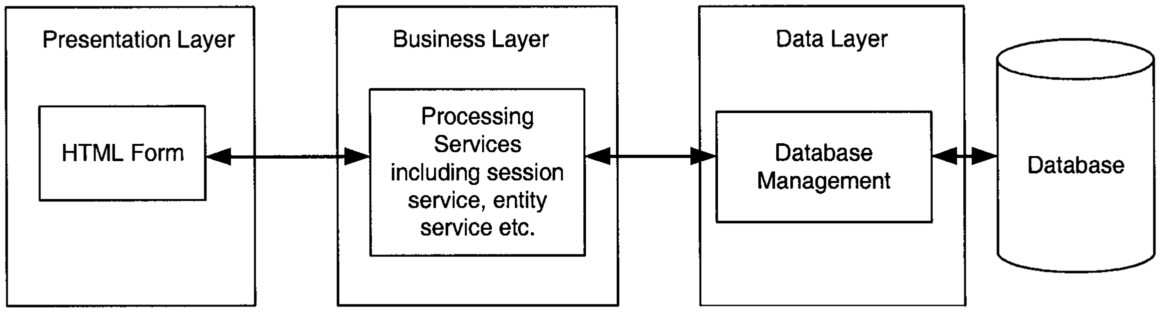


Figure 2.3 The logical view of the 3-tier system

As a web-based application, the CINDI system adopts the three-tier architecture, which supports the multi-user, multi-thread environment. In order to optimize data usage, the distributed approach is applied through the distribution of the server backend. The logical view of the CINDI architecture is depicted in Figure 2.4.

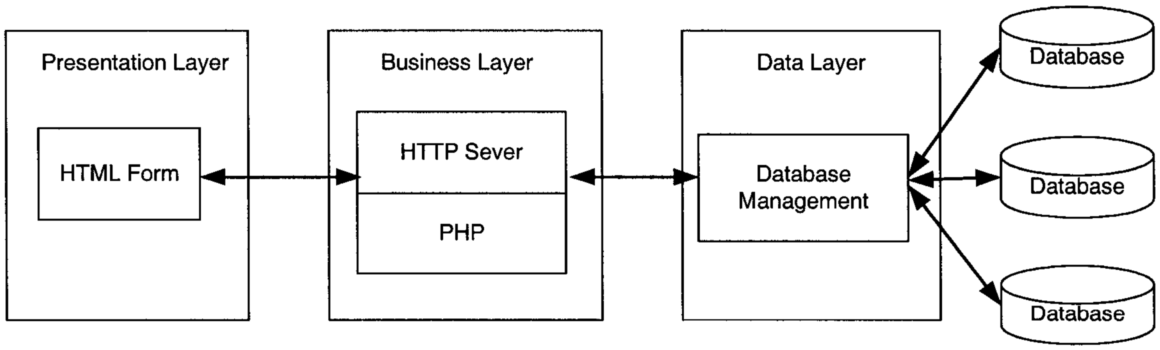


Figure 2.4 The logical view of the CINDI system architecture

Based on the architecture, the CINDI system functionally comprises five principal subsystems (Figure 2.5) as follows:

1. The register system, which allows contributors to provide resources and create bibliographic data records describing electronic resource by using automatic generator or by filling in and submitting HTML (Hypertext Markup Language) form.
2. A distributed and replicated system, which stores these records describing electronic information resources.
3. A search and annotation system, which allows the database to be searched for records corresponding to electronic resources of interest, and also allows users to make their comments on the resources of their interest.
4. The conference support system (CONFSYS) [GU02] [JIN03], which collects the papers submitted for a specified conference and allows the program committee members (reviewers) to evaluate those papers that are assigned by the program chair based on the reviewers' experience and interest. The papers would be included in the CINDI database thus improving its collection of the latest development.
5. The administrator system, which allows the system administrators to manage the information of the users as well as electronic resources, and to integrate the information of resources coming from the specified conference with the CINDI.

The CINDI system is built from standard Web components, including Apache server with the Secure Sockets Layer (SSL). The following sections will discuss the major subsystems in detail.

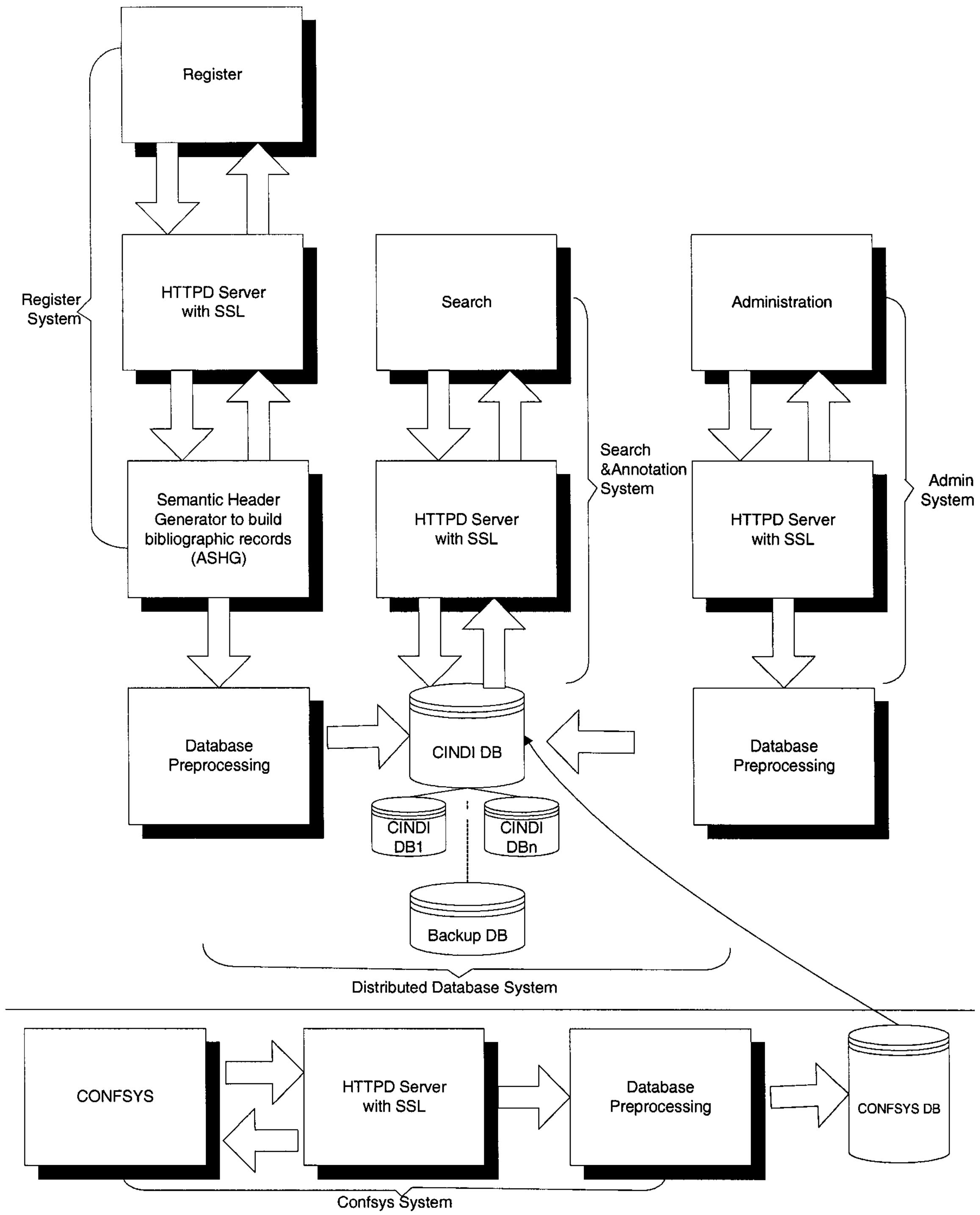


Figure 2.5 The structure of the CINDI system

2.4. Semantic header

The CINDI system provides a semi-automatic mechanism for generating a standardized index called “semantic header” [DESAI95]. The semantic header is designed to ensure homogeneity of syntax and semantics of such an index. It is the heart of the system that records the characteristics of a resource being indexed and provides the succinct information regarding the resource contents. The semantic header is intended to include those items (see Figure 2.6) that are most often used in the search of an information resource such as a title, name of one of the authors and subject as well as sub-subject; these items are used in the majority of search with 70% and 50% [DESAI95] for authors and subjects as well as sub-subjects, respectively. Meanwhile, the abstracts and annotations are included since they are useful for users to make decisions on the relevance of the resource.

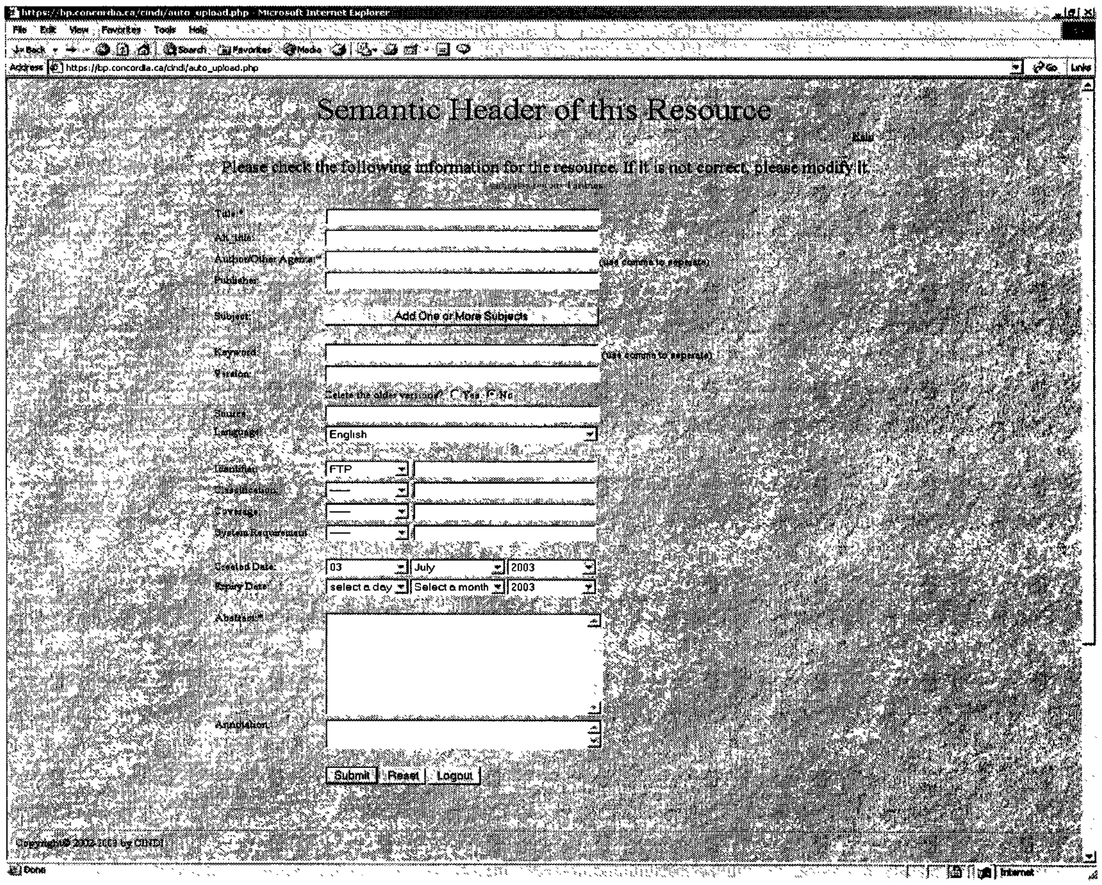
```
<semhdr>

<title> required </title>
<alt-title> OPTIONAL </alt-title>

<Subject> required: a list each of which includes fields for
subject and up to two levels of sub-subject: at least one entry
is required </Subject>

<language> OPTIONAL: language of the information resource
</language>
<char-set> OPTIONAL: character set used </char-set>

<author> required: a list each of which includes role, name,
organization, address, etc. of each person/institute responsible
for the information resource: at least the name or the
organization and address is required </author>

<Keyword> required: a list of keywords </Keyword>

<Dates>
<Created> required: </Created>
<Expiry> OPTIONAL: </Expiry>
<Updated> system generated </Updated>
```

```
</Dates>

<Version> OPTIONAL: version of the resource </Version>
<Supersedes>   OPTIONAL:   which   version   is   being   replaced
</Supersedes>

<Coverage> OPTIONAL: audience, spatial, temporal </Coverage>

<Classification> OPTIONAL: nature (legal, security level etc.) of
the resource </Classification>

<Identifier>  A  list  of  domains  for  identifiers  and  the
corresponding values: typical identifiers could be one of more
Unique Resource Locator(URL), Call No. for the resource, unique
name  of  the  resource  (URN),  site  where  the  item  is  to  be
archived: at least one required
</Identifier>

<Abstract> required: of the resource </Abstract>

<Annotation> OPTIONAL: </Annotation>

<SysReq>  OPTIONAL:  list  of  system  requirements  for  example
hardware  and  software:  the  component  and  the  corresponding
requirements are given
</SysReq>

<Source> OPTIONAL: gives the source or related list of resources
for each such resource it indicates a relationship and gives an
identifier which includes the domain and the corresponding value
</Source>

<size>  size  of  the  resource  in  appropriate  units  (e.g.,  bytes)
</size>

<Cost> OPTIONAL: cost of accessing the resource </Cost>

</semhdr>
```

Figure 2.6 The structure of semantic header [DESAI95]

2.5. The index registering subsystem

The index registering sub-system provides a graphic interface (see Figure 2.7) to facilitate the provider of a resource to register the bibliographic information for the resource. The interface allows the provider to enter the corresponding information with the restriction on the indexing standard by providing the pop-up selection windows.


https://bp.concordia.ca/cindi/auto_upload.php - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://bp.concordia.ca/cindi/auto_upload.php
Go Links

Semantic Header of this Resource

Help

Please check the following information for the resource. If it is not correct, please modify it.

Title:*		
Alt_title:		
Author/Other Agents:*		(use comma to seperate)
Publisher:		
Subject:	Add One or More Subjects	
Keyword:		(use comma to seperate)
Version:		
	Delete the older version? ( ) Yes [✓ No]	
Source:		
Language:	English	
Identifier:	FTP	
Classification:	—	
Coverage:	—	
System Requirement:	—	
Created Date:	03 July 2003	
Expiry Date:	select a day Select a month 2003	
Abstract:*		
Annotation:		

Submit Reset Logout

Copyright© 2002-2003 by CINDI

Figure 2.7 The interface of registering a resource

Figure 2.8 shows the function processing of the registration subsystem based on the logical structure of the CINDI system, in which S1, S2, S3, S4, and S5 are symbols of the PHP scripts that process the users' requests and servers' responses. In order to help the providers catalogue a resource with controlled terms, the subsystem offers a knowledge based expert system, which models the expertise of a reference librarian, to the resource providers with the context-sensitive help on the subject entry. In addition, for files in a number of formats, such as HTML, TEXT, $\LaTeX$, RTF, and PDF[1], an automated mechanism called Automatic Semantic Header Generator (ASHG)

[1] This work is in progress at the time of writing this thesis.

[HADDAD98] [ZHANG02], is used to generate a draft version of the semantic header for the new resource being uploaded to the system (see Figure 2.9). Once this draft semantic header is verified by the resource provider, the information entry can be registered into the database described in Section 2.6. Here, the project for this thesis integrated the ASHG for the PDF files with the previous work.

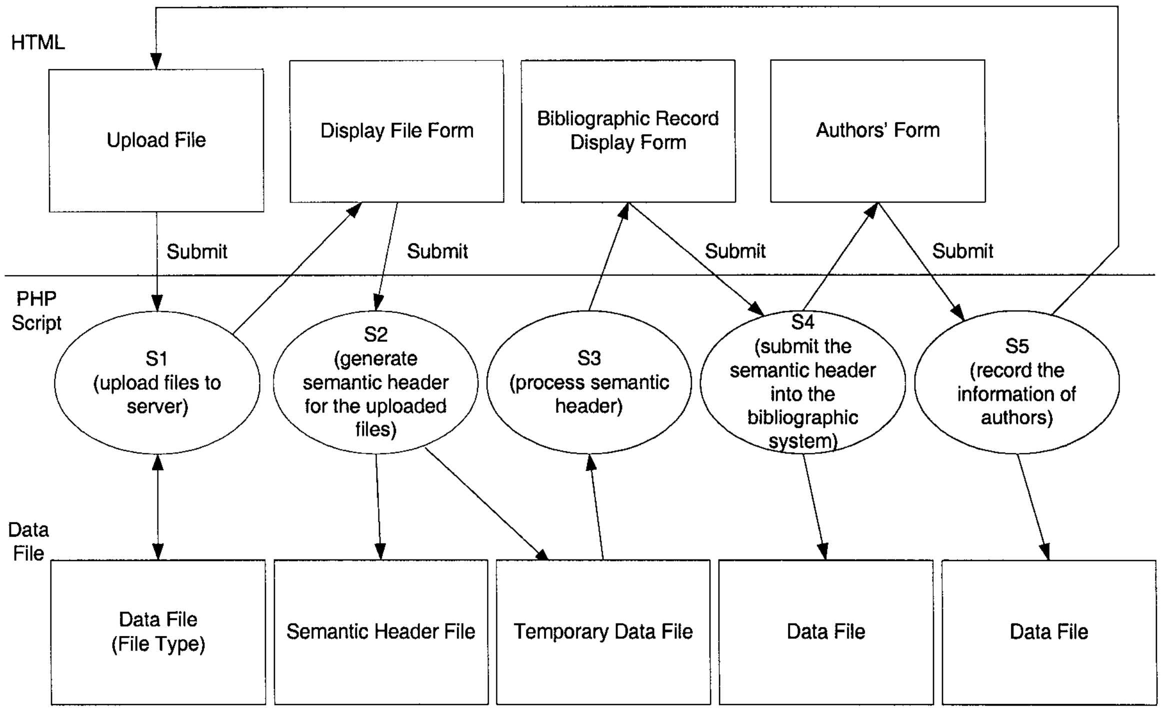
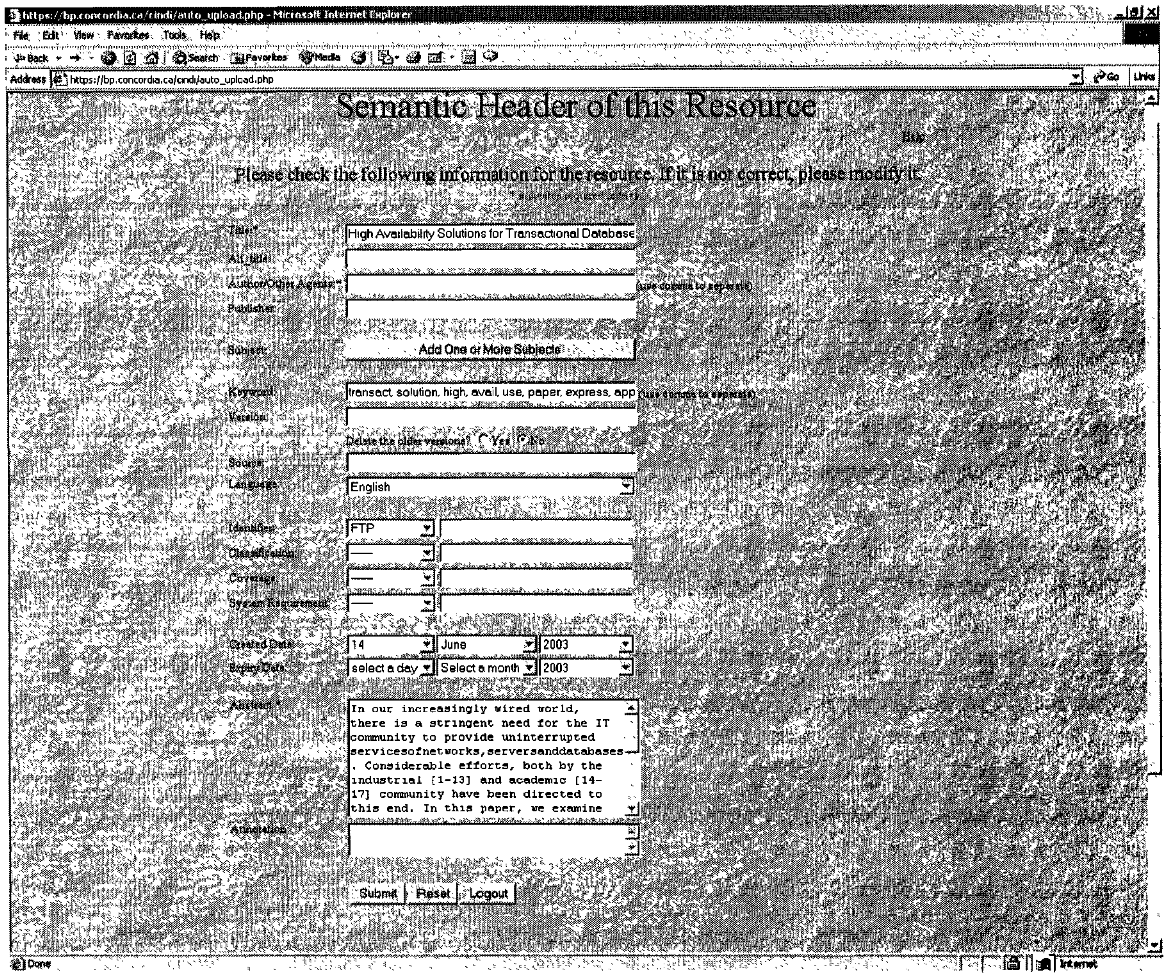

HTML
Upload File
Display File Form
Bibliographic Record Display Form
Authors' Form
Submit
Submit
Submit
Submit
PHP Script
S1 (upload files to server)
S2 (generate semantic header for the uploaded files)
S3 (process semantic header)
S4 (submit the semantic header into the bibliographic system)
S5 (record the information of authors)
Data File
Data File (File Type)
Semantic Header File
Temporary Data File
Data File
Data File

Figure 2.8 The logical structure of the index registering subsystem

```
<semhdrB>
<useridB> <useridE>
<passwordB> <passwordE>

<titleB> High Availability Solutions for Transactional Database
Systems <titleE>
<alttitleB> <alttitleE>

<subjectB>
<generalB> Computer Science <generalE>
<sublevel1B> Software <sublevel1E>
<sublevel2B> computer programs and softwares <sublevel2E>

<generalB> Computer Science <generalE>
<sublevel1B> Database management <sublevel1E>
<sublevel2B> transaction processing systems <sublevel2E>
```

```
<generalB> Computer Science <generalE>
<sublevel1B> Performance of computer systems <sublevel1E>
<sublevel2B>   reliability,   availability,   and   serviceability
<sublevel2E>
<subjectE>

<languageB> English <languageE>
<char-setB> <char-setE>

<authorB>
<aroleB> Author <aroleE>
<anameB>  <anameE>
<aorgB> <aorgE>
<aaddressB> <aaddressE>
<aphoneB>  <aphoneE>
<afaxB>  <afaxE>
<aemailB> Budrean@sita.int <aemailE>
<authorE>

<keywordB>  transact , solution , high, avail , use , paper ,
express , approach , database  <keywordE>

<identifierB>
<domain3B> FTP <domain3E>
<value3B> <value3E>
<identifierE>

<datesB>
<createdB> 2003/6/14 <createdE>
<expiryB>  <expiryE>
<datesE>

<versionB> <versionE>
<spversionB> <spversionE>

<classificationB>
<domain4B> <domain4E>
<value4B> <value4E>
<classificationE>

<coverageB>
<domain5B> <domain5E>
<value5B> <value5E>
<coverageE>

<system-requirementsB>
<componentB> <componentE>
<exiganceB> <exiganceE>
<system-requirementsE>

<genreB>
```

```
<formB> <formE>
<sizeB> 34193 <sizeE>
<genreE>

<source-referenceB>
<relationB> <relationE>
<domain-identifierB> <domain-identifierE>
<source-referenceE>

<costB> <costE>

<abstractB>
In our increasingly wired world, there is a stringent need for
the IT community to provide uninterrupted services of networks,
servers and databases. Considerable efforts, both by the
industrial [1-13] and academic [14-17] community have been
directed to this end. In this paper, we examine the requirements
for high availability, the measures used to express it and the
approaches used to implement this for databases.
We present a high availability solution, using off the Shelf
hardware and software components, for transactions based
applications and give our experience with this system.
<abstractE>
<annotationB> <annotationE>

<semhdrE>
```


https://bp.concordia.ca/cindi/auto_upload.php - Microsoft Internet Explorer

Address https://bp.concordia.ca/cindi/auto_upload.php

Semantic Header of this Resource

Please check the following information for the resource. If it is not correct, please modify it.

Title:*	High Availability Solutions for Transactional Database
Alt_title:	
Author/Other Agents:*	(use comma to separate)
Publisher:	
Subject:	Add One or More Subjects
Keyword:	transact, solution, high, avail, use, paper, express, app (use comma to separate)
Version:	
	Delete the older versions? Yes [✓ No]
Source:	
Language:	English
Identifier:	FTP
Classification:	—
Coverage:	—
System Requirement:	—
Created Date:	14 June 2003
Expiry Date:	select a day, Select a month, 2003
Abstract:*	In our increasingly wired world, there is a stringent need for the IT community to provide uninterrupted servicesofnetworks,serversanddatabases . Considerable efforts, both by the industrial [1-13] and academic [14-17] community have been directed to this end. In this paper, we examine
Annotation:	

File Edit View Favorites Tools Help
Back Search Favorites Media
Go Links
Submit
Reset
Logout
Done
Internet

Figure 2.9 An example of semantic header and its interface

2.6. The semantic header distributed database system

The bibliographic records registered by resource providers are stored in a semantic header distributed database system (SHDDBS). For the CINDI system users, the underlying semantic header database may be considered to be a unified centralized system. In fact, it would be distributed and replicated database system that supports reliable and failure-tolerant operations. This distribution is based on subject areas so that the database can be viewed to be horizontally partitioned.

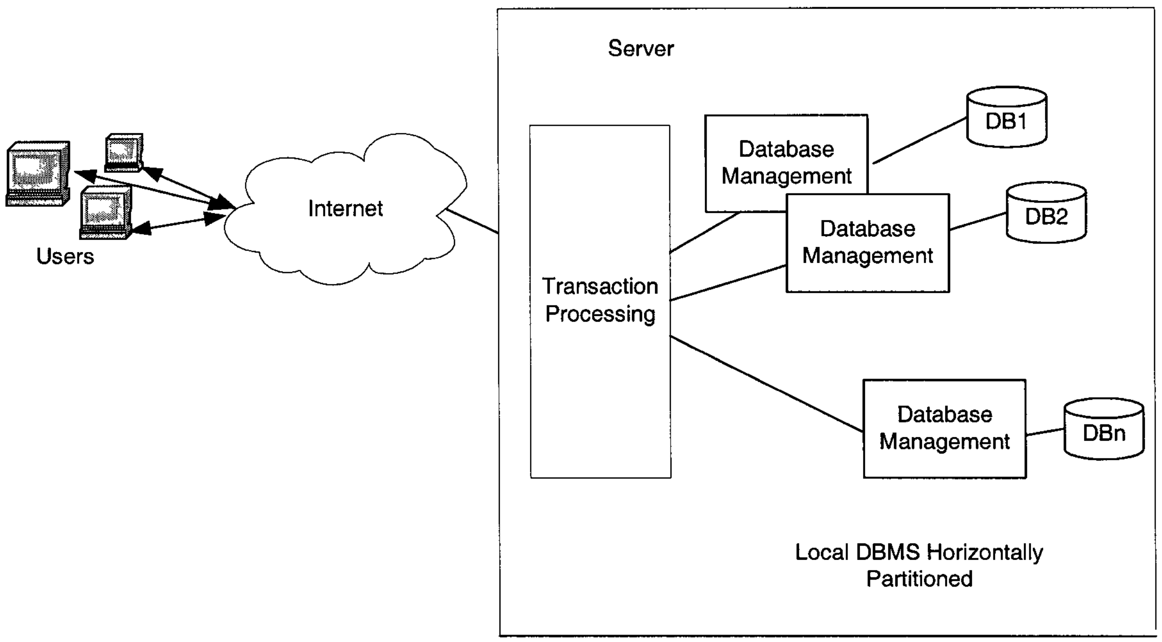


Figure 2.10 The view of the horizontally partitioned database system

It is possible that the semantic header database on different subjects will be maintained at the different nodes of the Internet as shown in Figure 2.10. The locations of such nodes need only to be known by the appropriate subsystem. A database catalog would be used to distribute the information. However, this catalog itself could be distributed and replicated as is done for a distributed database system [DESAI95] as described in Chapter 3.

Database catalogs will also be used to store information regarding subject areas maintained in the SHDDBS so that the users can select items for indexing and retrieving semantic headers. Thus, each node will contain a catalog consisting of all subjects as well as information relating to the locations of semantic headers, pertaining to a subject, in the distributed system.

The semantic header information entered by a contributor of a resource using a graphic interface is relayed from the user's workstation by a client process to the database server process at one of the nodes of the SHDDBS. The node is chosen based on its proximity to the workstation or on the subject of the index record. On receipt of the information, the server verifies the correctness and authenticity of the information and after finding everything in order, sends an acknowledgement to the client.

The server node is responsible for locating the partitions of SHDDBS where the entry should be stored and forwarding the replicated information to the appropriate nodes. In addition, the database server process is in charge of providing the catalog information for the search subsystem. In this way, the various sites of the database work in a cooperating mode to maintain consistency of the replicated portion, the replicated nature of database also ensures distribution of load and ensures continued access to the bibliography when one or more sites are temporarily nonfunctional.

2.7. The search and annotation subsystem

When making a search request, the client process communicates with the nearest catalog to determine the appropriate site of the SHDDBS. Subsequently, the client process communicates with the database and retrieves one or more semantic headers. The results of the query could then be collected and sent to the user's workstation. The contents of these headers are displayed, on demand, to the user who may decide to access one or more of the actual resources. In such a case, the best source is chosen based on optimum costs. The client process would attempt to use the appropriate hardware/software to retrieve the desired resources.

In traditional libraries, there are experienced reference librarians to help users locate the information of their interest. In the CINDI system, we use a graphical user interface and an expert system to model a human reference librarian in the search subsystem. By providing controlled items for a bibliographic record, the interface guides the CINDI users in entering the various search terms. The search subsystem generates the response to a search query by using the query manager, the error handler, the subject look-up manager, and the file manager. The query manager is responsible for interacting with the SHDDBS by sending queries and receiving search results. The subject look-up manager communicates with the catalog database by conducting standard subject term look-up. The error handler ensures that the query entered is valid otherwise presents the user with the relevant message. The file manager plays a role in managing the uploaded and downloaded files for the users who need them.

Since the scientific world depends on peer review of documents submitted for publication, the annotation sub-system encourages the users to make annotations on the existing resources; these serve the role of reviews and are stored along with the semantic header in the SHDDBS such that annotations could help the future users who are seeking for the resource to make a better decision on the relevance of the selected resource.

Based on the logical view of the CINDI system, the search and annotation subsystem is depicted as follows. Here, S1, S2, S3, and S4 represent the middle tier PHP scripts to process the requests of users.

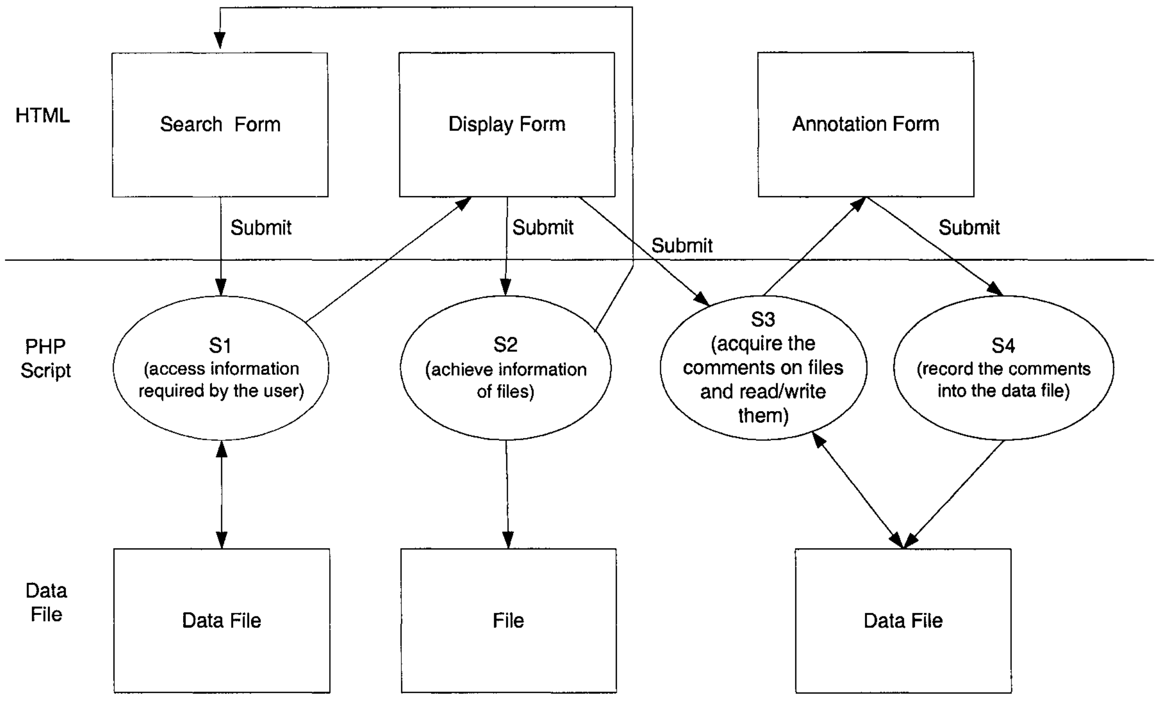

HTML
Search Form
Display Form
Annotation Form
Submit
Submit
Submit
Submit
PHP Script
S1 (access information required by the user)
S2 (achieve information of files)
S3 (acquire the comments on files and read/write them)
S4 (record the comments into the data file)
Data File
Data File
File
Data File

Figure 2.11 The logical structure of the search and annotation subsystem

2.8. CONFSYS: the conference support system

Since a conference is a major vehicle to report new research and development in human endeavors, the conference support system (CONFSYS) has been integrated as a subsystem of the CINDI. It thus provides a facility to collect the documents submitted to a conference supported by the CONFSYS and help in the evaluation process.

Authors of a conference are able to submit their papers and register their personal information via the Internet. The program chair can allocate the submitted papers to the program committee members (i.e. reviewers) manually or automatically on the Internet; the latter can review the assigned papers and make comments through the web. Not only the speed but also the accuracy of the evaluation process would be improved significantly.

Based on the previous work [GU02] [JIN03], which employs Apache and Tomcat with SSL as its web server to manage the requests from users and the responses from the relational database management system (RDBMS), we imported the ASHG into the CONFSYS whose structure is re-designed as follows:

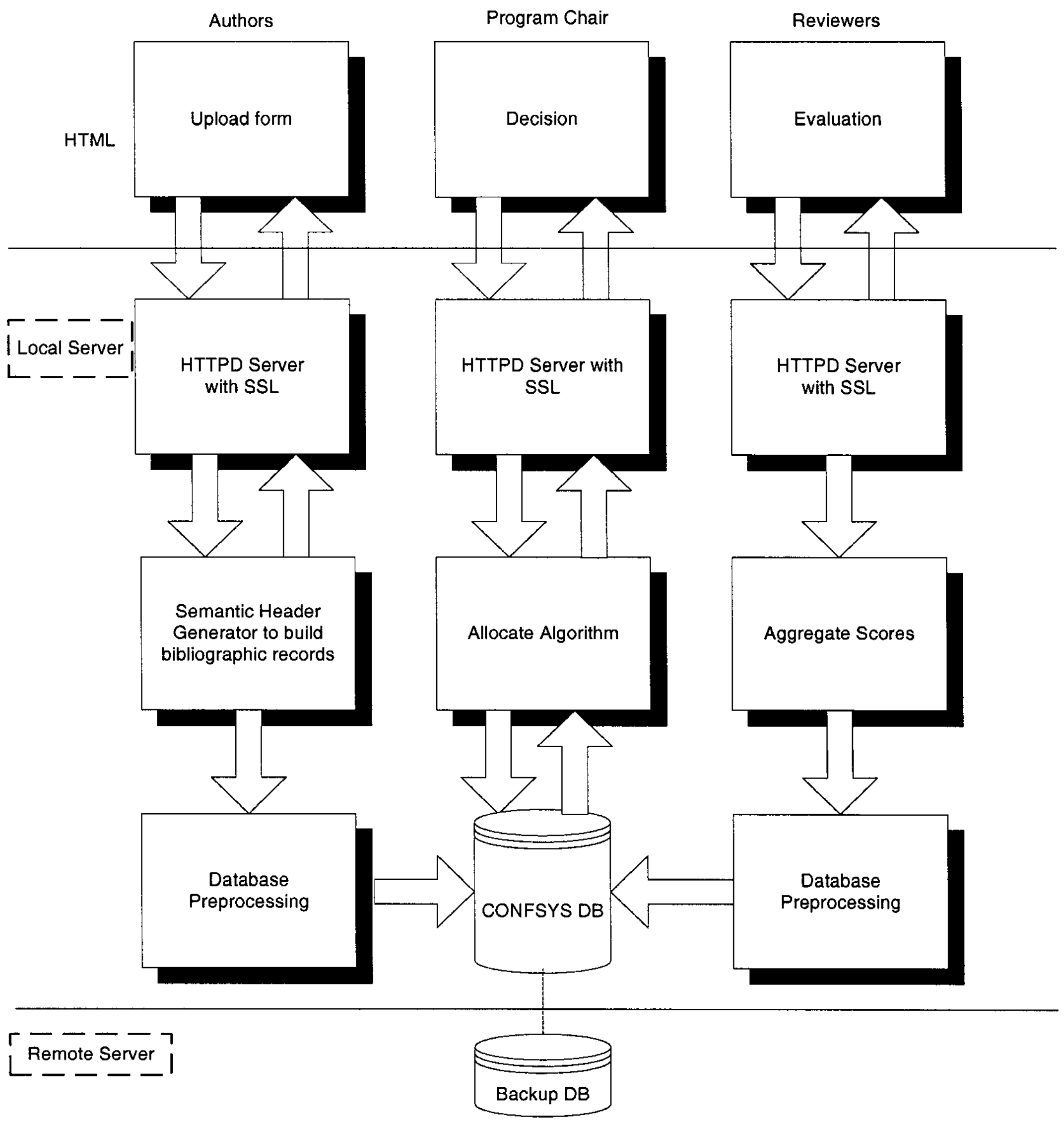


Figure 2.12 The structure of the CONFSYS

Here, the local server represents the server set up for a specific conference, and the remote server is used to backup the data for the conference in case the local server fails.

Meanwhile, it acts as a bibliographic subsystem of CINDI to be merged with it after the conference submission date.

2.9. The administration subsystem

This subsystem provides a graphical user interface for the system administrators to manage the registered resources, and the information about the providers and users of the CINDI system. The objective of the web interface is to provide a convenient way for an administrator to manage and monitor the bibliographic indexes and the subsystem for conferences, without actually selecting, and editing records directly from the database. Figure 2.13 shows the logical structure of the function.

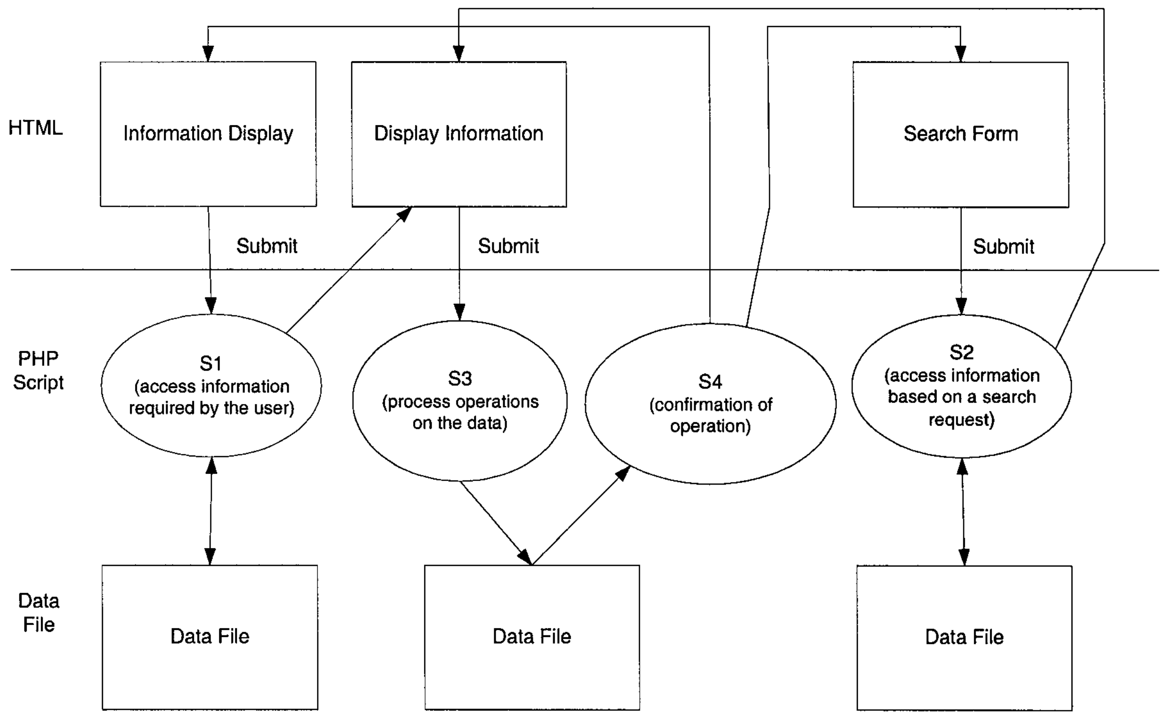

HTML
Information Display
Display Information
Search Form
Submit
Submit
Submit
PHP Script
S1 (access information required by the user)
S3 (process operations on the data)
S4 (confirmation of operation)
S2 (access information based on a search request)
Data File
Data File
Data File
Data File

Figure 2.13 The logical structure of the administration subsystem

In order to make this subsystem flexible, it offers a search function for the administrators for the personal information of users and the bibliographic information of resources, such as user name, email, author's name, and title of the resource and so on.

Since the CONFSYS is a subsystem of the CINDI, the information collected by a conference managed by the CONFSYS system should be integrated with the bibliographic database of the CINDI when the conference program is finalized. This helps enrich the digital library with the most recent development in the area covered by the conference. As for the details of integration, it will be discussed in Section 4.4.

Chapter 3 High Availability

3.1. Introduction to high availability

Rapid growth in the use of the Internet and other communication technologies has forced network service providers to offer server systems with high availability (HA) features. The goal of HA is to increase the availability of an application from the users' perspective. Improvements in availability are realized by providing more reliable components (including software) and by providing redundant components in parallel such that no single feature can cause a failure of the entire system. HA relies on the absence of single points of failure (SPOF) [DELANY00], so clusters are used to address the expectations of HA. Clustering ties several servers together so that they work as one, adding both redundancy and failover capacity to the system.

A central feature of a cluster is the ability for failover of its components and applications. Applications that were running on the failed node are restarted on its backup node(s). In order to support this failover mechanism, it is important that data, including text, is current on the backup node(s).

In the simplest failover situation (Figure 3.1), two systems are participating: it consists of the active primary or production server and the secondary or standby server, which will take over the processing tasks when the primary server fails. If the secondary server is sitting idle while waiting to take over, it is considered passive. If the secondary server is occupied with server tasks of its own while also waiting to take over, it is considered to be active as well.

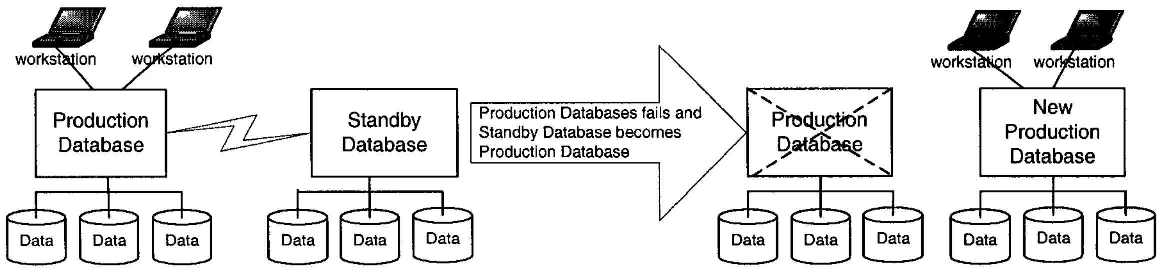


Figure 3.1 The failover situation with two systems

The CINDI system applies this primary/standby structure to achieve its high availability because it is the safest while being the simplest and cost-effective.

3.2. Methods of data replications

There are several methods used to replicate data. The common forms of replication include file-based replication, database data-file replication, and disk block replication. File-based replication is similar in function to a remote copy (i.e. using `rcp` or `scp`). The copy objects are at the file level. Database data-file replication is at the file level. However, the major difference between a standard file and a data-file is the state of file. The disk block copy method uses the disk block as the copy object. The replication approach chosen depends on the nature of the data to be replicated.

3.2.1. File-based replication

Under Linux, file-based replication can be provided by `rcp`, `scp`, and other remote user commands. Since this method is file-based, user permissions are very important to be considered when replication is set up. If a file or directory is not readable, it will be skipped over.

Because `rsh` and `rcp` are clear text protocols with passwords authentication only and without any encryption or shared secret between two connected computers, it is easy to be attacked by a cracker. Secure shell (ssh and scp) provides strong authentication and secure communication over insure channel, it is, therefore, ideal as a replacement for the clear text protocols, such as rsh and rcp. The significant advantages of ssh and scp are that they encrypt data not only before transmission but also all the way from the client to the destination server, and the session is not vulnerable to any of the security problems of telnet and rlogin, which belong to clear text protocols. Moreover, it allows users to log on to a server without ever being prompted for a password but with an encrypted pair of keys in form of a long text string, that is, a public key and a private key. The public key of username@hostname is required to be copied to the file $HOME/.ssh/authorized_keys of the system to be connected to, while the private key is stored on the system requesting the connection. This can be used for multiple connected systems since users can have multiple public keys for these systems in the file $HOME/.ssh/authorized_keys. Thus, ssh and scp on the host can be used to execute commands on and copy files to and from the remote host just as rsh and rcp, but without their vulnerability of passing passwords in clear text and with added encryption of all transmission.

3.2.2. Disk mirroring across nodes

Distributed disk block device (DRBD) copies disk blocks updated on the primary system to the backup system by sending a "heartbeat" to mirror whole disk blocks. No periodic replication is required since updates are made as they occur. Prior to failover, the active primary node is the master. After failover, the active backup node will become the master. Because both nodes are kept consistent, automatic failover can occur without the loss of data.

Each device has a state, which can be "primary" or "secondary". The application is supposed to run and to access the node with the device in the "primary" state. Every update is sent to the local "lower level block device" and to the node with the device in "secondary" state. The secondary device simply writes the data to its lower level block device. Reads are always carried out locally. If the primary node fails, the secondary device is switched into the "primary" state by heartbeat and starts the application. If the failed node comes up again, it becomes a new secondary node, and its content has to be synchronized to the primary one. In case the heartbeat fails, the secondary node will assume that the primary node failed, and take over services that were running on a primary machine.

3.2.3. Database replication

Databases are the most critical component of the Web and corporate systems, and they contain the most valuable information, without which IT-based activities could not be performed. Techniques used to replicate standard files do not work when replicating databases. Thus, database systems pose the toughest challenge for high availability and performance.

In general, two major approaches for database replication are available: asynchronous replication and synchronous replication. The asynchronous replication usually is a built-in database feature and makes use of the transaction logs that are sent to the backup nodes and applied online. Another method used for asynchronous replication is via triggers and/or snapshots, which are able to update objects in different databases. The synchronous replication takes advantage of the two-phase commit (2PC) protocol that can be a built-in feature of database, or a middle tier can be used to ensure that the transactions are committed or rolled back at all sites.

3.3. Solution for the CINDI system

In the course of operation, the CINDI produces and makes use of different types of data, which are the principle factors of the replication solution. It includes application text and uploaded/downloaded files, which must be the same on the backup node as those on the primary node, as well as transaction data with the ACID properties; this is the biggest challenge for the database replication.

3.3.1. Transaction data

Transaction data is derived from the operations of business. These data can reside in a relational database, file or other database formats, such as hierarchical repositories, object-oriented repositories, and flat files. Transaction data can be stored as a regular text or be controlled by a Relational Database Management System (RDBMS) where strict rules are applied. Transactions must adhere to the ACID properties depicted below, which must be maintained in case of a failure, as well as during normal operations.

Atomicity: Transactions are atomic (all or nothing). That is, all transactions must be complete. Once a transaction is completed, a “commit” is issued. This places the data permanently in the database. If an incomplete transaction is interrupted or cannot be completed, then the entire transaction is aborted. Thus, it has no effect at all.

Consistency: Data must be consistent within the database at all times. That is, a transaction takes the system from a consistent state to another consistent state.

For example, in an accounting or banking application where debits and credits must balance, the database remains consistent because all transactions are atomic. If a debit and its corresponding credit are included in the same atomic transaction, the database will remain consistent.

Isolation: Transactions must be isolated from one another. That is, even though there are many transactions running concurrently, any given transaction updates are invisible from the rest. Isolation provides consistent results irrespective of timing issues. Another way of saying this is that transactions are serial. Even though processing may occur in parallel, transaction effect is as if they had run one at a time. One example is the calculation of interest in a banking application. Processing timing should not result in a different balance of an account.

Durability: Finally, transactions must be durable. Durability ensures that the database remains consistent even during a failure of the system. Any data entered into the database prior to a failure must be removed after a failure if it would cause the database to be inconsistent. In other words, transactions in progress during the failure must be rolled back to restore the database to a consistent state.

3.3.2. Distributed and replicated scenarios

The CINDI database system adopts the master/slave structure for distribution and replication. The master/slave architecture is a means to create and maintain a remote copy of a production database so that critical data is protected from loss in case some harm befalls the production server or it requires planned maintenance that takes it offline. In that event, the standby database can take over processing from the primary production database, providing near continuous database availability. For the web-based applications needing high availability failover capacities, using master/slave databases is recommended because this architecture is the simplest-to-setup, lowest cost option for full high availability database failover.

In the CINDI system, since there are several file-based types of data produced by the operations of the system as mentioned previously, file-based replication should be considered; it is designed for replicating application text, including web pages and images referenced by the web pages. As for uploaded/downloaded files, they can be treated as application text except for privileges. Since application text as well as uploaded/downloaded files are less dynamic and resides in the directories accessible to all users, user/group permission is set to nobody/nogroup, which is the privilege of the web users. Also, replication frequency can be set as once per day (Table 5).

Table 5. Replication configuration

Directory	Contents	Replication methods	Frequency	Permissions
~/www	Application web pages Application scripts* Application images Application C/C++ files*	SCP	24 hours	User: nobody Group: nogroup
~/papers	Uploaded/downloaded files	SCP	24 hours	User: nobody Group: nogroup
mysql	MySQL database	MySQL Replication	Time configured	User: nobody Group: nogroup
NOTE: * excutable				

The replication work will be done by a cron job, which is a pretty standard feature of all Unix operating systems. It is simply a set of commands that are normally run using a shell and can be periodically run at times specified by users. Using the "crontab" command from the shell, a cron job can be created.

- **crontab -l** will show the currently set up cron jobs on the server.
- **crontab -r** will delete the current cron jobs.
- **crontab -e** will allow shell users to add or edit the current cron jobs by using the default text editor to edit the "crontab file".

For example, in the CINDI system, the time is set in the early morning everyday. Thus, a cron job will be set up with the command "**crontab –e**" as follows:

```
0 2 * * * shell_script
```

Here, "shell_script" refers to a shell command or a shell script to be executed for the replication; the first five numbers represent the specified time as described below:

- The first number is the minute of the hour for the command to run on.
- The second number is the hour of the day for the command to run on.
- The third number is the day of the month for the command to run on.
- The fourth number is the month of the year for the command to run on.
- The fifth number is the day of the week for the command to run on.

Since the replication is required to be done everyday, the last three numbers are replaced by "*".

3.3.3. Replication in MySQL database

Replication is the process of copying and maintaining database objects in multiple databases that make up a distributed database system [ORACLE99]. Basically, the main idea behind replication is that an update operation is performed first locally, and then, propagated to other remote replica databases belonging to a single whole system; therefore, the entire system becomes distributed. Theoretically, supposing there are no delays due to synchronization and no update transactions conflicts, this system would assure the end user a fast and consistent way of information retrieval with high availability because of alternate data access. Couloris and Dollimore [GJ88] summarize motivations for replication in the following way:

- **Performance enhancement:** Replication can reduce communication traffic in a distributed system and improve response time by providing clients with multiple destinations and switching their requests to a certain distributed database server depending on the origin of clients;
- **Enhanced availability and robustness:** Replication can increase system availability by making it possible to access the same file from more than one server so as to enable several clients' requests to be serviced by another server when one of the servers is not reachable, thereby reducing the effect of server and communication failures;
- **Consistency:** In a distributed database system, replication enables different replica databases to give the same results using the same logical set of data for the same request from the client at the same time. This concept has to deal with a trade-off when translated to a real system; when data are updated on a local database, replicas have to be synchronized. This propagation process can be done synchronously or asynchronously as mentioned above. The synchronous model is capable of maintaining the consistency of the system through an ordered processing of the requests where subsequent requests are "locked" until existent ones are done in order that all replicas are updated as consistent transactions; however, performance is reduced by locks. In contrast to synchronization, the asynchronous model does the data propagation process after updates, which will take place, instead of immediately,

with a delay that can vary from architecture to architecture. Therefore, it is possible that a client could access a data from a replica, which was not yet updated. Consistency is therefore not guaranteed, whereas performance is.

- **Replication transparency:** In a distributed database system, transparency is the property of hiding the structure and mechanisms of a system to the clients in order to make them view the system as a single rather than a collection of different components.

Since replication in MySQL is a pretty recent addition still under development, only the master/slave structure has been adopted as the replication model by MySQL. Basically, in a distributed system, a MySQL database server acts as the primary server (i.e. master) whereas other replica servers act as slaves, and updates propagation goes one-way only, from the master to the slave servers by using and saving information and timestamps in a binary log on the primary system. The master server keeps a binary log of updates; the slave, upon connecting, informs the master where it had left off since the last successfully propagated update, catches up on the updates, and then blocks and waits for the master to notify it of new updates. There are three threads that are involved in replication: one named `Binlog_dump` on the master, which is created on the master and required by the slave, and two on the slave [MYSQL03]; these are the I/O thread and the SQL thread. The former is issued on the slave when the slave starts, and connects to the master and asks it to send its binlogs. After the thread `Binlog_dump` sends the binlogs, the I/O thread reads it and simply copies it to some local files in the slave's data directory called relay logs. The latter one, i.e. the SQL thread, reads the relay logs and executes the queries it contains.

Basically, to the end users, replica offers a large benefit in terms of availability and robustness due to its capacity of failover. Another advantage of replica is the possibility of performance enhancement achieved by sending a part of the non-updating queries (i.e. read queries) to the replica server. Unfortunately, replication transparency is not granted by MySQL itself, and also consistency of data can be really compromised if privileges on tables and databases are not correctly set. That is why the configuration of the privileges becomes crucial in the system. The following steps are for configuring replication in MySQL [MYSQL03] for the CINDI system.

- **Step one**: Configure the master

  1. Set up the replication user

     Assuming two servers, A (10.1.1.1), the primary server as the master, and B (10.1.1.2), the standby server as the slave. MySQL's replication is done by having the slave server (B) connect to the master (A) and read the binary update log, and incorporating those changes into its own databases. The slave needs a user account to connect to the master; therefore, an account with only the FILE privilege on the master (A) requires creating with the following command:

     ```
     GRANT FILE ON *.* TO replicate@"%" IDENTIFIED BY
     '<password>';
     ```

     For example,

     ```
     GRANT FILE ON *.* TO replicate@10.1.1.2 IDENTIFIED BY
     'password';
     ```

  2. Shutdown the MySQL server

     ```
     mysqladmin -u root -p <password>  shutdown
     ```

  3. Create a snapshot

```
tar -cvf /tmp/mysql snapshot.tar /path/to/mysqldata-dir
```

4. Edit the "my.cnf" file to enable the binary update log by inserting the following lines:

```
[mysqld]
log-bin
server-id = 1
```
(uniquely, similar to IP address)

5. Restart MySQL on the master

- **Step two**: Configure the slave

1. Stop the MySQL server on the slave.

2. Copy the database files of the snapshot into the data directory on the slave server using the following command:

```
tar xzf snapshot.tar
```

Then, change the file mode and the directory to 660, which means that it is readable and writeable for owners and group only.

3. Start the MySQL server on the slave to ensure the data snapshot taken in the previous step successfully by running some selection queries. If the selection queries are successful, the MySQL server can be shutdown.

4. Edit the "my.cnf" file on the slave by adding the following lines:

```
[mysqld]
 master-host = <the hostname of the master>  e.g. 10.1.1.1
 master-user = replicate
 master-password = password
 master-port = <IP port for the master>
 server-id = 2
```

```
    default-character-set = <as for the master>
```

5. Start the slave server.

The slave server keeps track of what updates it has received from its master in the "master.info" file. The status of the slave thread can be seen through the SQL command "`SHOW SLAVE STATUS`". Any errors in processing the binary logs on the slave will cause the slave thread to exit, and generate a message in the *.err log. The errors can then be corrected, and the SQL statement '`SLAVE START`' can be used to restart the slave thread, where it will pick up where it left off in the binary log of the master.

For the CINDI system, the database replication by using MySQL's built-in replication is under consideration. Here, we propose the details of replication for the future work. In order to make the CINDI system highly available, in this thesis, we made use of the MySQL command `mysqldump` to create the data snapshot and then take advantage of the file-based approach to implement the database backup (see Appendix C). For the synchronous approach required by the CONFSYS system, we utilized the PHP functions `mysql_connect()` and `mysql_select_db()` to switch the access directions into the different data server so as to make the CINDI system intelligently manage the users' requests. Its implementation will be discussed in the following chapter.

Chapter 4 Query Processing

Since MySQL has a well-deserved reputation for being a very fast database server that is also quite easy to be set up and used, the CINDI system employs MySQL to build its distributed and replicated database system. MySQL, a client/server system, supports multiple platforms, such as Apple Macintosh OSX, IBM OS/2, Linux, Microsoft Windows, as well as countless flavors of Unix (for example, AIX, BSDI, DEC Unix, FreeBSD, HP-UX, Open BSD, Net BSD, SGI Iris, Sun Solars, SunOS4, SCO Unix). For the development of MySQL applications, there is also a host of APIs (i.e. Application Programming Interfaces) and libraries, for the languages including C, C++, Java, Perl, PHP, Python, as well as Tcl.

Compared to commercial relational databases, such as Oracle and DB2, MySQL does not support views, triggers and stored procedures. However, MySQL is conspicuous for its countless extensions. Depending on users' demand, MySQL can recognize some different types of tables, such as `ISAM`, `MyISAM` (default), `BDB`, `InnoDB`, `Gemini`, `MERGE` and `HEAP`. [KOFLER01] Moreover, temporary tables can be created by MySQL in order to assist in execution of `SELECT` queries, although they are not a separate table type.

4.1. Database design of the CINDI system

Based on the functional requirement and by optimizing the previous work [WANG02] [GU02] [JIN03], the entity-relationship (ER) data model is re-designed for the CINDI system in this thesis as shown in Figure 4.1(a)-(c). Each object namely an entity is presented as a rectangle, which is distinguishable from other objects. For example, the object resource is used to represent the uploaded documents, as an entity in the real world. Each entity has some attributes to be distinguished from the other entities. These attributes are called the keys of a table in the database design. Take resource as an example; the attributes title, keyword, abstract and so on are used to describe the resource, and are parts of the key fields represented using ellipses in the ER diagram. In Figure 4.1(b), the tables or entities on the left side of the line are stored on the CINDI system server, while the rest are applied on the conference local site server.

In light of the database normalization of 5NF, which requires reducing redundant data, the current database design of the CONFSYS subsystem needs improving in the near future. As shown in Figure 4.1(c), we give the improved design for the management of the authors' information.

To be useful, each entity must have at least an association with other entities; this association is called a relationship and represented by a diamond. Examples include the following: a resource is written by one or more authors; an author wrote one or more resource. Relationships are modeled as tables in the relational model. There are three possible types of relationships below between two entities:

- one-to-one (1:1), exists when the primary record in Table A will have only one related record in its associated Table B. Another distinguishing factor of a one-to-one relationship is both attributes used for the relationship are primary keys. In other words, the related data values in both tables must be unique; there can only be one matching record, in a one-to-one relationship. For example, an administrator has a unique authenticated record in the table `admin_authen`;

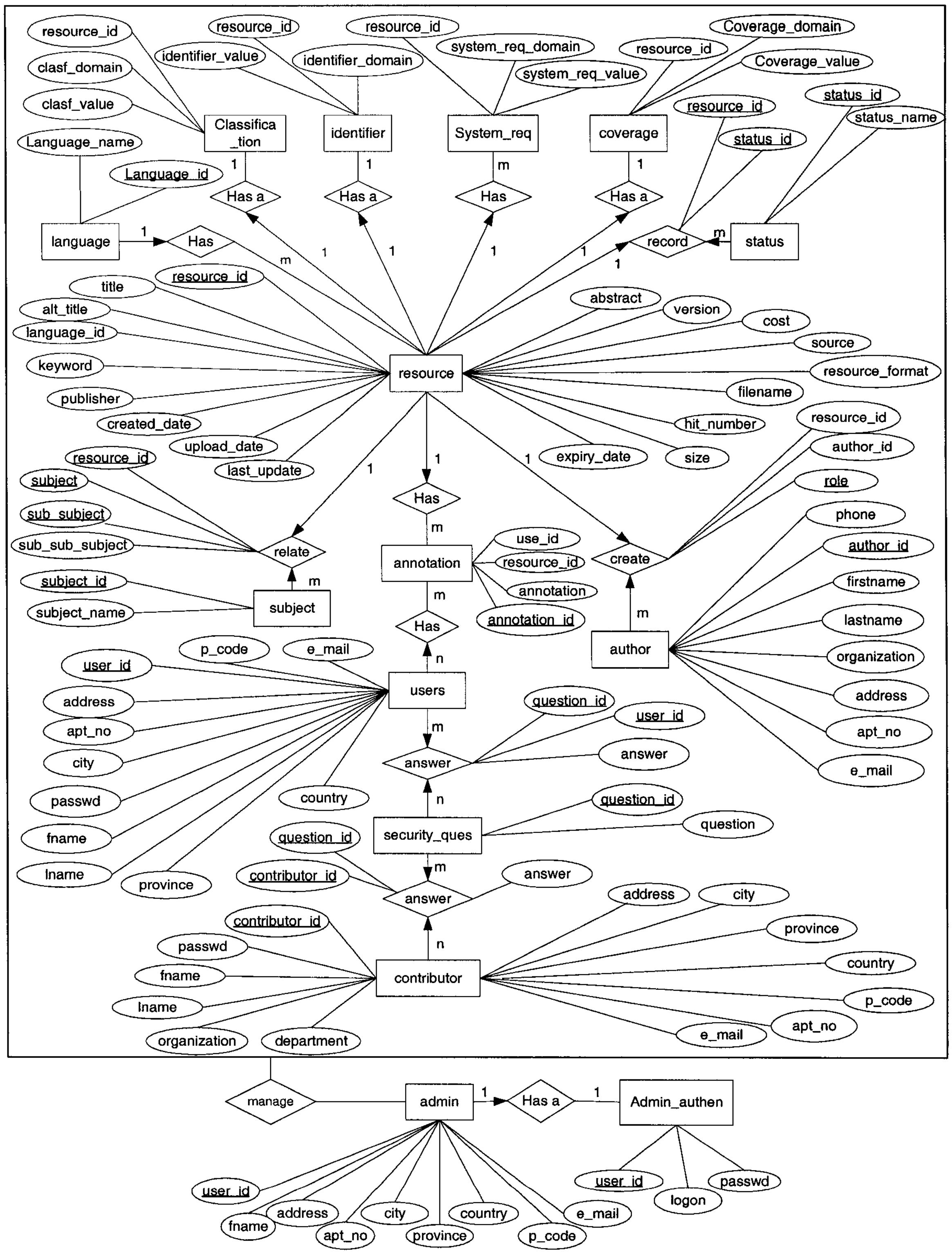


Figure 4.1(a) The ER model of the CINDI system

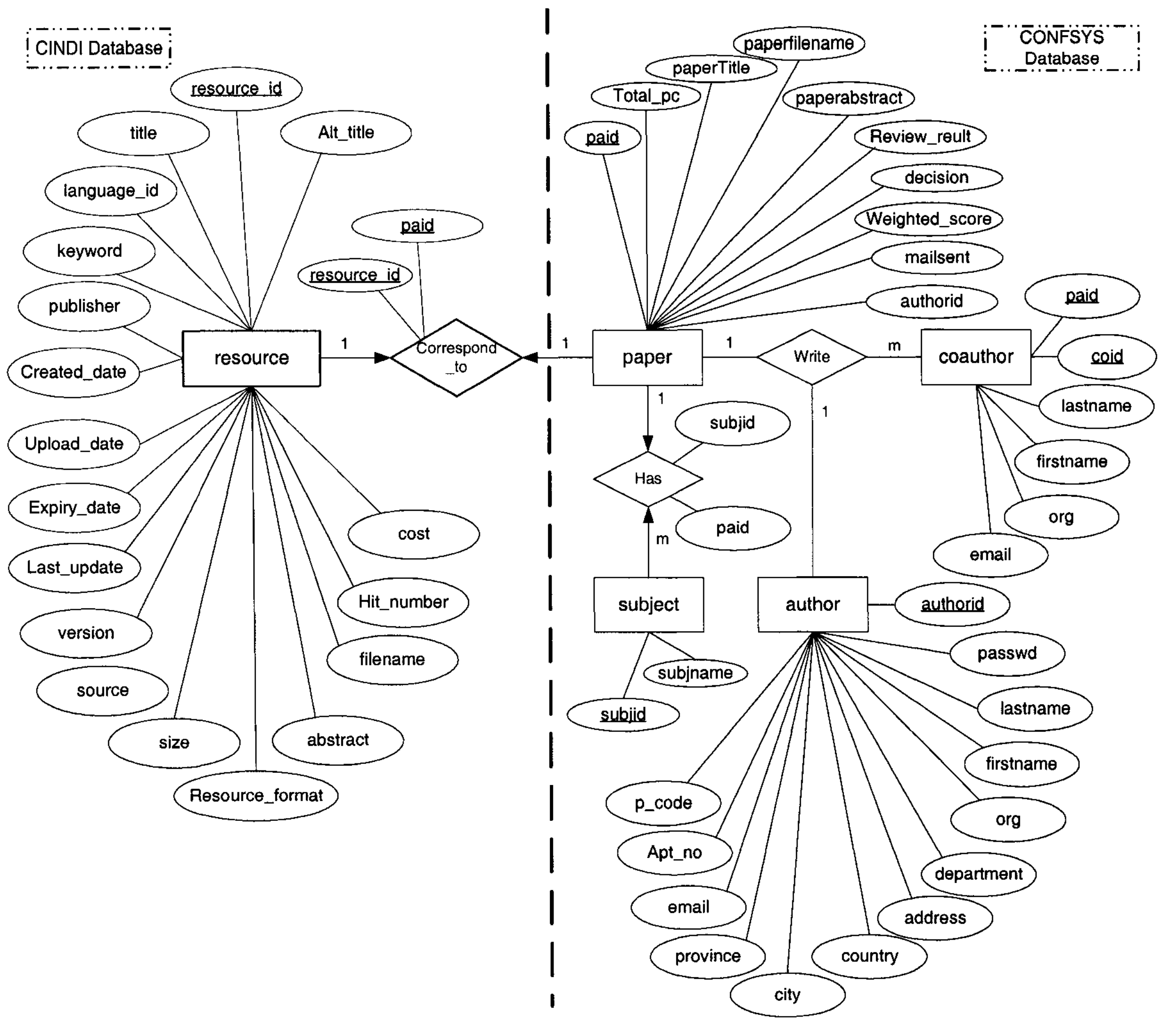


Figure 4.1(b) The ER model of the CINDI system including the CONFSYS

- one-to-many (1:m), the most common type of relationship, exists when the primary record in Table A can have many related records in its associated Table B. As an example, one resource is associated with several authors. many-to-one (m:1) is the same as one-to-many (1:m) only in the different direction;
- many-to-many (m:n), exists when a primary record in Table A has more than one matching record in its associated Table B; meanwhile, when a primary record in Table B has more than one matching record in its associated Table A. For example,

one user has several security questions to be answered, and a security question may be selected by several different users.

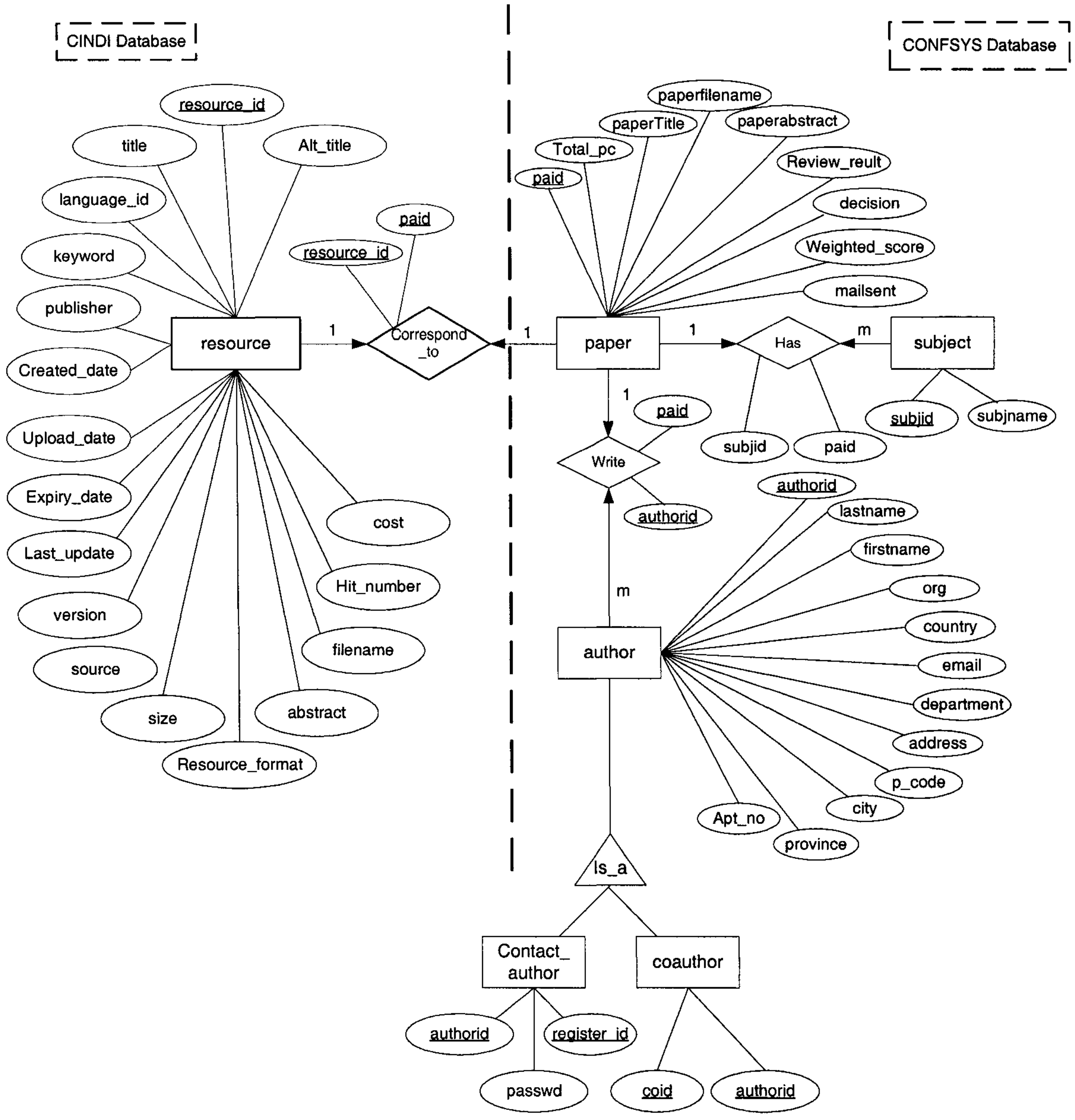


Figure 4.1(c) The improved ER model of the CINDI system including the CONFSYS

From the object-oriented point of view, some of the associations between entities can be viewed as virtual entities since these associations have descriptive attributes. Examples in the CINDI system consist of the relationship between the uploaded resources and their authors, as well as the association of the resource and subjects. In order to distinguish the relationships clearly, a concept of key constraints is introduced in database design. In general, the key constraint on an entity set in a relationship set is indicated with an arrow from the entity to the relationship. For example, the key constraint on Has in the ER diagram tells that a resource has at least one subject.

The key fields are involved in the associations, mainly via primary keys and foreign keys. Serving as a primary key, the combination of the values of one or more attributes must be unique such that no null values are accepted. Primary keys are used to locate a particular data record in a table as fast as possible. Foreign keys, being the primary keys of other tables, are used to enforce referential integrity on the relationship by referring to records in the referenced table. In order to keep the ACID properties, the primary keys in the referenced table must match the foreign keys of the referencing table. For example, `resource_id`, the primary key in the table resource, is a foreign key in the table `resource_author`, which is used to represent the relationship between the resource and its authors.

In the CINDI database, the tables with foreign key constraints are defined as type `INNODB` because `INNODB` provides MySQL with a transaction-safe (namely ACID compliant) table handler with commit, rollback, and crash recovery capabilities. In `INNODB` with its locking mechanism, all user activities take place within transactions. If the auto-commit mode is applied, each SQL statement will organize as a single transaction. If the auto-commit mode is switched-off by setting `autocommit` to zero, then a user always has a transaction open until it is either committed or rolled back. After the SQL `COMMIT` or `ROLLBACK` statement, the current open transaction will end and a new one will start. Both statements will release all `INNODB` locks that are set while the current transaction is processing. A `COMMIT` means that the changes made in the current transaction are made permanent and become visible to other users. In contrast, a `ROLLBACK` cancels all modifications made by the current transaction.

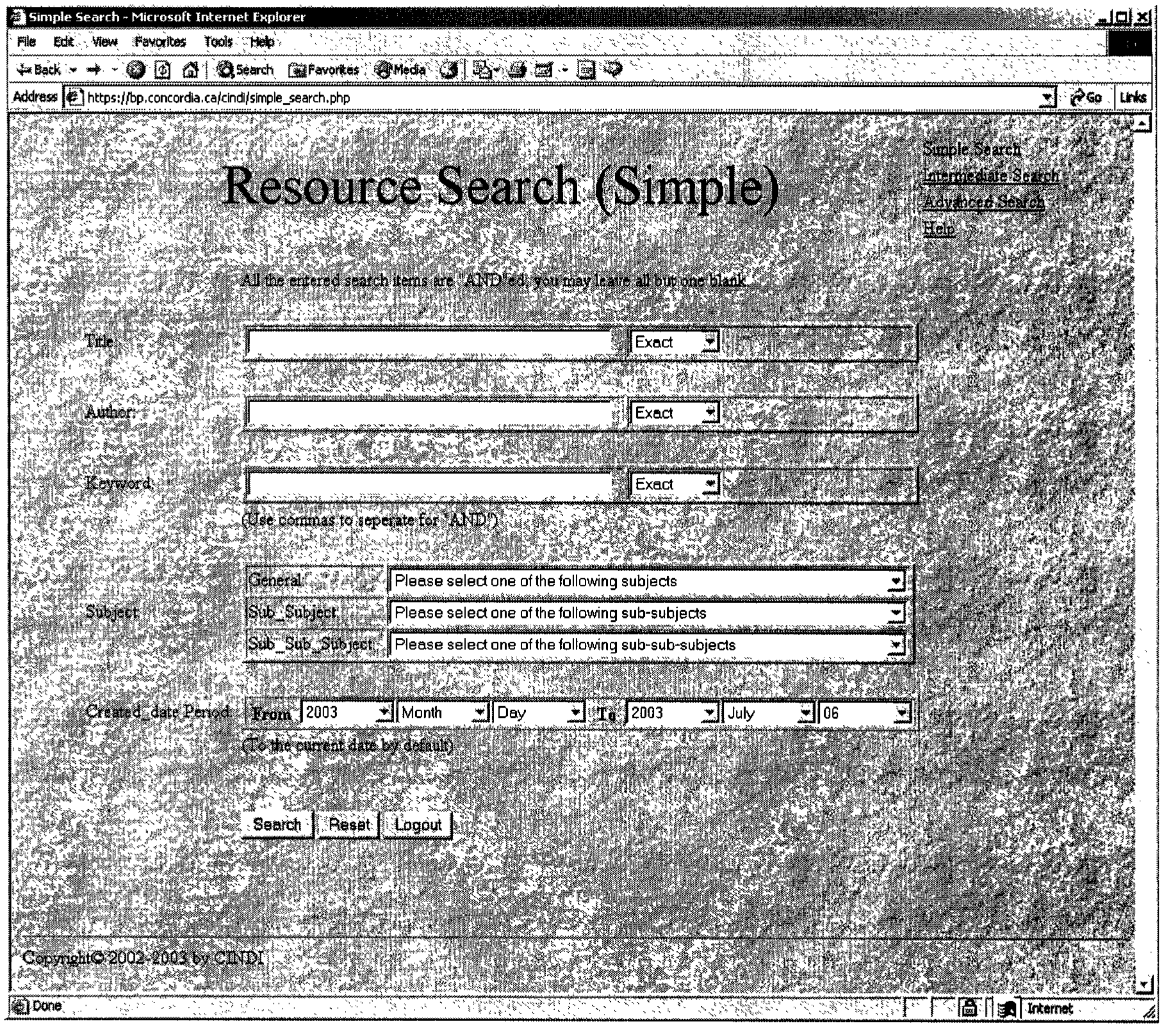
Because of the lack of triggers and saved procedures in MySQL, it is necessary that application implementation follow the foreign key restriction. That is, for insertion activity, a new record must be inserted into the referenced table first. Then, its corresponding information regarding the relationship between the others will be done in the referencing table(s). For deletion activity, the operations must be done in reverse as mentioned for the insert operation. For example, when a resource is registered, the transactions on insertion will be processed from one table to another in the following order: `language`, (`resource`, `author`, `subject`) and the other related tables (i.e. referencing tables), such as `resource_author`, `resource_subject`, `classification`, `system_req`, `identifier`, and `coverage`. Here, the tables `resource`, `author`, and `subject` are referenced tables. In contrast to insertion, the deletion activity for a specified record of a resource must occur in the following order: the referencing tables (e.g. `resource_author`, `resource_subject`, `classification`, `system_req`, `identifier`, and `coverage`), and then the referenced tables (e.g. `resource`, `author`). The details are given in Appendix A.

4.2. Search function

There are two search subsystems in the CINDI system. One is designed for users to locate digital resources of interest; the other one is embedded in the administration to assist system administrators in managing users' information and the bibliographic system.

4.2.1. Search function for users

The search subsystem for users of the CINDI system provides three levels of search for users depending on the users' search criteria: simple search, intermediate search, and advanced search.


Simple Search - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://bp.concordia.ca/cindi/simple_search.php
Go Links
Simple Search
Intermediate Search
Advanced Search
Help
Search Reset Logout
Done
Internet

Resource Search (Simple)

All the entered search items are "AND"ed, you may leave all but one blank

Title: Exact

Author: Exact

Keyword: Exact

(Use commas to seperate for "AND")

Subject: General: Please select one of the following subjects
Sub_Subject: Please select one of the following sub-subjects
Sub_Sub_Subject: Please select one of the following sub-sub-subjects

Created_date Period: From 2003 Month Day To 2003 July 06

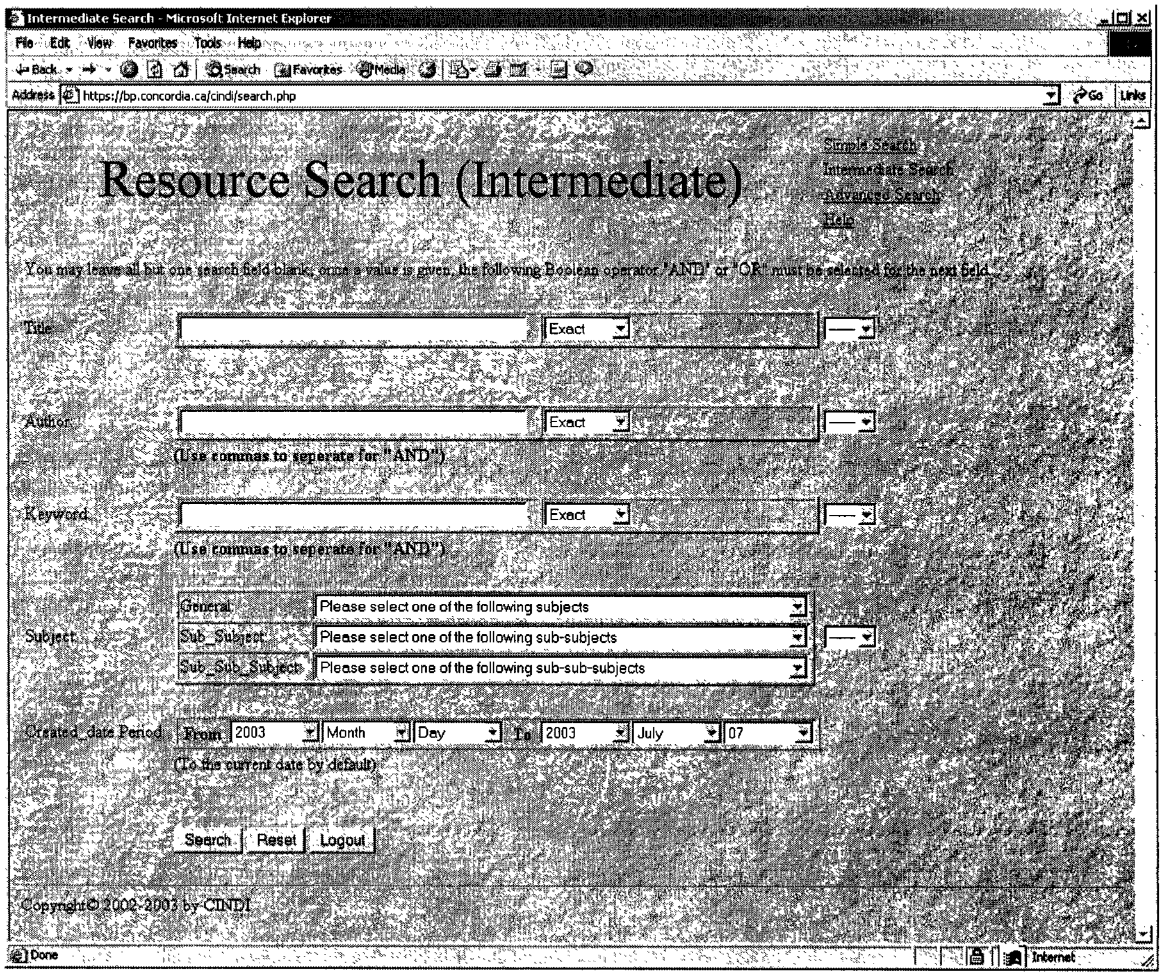
(To the current date by default)

Copyright© 2002-2003 by CINDI

Figure 4.2 The interface of simple search

Simple search allows users who do not know exactly the resource being looked for to enter at least one of the fields, namely title, author, keyword, subject, or a date period as shown in Figure 4.2. In order to make the system convenient, the system date is set as the default value of `Created_date To`, and the beginning of the current year will be the default value of Created_date From. For the users who do not know the exact title, author, and keyword of a resource, they can select "Substring" when they are looking for its related information; otherwise, "Exact" should be selected for the specified values of the fields. In the SQL statement for this type of entries, the Boolean expression for the SQL WHERE clause, these fields would be conjuncts. For the fields of Title and Author, depending on the use of "Exact" or "Substring", the corresponding match operator would be either "=" or "LIKE". For the field of Keyword, the selection of "Substring" will apply the operator "LIKE" for matching the keyword; the selection of "Exact" for keywords will require Full-Text searching; this would involve creating a temporary table of type MyISAM rather than using the table resource itself of INNODB, which does not support Full-Text searching.

Intermediate search enhances simple search facility for users who are familiar with the information they are looking for or would like to narrow the search results. Herein, users can provide more than one value of the fields for authors and keywords, which require to be separated by commas for a conjunction. Boolean search is a search that allows users to narrow their results through the use of Boolean operators. Boolean operators include AND, OR, XOR, and other operators; a user can narrow results by using them appropriately. In the search subsystem, the relationship among two or more comma separated values is considered to be the Boolean operation AND. For the relationship between two separate fields, it can be chosen as conjunct or disjunct by choosing the Boolean operator AND or OR.


Resource Search (Intermediate)

Simple Search
Intermediate Search
Advanced Search
Help

You may leave all but one search field blank; once a value is given, the following Boolean operator "AND" or "OR" must be selected for the next field.

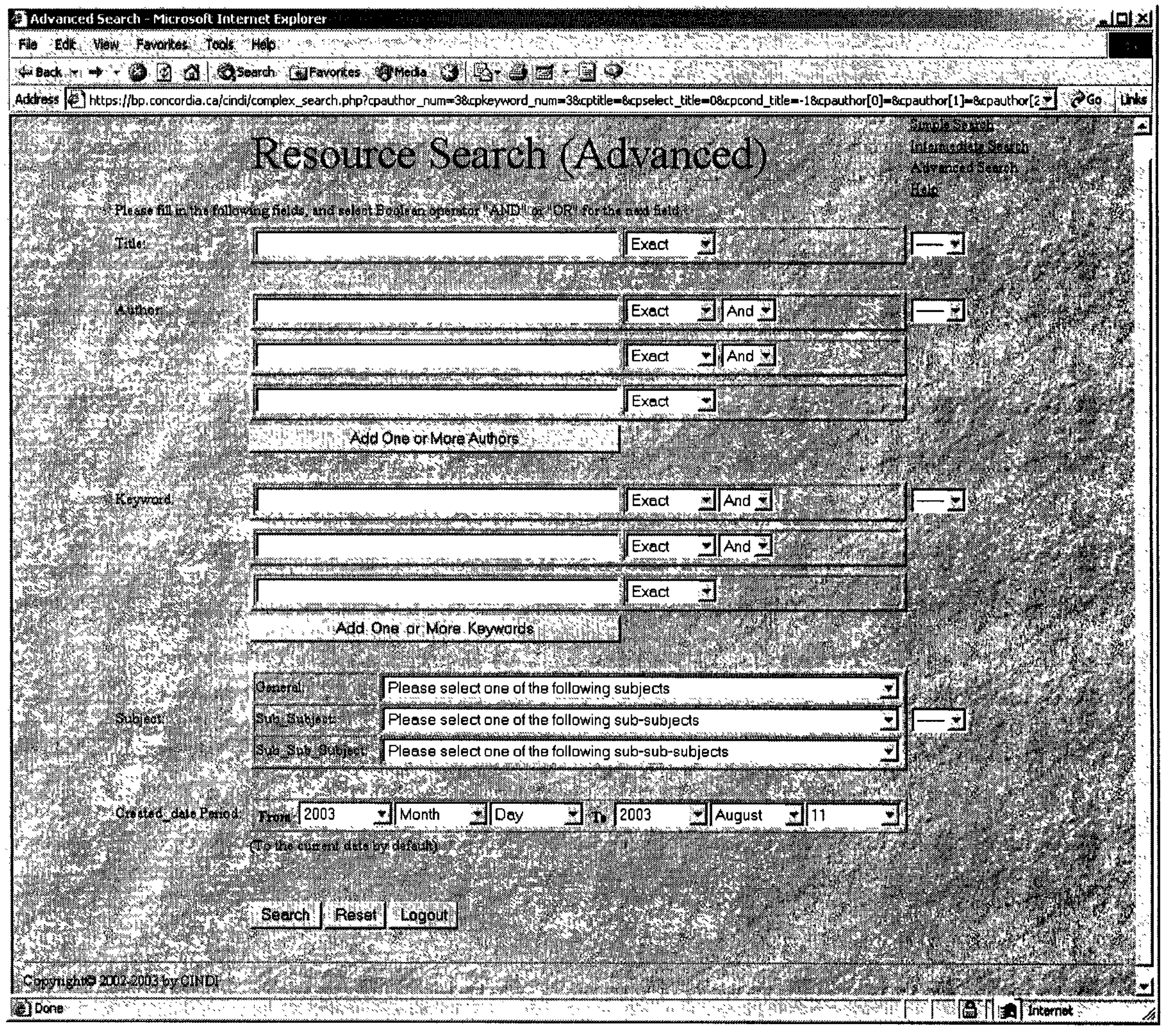
Title		Exact	—
Author	(Use commas to seperate for "AND")	Exact	—
Keyword	(Use commas to seperate for "AND")	Exact	—
Subject	General: Please select one of the following subjects Sub_Subject: Please select one of the following sub-subjects Sub_Sub_Subject: Please select one of the following sub-sub-subjects		—
Created_date Period	From 2003 Month Day To 2003 July 07 (To the current date by default)		

Search Reset Logout

Copyright© 2002-2003 by CINDI

Figure 4.3 The interface of intermediate search

Advanced search requires users to be more familiar with what they are interested in so as to enter more details of the resource by filling in the form and choosing the appropriate Boolean operators not only between the entered values but also among the fields. Compared with intermediate search that only a single Boolean operation can be done between the values of the fields for authors and keywords, advanced search makes it more flexible by allowing users to define the association among three or more values of these fields as the combination of conjunct and disjunct as shown in Figure 4.4.


Advanced Search - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://bp.concordia.ca/cindi/complex_search.php?cpauthor_num=3&cpkeyword_num=3&cptitle=&cpselect_title=0&cpcond_title=-1&cpauthor[0]=&cpauthor[1]=&cpauthor[2
Go Links
Simple Search
Intermediate Search
Advanced Search
Help
Add One or More Authors
Add One or More Keywords
Search
Reset
Logout
Done
Internet

Resource Search (Advanced)

Please fill in the following fields, and select Boolean operator "AND" or "OR" for the next field.

Title: Exact

Author: Exact And

Exact And

Exact

Keyword: Exact And

Exact And

Exact

Subject: General: Please select one of the following subjects

Sub_Subject: Please select one of the following sub-subjects

Sub_Sub_Subject: Please select one of the following sub-sub-subjects

Created_date Period: From 2003 Month Day To 2003 August 11

(To the current date by default)

Copyright© 2002-2003 by CINDI

Figure 4.4 The interface of advanced search

Based on the previous work [SHAYAN97] [WANG02], we rewrote the overall search query grammar in BNF for the three levels of search as follows:

```
<simple search>         ::= <operand> [ <op> <operand> [ <op> <operand>
                            [ <op> <operand> [ <op> <operand> ] ] ] ]
<intermediate search>   ::= <operand> [ <op> <operand> [ <op> <operand>
                            [ <op> <operand> [ <op> <operand> ] ] ] ]
<advanced search>       ::= <operand> <op> <operand> <op> <operand>
                            <op> <operand> <op> <operand>
<operand>               ::= <title> | <subject> | <author_unit> |
                            <keyword_unit> | <date>
<op>                    ::= <AND> | <OR>
<title>                 ::= <exacttitle> | <substringtitle>
<exacttitle>            ::= <string>
```

```
<substringtitle>          ::= <string>
<author_unit>             ::= <author> | <AND | OR> <author_unit>
<author>                  ::= <exactauthor> | <substringauthor>
<exactauthor>             ::= <string>
<substringauthor>         ::= <string>
<keyword_unit>            ::= <keyword> | <AND | OR> <keyword_unit>
<keyword>                 ::= <exactkeyword> | <substringkeyword>
<exactkeyword>            ::= <string>
<substringkeyword>        ::= <string>
<subject>                 ::= <general> | <general> <AND> <sub_subject> |
                              <general> <AND> <sub_subject>
                              <AND> <sub_sub_subject>
<general>                 ::= <string>
<sub_subject>             ::= <string>
<sub_sub_subject>         ::= <string>
<date>                    ::= <from_date> | <to_date> |
                              <from_date> AND <to_date>
<from_date>               ::= <year> - <month> - <day>
<to_date>                 ::= <year> - <month> - <day>
<string>                  ::= <character> | <character> <string>
<character>               ::= a|A|b|B|c|C| … |x|X|y|Y|z|Z|0|1|2| … |7|8|9
<day>                     ::= 1|2|3|4| … |28|29|30|31
<month>                   ::= 01|02|03| … |10|11|12
<year>                    ::= current year - 10|current year - 9|
                              current year - 8| … |current year - 2|
                              current year - 1|current year
```

where the meta-symbols of BNF are:

::= meaning "is defined as"

| meaning "or"

< > angle brackets used to surround category names.

Considering the search scenario, we correct some grammar representations in the previous work [WANG02], such as the description of year that was defined as a fixed value and will be changed based on the current date.

To process a selection operation with a general selection condition, we first express the condition in conjunctive normal form (CNF); that is, as a collection of conjuncts that are connected through the use of the ∧ operator (i.e. AND). Each conjunct consists of one or more terms connected by ∨ (i.e. OR). Conjuncts that contain ∨ are said to be disjunctive, or to contain disjunction.

Simple search has a selection of the form below where op is the conjunction operator (∧):

```
[title = 'string'  |  title like '%string%']
op [author_name = 'string' |  author_name like '%string%']
op [keyword = ' string ' | keyword like '%string%']
op [subject = 'subject']
op [sub-subject = 'sub-subject']
op [sub-sub-subject = 'sub-sub-subject']
op [created_date_from >= 'from date']
op [created_date_to <= 'to date']
```

Intermediate search has a selection with the following condition where op is either conjunction (∧) or disjunction (∨):

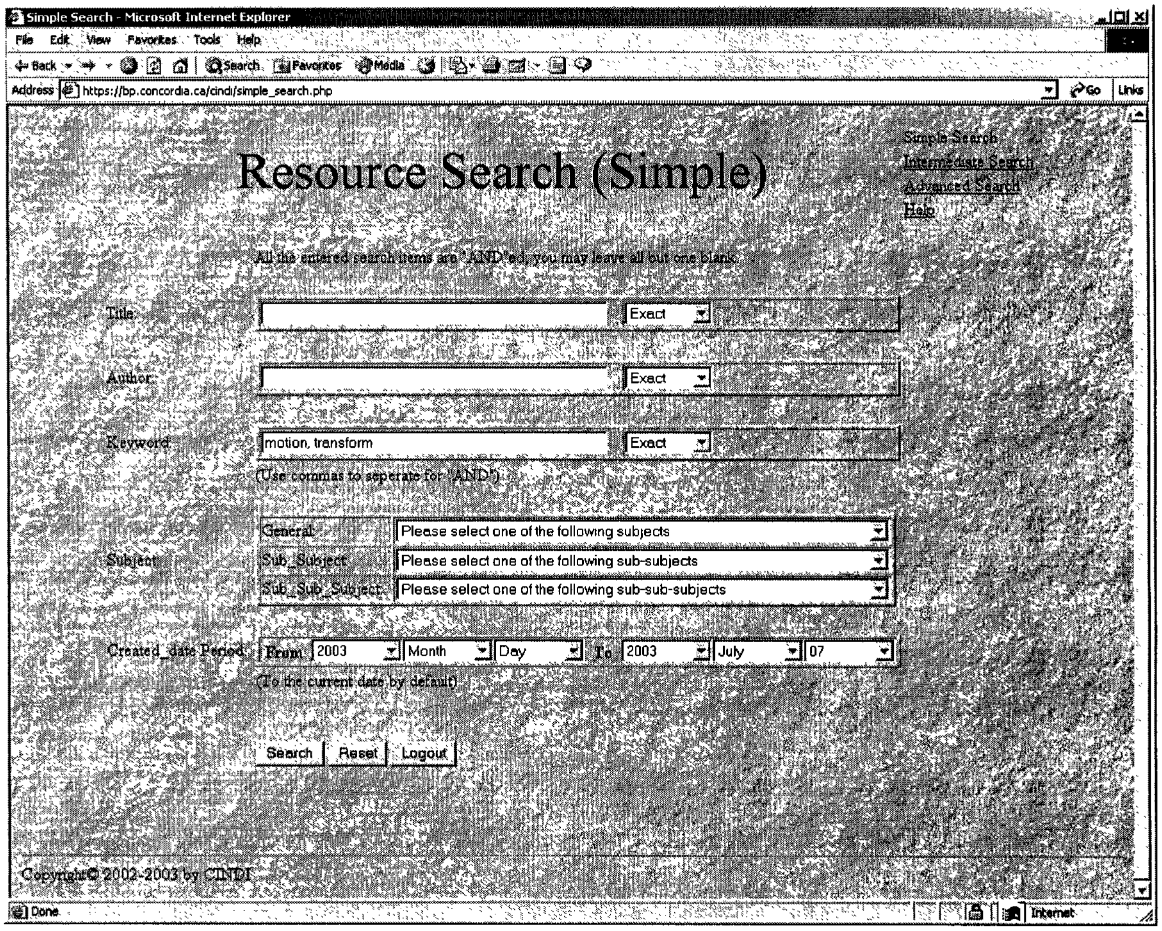
```
op ::= ∧ | ∨
[title = 'string' | title like '%string%']
op [author_name = 'string' |  author_name like '%string%']
op [keyword = 'string' | keyword like '%string%']
op [subject = 'subject' ∧ [sub-subject = 'sub-subject' ∧ [sub-
sub-subject = 'sub-sub-subject']]]
op created_date_from >= 'from date' ∧ created_date_to <= 'to
date'
```

Advanced search has a selection with the following condition where `op` is either conjunction (∧) or disjunction (∨):

```
op ::= ∧ | ∨
(title = 'string' | title like '%string%')
op (author_name = 'string1' | author_name like '%string1%') op
(author_name = 'string2' | author_name like '%string2%') op ...
op (author_name = 'stringn' | author_name like '%stringn%')
op ((keyword = 'string1' | keyword like '%string1%') op (keyword
= 'string2' | keyword like '%string2%') op ... op (keyword =
'stringn' | keyword like '%stringn%'))
op (subject = 'subject' ∧ sub-subject = 'sub-subject' ∧ sub-sub-
subject = 'sub-sub-subject')
op (created_date_from >= 'from date' ∧ created_date_to <= 'to
date')
```

For simple search, the users are allowed to enter more than one value of keywords separated with commas (see Figure 4.5 (a)-(b)). For instance, Figure 4.5(a) indicates that the query is based on the "motion" AND "transform" as keywords, and the created period by default from "Jan. 1, 2003" to "Jul. 7, 2003".

For intermediate and advanced search, in addition to keywords, one or more author names are accepted by using commas and by clicking button to separate them, as shown in Figure 4.6 (a)-(b) and Figure 4.7 (a)-(b), respectively. For example, Figure 4.6(a) shows that the substring title is "fast", the substring of author names is "Tu, Wang", the keyword is "motion, transform", the subject is "Electrical Engineering", and the created period from "Jan. 1, 2003" to "Jul. 7, 2003". Figure 4.7(a) gives an example of advanced search to search for the resources by "Tu" and "Wang" created from "Jan. 1, 2003" to "Jul. 7, 2003" with the substring title "fast" and the keywords "motion" or "transform", as well as the subject "Computer Science" and the sub-subject "Numerical analysis in mathematics of computing" and the sub-sub-subject "Fast fourier transforms (fft) approximation". For advanced search, if the Boolean operator is OR, then the selection with the condition on authors' names follows the above CNF. However, when the AND Boolean operator is chosen, it adds the complexity of the selection. Here, two solutions are considered and applied to the system for searching function.



Simple Search - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://bp.concordia.ca/cindi/simple_search.php
Go Links
Simple Search
Intermediate Search
Advanced Search
Help
Search Reset Logout
Done
Internet

Resource Search (Simple)

All the entered search items are "AND"ed, you may leave all but one blank.

Title: [ ] Exact

Author: [ ] Exact

Keyword: motion, transform Exact

(Use commas to seperate for "AND")

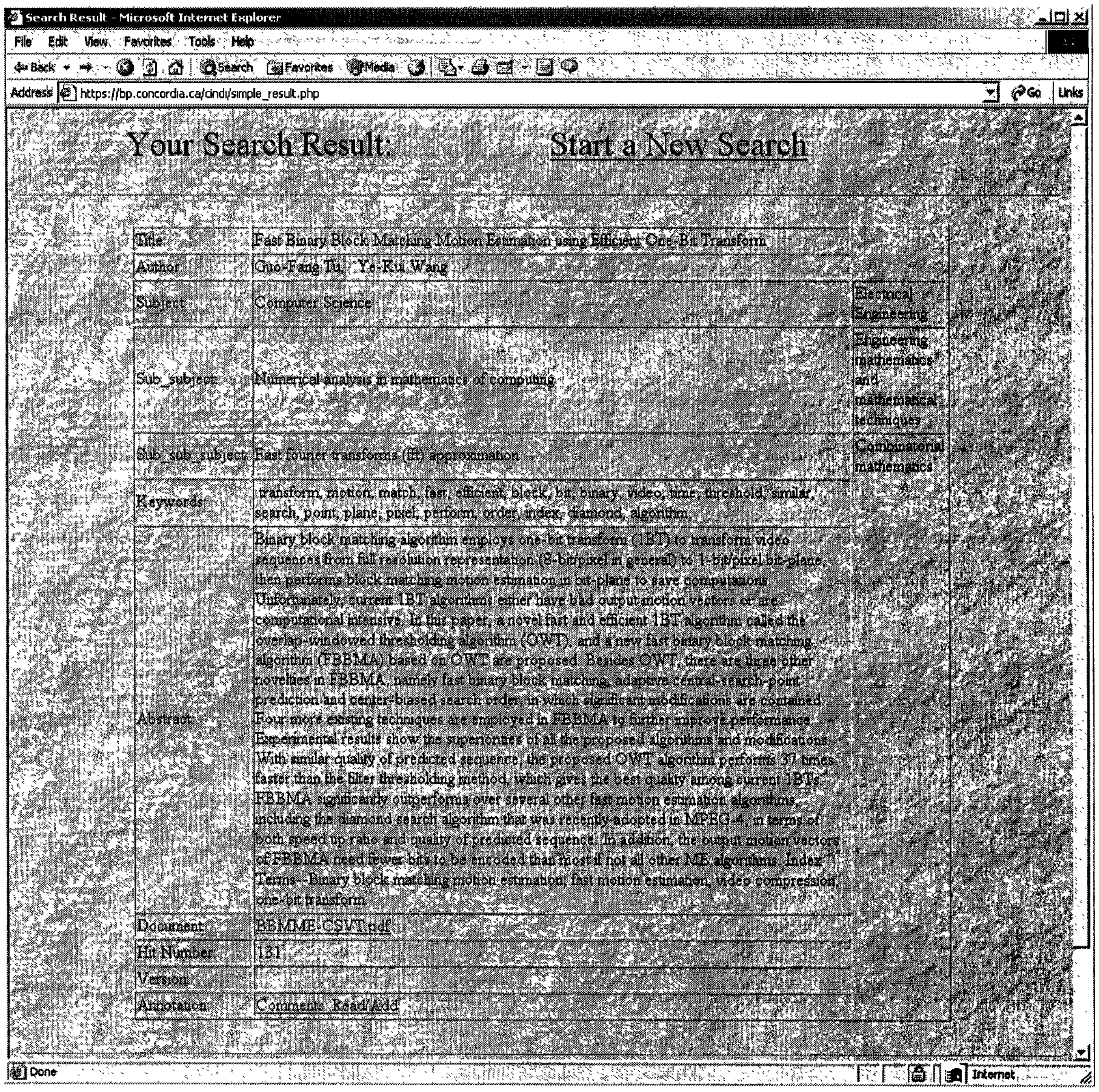
Subject:
General: Please select one of the following subjects
Sub_Subject: Please select one of the following sub-subjects
Sub_Sub_Subject: Please select one of the following sub-sub-subjects

Created_date Period: From 2003 Month Day To 2003 July 07

(To the current date by default)

Copyright© 2002-2003 by CINDI

Figure 4.5 (a) An example of resource simple search


Search Result - Microsoft Internet Explorer

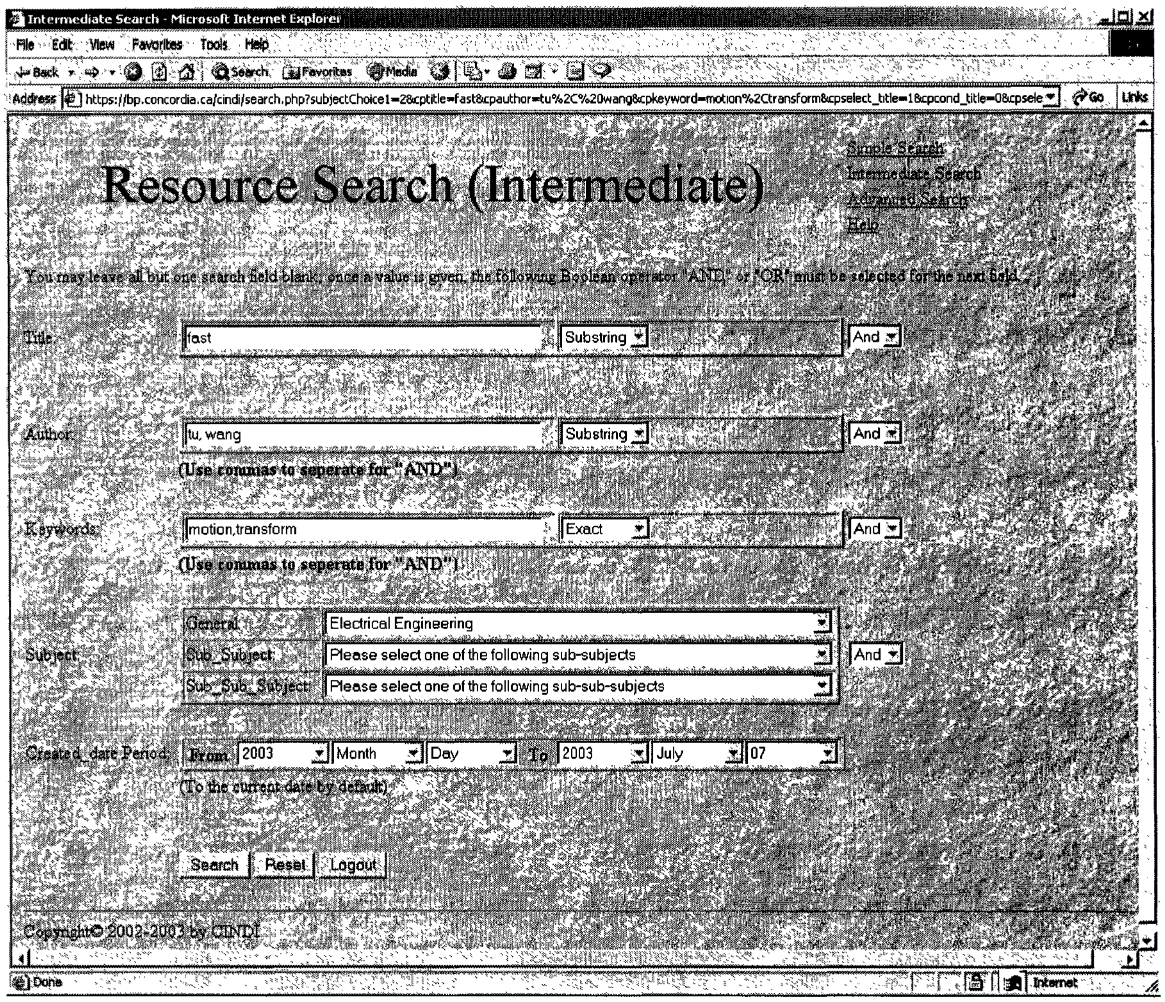
Address https://bp.concordia.ca/cindi/simple_result.php

Your Search Result: Start a New Search

Title:	Fast Binary Block Matching Motion Estimation using Efficient One-Bit Transform	
Author:	Guo-Fang Tu, Ye-Kui Wang	
Subject:	Computer Science	Electrical Engineering
Sub_subject:	Numerical analysis in mathematics of computing	Engineering mathematics and mathematical techniques
Sub_sub_subject:	Fast fourier transforms (fft) approximation	Combinatorial mathematics
Keywords:	transform, motion, match, fast, efficient, block, bit, binary, video, time, threshold, similar, search, point, plane, pixel, perform, order, index, diamond, algorithm	
Abstract:	Binary block matching algorithm employs one-bit transform (1BT) to transform video sequences from full resolution representation (8-bit/pixel in general) to 1-bit/pixel bit-plane, then performs block matching motion estimation in bit-plane to save computations. Unfortunately, current 1BT algorithms either have bad output motion vectors or are computational intensive. In this paper, a novel fast and efficient 1BT algorithm called the overlap-windowed thresholding algorithm (OWT), and a new fast binary block matching algorithm (FBBMA) based on OWT are proposed. Besides OWT, there are three other novelties in FBBMA, namely fast binary block matching, adaptive central-search-point prediction and center-biased search order, in which significant modifications are contained. Four more existing techniques are employed in FBBMA to further improve performance. Experimental results show the superiorities of all the proposed algorithms and modifications. With similar quality of predicted sequence, the proposed OWT algorithm performs 37 times faster than the filter thresholding method, which gives the best quality among current 1BTs. FBBMA significantly outperforms over several other fast motion estimation algorithms, including the diamond search algorithm that was recently adopted in MPEG-4, in terms of both speed up ratio and quality of predicted sequence. In addition, the output motion vectors of FBBMA need fewer bits to be encoded than most if not all other ME algorithms. Index Terms--Binary block matching motion estimation, fast motion estimation, video compression, one-bit transform	
Document:	BBMME-CSVT.pdf	
Hit Number:	131	
Version:		
Annotation:	Comments Read/Add	

File Edit View Favorites Tools Help
Back Search Favorites Media
Go Links
Done
Internet

Figure 4.5 (b) An example of simple search



Intermediate Search - Microsoft Internet Explorer

Address https://bp.concordia.ca/cindi/search.php?subjectChoice1=2&cptitle=fast&cpauthor=tu%2C%20wang&cpkeyword=motion%2Ctransform&cpselect_title=1&cpcond_title=0&cpsele

Resource Search (Intermediate)

Simple Search
Intermediate Search
Advanced Search
Help

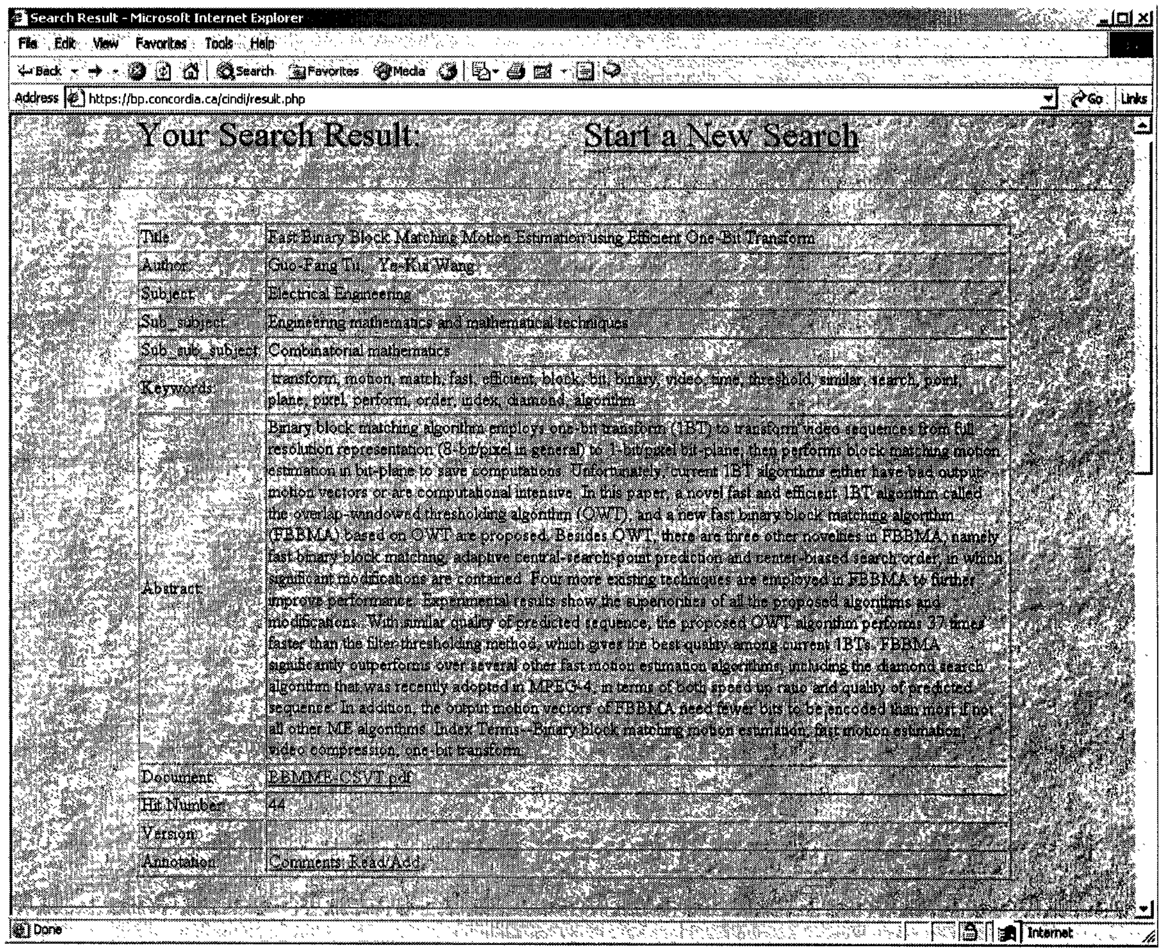
You may leave all but one search field blank, once a value is given, the following Boolean operator "AND" or "OR" must be selected for the next field.

Title	fast	Substring	And
Author	tu, wang (Use commas to seperate for "AND")	Substring	And
Keywords:	motion,transform (Use commas to seperate for "AND")	Exact	And
Subject	General: Electrical Engineering; Sub_Subject: Please select one of the following sub-subjects; Sub_Sub_Subject: Please select one of the following sub-sub-subjects		And
Created_date Period:	From 2003 Month Day To 2003 July 07 (To the current date by default)		

Search Reset Logout

Copyright© 2002-2003 by CINDI

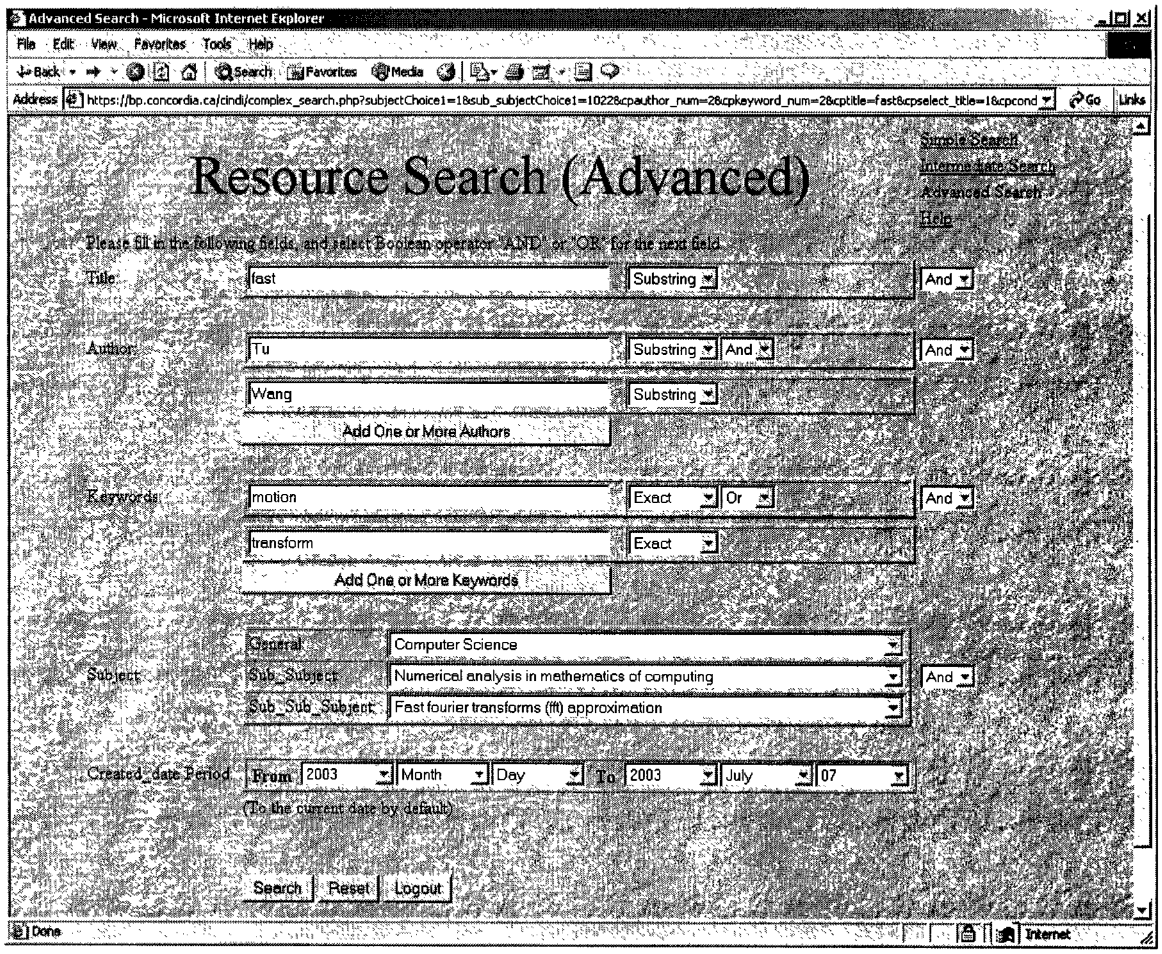
Figure 4.6 (a) An example of intermediate search


Search Result - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://bp.concordia.ca/cindi/result.php
Go Links
Done
Internet

Your Search Result: Start a New Search

Title:	Fast Binary Block Matching Motion Estimation using Efficient One-Bit Transform
Author:	Guo-Fang Tu, Ye-Kui Wang
Subject:	Electrical Engineering
Sub_subject:	Engineering mathematics and mathematical techniques
Sub_sub_subject:	Combinatorial mathematics
Keywords:	transform, motion, match, fast, efficient, block, bit, binary, video, time, threshold, similar, search, point, plane, pixel, perform, order, index, diamond, algorithm
Abstract:	Binary block matching algorithm employs one-bit transform (1BT) to transform video sequences from full resolution representation (8-bit/pixel in general) to 1-bit/pixel bit-plane, then performs block matching motion estimation in bit-plane to save computations. Unfortunately, current 1BT algorithms either have bad output motion vectors or are computational intensive. In this paper, a novel fast and efficient 1BT algorithm called the overlap-windowed thresholding algorithm (OWT), and a new fast binary block matching algorithm (FBBMA) based on OWT are proposed. Besides OWT, there are three other novelties in FBBMA, namely fast binary block matching, adaptive central-search-point prediction and center-biased search order, in which significant modifications are contained. Four more existing techniques are employed in FBBMA to further improve performance. Experimental results show the superiorities of all the proposed algorithms and modifications. With similar quality of predicted sequence, the proposed OWT algorithm performs 37 times faster than the filter-thresholding method, which gives the best quality among current 1BTs. FBBMA significantly outperforms over several other fast motion estimation algorithms, including the diamond search algorithm that was recently adopted in MPEG-4, in terms of both speed up ratio and quality of predicted sequence. In addition, the output motion vectors of FBBMA need fewer bits to be encoded than most if not all other ME algorithms. Index Terms--Binary block matching motion estimation, fast motion estimation, video compression, one-bit transform.
Document:	BBMME-CSVT.pdf
Hit Number:	44
Version:	
Annotation:	Comments: Read/Add

Figure 4.6 (b) An example of intermediate search


Advanced Search - Microsoft Internet Explorer

Address https://bp.concordia.ca/cindi/complex_search.php?subjectChoice1=1&sub_subjectChoice1=1022&cpauthor_num=2&cpkeyword_num=2&cptitle=fast&cpselect_title=1&cpcond

Resource Search (Advanced)

Simple Search
Intermediate Search
Advanced Search
Help

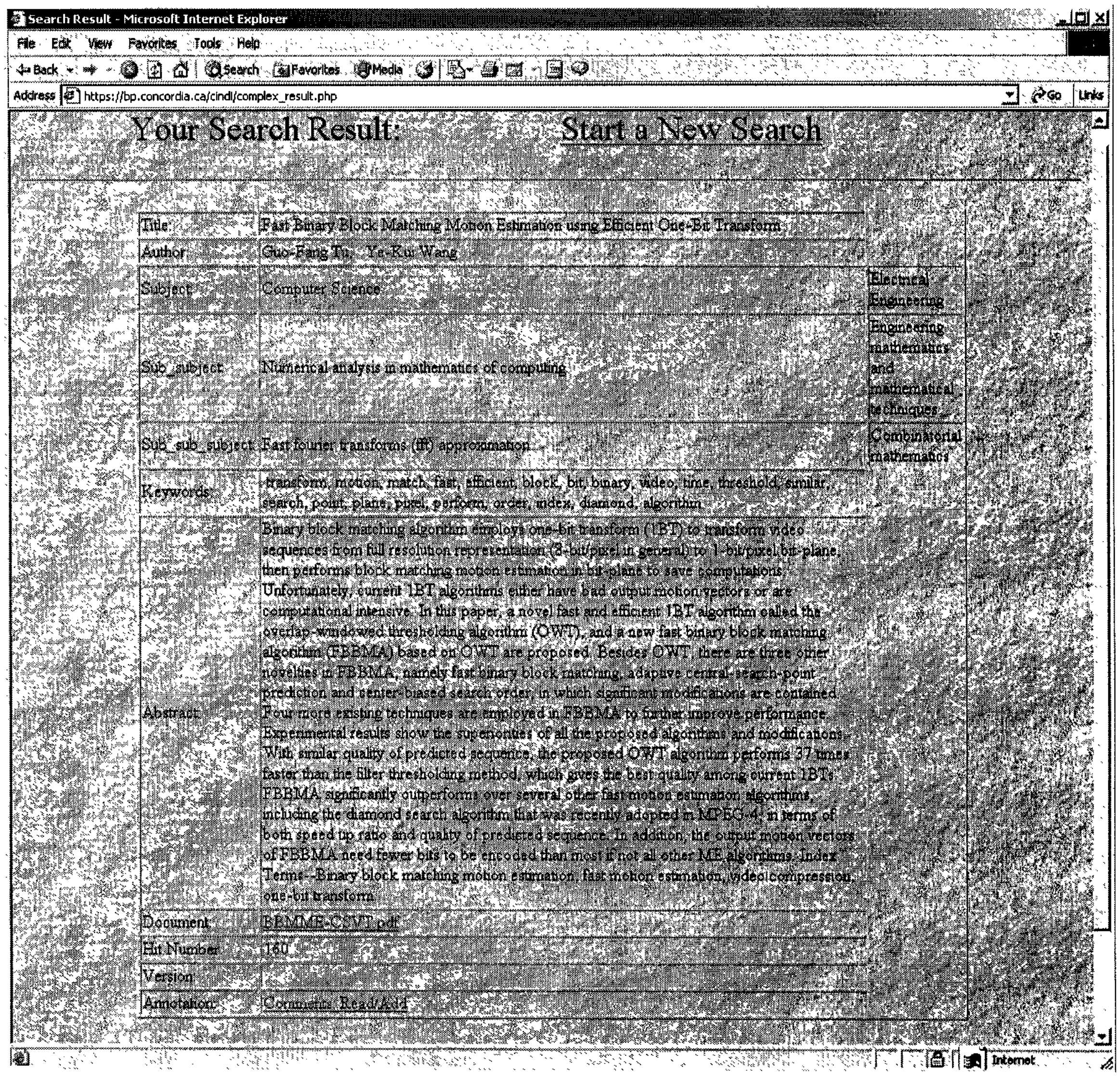
Please fill in the following fields, and select Boolean operator "AND" or "OR" for the next field.

Title:	fast	Substring		And
Author:	Tu	Substring	And	And
	Wang	Substring		
Keywords:	motion	Exact	Or	And
	transform	Exact		

Subject:	General	Computer Science	
	Sub_Subject	Numerical analysis in mathematics of computing	And
	Sub_Sub_Subject	Fast fourier transforms (fft) approximation	

Created_date Period: From 2003 Month Day To 2003 July 07
(To the current date by default)

Figure 4.7 (a) An example of advanced search



Search Result - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://bp.concordia.ca/cindi/complex_result.php
Go Links
Internet

Your Search Result: Start a New Search

Title:	Fast Binary Block Matching Motion Estimation using Efficient One-Bit Transform	
Author:	Guo-Fang Tu; Ye-Kui Wang	
Subject:	Computer Science	Electrical Engineering
Sub_subject:	Numerical analysis in mathematics of computing	Engineering mathematics and mathematical techniques
Sub_sub_subject:	Fast fourier transforms (fft) approximation	Combinatorial mathematics
Keywords:	transform, motion, match, fast, efficient, block, bit, binary, video, time, threshold, similar, search, point, plane, pixel, perform, order, index, diamond, algorithm	
Abstract:	Binary block matching algorithm employs one-bit transform (1BT) to transform video sequences from full resolution representation (8-bit/pixel in general) to 1-bit/pixel bit-plane, then performs block matching motion estimation in bit-plane to save computations. Unfortunately, current 1BT algorithms either have bad output motion vectors or are computational intensive. In this paper, a novel fast and efficient 1BT algorithm called the overlap-windowed thresholding algorithm (OWT), and a new fast binary block matching algorithm (FBBMA) based on OWT are proposed. Besides OWT, there are three other novelties in FBBMA, namely fast binary block matching, adaptive central-search-point prediction and center-biased search order, in which significant modifications are contained. Four more existing techniques are employed in FBBMA to further improve performance. Experimental results show the superiorities of all the proposed algorithms and modifications. With similar quality of predicted sequence, the proposed OWT algorithm performs 37 times faster than the filter thresholding method, which gives the best quality among current 1BTs. FBBMA significantly outperforms over several other fast motion estimation algorithms, including the diamond search algorithm that was recently adopted in MPEG-4, in terms of both speed up ratio and quality of predicted sequence. In addition, the output motion vectors of FBBMA need fewer bits to be encoded than most if not all other ME algorithms. Index Terms--Binary block matching motion estimation, fast motion estimation, video compression, one-bit transform	
Document:	BBMME-CSVT.pdf	
Hit Number:	160	
Version:		
Annotation:	Comments: Read/Add	

Figure 4.7 (b) An example of advanced search

Solution 1: use logical condition to process the data

```
IF (author != NULL)
   count the number of commas
IF (the number of commas > 1)
{
      Token commas to separate the authors' name and keep them
into an array author[]
      FOR (i = 0; i < the number of commas; i++)
      {     query to get the distinct IDs of resource by author[i]
            keep them into an array resource resource_id[]
```

```
        }
}
FOR (i = 0; i < the number of commas; i++)
            FOR (j = the number of commas - 1; j > i; j--)
         {  IF (resource_id[i] is equal to resource_id[j])
                  Record it into an array resourceID[i]
         }
         IF (array resourceID is empty)
            Message "No resource meets the search condition"
        ELSE
            Query and output the search results
```

Solution 2: create a temporary table for the join of resource and author.

```
CREATE TEMPORARY TABLE IF NOT EXISTS TEMPresource_authorname (

resource_id int (10) unsigned not null,

author_name varchar(255) not null,

primary key (resource_id));
```

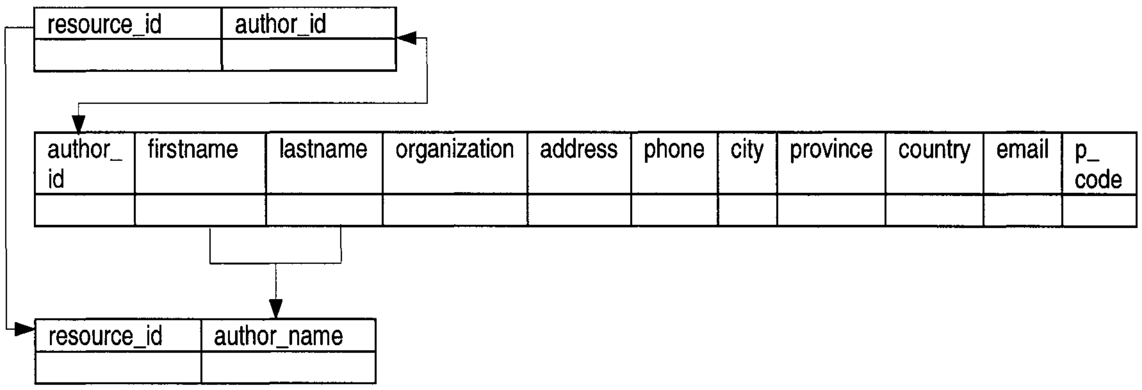


Figure 4.8 The logical view of temporary table `TEMPresource_authorname`

The following are instances of creating a temporary table `TEMPresource_authorname` for the join of the table `resource` and the table `author`. Details of temporary tables in MySQL will be discussed in Section 4.5.

Table 6. An instance of the table `resource_author`

resource_id	author_id
2195	570
2195	571

Table 7. An instance of the table `author`

author _id	first name	last name	organi_ zation	add_ ress	phone	city	province	country	e_ mail	p_code
570	John	Smith	University	N/A	NULL	NULL	QC	Canada	John @u.edu	H3H 1M8
571	Jean	Chin	University	N/A	NULL	NULL	QC	Canada	Jean @u.edu	H3H 1M8

Table 8. The join result of the above tables: the table `TEMPresource_authorname`

resource_id	author_name
2195	John Smith, Jean Chin

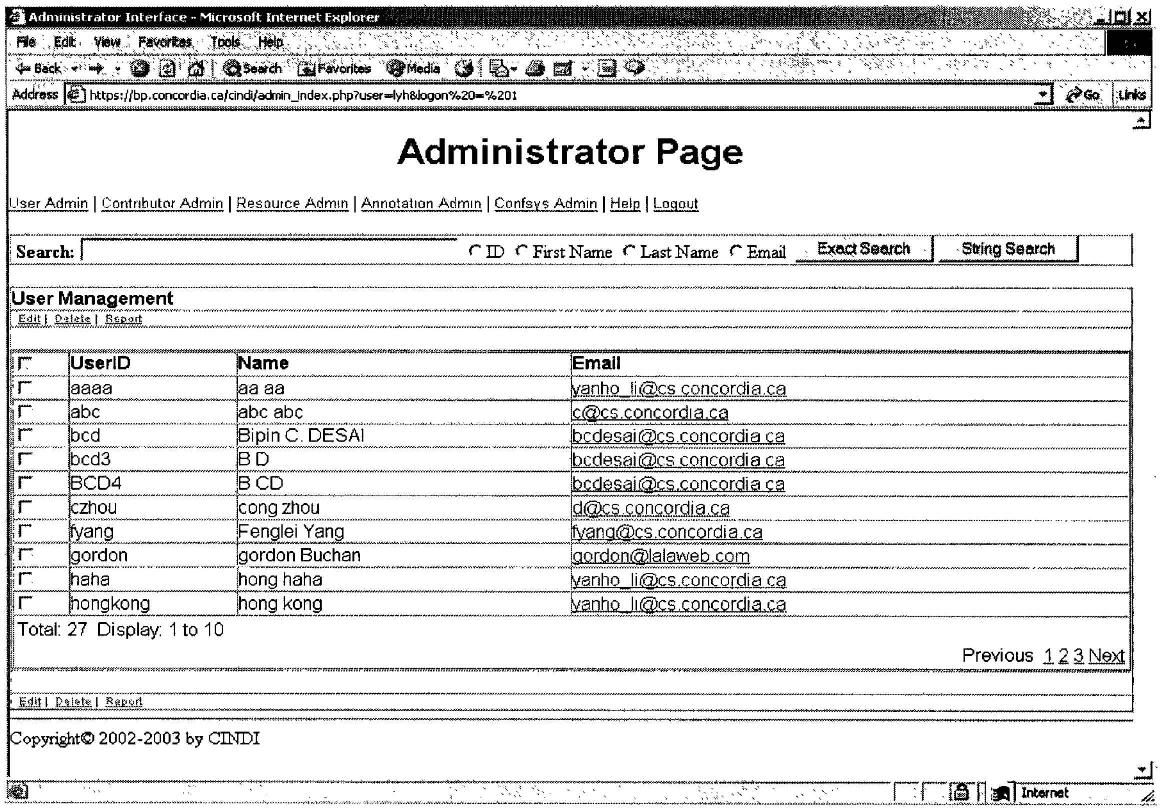
4.2.2. Search function for administrators

The administration search subsystem of CINDI allows the system administrators to enter a value of a specified field, which varies from interface to interface. For users' and contributors' information management as shown in Figure 4.9, the administrators can look for the related information based on the values of `user ID`, `first name`, `last name` and `email`. For the bibliographic system administration including resource management and annotation management, some key fields are provided by the system to let the administrators search the information based on their values (Figure 4.10).

Based on the different search criteria, the system provides the following types of search for the administrators:

- **Exact Search**, which takes the administrators' inputs as the precise condition and makes use of the equivalent operator "=" in the WHERE clause of SQL. For example,

```
SELECT * FROM table WHERE field = 'input';
```


Administrator Interface - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://bp.concordia.ca/cindi/admin_index.php?user=lyh&logon%20=%201
Go Links

Administrator Page

User Admin | Contributor Admin | Resource Admin | Annotation Admin | Confsys Admin | Help | Logout

Search: ID First Name Last Name Email Exact Search String Search

User Management

Edit | Delete | Report

	UserID	Name	Email
☐	aaaa	aa aa	yanho_li@cs.concordia.ca
☐	abc	abc abc	c@cs.concordia.ca
☐	bcd	Bipin C. DESAI	bcdesai@cs.concordia.ca
☐	bcd3	B D	bcdesai@cs.concordia.ca
☐	BCD4	B CD	bcdesai@cs.concordia.ca
☐	czhou	cong zhou	d@cs.concordia.ca
☐	fyang	Fenglei Yang	fyang@cs.concordia.ca
☐	gordon	gordon Buchan	gordon@lalaweb.com
☐	haha	hong haha	yanho_li@cs.concordia.ca
☐	hongkong	hong kong	yanho_li@cs.concordia.ca

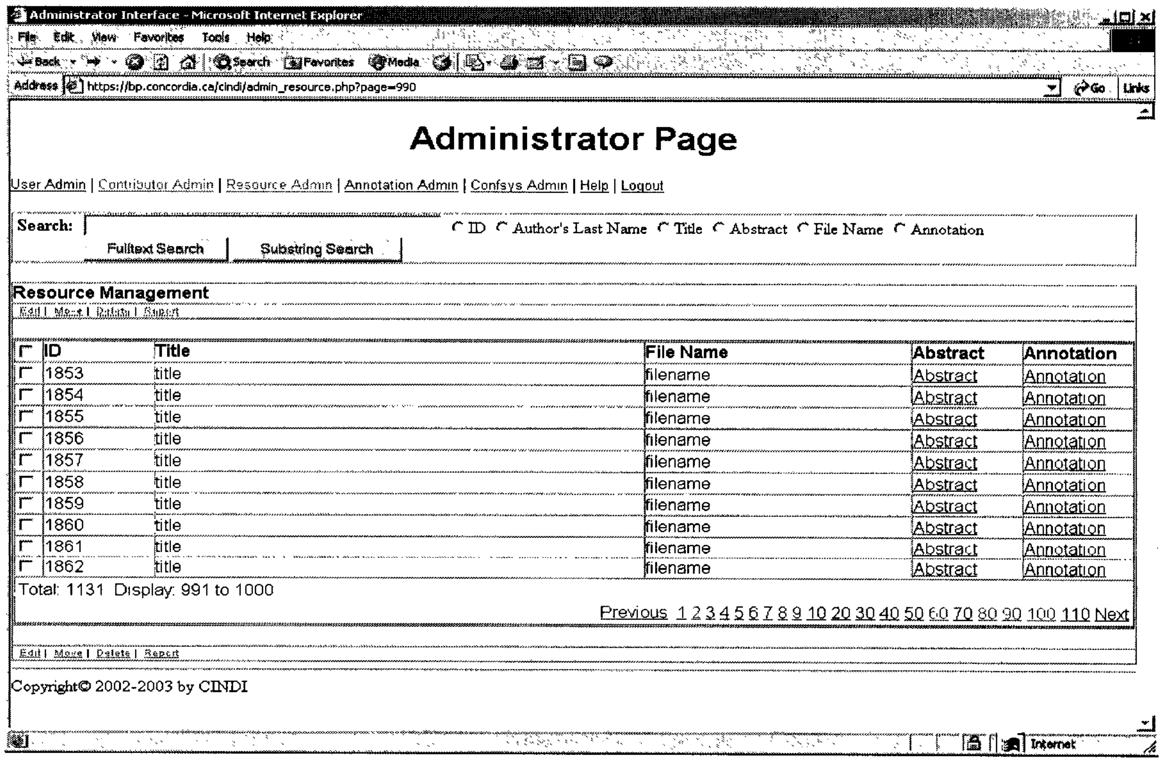
Total: 27 Display: 1 to 10

Previous 1 2 3 Next

Edit | Delete | Report

Copyright© 2002-2003 by CINDI

Figure 4.9 The interface of user management


Administrator Interface - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://bp.concordia.ca/cindi/admin_resource.php?page=990
Go Links

Administrator Page

User Admin | Contributor Admin | Resource Admin | Annotation Admin | Confsys Admin | Help | Logout

Search: ID Author's Last Name Title Abstract File Name Annotation

Fulltext Search Substring Search

Resource Management

Edit | Move | Delete | Report

	ID	Title	File Name	Abstract	Annotation
☐	1853	title	filename	Abstract	Annotation
☐	1854	title	filename	Abstract	Annotation
☐	1855	title	filename	Abstract	Annotation
☐	1856	title	filename	Abstract	Annotation
☐	1857	title	filename	Abstract	Annotation
☐	1858	title	filename	Abstract	Annotation
☐	1859	title	filename	Abstract	Annotation
☐	1860	title	filename	Abstract	Annotation
☐	1861	title	filename	Abstract	Annotation
☐	1862	title	filename	Abstract	Annotation

Total: 1131 Display: 991 to 1000

Previous 1 2 3 4 5 6 7 8 9 10 20 30 40 50 60 70 80 90 100 110 Next

Edit | Move | Delete | Report

Copyright© 2002-2003 by CINDI

Figure 4.10 The interface of resource management

- **Substring Search**, which considers the search items input by the administrators as part of the value of the specified field that is used for the queries. Here, with the WHERE clause of SQL, the match operator `LIKE` is used as follows:

```
SELECT * FROM table WHERE field LIKE '%input%';
```

- **Fuzzy Search**, which supports the search for the text fields of title, abstract and annotation, will be described in the following section.

4.2.3. Fuzzy search

It is well known that people easily mistype a word due to human "optical character recognition" (OCR) errors. Fuzzy search, therefore, is provided by the administration sub-system of the CINDI to help the administrators find documents even if the words entered are mistyped or misspelled. For example, a document containing the phrase "United states of America" could be found even though "United" might have been OCR'ed as "unlted", "Vnlted", or "Vnited", and "America" might have been entered as "america" or "Amereca". Here, GNU Aspell, a Free and Open Source spell checker designed to eventually replaced Ispell [ATKINSON02], is applied in the CINDI system as a spell checker because it does a much better job of coming up with possible suggestions than just about any other spell checker for the English language, including Ispell and Microsoft Word. [ATKINSON02] Furthermore, it has many technical enhancements over Ispell, such as intelligently handling personal dictionaries when more than one Aspell process is opened at once. Users can customize the personal dictionaries based on their requirements. For instance, some of words in database such as MySQL and Informix can be built into the personal dictionaries while being checked by Aspell so that these words will not be treated as OCR errors by the Aspell spell checker.

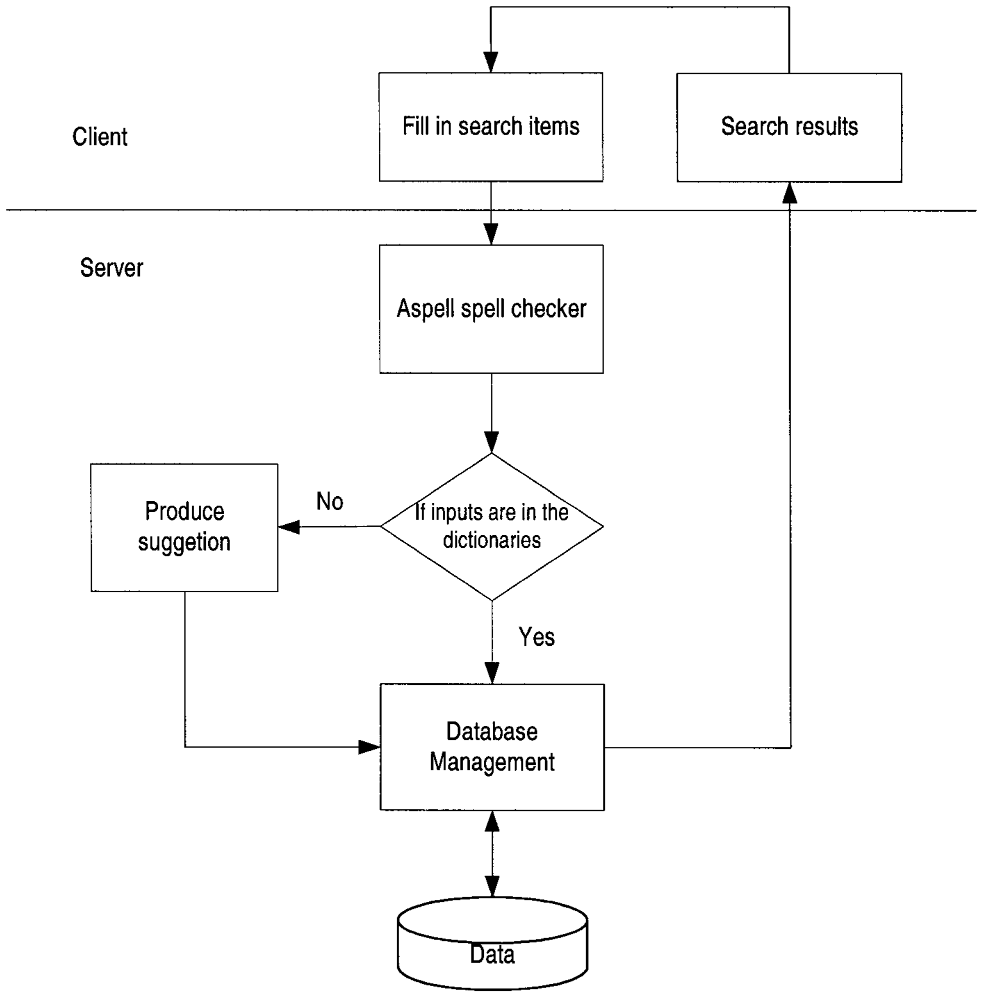
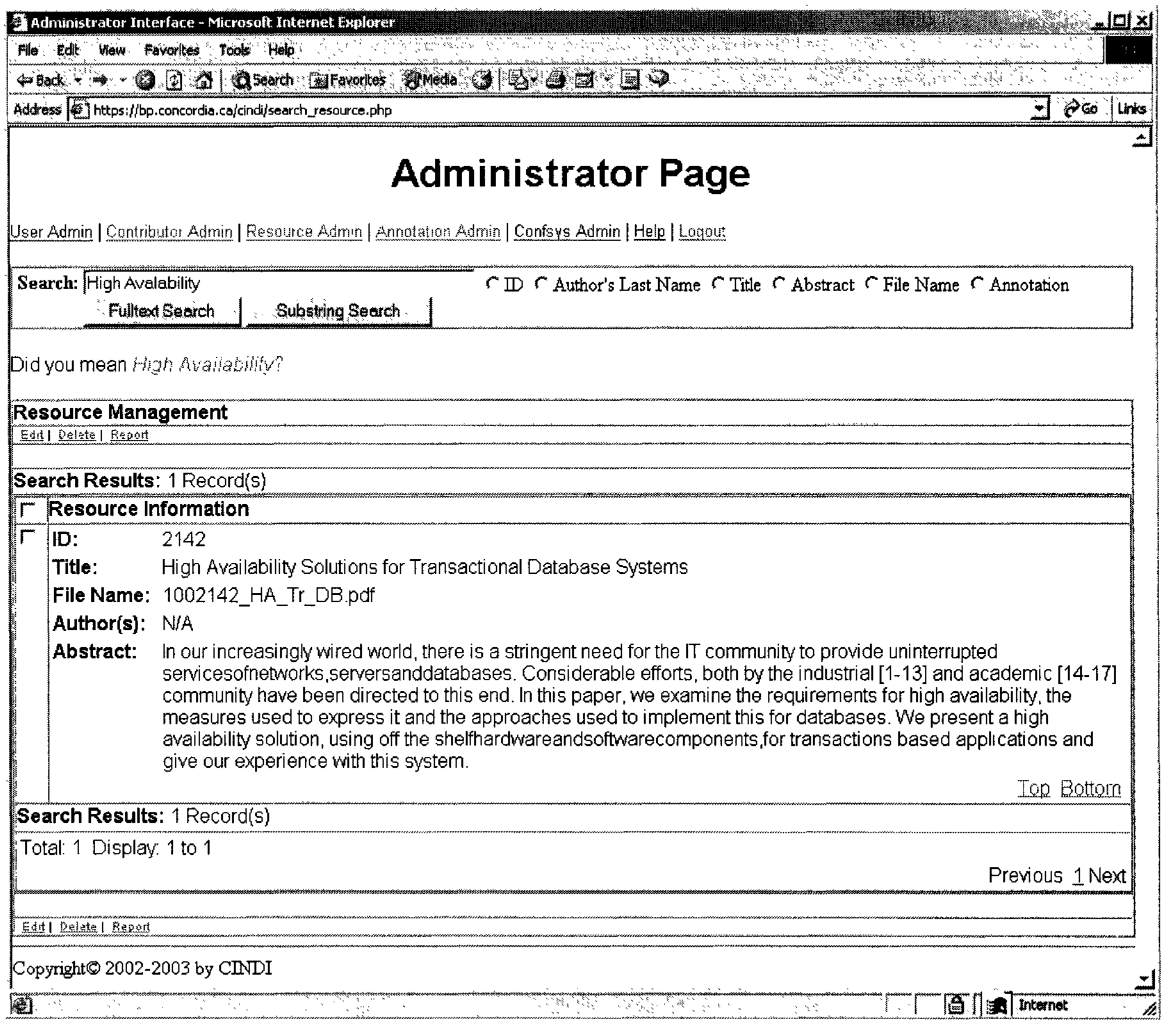


Figure 4.11 The workflow of the fuzzy search

Figure 4.11 shows the system workflow of the fuzzy search. For example, the search item "High Availability" for substring of the title field is OCR'ed as "high availibiliy" or "High Avalability". By providing the suggestion from the Aspell spell checker, the search result will come out based on the closest suggestion as Figure 4.12.


Administrator Interface - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address https://bp.concordia.ca/cindi/search_resource.php

Administrator Page

User Admin | Contributor Admin | Resource Admin | Annotation Admin | Confsys Admin | Help | Logout

Search: High Avalability ◯ ID ◯ Author's Last Name ◯ Title ◯ Abstract ◯ File Name ◯ Annotation

Fulltext Search | Substring Search

Did you mean *High Availability*?

Resource Management

Edit | Delete | Report

Search Results: 1 Record(s)

Resource Information

ID:	2142
Title:	High Availability Solutions for Transactional Database Systems
File Name:	1002142_HA_Tr_DB.pdf
Author(s):	N/A
Abstract:	In our increasingly wired world, there is a stringent need for the IT community to provide uninterrupted servicesofnetworks,serversanddatabases. Considerable efforts, both by the industrial [1-13] and academic [14-17] community have been directed to this end. In this paper, we examine the requirements for high availability, the measures used to express it and the approaches used to implement this for databases. We present a high availability solution, using off the shelfhardwareandsoftwarecomponents,for transactions based applications and give our experience with this system.

Top Bottom

Search Results: 1 Record(s)

Total: 1 Display: 1 to 1

Previous 1 Next

Edit | Delete | Report

Copyright© 2002-2003 by CINDI

Figure 4.12 An example of the fuzzy search

4.2.4. Application of indexes

Computing joins with fast response times is extremely difficult to achieve for very large relations. One approach to this problem is to create an index that is designed to speed up specific join queries. An index is an auxiliary structure designed to speed up operations on search. All indexes can be viewed as a collection of data entries with an efficient way to locate all data entries with a given search value. [RG00] Indexing terms consist of the following types:

- **Single-column index:** which considers the values in just one column.

- **Multiple-column index:** which considers the values in more than one column.
- **The leftmost prefix:** which is the leftmost set of columns acting as an index.

The first two are self-explanatory; the third one is more complex. A multiple-column index has the potential to serve as several prefixes because any contiguous set of columns, starting with the leftmost column in the index can be used to match values. For instance, assuming a three-column index based on the following columns of table resource: `title, keyword`, and `abstract`. Besides the original three-column index, the two more potential candidates for indexes are:

- `title, keyword`
- `title`

There are two basic rules, given below when defining a prefix:

- The index must include the leftmost column; therefore, there is no index on:

  `keyword`

  `abstract`

  `keyword, abstract`
- The index can not skip columns; hence, there is no index on:

  `title, abstract`

As a result, it is important to use the leftmost prefix if possible when a WHERE clause extends across a multiple-column index. Otherwise, the query optimizer will not make use of an index at all; thus, performance will suffer.

In order to improve the performance, single-column index and multiple-column index are considered in the CINDI system. In order to accelerate the join in resource searching, simple-column index on fields of `resource_id, author_id, subject_id, user_id`, and `language_id`, are used for joining between the corresponding tables, such as `resource`, `author`, `subject`, `users`, and `language`. Moreover, multiple-column index on fields, such as `lastname` and `firstname` in the author table, is created to facilitate the query based on author name criteria.

As of version 3.23.23, MySQL supports full-text index, a built in functionality that allows users to search through certain tables for a string matching. Substring search as mentioned above, using the `LIKE` operator in the WHERE clause of SQL statements, works slowly and inefficiently when searching text fields; especially as the database grows the system will become downright ineffective. The solution to this problem is full-text searching. According to the MySQL document [MYSQL03], full-text search is a "natural language search"; it indexes words that appear to represent a row, made up of the specified columns. `MATCH() AGAINST()` is used as the condition of the WHERE clause of SQL statement as illustrated below:

```
SELECT *
FROM  table
WHERE MATCH(column1, column2) AGAINST ('text');
```

In general, when most people use a search box, they type in a number of words they consider as likely keywords rather than only one word. To deal with it, MySQL takes all the words, splits them up, and then matches by using a natural language search. As of version 4.0.1, MySQL can also perform Boolean full-text searches using the IN BOOLEAN MODE modifier. Since the CINDI system utilizes Version 3.23.56 that provides the simple full-text searching, the Boolean mode will not be discussed here but hopefully will be applied in the future.

It is important to note that there are several limitations on full-text searching as follows:

- The data set should be large enough; otherwise, the search accuracy will be lower because of the 50% threshold. For example, if there is only one record in the table, no matter what is being searched for, it is in 50% or more of the row in the table, thus disregarded.
- The words of less than three characters will be omitted since MySQL views them as the noise words.
- Full-text index only supports the table type of `MyISAM`; the other types of table, hopefully, will be implemented in the near future [MYSQL03].

Since the data stored in the CINDI system is large, the accuracy caused by a small data set is not a problem. In order to cope with the limitation on the table type, some revisions need to be made on the tables. In the original database of the CINDI system, the data type for the field of abstract in the relation resource was defined as `BLOB`, which are meant primarily for binary data. Because MySQL does not support indexing `BLOB` data types for full-text searching, the definition must be changed by switching data type from `BLOB` to `TEXT` as follows so as to make the abstract column useful for searching:

```
ALTER TABLE resource MODIFY abstract TEXT;
```

Since the table type of resource is defined as `INNODB` adhering to the properties of ACID, it is impossible to build a full-text index for the table `resource` due to the limitation. There are two solutions to be considered according to the system architecture:

- For the system configuration with only one database server as well as the one with master and slave, the solution is to create a temporary table for resources with the type `MyISAM` that supports full-text searching. To follow the ACID properties, all write queries will be run against the table `resource` of type `INNODB`, while search queries will be run against the temporary table. For the system configuration with master and slave, updates will be copied from the master to the slave as a backup through binary log files to keep consistent.

- The other solution is designed for the distributed database system where queries will be distributed into different servers based on the privilege of these queries. That is, all write queries, such as inserting data and deleting data, will visit the production server; all read queries will be forced to access the backup server on which the table type will be transformed into `MyISAM` for searching because only `MyISAM` tables support full-text indexes as mentioned before. In this case, it requires that both servers are active. Meanwhile, the middle tier, being workflow controller, will be responsible for switching the queries with PHP scripts based on the privileges granted for users. For example, a contributor of the CINDI system can access the production server while a user who is searching for some resource is able to approach the backup server.

So far, the first solution is adopted by the CINDI in the thesis due to its architecture. With the development of the CINDI system, the second one has been considered for the future.

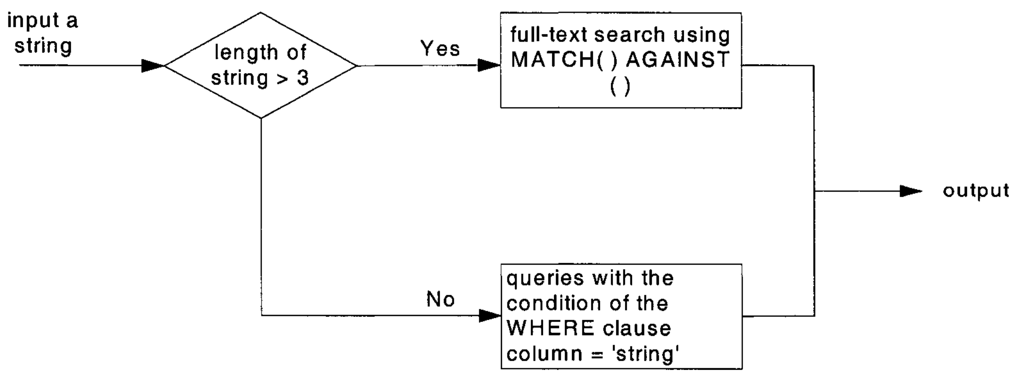

input a string
length of string > 3
Yes
full-text search using MATCH( ) AGAINST ( )
No
queries with the condition of the WHERE clause column = 'string'
output

Figure 4.13 The logical flowchart for Full-text searching

To address the limitation on the length of a string, checking the length of a string is taken into account before processing queries as shown in Figure 4.13.

4.3. Backup and retrieve the CONFSYS system

As mentioned previously, the CONFSYS is a sub-system of the CINDI system. That is, all the information collected in the CONFSYS will be automatically merged into the bibliographic system of the CINDI in order to provide the latest information for users. Another benefit is that a remote backup is built for a specified conference in case the local server fails.

In the original version [GU02] [JIN03] of the CONFSYS, the workflow of uploading a paper to a conference is depicted below:

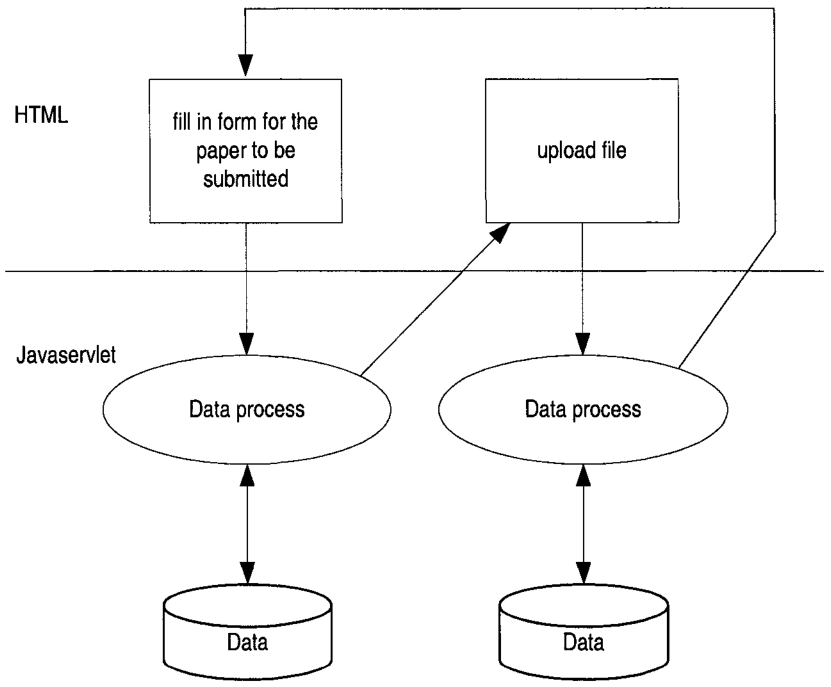


Figure 4.14 The workflow of uploading a paper in original version

In the thesis, we introduced the ASHG into the CONFSYS and improved the functional performance for authors, such as re-upload file and check status. In addition, in case of the failure of the CONFSYS server, its backup was built in this project. Moreover, being a subsystem of the CINDI, the CONFSYS was integrated into the CINDI system, whose workflow is shown in Figure 4.15 below. In the CINDI system, the submitted papers will be processed by the semantic header generator for the CONFSYS and the bibliographic system of the CINDI. It makes the CONFSYS more convenient for authors because they will just need to check the corresponding information and make required correction rather than to fill in an entire form for the paper to be submitted. Meanwhile, the information regarding the uploaded paper and its authors are written into the local server and the remote server concurrently. The local server is the system at the site of the organizers of the conference; the remote server is the CINDI server.

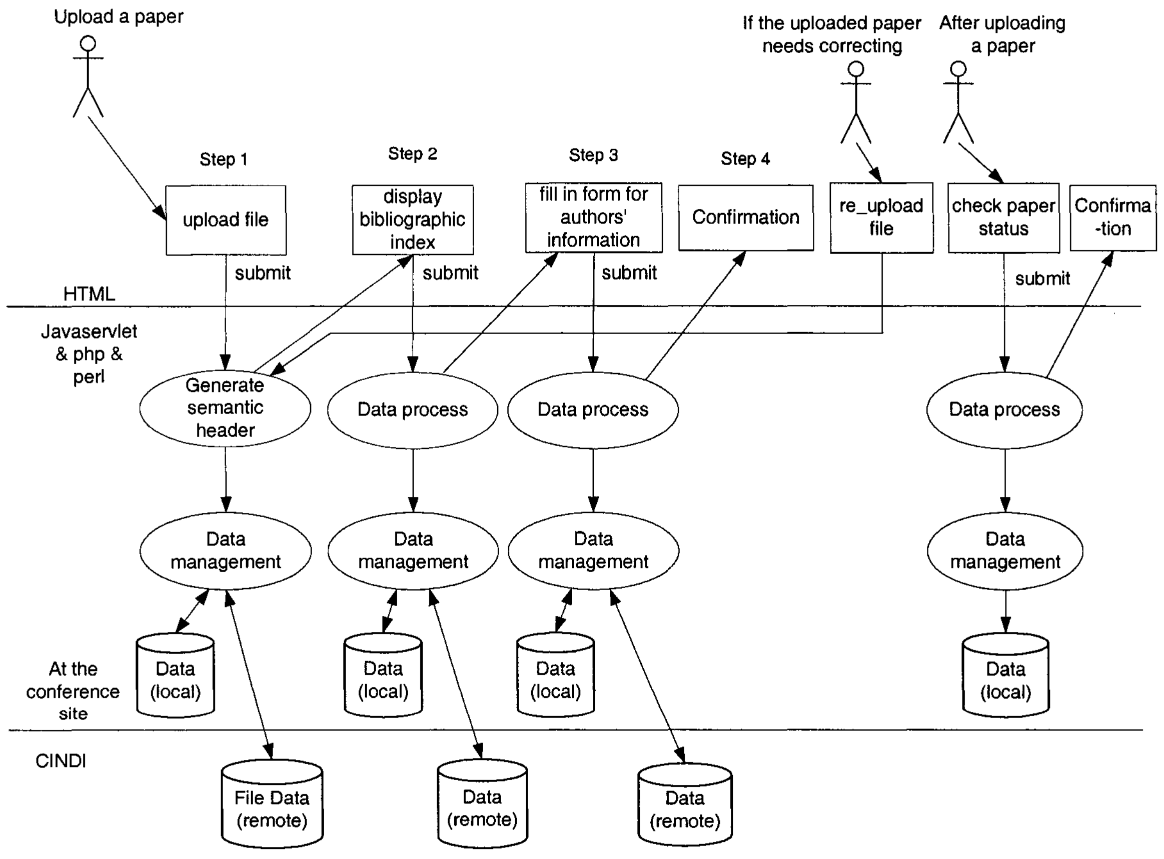


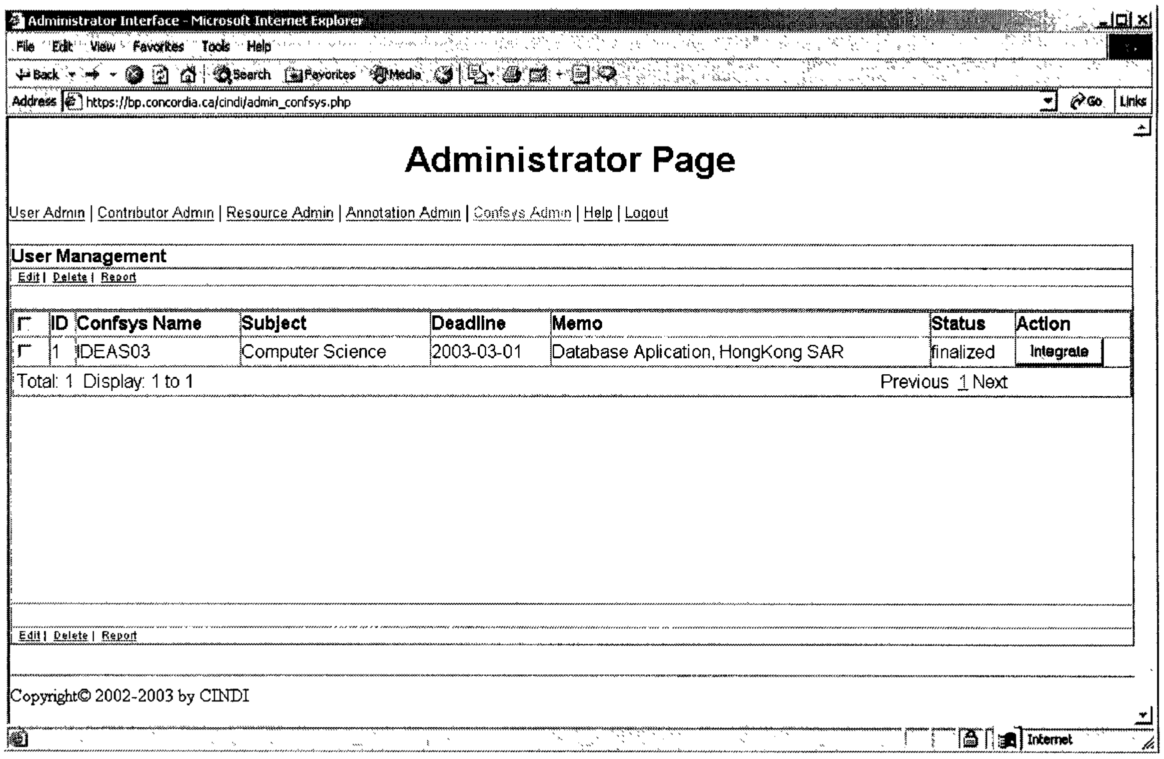
Figure 4.15 Improvement version of the CONFSYS with distributed structure

For each conference to be integrated into the CINDI system, a corresponding backup database will be set up on the remote CINDI server in advance. A new user account will be created for the conference to access the remote CINDI server for transferring the uploaded files as well. Using the user's account and password, a pair of encrypted public key and private key will be generated so as to access the remote server without presenting password. These keys are useful for programming the file-based backup between servers because no passwords are required while the pair of encrypted keys guarantees the transmission security.

In order to ensure users of the CONFSYS to connect to both the local server and the remote CINDI server in parallel, the function `mysql_connection()` provided by PHP helps to create the links between the two servers at the same time. With the links, each query run against the two servers will be processed by the database management systems on both servers concurrently. As for the file-based backup, `scp` provided by the secure shell is used.

4.4. Solution for integrating the CONFSYS with the CINDI

Being a backup for a conference on the remote server, different backup databases must be created for each conference. As a result, these databases must be merged into the CINDI database system after each conference program is finalized, as shown Figure 4.16 (a). This work will be performed by the system administrators when the status of a special conference is "finalized". After the integration shown in Figure 4.16 (b) is done, this status will be changed as "integrated" shown in red font color (see Figure 4.16 (c)).


Administrator Interface - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address https://bp.concordia.ca/cindi/admin_confsys.php

Administrator Page

User Admin | Contributor Admin | Resource Admin | Annotation Admin | Confsys Admin | Help | Logout

User Management

Edit | Delete | Report

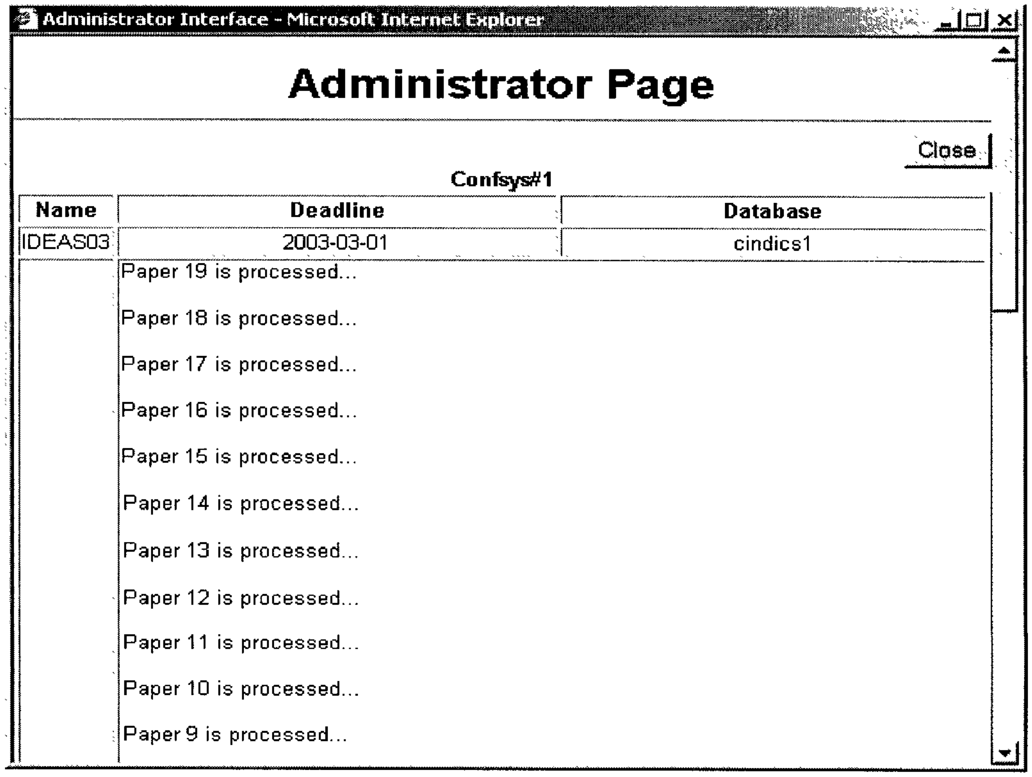
	ID	Confsys Name	Subject	Deadline	Memo	Status	Action
☐	1	IDEAS03	Computer Science	2003-03-01	Database Aplication, HongKong SAR	finalized	Integrate

Total: 1 Display: 1 to 1 Previous 1 Next

Edit | Delete | Report

Copyright© 2002-2003 by CINDI

Figure 4.16 (a) Interface for integration of the CONFSYS with the CINDI



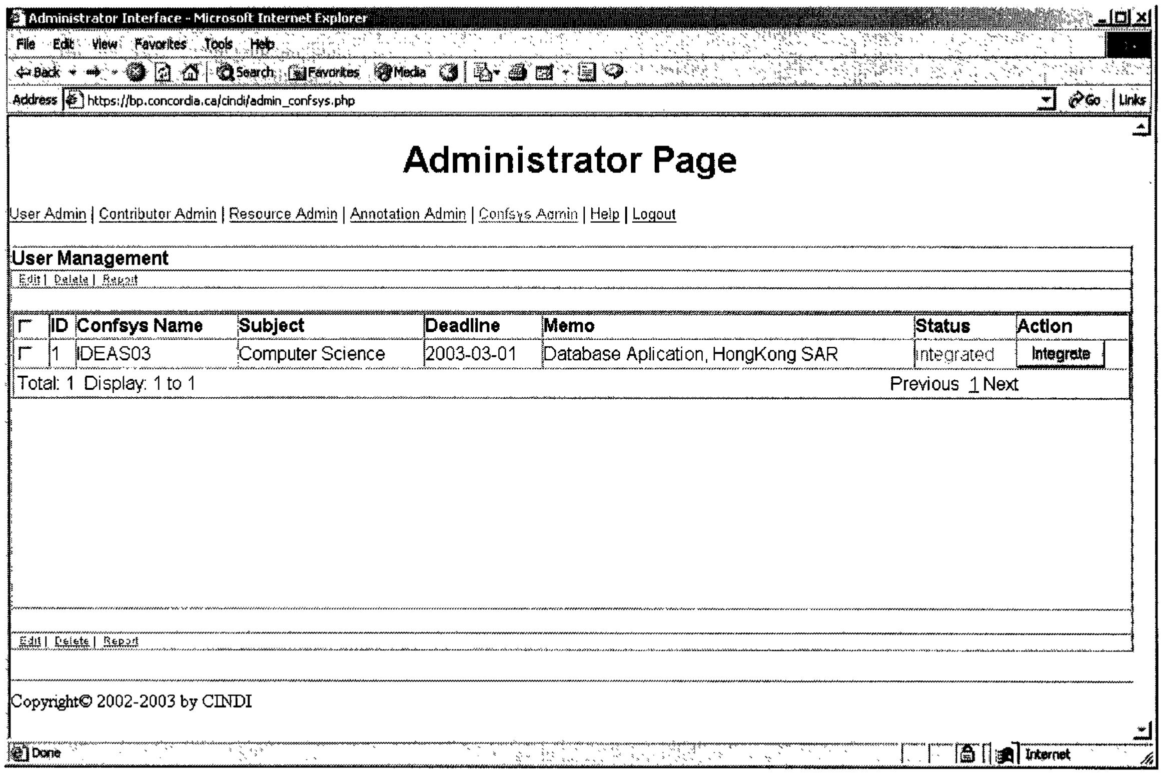
Administrator Page

Confsys#1

Name	Deadline	Database
IDEAS03	2003-03-01	cindics1

```
Paper 19 is processed...
Paper 18 is processed...
Paper 17 is processed...
Paper 16 is processed...
Paper 15 is processed...
Paper 14 is processed...
Paper 13 is processed...
Paper 12 is processed...
Paper 11 is processed...
Paper 10 is processed...
Paper 9 is processed...
```

Figure 4.16 (b) Interface for integration of the CONFSYS with the CINDI


Administrator Interface - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://bp.concordia.ca/cindi/admin_confsys.php
Go Links

Administrator Page

User Admin | Contributor Admin | Resource Admin | Annotation Admin | Confsys Admin | Help | Logout

User Management

Edit | Delete | Report

	ID	Confsys Name	Subject	Deadline	Memo	Status	Action
	1	IDEAS03	Computer Science	2003-03-01	Database Aplication, HongKong SAR	integrated	Integrate

Total: 1 Display: 1 to 1 Previous 1 Next

Edit | Delete | Report

Copyright© 2002-2003 by CINDI

Figure 4.16 (c) Interface for integration of the CONFSYS with the CINDI

In order to handle concurrency of multiple conferences, a unique identifier is required by the bibliographic system. Since the same identifiers for different conferences would be used if they were opened at the same time and it would be difficult to manage the data. In order to trace the linkage between an uploaded paper and its bibliographic index, a table called `resource_paper` is created to represent this relationship for each conference in its backup database. Based on the relationship, all the submitted papers will be processed one by one (Figure 4.16 (b)) as if they were contributed by the contributors who are the authors of the papers, although the integration is batched from the point of the view of users. The authors will be automatically registered as contributors, and then they will receive an email from the CINDI system to notify them of the semantic header registration information by accessing the CINDI system.

Since the comments on the uploaded papers given by their authors are useful, in this project we give an opportunity to authors to make comments on their works when they upload their papers. To record the annotations by authors, a relation should be created to store them for users. However, the table `annotation` is created to record the comments made by readers. To keep the ACID properties, this table is referencing the table `users` by using the foreign key `user_id`, which is the primary key of the table `users`. Consequently, a new table named `annotation_confsys` is built for authors' comments, with the same structure as that of the table `annotation` but referencing the table `author`. The details will be discussed in the following section.

4.5. Temporary tables

As mentioned in Section 4.4, two tables to register the annotations of the digital resources, are created separately based on the roles of the annotators, contributors (i.e. authors) or users (i.e. readers). However, accessing the annotations on a specified resource, by authors themselves or by the readers of interest, creates a problem. To resolve this, a temporary table is introduced in the CINDI system. Temporary tables in MySQL are used to supply the functionalities similar to views in the other commercial RDBMS. One of the advantages is to speed up the search. Another one is to save the disk space because MySQL will remove the temporary tables automatically and free up the space that the table used when the database is disconnected. Moreover, if the type of a temporary table is defined as `HEAP`, queries run against the table may be much faster than on-disk temporary tables. However, it does not support the data types of `text` and `BLOB`. Therefore, a temporary table for searching for resources will be created as the type of `MyISAM` as the default in the CINDI system. Another temporary table created for searching for the relationship resource and its authors' names can be created as the type of `HEAP`. To make the management of annotations convenient, a temporary table called `TEMPannotation` is built when a CINDI user searches for annotations; this table is used to merge the table `annotation` with the table `confsys_annotation` associated with the resources. Thus, when the information of annotations is required, regardless of the roles of annotators, either authors or readers, the information will be retrieved from the same table. With this approach, the search speed is accelerated and the users of the CINDI system can get all comments. (See Appendix B)

Since full-text searching is used in the CINDI system as discussed in the previous sections, and because only the table type `MyISAM` supports full-text searching, and since the table type of the table `resource` is defined as `INNODB` for protecting the ACID properties, a temporary table of type `MyISAM` is built for speeding up search involving the table `resource`. The implementations are given in Section 6.2.

Chapter 5 Security of the CINDI System

With the development of the Internet, an increasingly common problem for an organization is to make sensitive, internal information securely available on a WWW server that is accessible over the Internet. The comparatively simple technicalities including firewall and constant security patches etc. are applied to protect the WWW servers. Meanwhile, users are required to authenticate with secure registration to the servers.

Users can be authenticated by a server in a number of ways; the most common one being a username/password scheme where a user supplies a user name and the corresponding password; other schemes include the use of a smart card with encrypted authenticated information, and biometrics such as affordable finger scanners and expensive retinal scanners. With these logically simple and cost effective measures, users will be able to access servers without having to re-authenticate. For instance, once users access the authenticated location with privilege granted by the server, then, the subsequent server pages are available to them without re-authentication. Therefore, it requires that each transaction should be authenticated or some form of semi-permanent tokens should be passed so that they can be shown to the server as needed. Secondary considerations consist of the ability to generate an audit trail, track usage in real time, and timeout check to force users log out of the system after a period of inactivity. [SEIFRIED01] There are some existing authentication protocols that can be taken advantage of for the secure web servers.

5.1. Authentication methods

5.1.1. Username and password

This is the most common and one of the cheapest approaches for authentication by taking advantage of people's memory as the storage mechanism. It is so easy to manage and to deploy that most systems use it. However, the weakness is that it is relatively insecure not only because users may choose bad passwords but also passwords get stored in forms' fields, which are easy to be cracked by the invalidated crackers on the Internet.

On Unix, username and password are typically kept in flat text files, namely /etc/passwd and /etc/shadow, in the form username:hashed_password, where the password is hashed so that it is not possible to simply read and use it. Typical hashes used are crypt and MD5 [MD5]; these are much more reliable than checksum and many other commonly used methods. When a username and password need to be checked, the system goes down the list until it finds a matching user name; and then, it hashes the password that was supplied and compares it to the listed value, if they match, then, that means the username and password supplied were correct.

For the web application, it is recommended to use a database file so as to increase the speed of access. It is quite workable for a single server but not easy for the distributed database system unless the data files are kept replicated and up-to-date synchronously across multiple servers.

5.1.2. Cookies and sessions

Cookies are small amount of information in the form of plain text files, which are created and sent to the web browser by the server along with an HTML page when clients access a particular site. Cookies can be stored on the users hard drive and could be kept persistently; this function can be disabled or the stored cookies may be deleted by users. Also, cookies can be stored in memory for the length of session, and will be lost when the browser is closed. The latter form is safer than the former, and it can be easily set as a timeout checker by issuing a time limit on accessing the server. As shown in Figure 5.1, once cookies are saved on the client machine when they arrive, some of them will be returned to the web server in each subsequent request from the clients by passing the following six parameters:

1. The name of the cookie,
2. The value of the cookie,
3. The expiration date of the cookie,
4. The path the cookie is valid for,
5. The domain the cookie is valid for, and
6. The need for a secure connection to exist to use the cookie

Among them, the first two parameters are mandatory; the other four can be set manually or automatically. These parameters are separated by semicolons when they are set explicitly. With these parameters, users do not need to re-authenticate after logging into a system. However, cookies are not the answer to the security of web application's state management not only because cookies are easy to be copied off the system and put on another system to gain access but also because cookies hold a small amount of data that most web browsers allow; typically around 300 bytes including only 20 bytes per domain. [AH02]

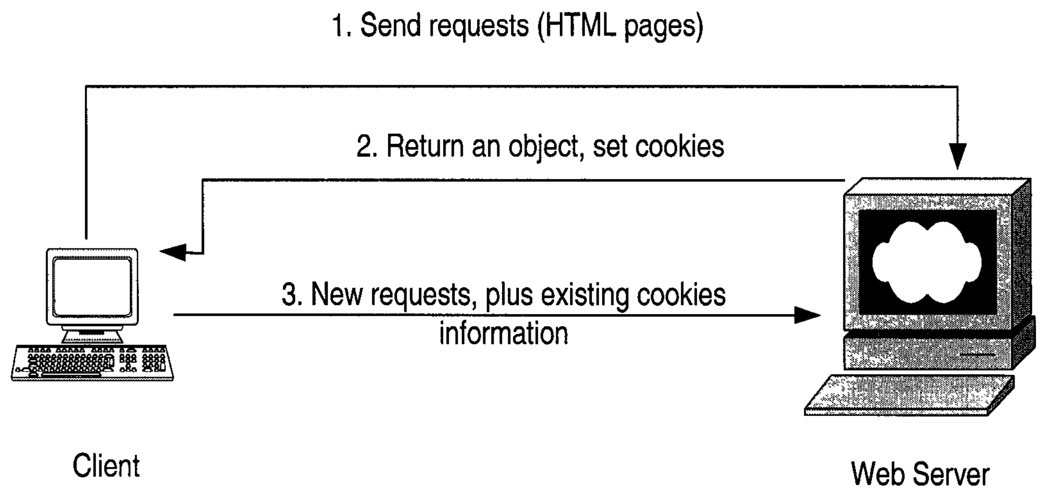
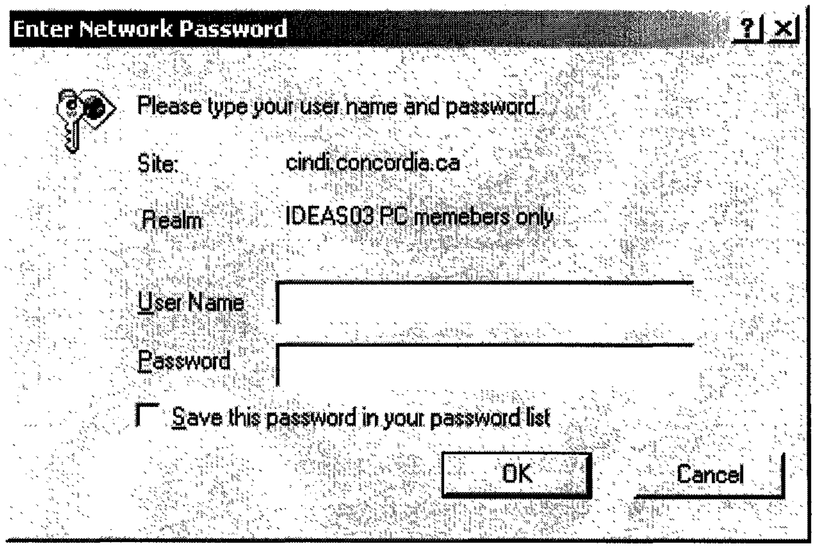

1. Send requests (HTML pages)
2. Return an object, set cookies
3. New requests, plus existing cookies information
Client
Web Server

Figure 5.1 The logical view of cookies

To solve the above limitations on cookies, the better way is to store the state information on the server side rather than the client side. That is what sessions are about. Sessions, or session objects, are server-side collections of variables that make up the state. To associate each set of data with the correct client, a unique session-id identifies one session object on the server side and is transmitted on each request by the client. The most convenient way to send the session-id on each request from the client is to store it in a cookie once the session is initiated. In general, many web sites use a session based on the user login. Once a user logs in the system with valid username and password, a session is initiated by using username and password. Unfortunately, this session-id is easy to be stolen by an attacker who will use the session-id as a short time password to access the web server successfully. Therefore, it is highly recommended that session-id must be encrypted or be a random number that could not be easily duplicated.

5.1.3. HTTP authentication

The basic authentication scheme provided by HTTP (Hypertext Transfer Protocol) is based on the model that the client must authenticate itself with a username and a password for each realm. Realms are referred to be a set of protection spaces into which protected server resources can be partitioned, each with its own authentication scheme and/or authorization database. The realm value should be considered an opaque string passed in an HTTP request during basic authentication that can only be compared for equality with other realms on that server (Figure 5.2). The server will service the request only if it can validate the username and password for the protection space of the Request-URI without any other optional authentication parameters. As the username and password are transmitted over the Internet as the plain text, it is not a secure approach for user authentication unless it is combined with other secure system such as SSL. [FRANKS99]


Enter Network Password

Please type your user name and password.

Site: cindi.concordia.ca

Realm IDEAS03 PC memebers only

User Name

Password

Save this password in your password list

Figure 5.2 An instance of HTTP authentication

The digest authentication is a new feature that is similar to the basic authentication except that the authentication credentials pass through a hashing algorithm. As a result, the encrypted message will be sent to the server instead of the plain text that the basic authentication uses. Therefore, it is much safer than the basic authentication. However, since it still belongs to the password-based systems, it cannot get rid of the drawbacks of this approach, namely being easy to be attacked by using programming tools. Also, some web browsers limit the use of the digest authentication.

HTTP header fields [W3C92] are used to specify the information of an object (i.e. an HTML page) to be transmitted between clients and servers. Using HTTP header based authentication can easily pass through multiple servers since HTTP header authentication data is cached in the browser session without the need to constantly re-authenticate users. However, this means that it is easy for someone else to go back to the sites if the user leaves the browser running and get authenticated with the credentials that are stored in browser memory. On the other hand, since HTTP is stateless, the servers do not know the information about the users and the web sites they visit. It is especially dangerous because the transmitted data including the clients' usernames and passwords would be automatically sent even if the clients had logged off the web sites several days ago. Also, because the authentication happens at the web server, it is hard to track what is going on.

In order to maintain the state information on HTTP servers, the following methods are generally applied:

- Storing the state information in a cookie, which passes the six parameters as mentioned in Section 5.1.2.
- Encoding the state information in URL links, which appends a unique string of data to the URL, or uses the Domain Name System (DNS) and BIND [AL01] to secure the server name. For example, once a user clicks the "Webmail" link on the home page

of Department of Computer Science at Concordia University, s/he goes to it and gets a URL ended with an encoded string as follows:

```
https://mailhost.cs.concordia.ca/horde/imp/login.php?Horde=cf4
44fbbda2f506cb234ecffbcee9739
```

- Sending it in hidden form variables, which is universally supported by web browsers and is similar to using the cookies in concept. However, because it is easy to be accessed by viewing the source of the web pages in the web browsers, it is a bad idea for sensitive data.

5.2. Solution for the CINDI system

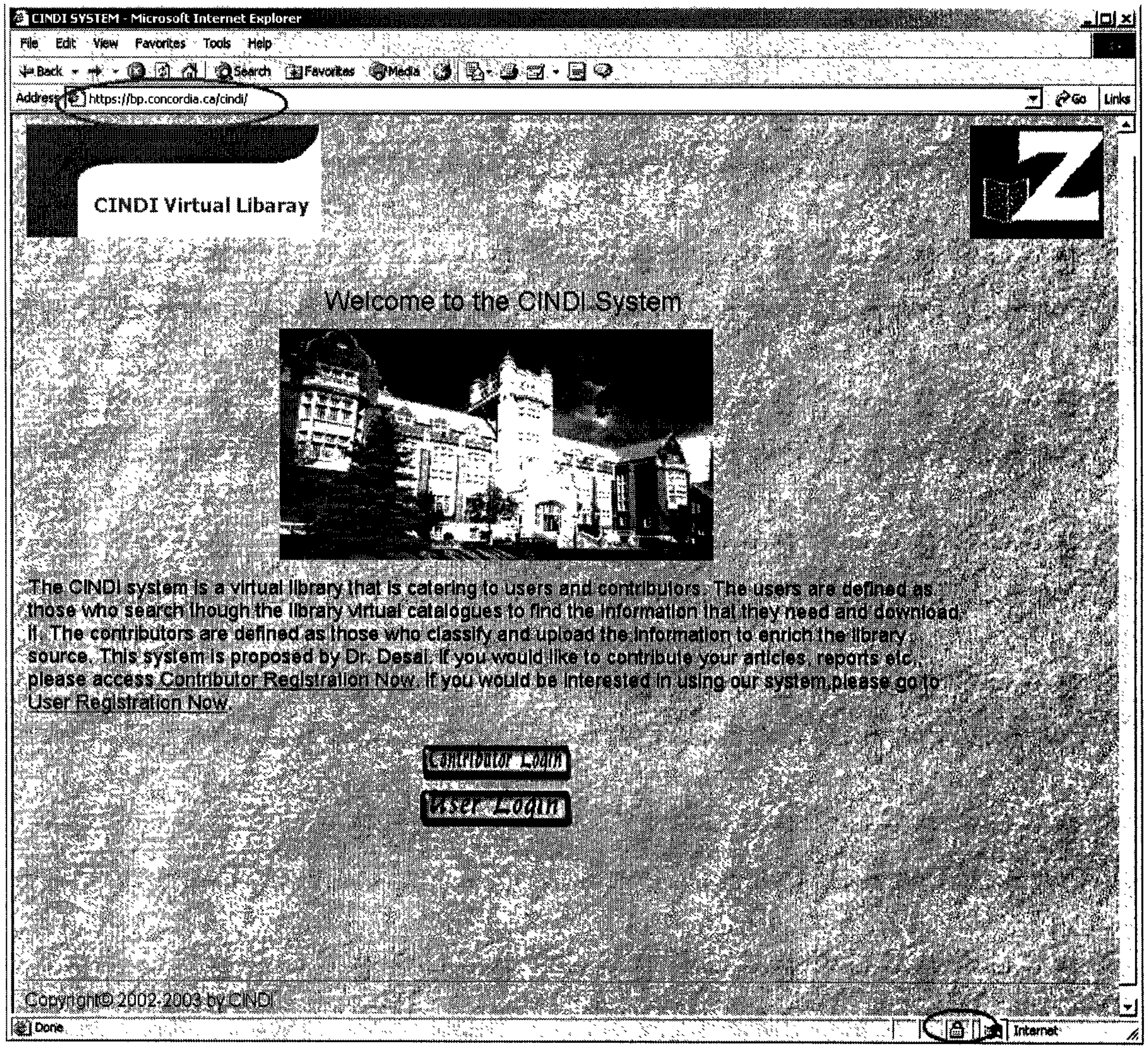
5.2.1. Apache server with SSL

There are many different web servers that support the basic functionality to transfer information via HTTP. Apache is one of the most popular web servers with 62.53% [NETCRAFT03] of the market in the non-secure web serving and a good chunk of the secure market so far. Apache has a large number of features to help handle user sessions and otherwise track them using Perl, PHP, Java servlets and other methods.

The SSL protocol developed by Netscape Communications Corporation is a security protocol that provides privacy between two communicating applications (i.e. a client and a server) over the Internet through a private, authenticated and reliable communication channel with digital certificates. First, the SSL relies on a reliable transport protocol such as TCP for data transmission and reception. Furthermore, the SSL protocol is application independent, allowing higher level application protocols such as HTTP, FTP (File Transfer Protocol), and Telnet to be layered on top of it transparently. Moreover, the SSL protocol is able to negotiate an encryption algorithm and session key as well as to authenticate the server before the application protocol transmits or receives its first byte of data. The SSL protocol maintains the security and integrity of the transmission channel by using encryption, authentication and message authentication codes. [HICKMAN95]

Apache-SSL is a secure Web server, based on Apache and SSLeay/OpenSSL [LL03]. The SSL can be adapted as an add-on module, which can be added to the Apache server to improve the security without changing a single line of code. The SSL uses an encryption technique called the Rivest-Shamir-Adleman (RSA) public key cryptography [RSA] during the handshaking process, which offers a pair of asymmetric keys for encryption and decryption. Each pair of keys consists of a public key and a private key, of which the latter is always linked mathematically to the former. The private key is uniquely derived from the public key with a large number factor that is infeasible to be computed by an attacker. The information to be sent is encrypted with the public key of the receivers while the encrypted data received can be decrypted only with the private key of the intended receivers. Meanwhile, data encrypted with the private key as a digital signature must be decrypted only with the associated public key and be verified according to a simple, prescribed mathematical relation. Consequently, it is practically impossible to derive the clear text information between the connected parties. In view of the above, more and more web applications use secure Apache server with the SSL to prevent the information from being intercepted and to keep secure communication between servers and clients.

From the viewpoint of users, having a secure web server Apache with the SSL, the URL is expressed using a protocol `https` (Figure 5.3) instead of `http`. A lock icon in the web browser as shown in Figure 5.3 is used to indicate the use of secure interaction. Considering the security of the entire system, the CINDI system takes advantage of this feature of the Apache and the SSL protocol to build a secure web server for its users.


CINDI SYSTEM - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://bp.concordia.ca/cindi/
Go Links
CINDI Virtual Libaray
Welcome to the CINDI System
Contributor Login
User Login
Done
Internet

The CINDI system is a virtual library that is catering to users and contributors. The users are defined as those who search though the library virtual catalogues to find the information that they need and download it. The contributors are defined as those who classify and upload the information to enrich the library source. This system is proposed by Dr. Desai. If you would like to contribute your articles, reports etc., please access Contributor Registration Now. If you would be interested in using our system,please go to User Registration Now.

Copyright© 2002-2003 by CINDI

Figure 5.3 The window browsing an Apache-SSL web system

5.2.2. Application of PHP

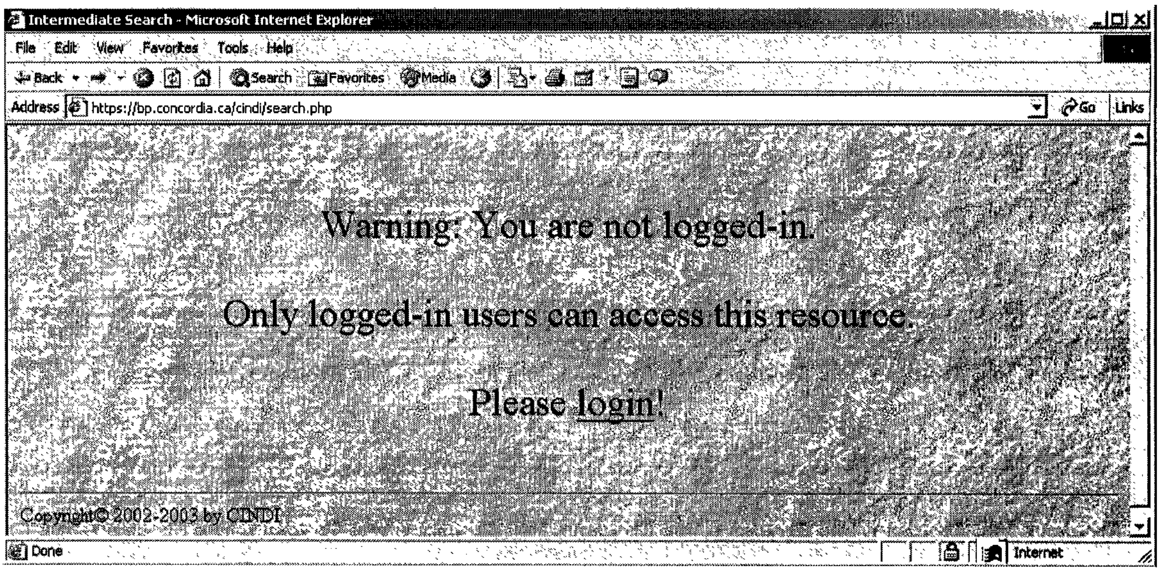
PHP (PHP Hypertext Preprocessor) is a server-side scripting language that facilitates the creation of dynamic Web pages by embedding PHP-coded logic in HTML documents. It combines many of the finest features of Perl, C, and Java, and adds its own elements to the concoction to give Web programmers great flexibility and power in designing and implementing dynamic, content-directed Web pages.

Web applications have become a popular way to provide global access to data, services, and products. While this global access is one of the Web's underlying advantages, any security holes in these applications are also globally exposed and frequently exploited. It is extremely easy to write applications that contain unintentional security holes. To implement a secure web application by using PHP in this thesis, we use the secure configuration of the PHP installation that turns `register_globals` [PHP03] off in the CINDI system. By turning off the ability for any user-submitted variable to be injected into PHP code, the amount of variables poisoning a potential attacker may inflict will be reduced because the internal variables are effectively isolated from user submitted data.

As stated previously, the basic HTTP authentication uses a simple challenge-and-response mechanism to obtain credentials from users. In order to have more control over user authentication in the form of whether usernames and passwords are stored and how they are looked up, a PHP-based solution is utilized in the CINDI system. Here, session cookies are used to store the state information. Also, with the `header()` function provided by PHP, the HTTP response header is specified as follows so that no state information will be stored in the local machine and thus disable the “Back” function of the web browser. Consequently, by using session control, it prevents subsequent users on the same computer from viewing the sensitive information of the users of the CINDI system. Furthermore, the "stale" page will be logged out off the system automatically by setting a cookie with 20 minutes of the lifetime.

```
$expiry = 60*20; // 20 mins
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");     // Date in
the past
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
header("Cache-Control: no-cache, must-revalidate");
header("Pragma: no-cache");
session_start();
setcookie(session_name(), session_id(), time()+$expiry, "/");
```

To transmit some variables across requests, the CINDI system either relies on registering them in sessions with session handling functions or hides them in the HTML form. For each user, the CINDI system will register the username and password with login sessions so that each one will have a unique identifier. Since the session module cannot guarantee that the information stored in a session is only viewed by the user who created the session, more protection is applied to the sensitive data, especially the authenticated usernames and passwords. In order to keep the user information secure, some important data carried by sessions, such as passwords, will be encrypted with the MD5 hashing function. With the session control in the PHP programs, even if someone knows the URL that the authenticated user requests, s/he cannot access it. For example, without logging into the CINDI system, if a user just inputs the URL `https://bp.concordia.ca/cindi/search.php`, then a warning message will be displayed as follows:


Intermediate Search - Microsoft Internet Explorer

https://bp.concordia.ca/cindi/search.php

Warning: You are not logged-in.

Only logged-in users can access this resource.

Please login!

Copyright© 2002-2003 by CINDI

Figure 5.4 A warning message for invalid users

5.2.3. Cryptography

Cryptography is the art and science of keeping message secure, which at its best facilities the following: [HORVATH02]

- Confidentiality: the sender of a message should be able to limit access to a message.
- Authentication: the receiver of a message should be able to ascertain the origin of the message.
- Integrity: the receiver of a message should be able to make certain whether a message has been modified in transmission.

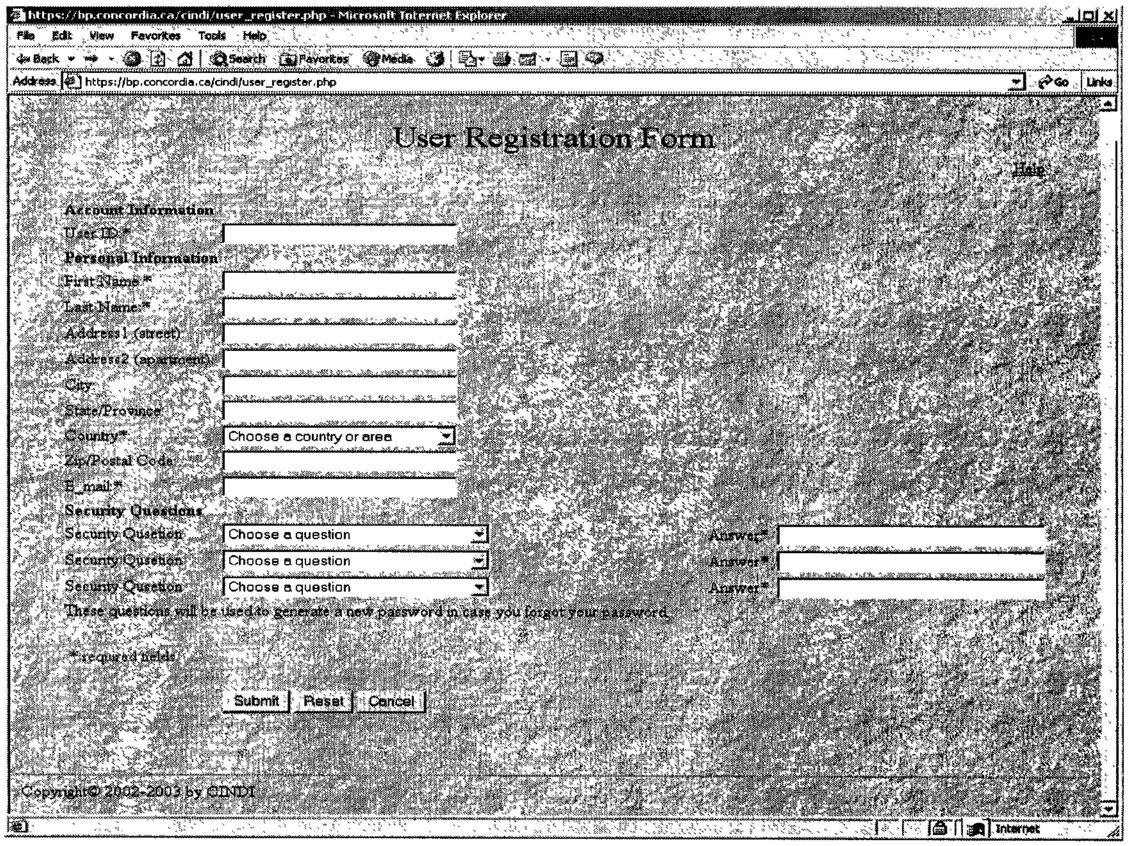
Encryption plays a key role in providing confidentiality. Encryption disguises the contents of a message by applying a mathematical algorithm to it. Not only is encryption valuable in ensuring the confidentiality of a message, but also it is useful in maintaining the privacy of any data, either stored in files or recorded in databases.

There are many encryption algorithms providing varying degrees of security, such as DES (Data Encryption Standard) and RSA. DES is a U.S. and international standard, being a symmetric key algorithm using the same key (i.e. the secret key) to encrypt a message and to decrypt it. In contrast, RSA, as stated above, is an asymmetric cryptosystem using one key (i.e. the public key) to encrypt a message and a different key (namely, the private key) to decrypt it. The security of the encrypted string depends on maintaining the secrecy of the private key.

A one-way hash function, also known as a message digest, fingerprint or compression function, is a mathematical function that takes a variable-length string as the input and converts it into a fixed-length binary value as the output. Furthermore, a one-way hash function is designed in such a way that it is technically infeasible to reverse the conversion, that is, to find a string that hashes to a given value, hence the name **one-way**. It should also be hard to find two arbitrary strings that would produce the same hash value. Therefore, the one-way hash function is important in securing information as well as in providing some measure of data integrity.

Since the CINDI system applies usernames and passwords as the authentication identifiers transmitted over the Internet, using a clear text is not a safe way as stated previously. In this project, the MD5 hash function is used in the CINDI system; the MD5 converts a variable-length string to a fixed-length string of 32 bytes and is used for passwords, thus creating a "fingerprint" of the passwords that can be stored in a database. Because no one can reverse the conversion, storing a password in this fashion is very secure.

5.2.4. User Registration


https://bp.concordia.ca/cindi/user_register.php - Microsoft Internet Explorer

User Registration Form

Help

Account Information

User ID:*

Personal Information

First Name:*

Last Name:*

Address1 (street)

Address2 (apartment)

City

State/Province

Country* Choose a country or area

Zip/Postal Code

E_mail:*

Security Questions

Security Question	Choose a question	Answer*
Security Question	Choose a question	Answer*
Security Question	Choose a question	Answer*

These questions will be used to generate a new password in case you forgot your password.

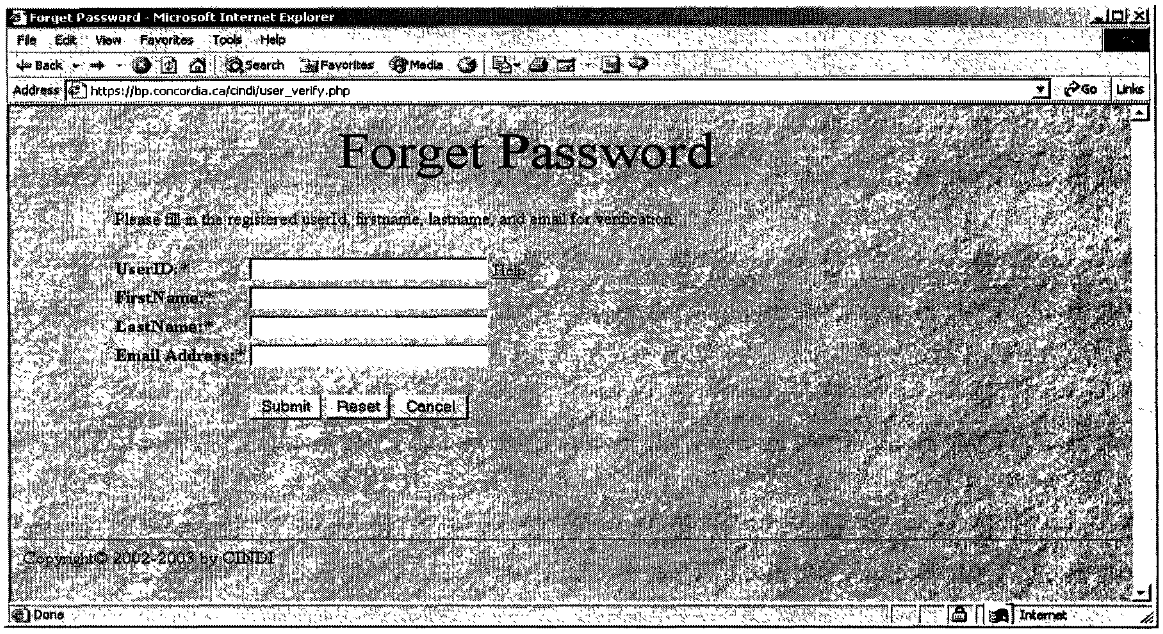
* required fields

Submit Reset Cancel

Copyright© 2002-2003 by CINDI

Figure 5.5 The interface for user registration

The CINDI system supports two classes of users: the users who are able to search for the information of their interest and to make comments, and the contributors who provide the resources along with the semantic header information of the resources to be maintained by the CINDI. Both classes of users are required to input their personal information in order to register with the system by using the graphical interface as shown in Figure 5.5. The required fields for users' personal information include username (i.e. User ID), first name, last name, address, country and email. The email is used to send the user a randomly generated password. In case the user gave an invalid faked email addresses, the password would not be received by the user. The email addresses registered by the users will also be used for their personal information management as identifiers in case users forgot their passwords (Figure 5.6). Based on the previous registration [WANG02], we added the security questions in this project for users during registration. Then the questions and their corresponding answers will be used to validate a user if s/he forgets her/his password and wants to obtain a new one. (Figure 5.7)


Forget Password - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address https://bp.concordia.ca/cindi/user_verify.php

Forget Password

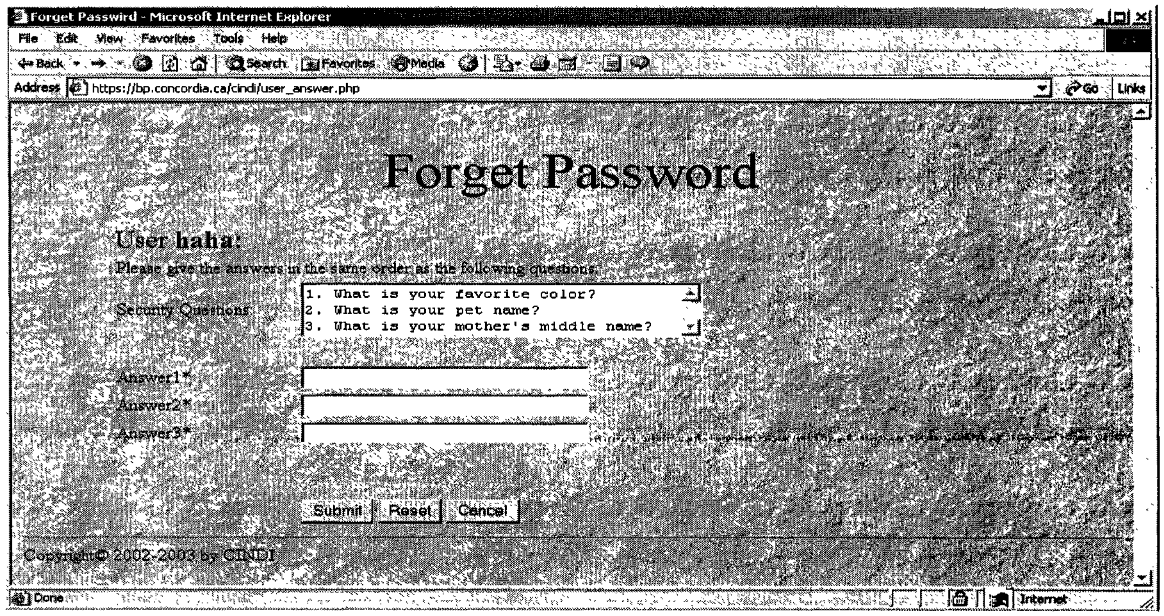
Please fill in the registered userId, firstname, lastname, and email for verification.

UserID:* Help
FirstName:*
LastName:*
Email Address:*

Submit Reset Cancel

Copyright© 2002-2003 by CINDI

Figure 5.6 The interface for forgetting passwords


Forget Passwird - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address https://bp.concordia.ca/cindi/user_answer.php

Forget Password

User haha:

Please give the answers in the same order as the following questions:

Security Questions
1. What is your favorite color?
2. What is your pet name?
3. What is your mother's middle name?

Answer1*
Answer2*
Answer3*

Submit Reset Cancel

Copyright© 2002-2003 by CINDI

Figure 5.7 The interface for verifying users' security questions

Chapter 6 Implementation, Test and Results

The results of the work done for this thesis are presented in this chapter. Since the work involved many fields, such as system backup, the CONFSYS integration with PHP, searching process, testing and debugging, we provide the implementation and results for these subtasks under separate categories.

6.1. System backup

As discussed in Chapter 3, there are several approaches for system backup, including file-based replication and database replication. These approaches are utilized by the CINDI system depending on the data types and the functional requirement.

In order to keep a backup of the current CINDI system, a backup server is set up, and the data and files are replicated automatically everyday by using the cron job. However, for the data stored in MySQL on the CINDI server, a snapshot is required if the replication is implemented asynchronously. This snapshot is done by using the shell script that embeds `mysqldump` [MYSQL03], which is a command-line script to dump a database or a collection of database for backup or for transferring the data to another SQL server (not necessarily a MySQL server). The details are listed in Appendix C. The data file that is dumped from the MySQL database is replicated to the backup server with the file-based method by using `scp`.

The data in the CONFSYS database on the conference site is replicated synchronously by using PHP functions `mysql_connect()` and `mysql_select_db()` to the remote server CINDI. As stated below, the function `mysql_connect()` is used to build two separate links to the local database server (i.e. the conference server called “localhost”) and the remote database server (the CINDI server namely `bp.cocnordia.ca`), respectively. Based on the links with the corresponding servers, the associated databases, namely `cindi` and `confsys`, will be selected by using `mysql_select_db()`:

```
$db = mysql_connect("bp.cocnordia.ca", "cindidatabase",
"password")
 or die("unable to connect to mysql server");
mysql_select_db("cindi", $db)
   or die("unable to select database 'cindi': ".mysql_error());
$db_confsys = mysql_connect("localhost","confsysdatabase",
"password")
 or die("unable to connect to mysql server");
mysql_select_db("confsys", $db_confsys)
   or die("unable to select database 'confsys': ".mysql_error());
```

With the two links, the queries against the two databases are run separately and concurrently. For example, the author’s information is recorded into the two databases when the author registers the personal information for the submitted paper.

```
$result_confsys = mysql_query("insert into author values
(NULL,'$author_fname','$author_lname',
'$organization','$address','$phone','$email',null, null, null,
null)", $db_confsys);
$result_cindi = mysql_query("insert into author values
(NULL,'$author_fname','$author_lname',
'$organization','$address','$phone','$email', null, null, null,
null)", $db);
```

With this method, the CINDI database is a remote backup of the local conference database. Meanwhile, it makes the integration of the CONFSYS into the CINDI convenient.

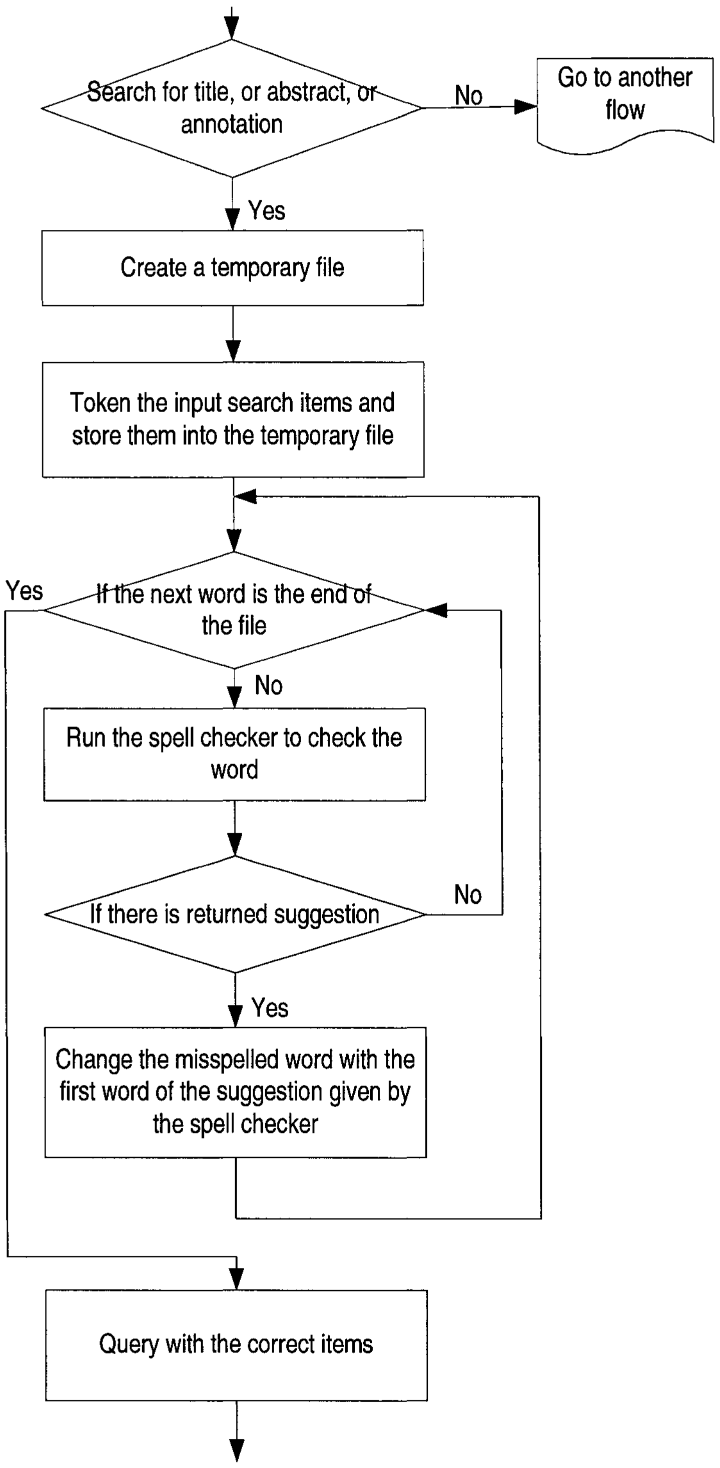
6.2. Searching process

6.2.1. Implementation of fuzzy search

In the administration sub-system, fuzzy search is utilized for the text search items including title, abstract, and annotation in order to address the human recognition problem and to provide suggestions. As introduced previously, the Aspell spell checker is adopted in the searching system, which is able to give ten suggested words, as shown in the following example: if "united states of America" is mistyped as "vnited states of America". By using the spelling checker, we get the suggestion below:

```
1) united                                  6) wonted

2) vented                                  7) noted

3) vaunted                                 8) voted

4) vined                                   9) vended

5) vomited                                 0) dinted


i) Ignore                                  I) Ignore all

r) Replace                                 R) Replace all

a) Add                                     x) Exit
```

In this case, it requires that users interact during the processing of spell check. For the searching function, in order to avoid this interaction from users, we consider the first word given by the spell checker as the best and closest choice. With the corrected words, the queries will be run against the tables. The details are shown in the following workflow chart.



Search for title, or abstract, or annotation
No
Go to another flow
Yes
Create a temporary file
Token the input search items and store them into the temporary file
Yes
If the next word is the end of the file
No
Run the spell checker to check the word
If there is returned suggestion
No
Yes
Change the misspelled word with the first word of the suggestion given by the spell checker
Query with the correct items

Figure 6.1 The workflow of the application of Aspell

6.2.2. Implementation of full-text searching

As mentioned in Section 4.2.4, to keep the ACID properties and to speed up the searching, a temporary table is created based on the table `resource` in the searching subsystem. In general, people can provide several keywords when looking for a relevant resource, and these keywords may exist in the abstract and keyword separately. In this case, the simple match operators "=" and "`LIKE`" are not used in the simple Boolean condition to search for these keywords. The full-text searching is the right answer to this problem.

First of all, a temporary table called `TEMPresource` is created with the default table type `MyISAM` that supports full-text indexes. Based on the temporary table, a full-text index is added for the `abstract` and `keyword` fields as follows:

```
include_once("dbConnect.php");
$sql_temp = "CREATE TEMPORARY TABLE IF NOT EXISTS TEMPresource ";
$sql_temp .= "select * from resource";
$sql_temp_query = mysql_query($sql_temp, $db);
if (!$sql_temp_query)
  error_message(mysql_error());
$sql_temp_index    =    "ALTER    TABLE    TEMPresource    ADD
FULLTEXT(abstract, keyword)";
$sql_temp_index_query = mysql_query($sql_temp_index, $db);
if (!$sql_temp_index_query)
  error_message(mysql_error());
```

Afterwards, the queries run against keywords can be implemented by using the full-text match operator `match()against()` for the SQL WHERE condition as shown below.

```
$kword .=  "match(abstract,keyword) against ('";
for($i = 0; $i < $count_comma; $i++)
{
  $kword .= $keyword_array[$i];
  if ($i < $count_comma - 1)
    $kword .= " ";
}
$kword .= "') ";
```

Here, `$count_comma` is a variable to count the number of commas that are used to separate the entered keywords. With the above method, the SQL WHERE condition is written as `match(abstract,keyword) against ('$keyword_array[0] $keyword_array[1] $keyword_array[2]')` if there are three keywords input by users.

6.3. Test results

In order to test the functions of the CINDI system, we implemented and tested it by using a number of different scenarios. The goal is to improve the system performance and to fix the existing problems. The following tables show some key scenarios and the results obtained by testing. As shown in Table 9, there were several problems that needed debugging in the previous CINDI system. Through our analysis and improvement, we fixed these problems and obtained the test results shown in Table 10.

Table 9. Test scenarios and results of the original CINDI system [WANG02]

Test Contents	Scenario Descriptions	Results
Security	The system provides a warning message for a user who wants to enter the system without login by entering one of the URLs of the system.	Passed. (See Figure 5.4)
	With "Back" button in the web browser to enter the system after a user logs out the system. In this case, there is a warning message for invalid access.	Failed. Because the session cookies store in the local machine, a user can access the system by using the "Back" button after logging out
	A stale page is logged out of the system automatically after a period.	Failed. Without any timeout control, a web page can keep alive even though it is keeping "stale".
Database design	Database design follows the basic normalization to reduce data redundancy.	Failed. There is a redundant table called `resource_authorname`.
Query processing	A resource is not registered in the system if the registration is not finished when the contributor log out.	Failed. Since the insertion begins at the beginning of the registration without monitoring the registration process, when the registration of a resource is not finished, there is no action to process the inserted record so as to make some redundant data in the database.
	By using the searching interface, users can select the subjects stored in the database.	Failed. Because of the wrong queries run against the table subject, for the sub-sub-subject, users can only get a subset of the subjects. For example, the number of the sub-sub-subjects that belong to the sub-subject with the code of 1022 should be 97 instead of 10 given by the current query. (See Appendix D)
File operation	On the command line, the semantic header is generated automatically and stored in the directory.	Passed.
	Via the web, the semantic header is generated automatically and stored in the directory.	Failed. Since the web users belong to nobody/nogroup, the lowest permission, they are illegible to access the directory with the higher privilege cindi/users. Thus, the semantic header generator cannot create correctly the same file as on the command line.

Table 10. Test scenarios and results of the enhanced CINDI system

Test Contents	Scenario Descriptions	Approaches to fixing the problems	Results
Security	With "Back" button in the web browser to enter the system after a user log out the system. In this case, there is a warning message.	By using the PHP function `header()` and session cookies, as discussed in Section 5.2.2.	Passed. A warning message is shown as Figure 5.4.
	A stale page is logged out of the system automatically.	By setting the timeout with the session cookies	Passed. The system will log out after 20 minutes.
Database design	Database design follows the basic normalization to reduce data redundancy.	As stated in Section 4.2.1, we created a temporary table to play the same role of the redundant table to reduce the redundant data and to free up the space so that the design meets the database normalized forms.	Passed.
Query processing	A resource is not registered in the system if the registration is not finished when the contributor logs out.	By creating a table that records the process status of a resource, the related data will be deleted automatically when the contributor logs out the system, before the registration is not finished.	Passed.
	By using the searching interface, users can select the subjects stored in the database.	By modifying the query against the subject. (See Appendix D)	Passed.
File operation	Via the web, the semantic header is generated automatically and stored in the directory.	By correcting the privilege of the directory as nobody/nogroup.	Passed. The semantic header generator can create correctly the same file as on the command line.

The overall test results bring us the conclusion that some key issues, such as security, database design and query processing, are solved in this thesis. However, a real life scenario with large amount data, such as a number of concurrent users, should extend the test in order to assess the performance of the system.

Chapter 7 Conclusion and Future Work

7.1. Conclusion

The web-based CINDI system is being developed to launch a searchable distributed database system of material documenting scholarly works for users who are looking for some specific topics over the Internet. Its goal is to build an efficient and effective virtual library containing a wide range of digital document collection described by a data structure to record the bibliographic information of the resources, i.e. the Semantic Header. It aims to help users without any cataloging knowledge to search for this up-to-date information through the Internet.

Making use of the Semantic Header meta-data entry, contributors of resources are able to register the resources on the web either by themselves manually or by invoking the ASHG automatically. The former method would be more accurate and support any format of the resources, while the latter is much faster and can process some standard styles of files. With the ASHG, the CONFSYS, a subsystem of the CINDI, provides the authors who submit their papers to a specific conference with the semantic header generated automatically for the uploaded papers. As a result, it not only improves the submission of papers but also enriches the collection of the CINDI system. With the presence of abstracts and annotations in the semantic header, not only does the CINDI system provide a summary of documents but also it can aid to making better decision regarding the relevance of the resource without actually downloading the enlisted resource.

Nowadays, keywords are the most popular search criterion utilized by most of the existing search engines to retrieve the online information. As stated and tested previously, this scheme results in lower search precision as well as recall because the context of the meaning of a word is lost. To address this context problem in the search engines, the CINDI system provides the user with several different search criteria: title, author name(s), keyword(s), subject and the period of created date, which are offered in three levels of search depending on the users' knowledge of the resources. Since users may make mistakes in typing and spelling caused by human OCR, fuzzy search is implemented using the Aspell spell checker to improve the usability of the system.

In order to facilitate the system administrators to manage the information of the CINDI system, the administration subsystem has been designed and implemented. It assists the administrators in editing and managing the user/contributor personal information for registration as well as the bibliographic index of the registered resources. Moreover, to provide the up-to-date information for users, the administrators can take advantage of the administration subsystem to merge the latest conference information regarding the submitted papers into the CINDI system easily and quickly.

Considering the usability of the CINDI system, the graphical user interfaces for its five subsystems are designed based on the knowledge and experience of users. Since users generally have no knowledge of cataloging in a library, the registration subsystem provides a standard semantic header metadata entry for contributors to help them index the contribution. Furthermore, to help users to find the needed documents efficiently from the CINDI virtual library, the system offers the style of searching interface from users to users based on the users' knowledge about the documents they need. Moreover, based on the Window's interface style, online help is offered for users by the CINDI system, which can guide users' operations on the associated subsystems.

7.2. Contribution to the CINDI system

The contribution of the thesis to the CINDI system falls into the following categories:

- Refine the database design for the CINDI system to make the system more effective and efficient.
- Introduce the replication of MySQL to the CINDI system to supply high availability for the system, and backup the CONFSYS in the remote server so as to implement the distributed database system for the CINDI.
- Improve the security of the CINDI system by upgrading the PHP to turn the register_global variables off, improving the security of the PHP programming with session and cookies controls, and introducing MD5, the one-way encryption algorithm for passwords.
- Design and implement the administration subsystem in order to make the system management convenient and efficient.
- Take advantage of temporary tables of MySQL to speed up searching and to free up disk space as well as to reduce the redundant tables from the system.
- Introduce the Aspell spell checker to the CINDI system to support fuzzy search in case of users' mistyping.
- Implement the personal information management for the users and contributors.
- Integrate the CONFSYS to be a subsystem of the CINDI so that the CINDI system can include the latest information in a rapid manner for users.

- Improve the registration subsystem by developing the registration function with security questions.
- Integrate the semantic header generator for PDF files with the CONFSYS to improve its performance.
- Enhance the functionalities of the CINDI system by processing the compressed files and by integrating the semantic header generator for PDF files.
- Improve the performance of the CINDI system by revising the interfaces of the system, testing the system with black-box and white-box methods as well as debugging.

7.3. Future work

In order to improve the performance of the CINDI system further and to provide an efficient and effective digital library for the users on the Internet, the following are some suggestion that will be helpful for the future work:

- Uniform and improve the interfaces of the CINDI system

  A good interface enables users to quickly and effortlessly access any place of the web site by providing a well-defined, clearly-named entry point. This requires that the interfaces have a distinctive fashion, including the background, the icons, and the text fields etc., which are used in the same way to identify the major function sections. In the current CINDI system, the backgrounds and text colors need improving since the five subsystems have different colors and backgrounds, although each subsystem has its own unified colors and background.

In order to make the use of the CINDI system efficiently and effectively, the CINDI system provides an online help on each accessible page. However, with the future development of the CINDI, more details would be required.

- Improve the accuracy and functional performance of the semantic header generator

  So far, the semantic header generator can support the standard styles of files, such as HTML, RTF, PDF, $\LaTeX$ and TXT. Since the document files PostScript and Windows DOC are also used for documents, the semantic header generator should be enhanced so as to be able to generate the bibliographic indexed for these files. During the implementation of this thesis, a project to generate PDF files from RTF and DOC is underway.

  By testing the semantic header generator, we find that authors' information and subject fields are key points in measuring the accuracy of the bibliographic indexes. To improve its performance, while enlarging the subject data set, a controlled word dictionary of the exactly associated subjects should be built such that the most possible keywords can be linked with the related subjects based on their context. In addition, the efficiency of the semantic header generator should be improved by optimizing the C++ and C programs in its implementation.

- Improve fuzzy search

  By utilizing the open source Aspell spell checker, the CINDI system provides the simple fuzzy search based on the word list provided by the Aspell, which is a common English dictionary. However, for a user-oriented system, the common dictionary is limited to looking up the specific words that may be considered misspelling. In order to improve this function, a dictionary based on the information of existing resources in the CINDI system should be set up by taking advantage of the customization of the Aspell. Also, fuzzy logics should be considered in the future.

- Add the intermediate steps for search

  After the users click the button for search in Figure 4.2, 4.3 and 4.4, we should give the short introduction to the search results along with the corresponding URLs. The users would browse the information on the current screen to decide which of the entries best fit their needs. Then, using the relevant URL, the semantic header of the resource would be displayed.

- Tune the MySQL databases

  Basically, from the viewpoint of functionality, the CONFSYS subsystem can be smoothly applied to a conference. However, from the position of efficiency and performance, it requires improving the relationships between the tables or relations. For example, the linkage between the submitted papers and their authors should be represented by creating a distinctive table, instead of by using the table containing the information of the papers, as shown in Figure 4.1(c).

- Improve and implement the distributed database system

  The current CINDI system supplies the replication of the MySQL databases to improve the performance and protect the availability of applications because alternate data access option exists; therefore, if the production server failed, the replica can be used as the production system to ensure high availability. With a larger number of users, a "bottle-neck" problem will be caused by the increased traffic. Therefore, load balancing should be considered. One of the solutions is to horizontally partition the database system based on subject areas. That is, one site will accept the requests on a subject such as Computer Science; another one will process the visits on another subject, for example, Electrical Engineering. One more solution is to switch the users' requests based on their goals. That means, the contributors who will upload some resources can access one server while the users who are going to search for some resources can visit another one. In order to keep all sites synchronized, it is better to distribute the write queries to all the servers.

References

[AD02] Allen, Susan and Davis, Eric, *A Practical Application of Dublin Core: Worthington Memory,* COASIS&T, September 2002

http://www.worthingtonmemory.org/PDFfiles/CO-ASIST_DublinCore_Presentation_files/frame.htm

Archive copy: http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/frame.htm.zip

[AH02] Allen, Jeremy and Hornberger, Charles, *Mastering PHP 4.1*, pp333-356, SYBEX Inc., 2002

[AL01] Albitz, Paul and Liu, Cricket, *DNS and BIND*, chapter 11, 4th Edition, O'REILLY, 2001

[ARMS02] William Y. Arms, *Digital Libraries*, pp40-48, The MIT Press, 2000

[ATKINSON02] Atkinson, Kevin, *GNU Aspell*, 2002

http://aspell.net

Archive copy: http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/aspell.zip

[DELANY00] Delany, Doug, *Implementing Highly Available Commercial Applications under Linux using Data Replication*, PolyServer, Inc., December 2000

[DESAI94] Desai, Bipin C., *A System for Seamless Search of Distributed Information Sources*, May 1994.

http://www.cs.concordia.ca/~faculty/bcdesai/web-publ/w3-paper.html

[DESAI95] Desai, Bipin C., *The Semantic Header and Indexing and Searching on the Internet,* 1995

http://www.cs.concordia.ca/~faculty/bcdesai/web-publ/cindi-system-1.1.html

[DESAI02] Desai, Bipin C., *Search and Discovery on the Web,* 2002 http://www.ssgrr.it/en/ssgrr20025/papers/74.pdf

Archive copy:

http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/searchDiscoveryonWeb.pdf

[FRANKS99] Franks, J. et al., HTTP Authentication: Basic and Digest Access Authentication, The Internet Society, 1999

http://www.ietf.org/rfc/rfc2617.txt

Archive copy: http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/rfc2617.txt

[GJ88] George, Coulouris and Jean, Dollimore, *Distributed Systems: Concepts and design*, AddisonWesley, 1988

[GU02] Gu, Zhengwei, *CONFSYS: The Cindi Conference Support System,* Department of Computer Science, Concordia University, 2002

http://www.cs.concordia.ca/~faculty/bcdesai/grads/zgureport.pdf

[HADDAD98] Haddad, Sami, *Automatic Semantic Header Generator*, Department of Computer Science, Concordia University, September, 1998

http://www.cs.concordia.ca/~faculty/bcdesai/grads/haddad-thesis.pdf

[HICKMAN95] Hickman, Kipp E.B., *SSL 2.0 PROTOCOL SPECIFICATION,* Netscape Communications Corp. 1995

http://wp.netscape.com/eng/security/SSL_2.html

Archive copy: http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/SSL.zip

[HORVATH02] Horvath, Mary, *Understanding Encrypt, ToBase64, and Hash*, Allaire Documentation Group, Allaire Corporation, 2000

http://www.macromedia.com/devnet/server_archive/articles/understanding_encrypt.html

Archive copy:

http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/understanding_encrypt.zip

[JIN03] Jin, Xin, *CONFSYS:The Cindi Conference Support System*, Department of Computer Science, Concordia University, 2003

[KOFLER01] Kofler, Michael, *MySQL*, Spinger-Verlag GmbH&Co. KG, 2001

[LL03] Laurie, Ben and Laurie, Adam,*Apache-SSL*, 2003

http://www.apache-ssl.org/

Archive copy: http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/Apache-ssl.zip

[MARC01] Network Development and MARC Standards Office, *MARC to Dublin Core Crosswalk*, Library of Congress, February, 2001

http://www.loc.gov/marc/marc2dc.html

Archive copy: http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/marc2dc.zip

[MD5] MD5 Homepage (unofficial)

http://userpages.umbc.edu/~mabzug1/cs/md5/md5.html

Archive copy: http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/md5.html

[MILLER96] Miller, Paul, *An application of Dublin Core from the Archaeology Data Service (draft),* July 1996

http://intarch.ac.uk/ahds/project/metadata/dublin.html

Archive copy: http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/dublin.zip

[MYSQL03] *MySQL Document*, MySQL AB, 1995-2003

http://www.mysql.com/documentation/index.html

[NETCRAFT03] *Web Server Survey*, Netcraft Ltd., 2003 http://news.netcraft.com/

Archive copy: http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/netcraft.zip

[NP02] Nagarkar, Shubhada and Parekh, Harsha, *Development and Application of Dublin Core Metadata Standards in Marathi*, Proc. Int. Conf. on Dublin Core and Metadata for e-Communities 2002: 235-236

http://www.bncf.net/dc2002/program/ft/poster11.pdf

Archive copy: http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/poster11.pdf

[OKK01] Onyancha, Irene and Keizer, Johannes and Katz, Stephen, *A Dublin Core Application Profile in the Agricultural Domain*, International Conference on Dublin Core and Metadata Applications, October 2001

http://www.fao.org/agris/MagazineArchive/MetaData/DC2001_Japan.doc

Archive copy:

http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/DC2001_Japan.doc

[ORACLE99] *Oracle8i Concepts*, Release 8.1.5: Chapter 33, Distributed Databases – ORACLE, 1999

http://home.fms.indiana.edu/users/cshelton/oracle/server.815/a67781/toc.htm

Archive copy: http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/toc.zip

[PHP03] *PHP Manual,* The PHP Group, 2001-2003

http://php.lamphost.net/manual/en/

[RG00] Ramakrishnan, Raghu and Gehrke, Johannes, *Database Management Systems*, pp237-243, Mcgraw-Hill Higher Education, 2th edition, 2000

[RSA] *RSA*

http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci214273,00.html

Archive copy: http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/RSA.zip

[SEIFRIED01] Seifried, Kurt, *WWW Authentication*, 2001

http://www.seifried.org/security/www-auth/index.html

Archive copy:

http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/www-auth.html

[SHAYAN97] Shayan, Rajabihan Nader, *Cindi: The Database System for Subject Headings and Semantic Headers*, Department of Computer Science, Concordia University, May, 1997

http://www.cs.concordia.ca/~faculty/bcdesai/grads/nader-thesis.pdf

[UCR00] The University of California Regents, *Digital Library Research & Development,* 2000

http://sunsite.berkeley.edu/R+D/

Archive copy:

http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/digital_library.zip

[WANG02] Wang, Yuhui, *Enhanced Web Based Cindi System,* Department of Computer Science, Concordia University, 2002

[W3C92] *Object MetaInformation*, W3C, 1992

http://www.w3.org/Protocols/HTTP/Object_Headers.html

Archive copy:

http://www.cs.concordia.ca/~faculty/bcdesai/grads/yanho_li/Object_Headers.zip

[ZHANG02] Zhang, Zhan, *Cindi's ASHG*, Department of Computer Science, Concordia University, 2002

Appendix A: Application for foreign key constraints

In order to adhere to the ACID properties of the relational database, we make use of the foreign keys to keep the referenced tables (e.g. `resource` and `author`) and the referencing tables (e.g. `coverage` and `resource_author`) consistent. Thus, the insertion should begin from the referenced tables to the referencing tables while the deletion should be in the reverse order, from the referencing tables to the referenced tables. Based on the foreign key constraints, we give an example below of moving the resource records of the older version to the backup database, including deletion and insertion. Here, the primary key `resource_id` of the table `resource` is referenced by the referencing tables such as `identifier,coverage` and so on.

```
//get the resource records of the older version from the relevant
tables
if ( strcmp($author,  $older_author) == 0) { //delete the older
ones
        $query2 = mysql_query("SELECT * FROM identifier WHERE
resource_id = $required[0]") or die ("There is no older
version!");
        $old_identifier = mysql_fetch_row($query2);
        $query3 = mysql_query("SELECT * FROM coverage WHERE
resource_id = $required[0]") or die ("There is no older
version!");
        $old_coverage = mysql_fetch_row($query3);
        $query4 = mysql_query("SELECT * FROM classification WHERE
resource_id = $required[0]") or die ("There is no older
version!");
        $old_classifier = mysql_fetch_row($query4);
        $query5 = mysql_query("SELECT * FROM system_req WHERE
resource_id = $required[0]") or die ("There is no older
version!");
        $old_sys_req = mysql_fetch_row($query5);
        $query6 = mysql_query("SELECT * FROM annotation WHERE
resource_id = $required[0]") or die ("There is no older
version!");
        $old_annotation = mysql_fetch_row($query6);
        $query7 = mysql_query("SELECT * FROM resource_subject
WHERE resource_id = $required[0]") or die ("There is no older
version!");
```

```
        $old_re_subject = mysql_fetch_row($query7);
        $query8 = mysql_query("SELECT * FROM resource_author
WHERE resource_id = $required[0]") or die ("There is no older
version!");
        $old_re_author = mysql_fetch_row($query8);

        mysql_select_db("cindiold", $db)
            or die("unable to select database 'cindi':
".mysql_error());

 //insert related records of a resource, begin from the table
resource
        $ins_version1 = mysql_query ("insert into resource
values(null, $myArray1[0], '$myArray1[1]', '$myArray1[2]',
$myArray1[3],'$myArray1[4]', '$myArray1[5]', '$myArray1[6]',
'$myArray1[7]','$myArray1[8]', '$myArray1[9]', '$myArray1[10]',
'$myArray1[11]', $myArray1[12], '$myArray1[13]', '$myArray1[14]',
'$myArray1[15]',$myArray1[16], $myArray1[17])");
        $id = mysql_insert_id($db);
        $ins_version2 = mysql_query ("insert into identifier
values($myArray1[0], '$old_identifier[1]',
'$old_identifier[2]')");
        $ins_version3 = mysql_query ("insert into coverage
values($myArray1[0], '$old_coverage[1]', '$old_coverage[2]')");
        $ins_version4 = mysql_query ("insert into classification
values($old_classifier[0], '$old_classifier[1]',
'$old_classifier[2]')");
        $ins_version5 = mysql_query ("insert into system_req
values($old_sys_req[0], '$old_sys_req[1]', '$old_sys_req[2]')");
        $ins_version6 = mysql_query ("insert into annotation
values($old_annotation[1], '$old_annotation[2]',
'$old_annotation[3]')");
        $ins_version7 = mysql_query ("insert into
resource_subject values($old_re_subject[0],
$old_re_subject[1],$old_re_subject[2],$old_re_subject[3])");
        $ins_version8 = mysql_query ("insert into resource_author
values($old_re_author[0], $old_re_author[1],
'$old_re_author[2]')");
        $ins_version9 = mysql_query ("insert into
resource_authorname values($author_old[0], '$author_old[1]')");
     }

   mysql_select_db("cindi", $db)
      or die("unable to select database 'cindi':
".mysql_error());

//delete the releted records of a resource, the last one table
must be the table resource
   if ($required[0] != $paperid ) {
   $del_version9 = mysql_query ("delete from resource_authorname
where resource_id = $required[0]");
```

```
   $del_version8 = mysql_query ("delete from resource_author
where resource_id = $required[0]");
   $del_version7 = mysql_query ("delete from resource_subject
where resource_id = $required[0]");
   $del_version6 = mysql_query ("delete from annotation WHERE
resource_id = $required[0]");
   $del_version5 = mysql_query ("delete from system_req WHERE
resource_id = $required[0]");
   $del_version4 = mysql_query ("delete from classification WHERE
resource_id = $required[0]");
   $del_version3 = mysql_query ("delete from coverage WHERE
resource_id = $required[0]");
   $del_version2 = mysql_query ("delete from identifier WHERE
resource_id = $required[0]");

   $del_version1 = mysql_query ("delete from resource WHERE
resource_id = $required[0]");
   }
```

Appendix B: Application for temporary tables

Build a temporary table to merge the two tables regarding annotations given by authors and readers. (Section 4.5)

```
          include_once ("dbConnect.php");
           $temp_sql = "CREATE TEMPORARY TABLE tempAnnotation ";
           $temp_sql .= "select * from annotation ";
           $temp_query = mysql_query($temp_sql, $db);
           if (!$temp_query)
             error_message(mysql_error());
           if ($temp_query)
           {
            $tempSql = "select * from annotation_confsys";
            $tempSql_query = mysql_query($tempSql, $db);
            while ($temp_row = mysql_fetch_row($tempSql_query))
            {
              $temp_Sql = "insert into tempAnnotation values (";
              $temp_Sql .= "$temp_row[0], $temp_row[1],
$temp_row[2],";
              $temp_Sql .= " '$temp_row[3]')";
              $temp_Sql_query = mysql_query($temp_Sql, $db);
              if (!$temp_Sql_query)
                error_message(mysql_error());
            }
           }
          $result = mysql_query("SELECT * FROM tempAnnotation
   GROUP BY resource_id");
```

Appendix C: A shell script for dumping the database for backup

```
#!/bin/sh

# List all of the MySQL databases that will be backup here,
# each seperated by a space
databases="cindi"

# Directory where the backup files to be placed
backupdir=/home/cindi/cindi_dbbackup

# MySQL dump command
mysqldumpcmd=/usr/local/apache1327-mysql32352-php423-
openss196g/mysql3.23.52/bin/mysqldump

# MySQL Username and password
userpassword=" --user=username --password=password"

# MySQL dump options
dumpoptions=" --quick --no-create-db --no-create-info --add-
locks --extended-insert --lock-tables"

# Unix Commands
gzip=/bin/gzip
uuencode=/usr/bin/uuencode
mail=/bin/mail

# Create our backup directory if not already there
mkdir -p ${backupdir}
if [ ! -d ${backupdir} ]
then
   echo "Not a directory: ${backupdir}"
   exit 1
fi

# Dump all the listed databases
echo "Dumping MySQL Databases"
for database in $databases
do
   $mysqldumpcmd $userpassword $dumpoptions $database >
${backupdir}/${database}.sql
done

# Compress all of our backup files
echo "Compressing Dump Files"
for database in $databases
do
```

```
   rm -f ${backupdir}/${database}.sql.gz
   $gzip ${backupdir}/${database}.sql
done

# List what are done
ls -l ${backupdir}
echo "Dump Complete!"
exit
```

Appendix D: The queries regarding the subjects.

In the original version, the following query was used to achieve the records about sub-sub-subject.

```
$result_subject=mysql_query("select * from subject where
subject_id>=($cp_sub_subjectChoice1*1000) and
subject_id<($cp_sub_subjectChoice1*1000+1000)  order by
subject_name",$db);
```

Here, `$cp_sub_subjectChoice1` is the code of the sub-subject that is selected by users from the interfaces as shown in Figure 4.2, Figure 4.3 and Figure 4.4. For example, if the users selected the subject “Numerical analysis in mathematics of computing” with the code 1022, that is, `$cp_sub_subjectChoice1` is 1022, the query result was that only ten records were selected as follows:

```
select * from subject where subject_id>=1022*1000 and
subject_id<1022*1000+1000;

+-----------+------------------------------------------
|subject_id | subject_name
+-----------+------------------------------------------
|   1022000 | -----------------
|   1022001 | General
|   1022002 | Computer arithmetic
|   1022003 | Numerical analysis conditioning (and ill-
conditioning)
|   1022004 | Numerical error analysis
|   1022005 | Numerical analysis interval arithmetic
|   1022006 | Multiple precision arithmetic
|   1022007 | Numerical analysis algorithms
|   1022008 | Parallel numerical analysis algorithms
|   1022009 | Numerical analysis stability (and instability)
+----------+-----------------------------------------------
10 rows in set (0.00 sec)
```

In fact, since the left four digitals of a sub-sub-subject code is set to be the same as the sub-subject code, the query run against the table `subject` is modified as follows, which can achieve all the sub-sub-subjects with the same second level sub-subject.

```
$result_subject=mysql_query("select * from subject where
left(subject_id,4) = $cp_sub_subjectChoice1 and subject_id !=
$cp_sub_subjectChoice1 order by subject_name",$db);
```

Let `$cp_sub_subjectChoice1` be 1022, the query and its results can meet the requirement correctly.

```
select * from subject where left(subject_id,4)=1022 and
subject_id != 1022;
```

```
+------------+-----------------------------------------------
| subject_id | subject_name
+------------+-----------------------------------------------
|    1022000 | ------------------
|    1022001 | General
|    1022002 | Computer arithmetic
|    1022003 | Numerical analysis conditioning (and ill-
conditioning)
|    1022004 | Numerical error analysis
|    1022005 | Numerical analysis interval arithmetic
|    1022006 | Multiple precision arithmetic
|    1022007 | Numerical analysis algorithms
|    1022008 | Parallel numerical analysis algorithms
|    1022009 | Numerical analysis stability (and instability)
|   10220010 | Interpolation
|   10220011 | Extrapolation
|   10220012 | Interpolation formulas
|   10220013 | Smoothing interpolation
|   10220014 | Spline and piecewise polynomial interpolation
|   10220015 | Approximation
|   10220016 | Approximation of surfaces and contours
|   10220017 | Chebyshev approximation and theory
|   10220018 | Elementary function approximation
|   10220019 | Fast fourier transforms (fft) approximation
|   10220020 | Least squares approximation
|   10220021 | Linear approximation
|   10220022 | Minimax approximation and algorithms
|   10220023 | Nonlinear approximation
|    10220024 | Rational approximation
|    10220025 | Special function approximations
|    10220026 | Spline and piecewise polynomial approximation
|    10220027 | Wavelets and fractals approximation
|    10220028 | Numerical linear algebra
|    10220029 | Numerical linear algebra conditioning
|    10220030 | Numerical linear algebra eigenvalues and
eigenvectors
|    10220031 | Numerical linear algebra error analysis
|    10220032 | Numerical linear algebra systems (direct and
iterative methods)
|    10220033 | Numerical linear algebra matrix inversion
|    10220034 | Numerical linear algebra singular value
decomposition
|    10220035 | Numerical linear algebra sparse systems
|    10220036 | Numerical linear algebra structured systems
|    10220037 | Numerical linear algebra very large systems
|    10220038 | Quadrature and numerical differentiation
|    10220039 | Adaptive and iterative quadrature
|    10220040 | Automatic numerical differentiation
|    10220041 | Quadrature and numerical differentiation error
analysis
|    10220042 | Quadrature and numerical finite difference
methods
|    10220043 | Gaussian quadrature
|    10220044 | Quadrature and numerical differentiation
iterative methods
|    10220045 | Multidimensional (multiple) quadrature
|    10220046 | Roots of nonlinear equations
|    10220047 | Roots of nonlinear continuation (homotopy)
methods
|    10220048 | Roots of nonlinear convergence equations
|    10220049 | Roots of nonlinear error analysis
|    10220050 | Roots of nonlinear iterative methods
|    10220051 | Roots of nonlinear polynomials, methods for
|    10220052 | Systems of roots of nonlinear equations
|    10220053 | Optimization
|    10220054 | Constrained optimization
|    10220055 | Convex optimization programming
|    10220056 | Global optimization
|    10220057 | Gradient optimization methods
|    10220058 | Integer optimization programming
|    10220059 | Least squares optimization methods
|    10220060 | Linear optimization programming
|    10220061 | Nonlinear optimization programming
|    10220062 | Quadratic optimization programming methods
|    10220063 | Simulated optimization annealing
|    10220064 | Stochastic optimization programming
|    10220065 | Unconstrained optimization
|    10220066 | Ordinary differential equations
|    10220067 | Ordinary differential boundary value problems
|    10220068 | Ordinary differential chaotic systems
|   10220069 | Ordinary differential convergence and stability
|   10220070 | Ordinary differential-algebraic equations
|   10220071 | Ordinary differential error analysis
|   10220072 | Ordinary finite difference methods
|   10220073 | Ordinary differential initial value problems
|   10220074 | Ordinary differential multistep and multivalue
methods
|   10220075 | Ordinary differential one-step (single step)
methods
|   10220076 | Ordinary differential stiff equations
|   10220077 | Partial differential equations
|   10220078 | Partial differential equations: domain
decomposition methods
|   10220079 | Partial differential elliptic equations
|   10220080 | Partial differential equations: finite
difference methods
|   10220081 | Partial differential equations: finite element
methods
|   10220082 | Partial differential equations: finite volume
methods
|   10220083 | Partial differential hyperbolic equations
|   10220084 | Partial differential equations: inverse
problems
|   10220085 | Partial differential equations: iterative
solution techniques
|   10220086 | Partial differential equations: method of lines
|   10220087 | Partial differential equations: multigrid and
multilevel methods
|   10220088 | Partial differential parabolic equations
|   10220089 | Partial differential equations: spectral
methods
|   10220090 | Integral equations
|   10220091 | Integral delay equations
|   10220092 | Fredholm integral equations
|   10220093 | Integro-differential equations
|   10220094 | Volterra integral equations
|   10220095 | Applications
|   10220096 | Miscellaneous
+------------+-----------------------------------------
97 rows in set (0.00 sec)
```