Journal of Computer Science 1 (2): 157-163, 2005
ISSN 1549-3636

The Design and Implementation of Parallel Digital Library Management System

[1,2]Yang Yan, [1,2]Li Jianzhong and [2]Kan Zhongliang
[1]School of Computer Science and Technology, Harbin Institute of Technology
Harbin 150001, People's Republic of China
[2]School of Computer Science and Technology, Heilongjiang University
Harbin 150080, People's Republic of China

Abstract: As the number of documents in digital library grows, it becomes increasingly difficult to store, manage the large amount of documents and finding requested relevant documents by users. Parallel Digital Library management System (PDLS) is used for this purpose. PDLS is a general tool for building and managing digital libraries of all kinds of documents to meet user needs. It implements system management and query processing in parallel. The functions of PDLS include data collection, format standardization, information extraction, automatic classification, data loading, data maintenance, query processing, external data entering, personalized recommendation etc. This study introduces the design and implementation of PDLS.

Key words: Software Design and Implementation, Digital Library, Document, Parallel

INTRODUCTION

The amount of documents in digital library usually grows rapidly overtime. How to store, manage and search these documents within the digital library is an challenging problem. Documents in digital library are stored as semi-structured data, while in the traditional relational database it is stored as structured data. Relational database management system cannot manage semi-structured data efficiently and cannot satisfy the requirement of content-based text retrieval.

A lot of research works have been done about semi-structured data, such as data modeling, query language for text retrieval [1-3], index methods and text retrieval algorithms [4-6] and similarity search algorithms [7]. These research results have been used a lot in digital library systems. SSREADER Digital Library, the National Digital Library and WanFang Database are popular digital libraries in China. All the digital libraries classify the documents into several classes and support querying inside a given class. Metadata search and full-text search through a single keyword or expressions are both supported in these digital libraries. Other examples of digital libraries are Greenstone digital library [8-10], UC Berkeley Digital library, Tufts Digital library, ACM digital library, NCSTRL etc. Similar functions are supported in these digital libraries, such as metadata searching, full-text searching, documents classification and browsing. Greenstone digital library has a suite of software that provides management tools for creating and maintaining a digital library. Tufts Digital library is for the integration of collections that exist or may be developed in the future [11]. There is a system named Lore developed by Stanford. It is a database management system for managing semi-structured data [12]. The NCSTRL at Cornell University is a distributed technical report library developed by the ARPA-sponsored Computer Science Technical Report Project [13]. The NCSTRL collection is distributed among a set of interoperating servers operated by participating institutions. All of the digital libraries described above do not support the following functions: structure and content-based queries, automatic entries of external documents and parallel document processing.

The PDLS system described in this study has the following features. (1) Generalization. It is essentially a general document database management system. It can be used to build digital libraries for user needs and provides a suite of tools to maintain it. (2) Parallelism. PDLS uses a lot of processors to execute queries and manage documents, which improves both storage capacity and query efficiency. (3) Structure and content-based retrieval. Users can query inside a document for an element, e.g. a chapter of a book, which not only allows users to propose for a more accurate query, but also reduce the information transmission workload in networks. (4) Personalization. PDLS can query according to user's interest and recommend documents relevant to user. (5) Automatic external data entering. PDLS can combine with other search engines in finding and adding references automatically. (6) Multi-format supporting. DL collects a lot of document resources including books, journal papers, proceedings etc. and supports document

information retrieval for a lot of document formats. (7) DLSQL query. PDLS defines a query language like standard SQL, named DLSQL. By using DLSQL, users can program and do all the operations in PDLS. (8) Automatic document classification. It creates a classifier according to the sample documents loaded by the system manager and automatically classifies documents.

Architecture of PDLS: To meet the need of parallelization, we design the parallel and extendable architecture as shown in Fig.1, which is made up of three hierarchies: Client, Mediator and Server.

Client: There are two kinds of users, end user and system manager. Any user connected to the Web can access the web pages of PDLS through URL. These users are called end users. End users submit queries through the query interface and wait for the corresponding query results from the system. System manager accesses the system in local area network. Only the system manager authorizes system creation and maintenance. No matter the query operation submitted by the end users or maintenance operation submitted by the system manager, the Client accepts the operation, transmits it to DLSQL(which is the query language of PDLS), sends it to the Mediator and accepts the query results returned from the Server and displays them to the user.

Mediator: Mediator is in the middle layer of the PDLS architecture. It is the Web Server of the digital library. The Mediator accepts requests from Clients, analysis the requests, divides each request into sub-commands, creates the query execution plan, determines which Servers to execute these sub-commands, sends the commands to corresponding Servers and provides the results to Client. Mediator coordinates all the Servers to work together to execute any operation from the Client.

Server: Servers are composed of two kinds of processors, namely Query Processor and Data Collector. Each Query Processor is a node of a parallel computer. On one hand, it stores data in the digital library including the original documents, the index file and the metadata extracted. On the other hand, it executes queries on the stored data and returns the query results to end user or manager. Data Collector is simply named as a Collector. Collectors are personal computers on which the documents that need to be added in the digital library are stored. Collectors run on Windows operating system and are controlled by the Mediator. They collect documents, extract relevant information, create classifier, classify documents and load all the data into corresponding Query Processors.

Functions of PDLS: PDLS is a general system, which enables a person who has a lot of documents to use it to create a digital library. The system manager builds the digital library and defines the format, the metadata schema, and the classification schema of their documents. According to these definitions the manager builds the digital library and provides document retrieval service on it. People who query the digital library are end users. From this point of view, the functions of PDLS are composed of two parts, maintenance and query.

Maintenance: There are two interfaces for a manager to maintain the digital library: graphical interface and DLSQL interface. DLSQL is a language defined for operations in PDLS. All the operations either can be done in graphical interface or by DLSQL. The maintenance operation includes metadata definition, classification schema creation, information extraction, classifier creation, automatic document classification, data loading, metadata modification, document addition and deletion, Query Processor load balance coordination, classification schema maintenance, etc. Among them, metadata definition, classification schema creation and maintenance show the generalization of PDLS.

Metadata Definition: Metadata are data that describe the attributes of documents. PDLS allows managers to define their own metadata according to the kind of documents they own. PDLS defines a whole set of metadata, which includes all the metadata described in DublinCore and extends DublinCore to integrate other different applications. Through the manager interface, managers can select metadata that can describe their documents from the whole set of metadata. PDLS creates corresponding metadata schema in Query Processors and Data Collectors. The metadata extraction operation extracts only the metadata that the manager selected and the user queries these metadata.

Classification Schema Creation: A document usually belongs to a given class. All the classes of documents constitute the classification schema in a digital library. The classification schema is a tree structure. The root of the tree represents all documents in the system and the leaf nodes are the classes that have no subclasses, called the Smallest Class. Each node in the tree corresponds to a class in the classification schema and represents the documents set belonging to the class. The parent-child relation between nodes is an inclusion relation between classes. Each document belongs to a leaf node of the tree. Classification schema creation operation determines how many classes the documents in the system should be divided into and how many subclasses are included in a given class. An interactive interface is provided to managers, through which the managers can create the classification schema, which is stored by the Mediator. A user can specify the class in which the query should be executed. Browsing is dependant on the classification schema.

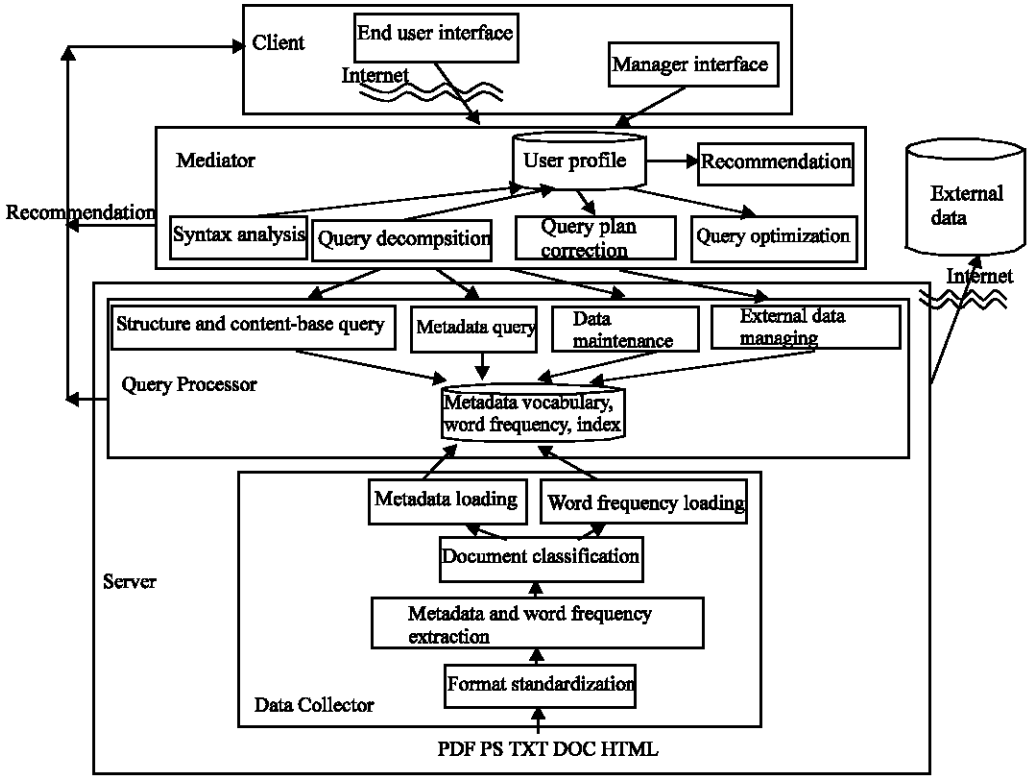

Fig. 1: Architecture of PDLS

Information Extraction: PDLS supports several document formats, including PDF, PS, DOC, HTML, TXT, etc. Information extraction operation analyzes different formats of documents, specifies the structure of a document, extracts metadata and words, and counts the frequency of each word. Metadata of a document includes title, author, keywords, publisher, publishing time, etc. Structure information of a document includes table of content, chapter, section, abstract, introduction, figure, table, references etc. The Collectors do information extraction. The extracted information is stored in the Collector and will be loaded into the Query Processors during the data loading process [14].

Data Loading: When building a digital library, huge amount of documents need to be added in. Data loading transmits these documents along with the metadata and other information produced by information extraction process from Collectors to corresponding Query Processors. The Mediator controls data loading process. It analyzes the amount of the documents in each class, determines how to distribute these data to Query Processors and stores them in each processor. The data stored in Query Processor is the data for a user query [15].

Classifier Creation: A document often belongs to a class in the classification schema. Determining which class each document belongs to manually is a boring operation. To create a classifier, a set of documents in each class is selected. Specialists manually specify the class of each sample document. The system extracts features of the sample documents in each class and stores them in Collectors as a classifier. Classifier can be used to classify documents automatically when documents are being entered into the digital library.

Automatic Documents Classifying: When a document is added to the digital library, a manager can specify the class of the document. If the manager does not specify the class, the classifier does it automatically. After extraction of metadata and word frequency in information extraction process, the classifier classifies the document according to the extracted information automatically [16].

Document Addition and Deletion: During the running stage, system manager can add or delete documents when needed. After the digital library is created, there are always new /old documents that need to be enrolled/deleted into/from the digital library. Document addition is different from data loading. Data loading is done at the initialization stage. Data loading needs to determine how to distribute the large amount of documents and store them in the Query Processors. Document addition is done at the running stage and can be added at any time. The amount of documents added each time is smaller. Document addition distributes data according to the distribution schema determined by data loading process. This operation need not redistribute data.

Metadata Modification: The extracted metadata is not accurate. The system provides two methods to modify the metadata then. The first one provided by the interactive interface during data loading and document addition process. When a document is added, the extracted metadata is displayed in the interactive interface. If it is wrong, the manager can modify them. Another method is done after the addition or loading process, whereby the manager queries for the documents whose metadata need to be modified and modifies them in batches.

Classification Schema Maintenance: Classification schema is created in the initialization stage. In the running stage, the amount of documents in the system increases successively, which may lead to some new classes in the system or the leaf node in the classification schema further divided into some subclasses. This might need to add some new class into the original classification schema. As the time pass, some classes of older documents can be deleted. Classification schema maintenance includes class addition, class deletion, and class name modification. On modifying the classification schema, the classifier should be modified and the data may be redistributed or loaded dependently.

Query Processor Load Balance Adjustment: When loading data, the data distribution schema is determined according to the classification schema and the amount of the documents belonging to a given class. One of the targets of data loading is the load balance problem among the Query Processors. The amount of documents in each class and the user hot spots changes a lot. Hence, the data should be redistributed to keep load balance of the Query Processors. Load balance adjustment model monitors the response time of each Query Processor periodically and redistributes the data appropriately [17].

External Documents Entering: One feature of PDLS is that references of a document can be found automatically. This involves two cases. When the reference is in the digital library, PDLS can find it and link the reference with the document automatically. When a user clicks on a hyperlink reference, the document corresponding to the reference can be opened automatically. When the reference is not in the digital library, PDLS can search for the document corresponding to the reference automatically through search engines and add it to the system if found.

Personalized Service: PDLS provides personalized service. User actions such as clicking, browsing and downloading are all recorded in the query logs, which are analyzed to capture the personalized information in user profiles. According to the user profile, personalized search returns documents that the user is really interested in and personalized recommendation recommends relevant documents also [18,19].

Query: Query interface is a Web page, which can be accessed through the Internet. Any user, no matter where he is, can access the main page of the digital library and submit requests. For different users, the system provides several different query interfaces, including browsing interface, simple query interface, complex query interface, structure and content-based interface, and DLSQL query interface. Among them, content and structure-based query and DLSQL query are special features of PDLS. Each query is sent to several Query Processors that store data relevant to the query and executed in parallel.

Browsing: Browsing is based on the classification schema. By browsing, users can click from the root node through a series of subclass nodes to a leaf node of the classification schema and finally find some relevant documents.

Simple Query: The characteristic of simple query interface is its ease of use. Users can submit a query just by filling in some keywords. The keywords are then send to some Query Processors that stores the documents in the classes the user selected.

Complex Query: Complex query allows users to input multiple query conditions using a combination of AND, OR and NOT. The attributes involved in the complex query are all metadata of documents, such as title, author and keywords. Simple query and complex query are both queries on metadata.

Structure and Content-based Query: Through the structure and content-based interface, users can submit a query about a part of a document. For example, in one scenario, a user might not finish reading a book at one time, while in another situation, he/she might want a specific chapter in a book having many chapters. A special part of a document, such as chapter, section, etc., is called an element of a document. If a document can be divided into several elements, then the system can only return the elements the user is interested in. Thus the amount of transmission is reduced and users can conveniently find a special element in reading.

DLSQL Query: To support programming, the system proposes a query language called DLSQL, which is like SQL. Through DLSQL interface, users can implement all the queries and the maintenance operations. DLSQL can be embedded in C to support programming.

Implementation of Key Techniques: Time and space cost are the most important parameters to evaluate a system, which are mainly determined by the storage structure of all the data and the query processing methods.

Storage Structure: Each Query Processor stores data in some given classes, which is shown in Fig. 2. Queries and maintenance operations are all based on these data [20].

Source file database is the set of books, papers and other documents, the format of which is PDF, PS, DOC, HTML etc. All the other data in digital library are extracted from the source files.

The HTML data contains some of the metadata and elements of source files, including title, authors, table of contents, abstract and references. The HTML data of a document is extracted during information extraction process and is stored in a HTML file, which is a short view of the document.

Each smallest class has a metadata table, a structure information table, a reference information table and a figure and table information table, as shown in the index and structured data in Fig. 2. Metadata includes title, author, keywords, publisher, etc. and are stored in the relational table. To reduce the response time of metadata query, several indices are built on it. The index includes author index, title index, keyword index, publishing time index and others if needed. Author index stores each author and the identifier of each document he published. Key word/title index stores each word in key word/title and the identifier

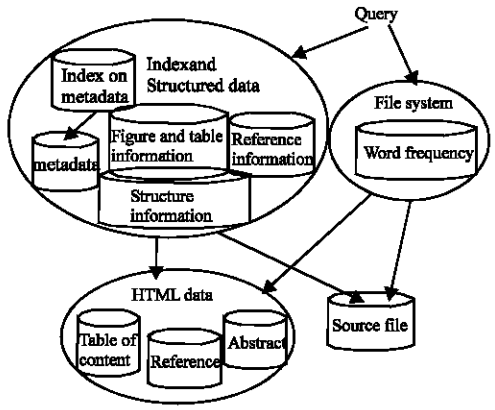

Fig. 2: Data in Digital Library

of the document corresponding to it. The author index, keywords index and the title index are implemented in an inverted index. The other indices can be implemented easily in a relational table. Structure information stored in relational table includes the document identifier, the identifier of element in the document, the title of the element, the hyperlink of the element, the maximal frequency, the beginning and the end page number of the element. The structure information is used to support structure and content-based query. The reference information stores referencing and referenced information among documents. This information is used to rank query results and support automatic reference linking. The figure and table information stores the title of a figure or a table in a document, and support figure-based and table-based query.

To support content-based query, word and word frequency should be extracted and stored at a reasonable structure. Each class in a processor has a vocabulary, which stores all the words in this class. Each word in the vocabulary corresponds to three inverted lists, which stores the element identifier, the position and frequency of the word, which appears in the element.

One inverted list, called big bucket, is used to store the element identifiers the word appears in and the frequency. The sequence of the data stored in an inverted list is a problem concern. In the big bucket, the identifiers of the elements sort the inverted list, which is convenient for multi-word query. For single word query, the second inverted list is built which stores the first N elements corresponding to the index word, which is sorted by the weight of the elements. This inverted list is called rank bucket. Different position a word appears in a document means different weight of the word. Obviously, a word in the title is more important than a word in the abstract, which is more important than a word in the body of a document. For this purpose, the third inverted list, called small bucket, is used to store the special position the word appears, such as title, keywords, etc. Each word in the vocabulary corresponds to three links. One points to the big bucket, one to the rank bucket and one to the small bucket.

The structure of the inverted index is shown in Fig. 3.

Sequential bucket is a backup of the word and frequency information. When a document is added to the digital library, its words and frequency are stored in the sequential bucket sorted by the document identifier. That is, the information in sequential bucket need not be sorted. Sequential bucket is used to maintain data in the system and to rebuild the inverted index when needed.

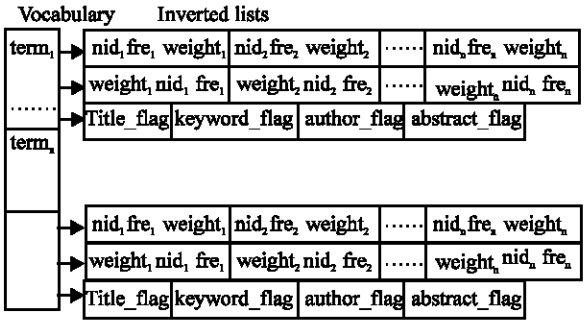

Fig. 3: Structure of Inverted Index

Query Processing: Query is the main function of a digital library. To execute query, the Client, the Mediator and the Query Processors should work together coordinated by the Mediator. Client receives the query request, transmits the request to DLSQL and sends it to the Mediator. Mediator fetches a query request from the request queue of all Clients and creates a process for the request. The process checks the syntax of the request, divides it into sub-requests, creates the query plan, optimizes the query plan, determines witch processors to execute the request and sends the sub-requests to these processors. Each Query Processor that received the sub-requests creates a corresponding process to execute the sub-requests. According to the request, the process determines how to execute the sub-requests and executes them. The results produced in each processor are merged to form the final results and returned to the user.

Each query is executed in parallel. The optimized query plan produced by the Mediator is the query plans on

some corresponding Query Processors. Query in each Query Processor may be a single word query or a multi-word query, may involve metadata or just a content-based query. The query optimization rules and the query execution methods in Query Processors are described below.

Query Optimization Rules

* Queries on structured data are executed first and the optimization is done by relational database;
* Different conditions about one word is combined into one query condition;
* A query that involves multiple conditions are converted into Disjunctive Normal Form;
* Save the query results of sub-requests to reuse them in a query;
* If a query involves only one word, the results of the query are sorted by weight of elements;
* If a query involves multiple words, the results of each sub-request are sorted by element identifier;
* The attributes that a user wants to see and the sorted attributes are reserved in the query result.

Query Processing Algorithm

Input: a user query request
Output: query results

* The query request is converted into DLSQL;
* DLSQL is checked for syntax and semantic errors;
* Divides DLSQL into optimized sequential sub-requests in different Query Processors;
* Each Query Processor executes query on itself in parallel:
  * If there are sub-queries on metadata or other structured data

  Executes them by RDBMS;
  * If the sub-query involves only one word and the position of the word is not pointed out in the query

  Executes the query in the rank bucket;
  * If the sub-query involves only one word and the position of the word is pointed out Executes the query in the small bucket;
  * Otherwise executes the query in big bucket;
  * Produces results list sorted by weight of elements;
  * Results are sent to merge processor;
* The results from each Query Processor are merged;
* Return results to user.

RESULTS AND DISCUSSION

PDLS is implemented on Linux and Windows operating systems. The Mediator and Query Processors are Linux and the Collectors are Windows. The programming languages used are standard C, VC, Pro C and Oracle.

Disk space and query response time are two key factors of PDLS. In PDLS, metadata, structure information, reference information, figure and table information are stored and managed by Oracle database system. Vocabulary, inverted index, sequential bucket are all managed by files system in the disk.

The number of different words in the set of documents determines the original size of the disk space occupied by inverted lists. A word occupies at least one block in the inverted list even if it is not full. With the amount of documents in the digital library increasing, each block of each word in the inverted list will become full. The number of different words increases when the number of documents increases. But when the document number reaches a given value, the number of different words becomes stable. The disk space of the inverted lists is about 30 percent of the size of the documents in the digital library. The size of small buckets is about 10 percent of the size of the big buckets. In fact, small buckets can answer a lot of user requests in digital library. The disk space of the various kinds of data in PDLS is shown in Table 1.

Table 1: Space Occupied by All the Data

Number of documents	1,342
Disk space of all the documents	1.33G
Number of different words in all the documents	115,230
Disk space of vocabulary	4.6 M
Disk space of big buckets	420 M
Disk space of small buckets	40 M
Disk space of sequential buckets	29 M
Disk space of metadata	53 K

Query response time is mainly determined by the time of disk I/O and data transmission. A buffer is used to store recently frequent accessed data. When the data needed is in the buffer, the response time will be greatly reduced. Increasing the size of the buffer or creating index for the frequently used data can improve the hit rate of the buffer. Hence the query response time is reduced. The documents in the Query Processors are distributed based on smallest class. The documents in the same smallest class are stored in one processor. The number of documents in a smallest class is usually not very large. When query is inside one smallest class, data transmission time is omitted. In the following experiments, there are 1342 documents in digital library. The query response time of different query words is shown in Table.2. The time is measured by second.

Table 2: Query Response Time (sec)

	The first query		The redundant query	
	CPU	Total	CPU	Total
Query words	Time	time	time	time
Parallel	0.04	1.13	0.01	0.73
Database	0.08	1.27	0.02	0.81
silicon	0.05	0.98	0.02	0.78
Parallel database	0.10	1.53	0.04	1.01
Algorithm design analyze	0.11	1.31	0.10	0.93

CONCLUSION

PDLS is a digital library management system. It can be used to build a document digital library corresponding to the features of the documents the manager owns. It provides maintenance functions including data collection, information extraction, documents classification, data loading, storage, etc. It also provides efficient query on metadata, structure and content of documents.

PDLS adopts parallel processing technique to solve the problem of large storage space and low query response time in digital library. A large amount of documents are distributed in various Query Processors. Increasing the number of Query Processors causes the increase of storage capacity. At the same time, each query is send to several processors to execute in parallel, which reduces the query response time.

ACKNOWLEDGEMENT

The work supported by the National Grand Fundamental Research 973 Program of China under Grant No. 1999.32704.

RERERENCES

1. Gonzalo Navarro and Ricardo Yates, 1997. Proximal nodes: A model to query document databases by content and structure. ACM Transactions on Information Systems, 15: 401-435.
2. Serge Abiteboul, 1997. Querying semi-structured data. Proceedings of the 6th International Conference on Database Theory.
3. Sihem AmerYahia, Chavdar Botev and Jayavel, 2004. TeXQuery: A fulltext search extension to Xquery. Proceedings of the 13th Conference on World Wide Web.
4. Brian, F., Cooper, Neal Sample, Michael J. Franklin, Gísli R. Hjaltason1 and Moshe Shadmon1, 2001. A fast index for semistructured data. Proceedings of the 27th VLDB Conference.
5. Dirk Bahle Hugh E. and Williams Justin Zobel, 2002. Efficient phrase querying with an auxiliary index. Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.
6. Dongwook Shin, Hyuncheol Jang and Honglan Jin, 1998. BUS: An effective indexing and retrieval scheme in structured documents. Proceedings of the third ACM Conference on Digital Libraries.
7. Christian Bohm, Bernhard Braunmuller, Hans-Peter Kriegel and Matthias Schubert, 2000. Efficient similarity search in digital libraries. IEEE Advances in Digital Libraries (ADL).
8. Ian, H., Witten, Rodger J. McNab, Stefan J. Boddie and David Bainbridge, 2000. Greenstone: A comprehensive open-source digital library software system. Proceedings of the 5th ACM Conference on Digital Libraries.
9. David Bainbridge, John Thompson and Ian H. Witten, 2003. Assembling and enriching digital library collections. Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL).
10. Ian, H., Witten, David Bainbridge and Stefan J. Boddie, 2001. Power to the people: End-user building of digital library collections. Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL).
11. Anoop Kumar, Ranjani Saigal, Robert Chavez and Nikolai Schwertner, 2004. Architecting an extensible Digital Repository. Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries (JCDL).
12. Jason McHugh, Serge Abiteboul, Roy Goldman, Dallan Quass and Jennifer Widom, 1997. Lore: A database management system for semistructured data. SIGMOD Record, 26: 54-66.
13. Naomi Dushay, James C. French and Carl Lagoze, 1999. A characterization study of NCSTRL distributed searching. Technical Report: TR99-1725, Cornell University Ithaca, NY, USA.
14. Li Guilin, Li Jianzhong and Yang Yan, 2003. Information extraction from PDF using plug-in. Comp. Appl., 23: 110-112.
15. Gu Xianrui, Li Jianzhong and Yang Yan, 2002. Parallel document data loading algorithm in digital library. Comp. Sci., 29: 104-106.
16. Ren Meirui, Li Jianzhong and Yang Yan, 2002. The implementation of an automated text categorization system based on Naïve Bayes method. Comp. Sci., 29: 285-287.
17. Yang Yan and Li Jianzhong, 2003. Cluster-based data allocation method of web servers in digital library. Comp. Eng. Appl., 39: 38-41.
18. Yang yan and Li Jianzhong, 2005. Interest-based recommendation in digital library. J. Comp. Sci., 1: 40-46.
19. Yang yan and Li Jianzhong, 2004. Topology-based paper recommendation in digital library. J. Comp. Res. Develop. (In press).
20. Yan Yang, Baoliang Liu and Zhaogong zhang, 2003. Partition based hierarchical index for text retrieval. In proceedings of the 4th International Conference of Web-Age Information Management.