Department of Electronic Engineering

PROJECT REPORT

BScIT-2001/02-(CS/HWC)-(03)

< A New City University Search Engine >

Student Name: Ng Zhun Wai
Student ID: 50183860
Supervisor: Dr CHUN, Andy H W
Assessor: Dr YUEN, Kelvin S Y

Bachelor of Science (Honours) in
Information Technology

Abstract

Imagine that you are going to collect information for some purpose. For example, what are the latest films on show? At the first sense, you may go to the front of a computer and search for films. It is a very easy job when you are familiar with the site or it is a small scale one. However, if you are not familiar with the site or it is very large, you may need help from a search engine. The time spends on searching, or efficiency of searching is highly depends on whether the search engine is good or not. For a good search engine, very desired results are output in the most beginning part of the results. From my research, the existing City University of Hong Kong search engine is not regarded as a good one from its students. Base on this point, I am trying to build a new City University of Hong Kong search engine.

The application was developed by using a system approach where analysis, design and development were carried out step by step. This search engine base on the technology of public search engine and is built specific for the structure City University of Hong Kong. So that, results from searching are relevant.

This report will be divided into two sections. Section A is a project overview. In this section, a project description will be made. Besides, there will be some relevant information about the project that should be known before going into Section B.

In section B, there is application documentation. In this part, there will be a development life cycle of the project. All information in this part are supported by the theories or technologies learnt in the past three years or those reference books stated in Section A.

The most difficult part in this project is the research part of search technique and implementation. For example, web-fetching technique called web crawling may not be difficult to understand but is difficult to implement. The time spent is long in these parts.

Acknowledgement

I would like to express my great gratitude to Dr. Andy Chun, my supervisor, for his kind advice on the project and precious information. He also made me feel confident in the Final Year Project presentation.

I would also like to express my sincere appreciation to Colt, my friend studying CS in City University of Hong Kong. He gives me lots of innovative and useful ideas on web technologies.

Last but not least, I would like to give my compliment to Mr. Vincent Mok, Laboratory Technician of the City University of Hong Kong for helping me overcome some technical problem during this project.

Ng Zhun Wai
20th March 2002

Table of Contents

SECTION A

PROJECT OVERVIEW

Introduction

Numerous people access City University of Hong Kong web site everyday. These people include students, staffs, professors of City University of Hong Kong or other people who are interested in City University of Hong Kong. City University of Hong Kong web site has more than 25000 pages. In order to search information from this large scale web site, a good search engine is essential.

Being an IT student in City University of Hong Kong, I would like to try my best to develop a new search engine for my University.

CHAPTER 1
PROJECT DESCRIPTION

1.1 Terms of Reference

Background

Numerous people access City University of Hong Kong (City University) web site everyday. These people include students, staffs, professors of City University or other people who are interested in City University. City University web site has more than 25000 pages.

The existing City University web search engine is AltaVista Search Engine 3.0. Although it is a popular search engine, it has the following main weak points:

- **Many outdated links**. I searched for some keywords; I discovered that many links of the search results are removed.
- **Not accurate results.** The results of search are not quite accurate. For example, search for ‘ELC’, first result is ELC STAFF DIRECTORY, but not the index page of ELC.

If the results of search are not good enough, people can’t get relevant information by search engine. If the user is an experienced user, e.g. students of City University of Hong Kong, he may find what he wants without difficulties. However, if the user is not clear about the structure of City University web site, he may spend a long time when he wants to find information from City University web pages. Therefore, in order to search information from this large scale web site, a good search engine is essential.

Reasons for Selecting this Problem

As we can see from above section, City University’s search engine has many outdated links and has not quite accurate results. If I am required to search in City University web, I prefer to use other public search engine to search in the domain of City University.

To solve this problem, a possible way is building a new search engine, which can have more accurate search results.
Besides, this existing City University search engine is powered by AltaVista. I would like to develop City University's own search engine.

Objectives

The objective of this project is to develop a new web search engine for City University. This search engine is named **New CityU Search Engine**. The characteristics of New CityU Search Engine should maintain the existing good features and improving:

- **Updates pages information in database frequently**. This can ensure that pages information in database is up-to-date information that we can see on the pages.
- **Delete outdated links**. If a page is outdated, the information for the page should be removed as soon as possible. So that, search results will not include that outdated page.
- **Search specific for City University web page**. Develop this search engine base on the structure of City University web page. So that, search results are more accurate.

Assumptions

The following issues are assumed in this project:

- **English search ONLY**. All the processes of search engine are tailor-made in English only. This project does not care other language. The results of other language are unexpected. However, use of other languages should not lead to system error.

Scope of the project

The following things will be provided:

- **Technical Review.** Review areas will be structure of existing City University web search engine, components of search engine and

methodology of search engine.

- **System Development**. This identifies the main activities of the project. Database server, web server, search engine java program construction.
- **Documentation**. It is the final report. Technical document such as DFD, ERD will be included.

The following things will NOT be provided:

- **Real host establishment**. During to security reasons of Electronic Engineering Department, my web server is not provided with a real IP address. In turns, although my web server has been set up, public users are not able access my web server for searching through Internet. Actually, I can still perform searching and test my results of search in local host. Without real IP address is not a matter indeed.
- **Security module**. Security module will not be included in the development of the system, as the security module is provided by database server and java packages. Their security protocols are enough when apply to this system. Since there are no transactions, very secure system is required, involved in this system. Also, firewall packages can easily be brought from outside.
- **User interface design**. User interface is not the focus of this project. Simple but clear user interfaces are adopted.

Challenges of The Project

Actually, this is a challenging work to build this search engine. First, City University of Hong Kong web site is very large, at least 25000 pages. Performance should be considered for searching query. Furthermore, search engine for such large scale web site, automatic and systematic database insertion method is require. It is not possible to create database for search manually. Second, for accurate results in search, very smart search method is required. It is also not an easy task. Finally, in order to extract content words from pages in different file format (html, cgi, asp), the automatic and systematic database insertion method should be able to support multi file format.

Furthermore, I have tried to ask for characteristics, which are related to search

methodologies, of City University web site. However, I was rejected. Without this information, the structure of City University is just by my own observation. Some characteristic that may favor or not favor to search may missed. The search results may be badly affected.

Constraints

There are constraints existed to limit the project development:

- **Human resources**. This project is an individual work. In some famous search engines, the system are developed and maintained by an engineer term. For instance, Google, one of the most popular search engines in the world, have dozens of experienced engineers working to maintain and improve performance. It is very difficult to develop a search engine is competitive with those search engine by only one person.
- **Hardware**. The search time is very short for famous search engines. Most of them can output search results within 0.5s. They have such great performance is due to their very high performance computing processors. For example, in Google, there is thousands of low cost PC's networked together to create a super fast search engine.
- **Time for researching structure of City University web site**. The success of famous search engine is attributed to the efficiency of search algorithm, hardware performance and also research teams contribution. Research team investigates the structure of sites, and in turns, improves search accuracy. This project’s develop time is so tight, about half a year, I do not enough have time to study structure of City University thoroughly. This may affect search accuracy directly, as my search engine not only depends on search algorithm, but also structure of City University web page.

Acceptance Criteria

From the constraint part, you can understand that it is nearly impossible to develop a search engine, which is competitive with famous public search engines, in this short time and limited resource.
One the day of completion of the project, all normal features of a search engine

must be provided. They are module of automatic insert contents of City University web into database, module of query search.

Improvement jobs may be done depending on available time.

1.2 Development Plan and Schedule

Major Technical Activities

The development will follow a typical development life cycle format.

System Analysis. An analysis of search engine methodologies and City University of Hong Kong web structure.

System Design. Data collection for system design. Database structure design. Preparation of function specification.

Systems Development. Database server, web server, and three modules of systems will be developed in sequence. The three modules are basic module, spider module and query module.

Systems Testing. Final systems testing and acceptance testing will be performed.

Also, documentation will be included. It is the final report including the application documentation.

Resource Required

There are some resources required to develop, test and demonstrate the project.

- **Database server**. A database is required to store information of City University of Hong Kong web pages. Database server like MySQL, SQL Server 2000 and Oracle can be configured.
- **Web server**. A web server is required serve search engine over the Internet using the Hyper Text Markup Language (HTML). The Web server accepts requests from browsers like Netscape and Internet Explorer and then returns the appropriate search pages.

Original project planning time chart

Stage	Period	Description
1	Sem A wk 4 - Sem A wk 8	Studying search engine technology, and structure of CityU web site. Decide work approach.

2	Sem A wk 9 - Sem A wk 11	Setup the server and database for my search engine. Decide work approach.
3	Sem A wk 12 - Sem B wk 3	Writing Web crawler java program package
4	Sem B wk 4 - Sem B wk 6	Writing Searching program
5	Sem B wk 7 - Sem B wk 8	Implementation and evaluation.
5	Sem B wk 9 - Sem B wk 11	Final conclusion and final report.

Production Breakdown

There will be products produced throughout the development process, they are:

- **Requirement specification**. An analysis of a search engine methodology. A brief description of the function provided in the final system.
- **Function specification**. A documented database structure and ER-Diagram. Data Flow Diagram will also be produced. Also, some kind of system structure diagram or function catalogue is included.
- **Progress report**. Progress report will be submitted to supervisor, to ensure the project is on schedule. It provides a framework for development and to make a more clear explanation of the project idea.
- **Final report**. Final report will be divided into two sections. The main section will consist of the documented application system, including a description of the system as well as technical documentation. The second section will explain the major design / idea of the project. Technical review will also be included

CHAPTER 2

TECHNICAL REVIEW

In this chapter, a brief understanding on the concepts applied in the development of the project is given.

In Section 2.1, some structures of City University web site are given and what my search engine can do in order to favor search results. In Section 2.2, there will be search engine methodologies. They are the major functions composing a search engine.

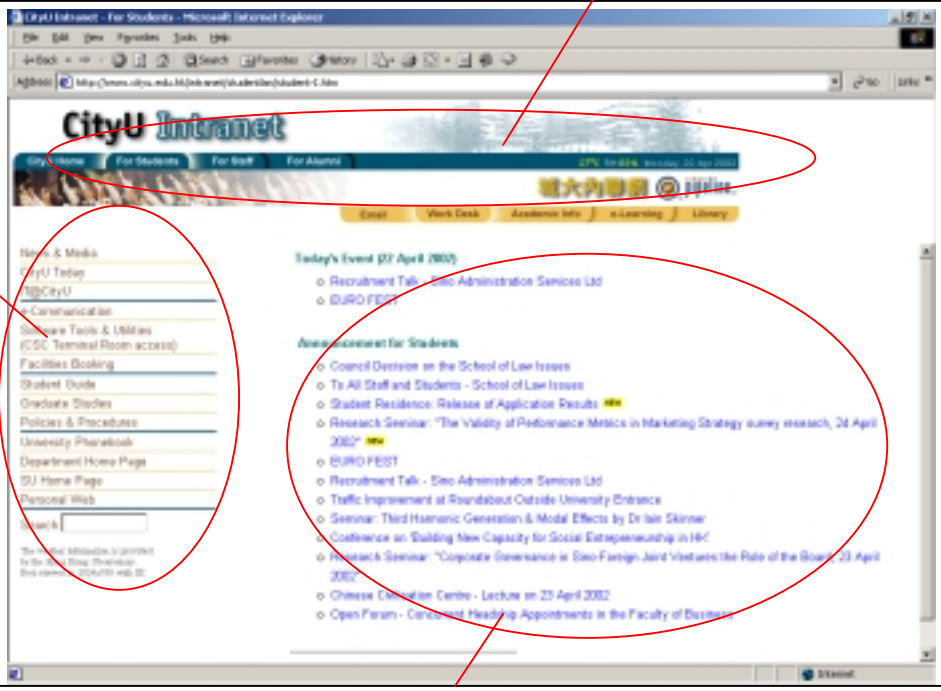
2.1 Structures of City University Web Site

Here are some structures that are related to this project, they should be considered in the development process. Frankly speaking, this search engine will be less accurate without considering these structures.

- **Keywords are used in url**. In the web of City University, keywords for that page can usually be found in the url.
  e.g. www.cciv.cityu.edu.hk , ***cciv*** is a keyword, since the contents in this page are related to issues about cciv.
  www.cityu.edu.hk/intranet , ***intranet*** is a keyword, since the contents in this page are related to City University's intranet. So **the search should relies more on the url.**
- **Frequently use frames and standard templates**. Frames and standard templates are used in City University's web pages.
  e.g. www.cityu.edu.hk/intranet/studentlan/student-I.htm, the standard template (layout format) of this master page (also called frameset page) is blended together by below three frames:

http://www.cityu.edu.hk/intranet/studentlan/header-stud.htm
This frame is the header frame. It contains the links of main pages, like the main page of Academic Information, library, etc.

http://www.cityu.edu.hk/intranet/studentlan/workdesk.htm
This frame is the index of intranet.



http://www.cityu.edu.hk/intranet/studentlan/stdumsg.htm
It is the content of this page.

Figure 2.1 Frameset: ww.cityu.edu.hk/intranet/studentlan/student-I.htm

In this example, the source code of the master page, www.cityu.edu.hk/intranet/studentlan/student-I.htm only contains the information of the three embedded frames besides url, title tag and meta tag. It does not contains any information about the contents of this page, as the contents are contained in the frame of http://www.cityu.edu.hk/intranet/studentlan/stdumsg.htm . In this case, the contents of this page can only determined by url, title tag and meta tag. These three parts are very important for describing what contents does this page contain. (The method that search engine get contents of a page is called spider, or web crawler, this technology will be explained in below section.) Therefore, search should rely heavily on **url, title tag and meta tag.**

- **Rare use of Meta Tag for describing contents or keywords**. In my observe, less than 5% of City University pages use Meta tag for describing

the keywords or contents in that page. **Search should rely heavily on url, title tag of that page.**

Sum up from above three points, the **contents of pages are most rely on url, title tag, then meta tag, and final body tag**. Since body tag is not mentioned above, so it is not as important as that three tags mentioned for high degree of importance. But it is important when pages show similar contents with url, title tag and meta tag. In turns, search algorithm is also base on this conclusion.

- **Do not use robots exclusion protocol to prevent being indexed by web crawler**. Robots exclusion protocol and web crawler will be explained in Section 2.2. The only thing to know here is that if there is robots exclusion protocol in City University web, I cannot get the contents of the pages automatically and this project cannot be developed.

2.2 Search Engine Methodologies

2.2.1 Overview

Generally speaking, search engine can be divided into two major modules. They are **Spider module** and **Query module**.

Spider module is the internal module of search that is only occupied by administrator. It part is hidden from external users, like students. Search engine searches base on the pages descriptions inside the database. It returns the pages in which the contents are relevant to the query inputted from external user. Actually, the information in database used for search is obtained and is inserted into database by Spider module. The functions will be described below: web crawler, string tokenization, inverted index and ranking, are functions of this module.

Query module is the external module that gets queries from external users, performs search action and finally returns output as search results back to user. The search function will be described below belongs to this module.

2.2.2 Robots Exclusion

Sometimes people find their web sites have been indexed (traveled within

their sites and obtained contents) by an indexing robot (i.e. web crawler, explained in section 2.2.3), or that a resource discovery robot has visited part of a site that for some reason shouldn't be visited by robots.
In recognition of this problem, many Web Robots offer facilities for Web site administrators and content providers to limit what the robot does. This is achieved through two mechanisms:

- **The Robots Exclusion Protocol.** A Web site administrator can indicate which parts of the site should not be visited by a robot, by providing a specially formatted file on their site, in http://.../robots.txt. For example: (actually, the "/robots.txt" file usually contains a record looking like this)

```
User-agent: *
Disallow: /cgi-bin/
Disallow: /tmp/
Disallow: /~andy/
```

In this example, three directories are excluded (disallow).
Note that you need a separate "Disallow" line for every URL prefix you want to exclude -- you cannot say "Disallow: /cgi-bin/ /tmp/". Also, you may not have blank lines in a record, as they are used to delimit multiple records.
Note also that regular expression is **not** supported in either the User-agent or Disallow lines. The '*' in the User-agent field is a special value meaning "any robot". Specifically, you cannot have lines like "Disallow: /tmp/*" or "Disallow: *.gif".

- **The Robots META tag** . A Web author can indicate if a page may or may not be indexed, or analyzed for links, through the use of a special HTML META tag. For example,
<META NAME="ROBOTS" CONTENT="NOINDEX, NOFOLLOW">
A robot should neither index this document, nor analyze it for links.

2.2.3 Web Crawler

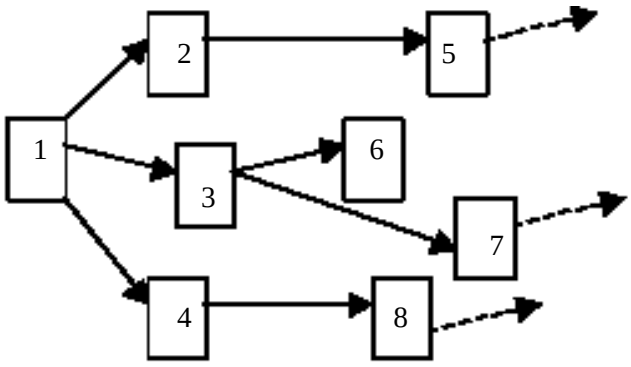
A web crawler is a program that systematically *fetches web pages*. Web crawlers are also known as ants, automatic indexers, bots, spiders, Web

robots, and worms [HREF1]. It can create an index of available Web pages is through an automatic Web page discovery and retrieval system. It is an application that “automatically traverses the Web's hypertext structure by retrieving a document, and recursively retrieving all documents that are referenced”.

An application that automatically traverses the Web might work as follows:

- A base set of known working hyperlinks is used as a starting point.
- Each of these hyperlinks is placed in a queue. The Web crawler will endeavor to retrieve each of the pages in this queue. Because the Web is a hyperlinked environment, each of these pages will in all likelihood have links to various other pages on various other Web servers.
- The Web crawler retrieves the first page in its queue and adds an entry for this page in the index.
- Then, it adds each of the hyperlinks that exist on that Web page to the queue.
- It repeats this process of retrieval and recording for each page in its queue.

Thus, for each page discovered, the list of pages that remain to be gathered will (ideally) grow, and so will the index. Indeed, it becomes evident that a Web crawler’s indexing process is one that builds upon itself: the more pages it retrieves, the more pages it discovers that need to be added to its index.



The sequence of crawler is 1,2,3,….,8….,
until the last document of the site.

Figure 2.2 Web crawler example

In this figure, a web crawler is discovering 7 pages from 1 initial page, moving left to right. The first three steps are depicted, the three dashed arrows at the end all point to further pages with more hyperlinks.

This process of automatic page discovery takes a great burden off of a search engines maintenance team. Rather than having to rely on announcements and user-submissions to grow the index, the indexing application is constantly and automatically discovering new pages for inclusion in the index. Day-to-day maintenance of the index is aided greatly by the use of such an algorithm, but skilled programmers are still required to maintain and update the actual indexing application, as opposed to manual indexing where the human labor is focused on directly maintaining the index.

To be effective, the implementation of a Web-traversal algorithm needs to be done at a rather large scale. Even with an automated system such as this, indexing a significant portion of the Web is a gigantic task. Imagine the indexing application were able to discover 1 million new pages a day (about 42,000 an hour or 700 per minute– possibly a generous estimate); it would take a bit over 2 years to index a static, non-changing body of 800,000,000 pages. Given that the turnover/change rate of the Web is significant–recall it is estimated to be around 40%. Many commercial search engines utilize variations on this approach to indexing.

Such a demanding application also requires a fair amount of data transmission bandwidth simply for the indexing of Web pages. The implementation of a scaleable and effective manual or automatic indexing scheme will almost certainly take on a large scale. There is an alternative: a scheme which makes no use of a persistent, local index.

In this project, the robots exclusion protocol is not provided. Web crawler can get contents (from now on, called index) from City University web site without block.

2.2.4 String Tokenization

Divide a string of characters into linguistically salient units. In java, there is build in string tokenizer class that allows an application to break a string into tokens. For example, string “City University of Hong Kong” can be broken into:
City, University, of, Hong and Kong five token, i.e. five single words.

With this string tokenization technique, the pages indexed by web crawler can be partitioned into tokens. This method is essential for the creating inverted index and then for ranking.

2.2.5 Inverted Index

A sequence of (key, pointer) pairs where each pointer points to a record in a database which contains the key value in some particular field. The index is sorted on the key values to allow rapid searching for a particular key value, using e.g. binary search. The index is "inverted" in the sense that the key value is used to find the record rather than the other way round. For databases in which the records may be searched based on more than one field, multiple indices may be created that are sorted on those keys.
An index may contain gaps to allow for new entries to be added in the correct sort order without always requiring the following entries to be shifted out of the way [HERF2].

2.2.6 Ranking

Ranking is the theory that calculates importance of each word in a page. It is the theory that keyword search based on.
Theory of ranking**:**
The following rules will affect the rank number of a keyword in a (importance of each rule is ordered in descending order):
Whether the

1. file name include the keyword.
2. title tag include the keyword.
3. meta tag include the keyword.
4. keyword exist in the first paragraph.

5. keyword exist in the last paragraph.
6. keyword exist in the body paragraph.

The above rule can be expressed in a term *prominence,* that is, how close to the beginning of the sentence the keyword is found.
Besides prominence, the *frequency* of the keyword appear in the page is also very important. [MARC]
e.g. a page with the keyword appear 10 times (page A) may get higher rank(rank number) than a page with the keyword only appear 4 times(page B). However, it is not always true when prominence of page B is much better than page A.
Frequency of keyword appear in a page is not fair when the size (no of words) of pages a not equal. For example, a page with 10000 words is probably having higher frequency than a page with only 1000 words. In turn, we have a better theory replace just counting the frequency. It is called *weight*.
Weight: = No. of keyword (frequency)/ No. of total words in the page.
Prominence and weight are 2 major factors for ranking a page.
The **output** of ranking is a list of keywords and each has the following information:

- this **keyword**
- **url** that this keyword is found
- **rank number** indicating the importance of this keyword in this url (document).

This information is used for keyword search.

2.2.7 Search Methods

- **Keyword search**. Keyword search is the most common and also the fundamental method of search. Ranking of a keyword is base on the idea of keyword search, since the concept of keyword search is determining which documents are most relevant to the query (keywords) and ranking is aimed at this determination.
  e.g. search for keyword - sport, the search engine will search for this word “sport” and output the results (url, descriptions) ordered

by the rank number in descending order. As mentioned above, the greater the rank number, the more important the keyword in that url (document).

In the case of multi-keywords search, e.g. search for - sport complex, keyword search perform **simple AND** search. That is, results of search are the url contains this two keywords and ordered by the sum of rank numbers of these two words in descending order.

- **Phrase search**. Search method is similar to keyword search, but for the exact phrase, exact word combination and position, inputted from the user.

  e.g. search -"City University", search for document with "City" and "University" similar to keyword search, moreover keywords "City" and "University" must be in consequent order, just as the input query.

- **Boolean search**. This search method is also similar to keyword search. Search is also base on keywords but and simple AND only. **And, OR and NOT keyword search** are performed.

  e.g. search – test OR examination NOT eec, urls and descriptions with keywords "test" or "examination" and do not have "eec" are returned.

- **Wildcard**. This search method is also similar to keyword search whereas character asterisk (*) can be used as don't care character.

  e.g. search – c*y, keywords city, candy, century and etc. are searched.

CHAPTER 3

SUMMARY AND CONCLUSION

Necessity of develop a new City University search engine. The existing City University search engine is not good enough as explained in background, in Section 1.1. Many outdated links, not so accurate results of it are the reasons for developing a new City University search engine.

It is necessary to develop a New CityU Search Engine.

Feasibility to develop new City University search engine. Building a new City University search engine is feasible. However, it is not easy to build one search engine better than existing one. Since the exist one, AltaVista, is a famous public search engine, it was developed and maintained by a group of experts. It is a **challenging work** indeed.

In this project, I am trying to develop a new search engine, which can replace the existing one.

The **base functions** to be provided are web crawling, words extracting, create inverted index, ranking, searching.

SECTION B

APPLICATION DOCUMENTATION

CHAPTER 4

SYSTEM ANALYSIS

4.1 Requirements Definition

In this section, requirements definition is given.

4.1.1 Problems Definition

As mentioned above, New CityU Search Engine is required to maintain the normal features of existing search engine, and furthermore, enhance the problems of many links outdated, and give more accurate results.

The functional requirements and non-functional requirements are given in section 4.1.2.and 4.1.3 respectively.

4.1.2 Functional Requirements

Function requirements mean the physical modules, which are going to be produced in the proposed systems. The functional requirements of the proposed system are described below:

- **Web crawling**. The system will provide web crawler. It can create an index of available City University web pages. It takes the index page of City University, i.e. www.cityu.edu.hk, as a starting point. After it starts, it get texts (contents) from each web page it reaches and also extracts the URLs of the hyperlinks in each page and put them into queue. This queue provides the web crawler with the URLs that it has to visit.
- **Words Extracting**. The system will provide string tokenizer. After web crawler have indexed the pages. The texts of each page have to be extracted from that page, if not, they are just rubbish mixing with HTML. String tokenizer method is used to extract content words from the HTML, ASP, or etc. pages of City University.

  e.g. <html><title>hello world</title><body>Hello World! This is my demo for words extracting. </body></html>

  The output will be:

Title – hello world

Body - Hello World! This is my demo for words extracting. Obviously, the actual file in City University web site is much more complex. It function is not easy to implement.

- **Create inverted index**. The system will provide a function to create inverted index. Using the extracted words from the string tokenizer, create inverted index for searching. Inverted index is created with three main attributes:
  a. Word
  b. URL. URL of word is found.
  c. Rank. Rank of word in this URL.
- **Ranking**. The system will provide a rank calculating function. The rank of a word is base on prominence and frequency, as explained before.
- **Search Query Analyzing**. Determine the search method used by the external user. Search function is based on this search method.
- **Search**. The system will provide search function. The external users, who pass query to search through this search engine, is provided four type of search method. They are:
  a. keyword search – search for keywords, like “academic” “skills”.
  b. phrase search – search for exact phrase, like “academic skills”.
  c. wildcard search – search with don’t care character, like “int*net”.
  d. Boolean search – search with Boolean operators AND, OR, NOT.
- **Search Results Ordering**. The search results from **search** function are not in order. This function can reorder the results by ranking number in descending order, i.e. the most relevant results are ordered at the most beginning.

4.1.3 Non-Function Requirements

Non-Function requirements mean the characteristics that are not related to system’s physical functions but are the characteristics that favorite to the

physical functions.

- **User Friendly**. The system should be easy to use, i.e. user friendly, for both administrator and external users. This can be done for providing function descriptions.
- **Secure**. The system should be secure. If not, hackers can access the database and undergo destruction.

4.2 Target Users

In this section, the target users of this project are defined and characteristics specified for the target users.

4.2.1 Internal Users

Internal users are the administrators of this system. Their jobs are monitoring the web crawling process, insert database actions and they can also view the queries from user. That is, they are response for monitor all internal process that web server and database server performed.

4.2.2 External Users

External users are people using this search engine to search for information in the City University web site. Most probably, they are City University student, professors, assistant or staff. Sometimes, this user may be a person who is not directly relate to City University. For example, a senior secondary student.

4.2.3 Characteristics Specified for target users

In order to make a search engine specific for City University, this project can make some adjustments, i.e. tailor made, for target users.

Easy to use admin tools. All the theories and technologies are hidden from administrator, so that administrator is not required to have much knowledge about java or search engine.

For instance, administrator is no need to know how to perform web crawler, no need to know how to create inverted index.

Also, simple but clear administration interface is provided; administrator can use the functions like crawling City University web easily by just pushing a button on the user interface.

Adopt simple but clear query format. Person use this search engine may not have a knowledge of search engine. Simple but clear query format is adopted.

For example, the Boolean search function do not acquire user to input brackets, ‘(‘and ‘)’, just simply put in keywords in the given text fields following the instructions.

4.3 Requirement Specification

4.3.1 Required Logical Data Flow Diagram

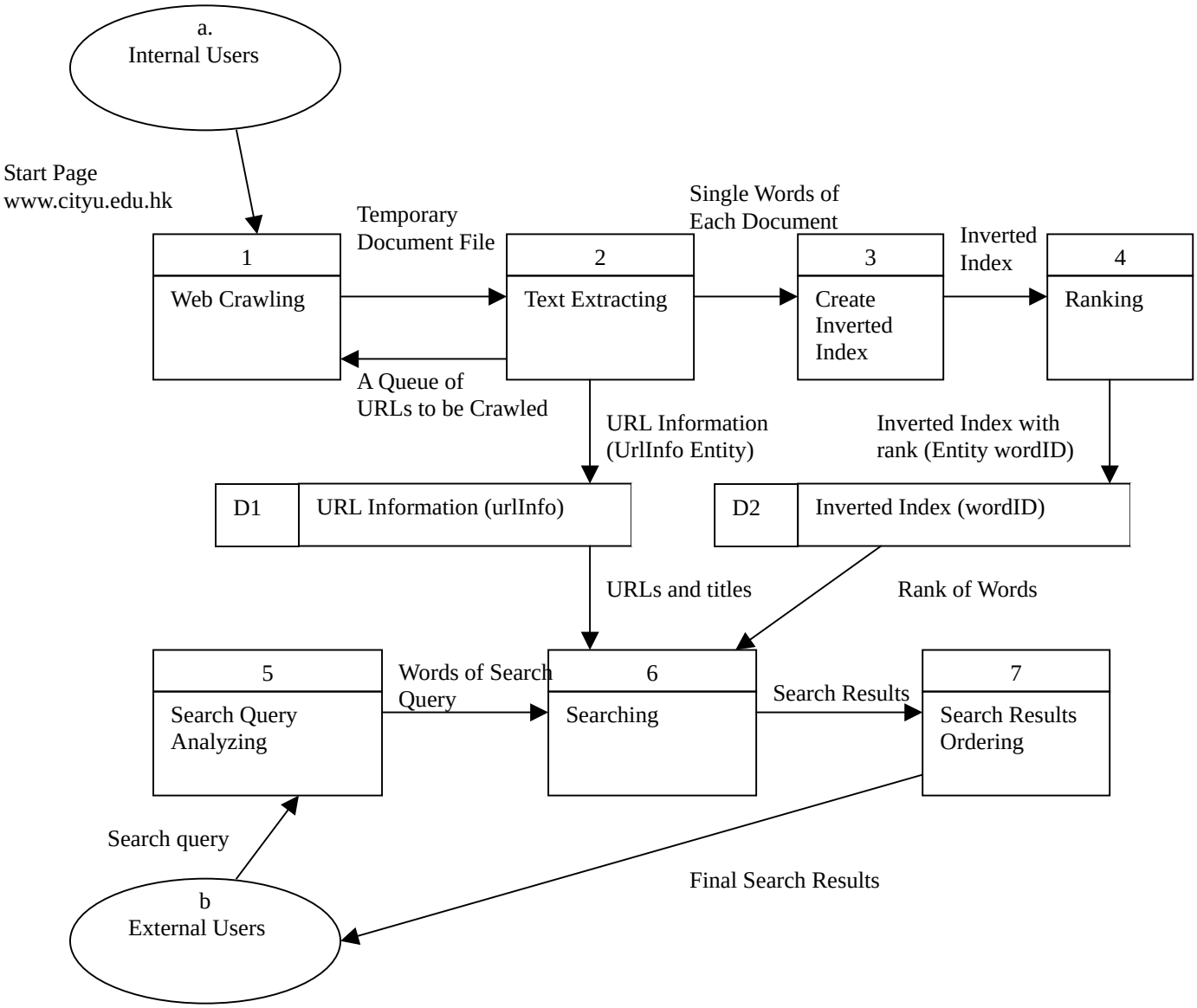


Figure 4.1 Required Logical Data Flow Diagram

The overview of the logical data flow model is drawn as above (figure 4.1). It show all the functions / processes required for the proposed system. In this section, only the system overview is provided. Detail of functions will be explained in Physical Design. There are totally 2 external entitles, 2 data stores and 7 processes in the logical data flow diagram. Detail description of these processes will be given in the following section.

4.3.2 Required Function Description

With the reference to the logical data flow diagram, there are 7 processes of the system and descriptions of these processes are stated below.

Process Description		
Process Description	*Output Data Flow*	*Description*
1. Web Crawling		
Start Page, Queue of URLs to be Crawled	Temporary Document Files	Downloading the files of URLs as temporary files.
2. Text Extracting		
Temporary Document File	A queue of URLs to be Crawled, Single Words of Each Document, URL Information (URL Entity)	It discards the HTML format in the document and extracts all the text words for each temporary file.
3. Create Inverted Index		
Single Words of Each Document	Inverted Index	It creates inverted index.
4. Ranking		
Inverted Index	Inverted Index with rank	It ranks each word of inverted index.
5. Search Query Analyzing		
Search Query	Words of Search Query	It determines which method of search is adopted in the searching. They are keyword search, phrase search, wildcard search and Boolean search. External user can also perform multi-method search. E.g. Boolean and wildcard search.
6. Searching		

Words of Search Query	Search Results	It searches for each partition of the query from the urlInfo and wordID tables.
7. Search Results Ordering		
Search Results	Final Search Results	It rearranges the order of search results by the ranking numbers (rank).

4.3.3 Required Logical Data Modeling

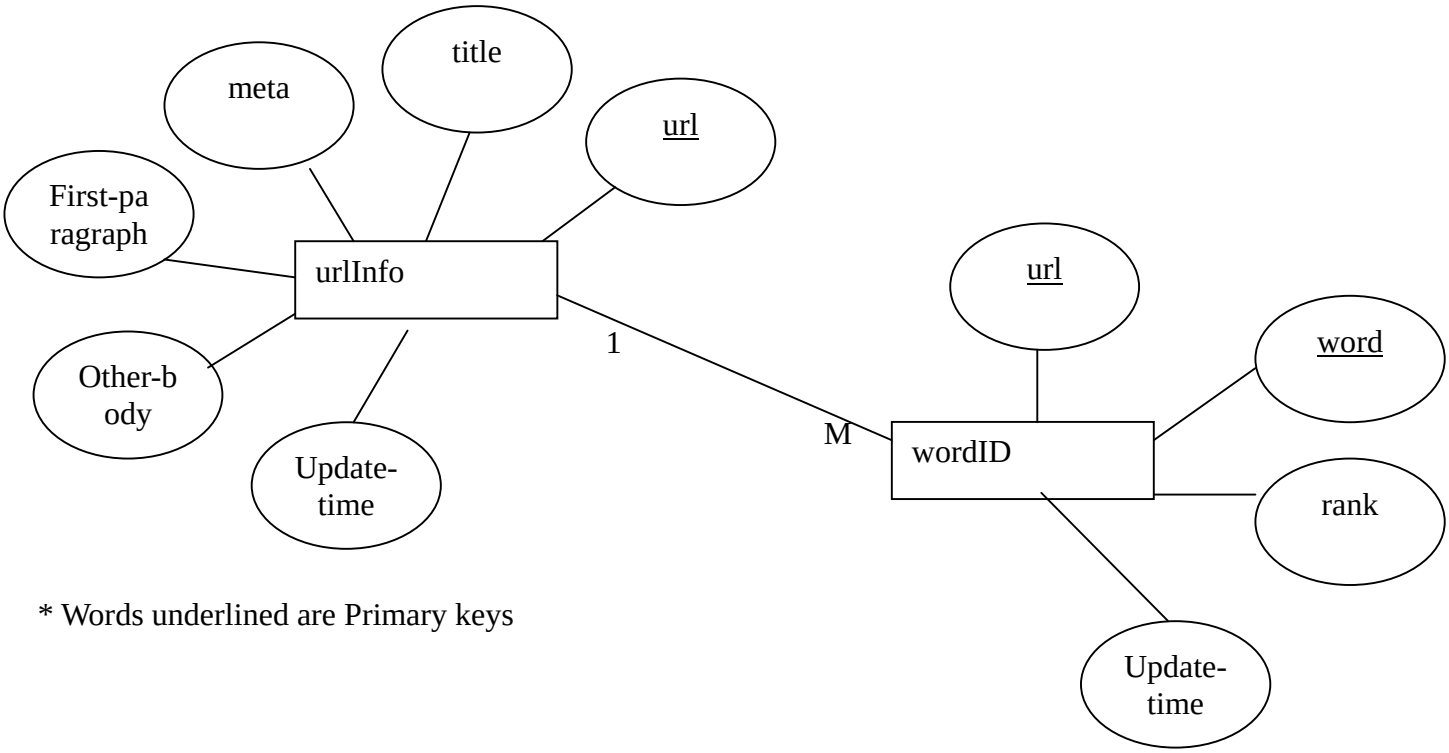


Figure 4.2 Required Logical Data Model

The required logical data model is drawn (figure 4.2) to show how data is processed through the system. Detail explanation will be given in, Composite Data Modeling.

4.3.4 Acceptance Criteria

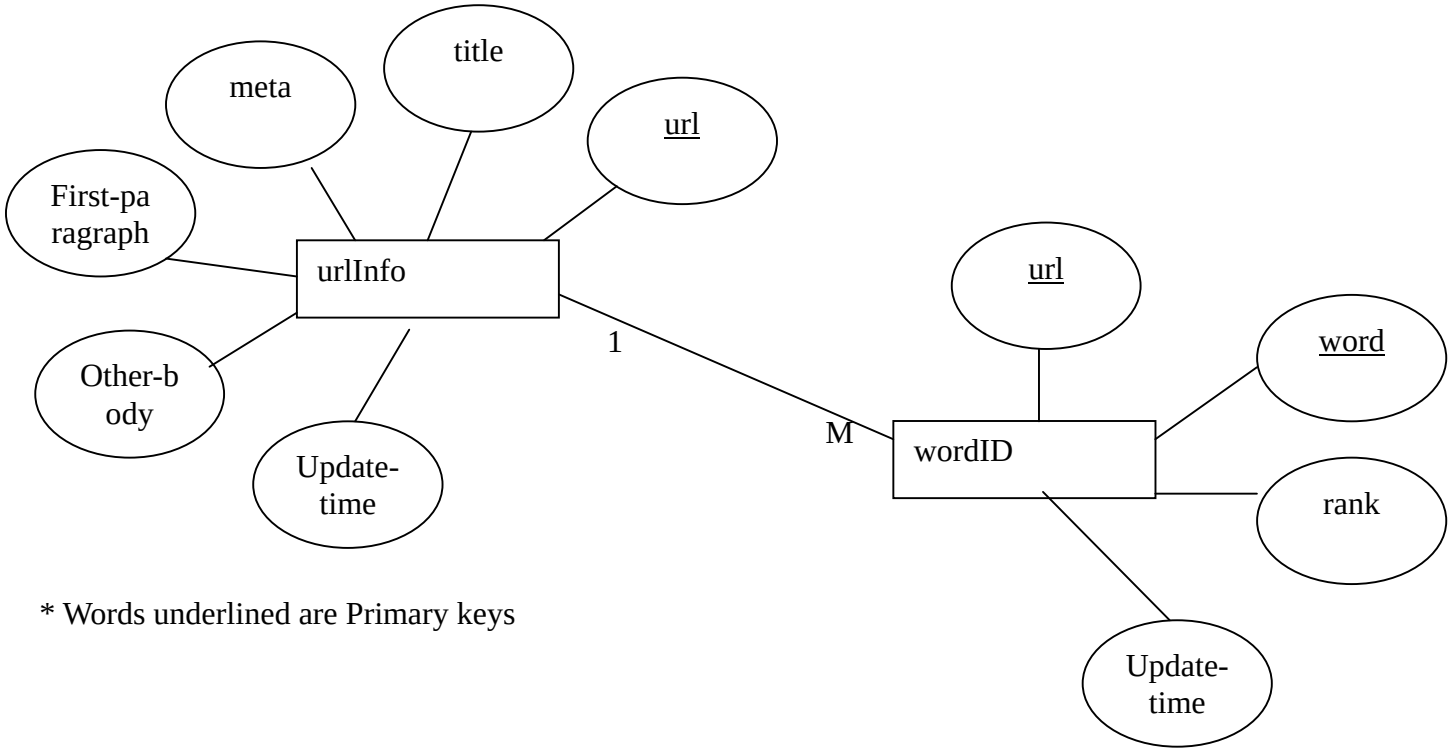
One the day of completion of the project, all features mentioned above of a search engine must be provided. They are web crawling, words extracting, create inverted index, ranking, search query analysis, searching, search results ordering. Improvement jobs may be done depending on available time.

CHAPTER 5

LOGICAL DESIGN

5.1 Composite Data Modeling

5.1.1 Entity Relationship Diagram



* Words underlined are Primary keys

Figure 5.1 Required Logical Data Model

5.1.2 System Assumption

i. **English search ONLY**. All the processes of search engine are tailor-made in English only. This project does not care other language. The results of other language are unexpected. However, use of other languages should not lead to system error.

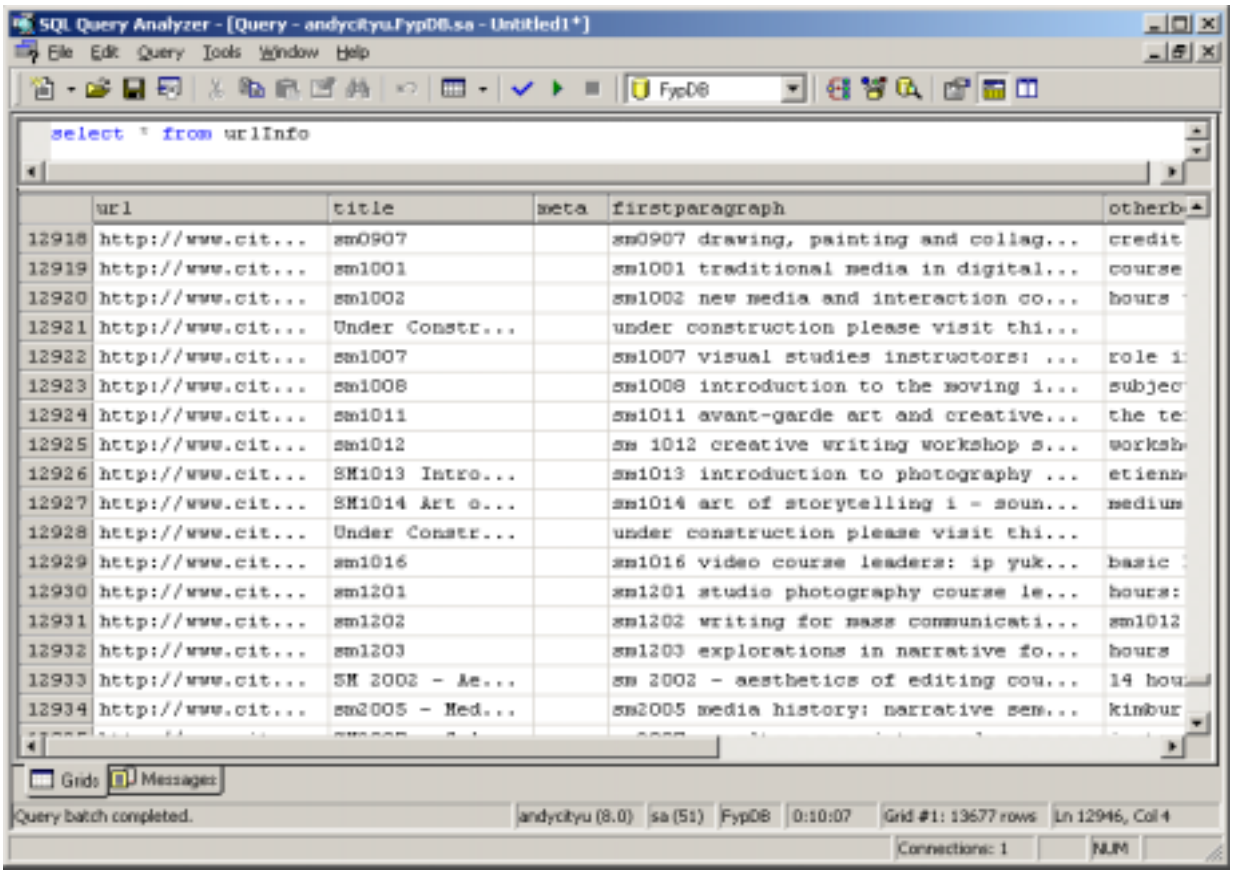
5.1.3 Entity Description

There are two entities in this system. They are listed below:

UrlInfo

UrlInfo divides the text of a page into six attributes:

- Url (primary key) – the URL of the page
- Title – text in title tag of the page
- Meta – text in the meta tag of the page
- Firstparagraph – the first 25 words in the body tag
- Otherbody – the other words after the firstparagraph in the body tag
- Updatetime – the update time



	url	title	meta	firstparagraph	otherb
12918	http://www.cit...	sm0907		sm0907 drawing, painting and collag...	credit
12919	http://www.cit...	sm1001		sm1001 traditional media in digital...	course
12920	http://www.cit...	sm1002		sm1002 new media and interaction co...	hours
12921	http://www.cit...	Under Constr...		under construction please visit thi...	
12922	http://www.cit...	sm1007		sm1007 visual studies instructors: ...	role i
12923	http://www.cit...	sm1008		sm1008 introduction to the moving i...	subjec
12924	http://www.cit...	sm1011		sm1011 avant-garde art and creative...	the te
12925	http://www.cit...	sm1012		sm 1012 creative writing workshop s...	worksh
12926	http://www.cit...	SM1013 Intro...		sm1013 introduction to photography ...	etienn
12927	http://www.cit...	SM1014 Art o...		sm1014 art of storytelling i - soun...	medium
12928	http://www.cit...	Under Constr...		under construction please visit thi...	
12929	http://www.cit...	sm1016		sm1016 video course leaders: ip yuk...	basic
12930	http://www.cit...	sm1201		sm1201 studio photography course le...	hours:
12931	http://www.cit...	sm1202		sm1202 writing for mass communicati...	sm1012
12932	http://www.cit...	sm1203		sm1203 explorations in narrative fo...	hours
12933	http://www.cit...	SM 2002 - Ae...		sm 2002 - aesthetics of editing cou...	14 hou
12934	http://www.cit...	sm2005 - Med...		sm2005 media history: narrative sem...	kinbur

Figure 5.2 View table urlInfo

This entity represent pages and their texts in each tag.

WordID is the inverted index of pages, it has the following attributes:

- Word (primary key) – the word
- url (primary key)– the URL of the page that the word is found
- rank – the rank number
- updatetime

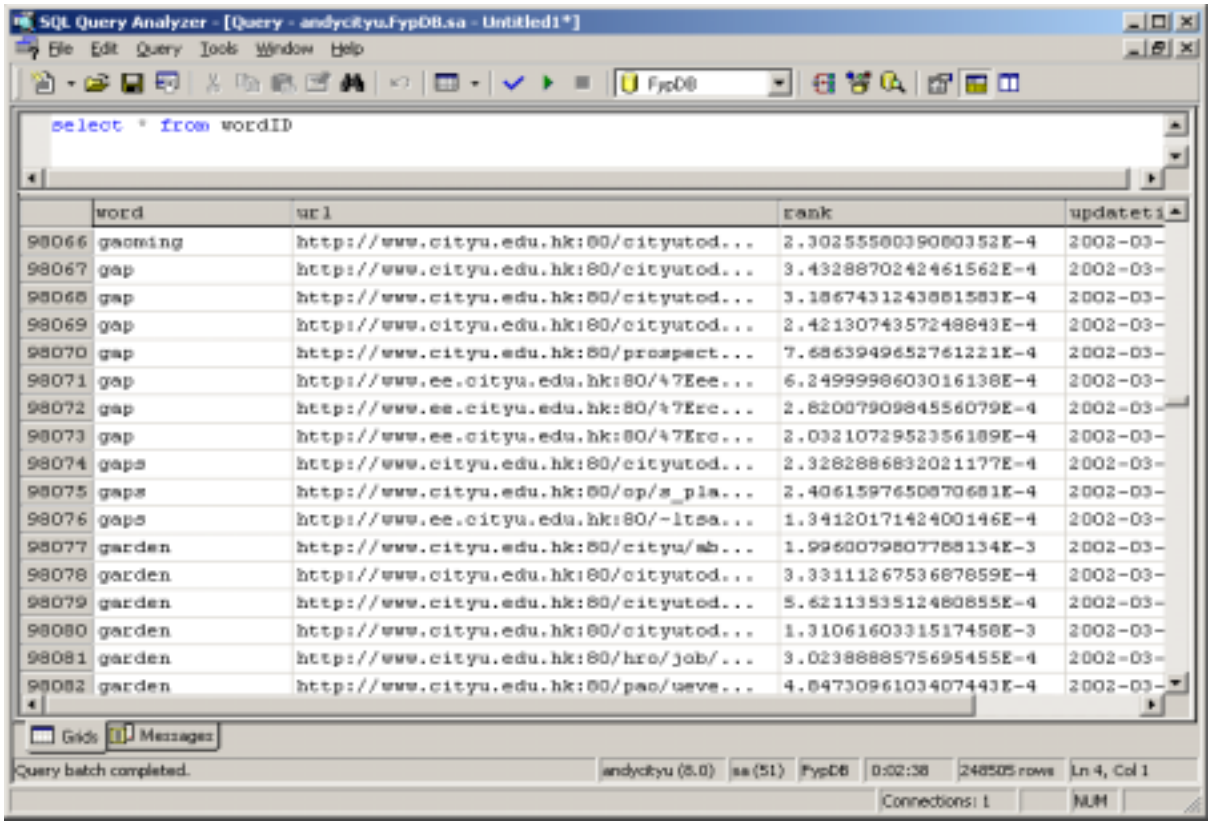


Figure 5.3 View table wordID

5.1.4 Relationship Description

Each page has many words. Each word in a page represents a wordID entity. The relationship of UrlInfo to wordID is **one** to **many** as you see in figure 5.1.

5.2 Logical System Design

5.2.1 System Overview

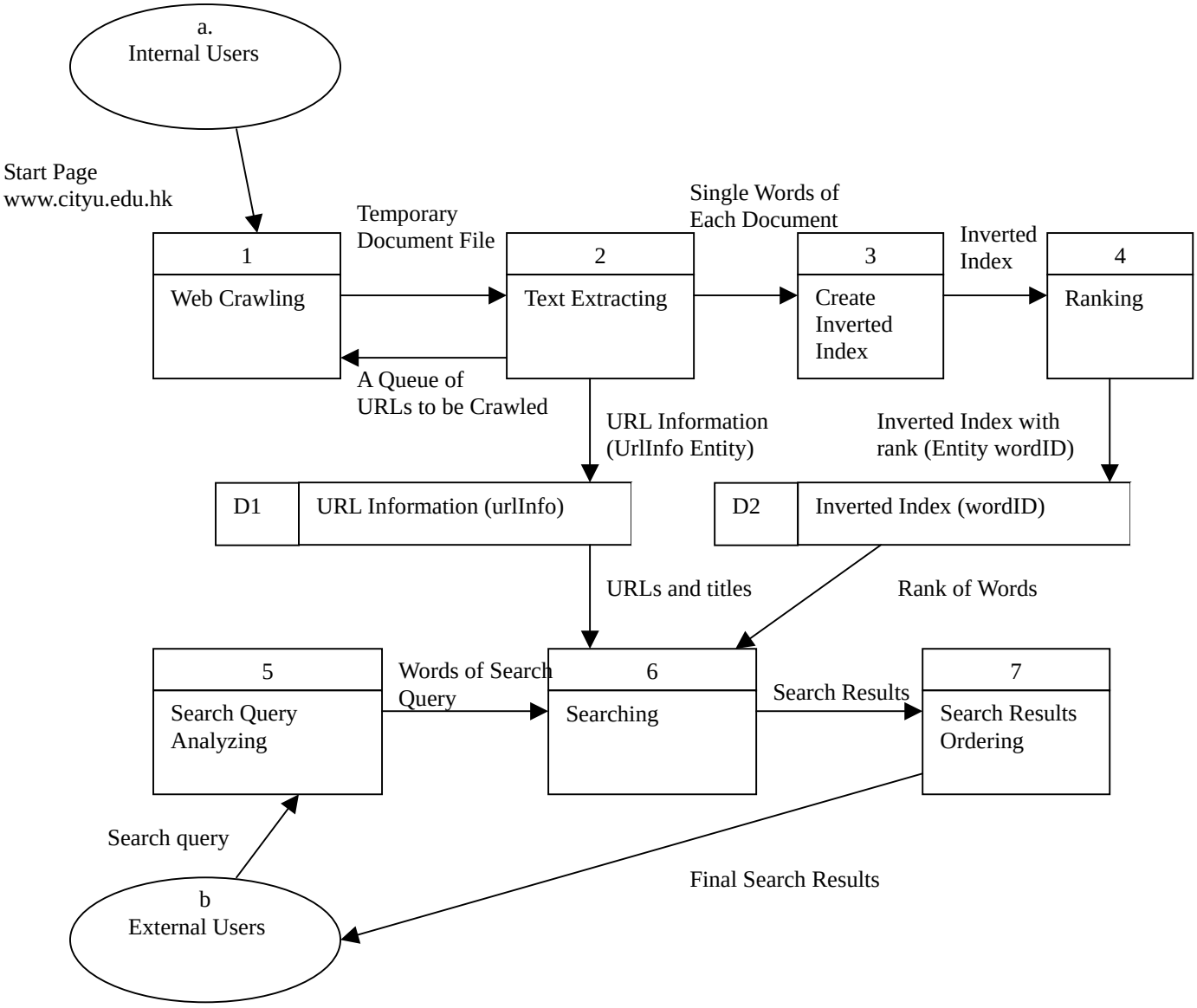


Figure 5.4 Required Logical Data Flow Diagram

The system overview contains 2 external entities, 2 data stores and 7 processes. Since the logical system design base on the required functions, I not explain it again here. The physical design will talk more on the functions.

CHAPTER 6

PHYSICAL DESIGN

6.1 Technical Configuration Definition

6.1.1 Base Technical Configuration

In this Section, there will be an integrated definition of the technical architecture for the operation of the system.

- **Platform** – **Windows 2000 Server edition** and it is required for running IIS.
- **Web server** – **Internet Information Server (IIS)** in windows 2000 Server edition.
- **Database server** – **SQL server 2000** edition. It stores all the data of the systems. They are urlInfo and wordID.

6.1.2 Development / Delivery Environment

After having the technical architecture for the operation of the system, in this section, we will have the technical architecture for the development of the system.

- **System Development Languages –** Java language, HTML, SQL are all used in the development. Java language is used to develop the whole system of search engine. SQL is only used for inserting, selecting and updating database. HTML is only used to develop a user interface for external user searching..
- **Database Connectivity Tool** – JDBC package in Java language is for used for building a JDBC/ODBC bridge. With Open Database Connectivity (ODBC) created by Microsoft for connecting database in Window 2000 server edition, Java application program is connected to the SQL server 2000 and execute queries.

6.2 Module Decomposition Chart

The below figure shows the overview of Java packages in CityU Search Engine:

- CityUSearchEngine.search
- CityUSearchEngine.search.spider
- CityUSearchEngine.search.query
- CityUSearchEngine.util

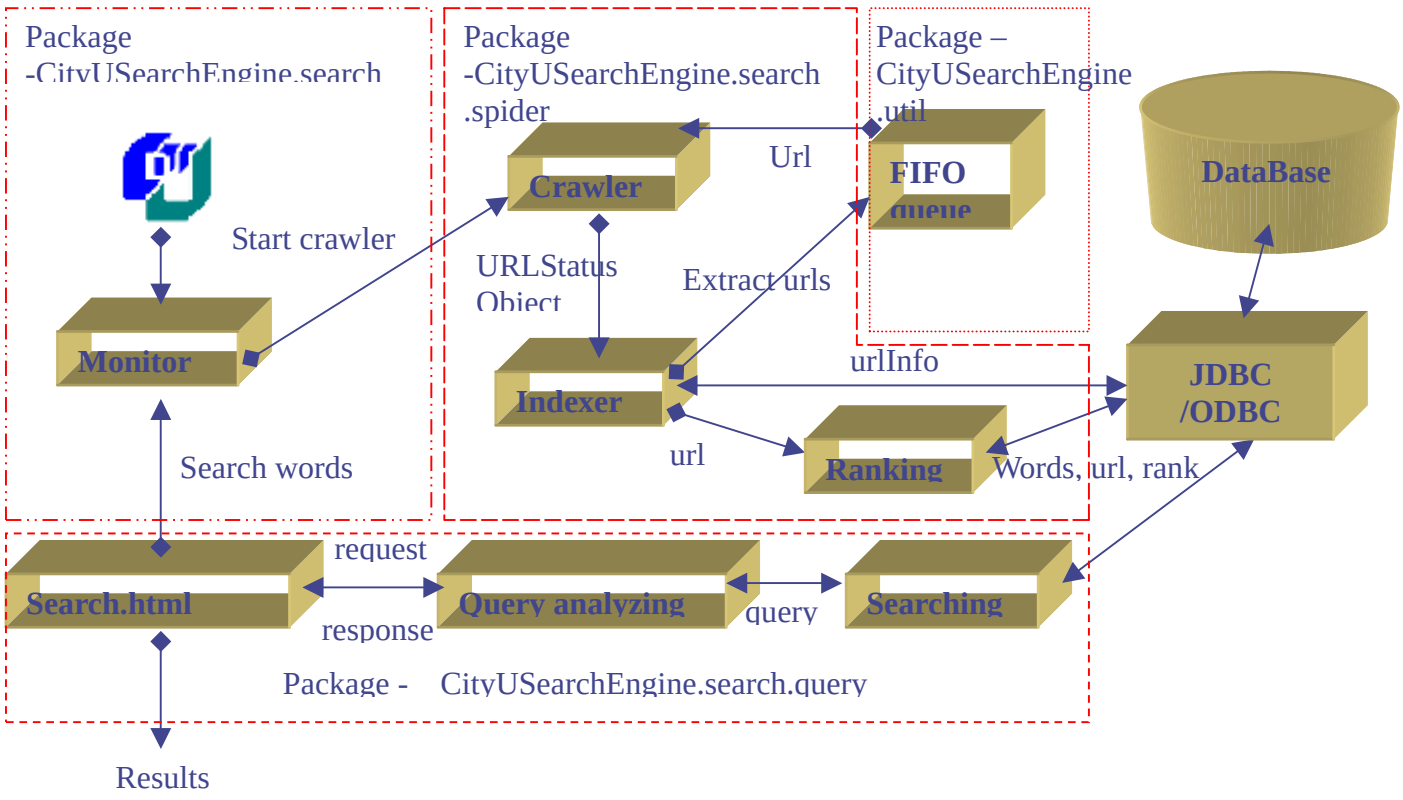


Figure 6.1 New CityU Search Engine Overview

Besides package CityUSearchEngine.util, the three packages represent three modules of CityU Search Engine. CityUSearchEngine.util.FIFOqueue is just the First In First Out (FIFO) queue for web crawling. The explanations of each package (module) will be shown in the below sections.

6.2.1 CityUSearchEngine.search

This package represents a module that provides an administrating environment for the administrator of CityU Search Engine. The functions of this module provides are not included in Required Function

Description since these functions are just providing interface (Monitor) and options for administrator but not actual functions related to search engine. It provides the following functions:

- Administrator starts the spider module.
- View queries inputted by external user.
- Monitor URL is crawling.
- View total file size crawled in bytes.
- Monitor URLs have been crawled.
- Monitor URLs have errors (exceptions) during crawling.

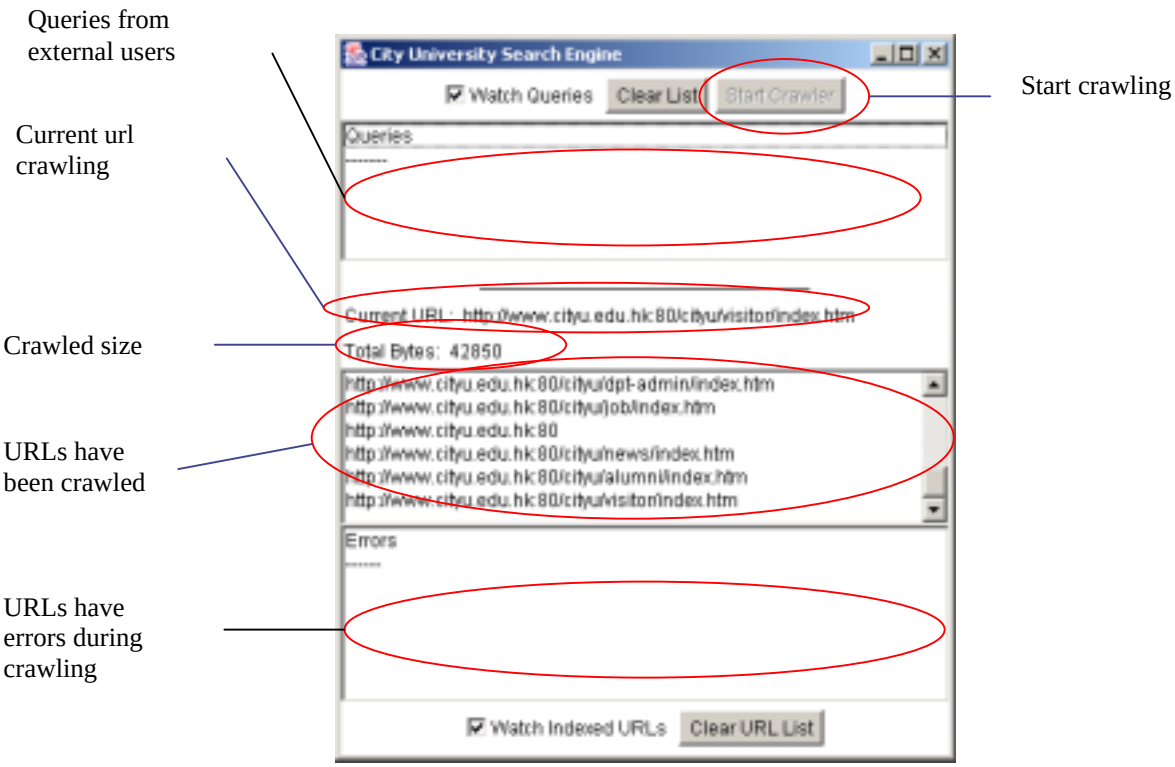


Figure 6.2 Monitor interface and functions

6.2.2 CityUSearchEngine.search.spider

This package is the spider module of search engine. It has the following functions:

- **Web crawler**.
  i. It obtains URLs from a queue of City University web pages.
  ii. Download each URL page as temporary file (tmp file)
  iii. Create objects called URLStatus which include:
  URL,
  tmp file name, and location.
  iv. Pass URLStatus object to Indexer.
  v. Send back URL of pages to Monitor to show crawler status.
  vi. After crawling, it deletes the records in urlInfo and wordID for those URLs have not been updated for one year (this parameter can be changed by administrator).
- **Indexer**.
  i. Extract URLs from tmp files, add to FIFO queue for crawling.
  ii. Extract tags and create inverted index from tmp files.
  Five parts are divided by extracting tags:
  URL,
  Title,
  Meta,
  Firstparagraph, and
  Otherbody.
  iii. Extract words from tags.
  iv. Pass to ranking.
- **Ranking**.
  Ranking of a word base on these three rules:
  i. Keywords from URL, title, and Meta tags are the most important.
  ii. Base on prominence (how close to the beginning).
  e.g. for title “CITYU Intranet – For Students”
  Importance: CITYU > Intranet > For > Students
  iii. Base on frequency (how many times that word appears).
  e.g. 4 times appear > 2 times appear

In CityU Search Engine, the ranking method is integrated from the above 3 points:

- For normal words in (otherbody), weight = 1
- For words in firstparagraph, weight = 5
- For words in meta tag, weight = 40
- For words in title tag, weight:
  - 1st word – 65
  - 2nd word – 60
  - 3rd word – 55
  - Others – 50
- For words in URL, weight = 50

Therefore, rank of a word (How important is that word refer to the page) is calculated from the below formula:

Rank = Total rank of that word / Total ranks of all the words in the page

e.g. For “intranet” occurs once in URL, once in title(position 2), 2 in otherbody and the total ranks of all the words in that page is 1000.

- Rank of “intranet” = (50 + 60 + 2*1)/1000
  = 0.112

If this rank number is *large*, content of the page will be regarded as *fit* to the keyword.

- **Database Insert Agent**

This agent is used to insert or update db for both urlInfo and wordID tables.

For urlInfo table:

- If url do not exist → insert
- If url exist and has been updated within update period → do not update
- If url exist and has not been updated within update period → update

For wordID table:

- If word in url do not exist → insert
- If word in url exist and has been updated within update period → do not update

- If word in url exist and has not been updated within update period → update

6.2.3 CityUSearchEngine.search.query

this package is the query module of CityU Search Engine. It has the following functions:

- **Search query analyzing –** determine what types of search methods are posted by the external user. The four search methods are explained below.
- **Searching –** search for each single word from table wordID and perform AND, OR, NOT base on the query, except phrase search. Then reorder base on the rank is performed. Finally, results of search are returned with URLs, ranks and descriptions.
  - Keyword search is simple AND search for all single words from in the query. Its format is just input the keywords.

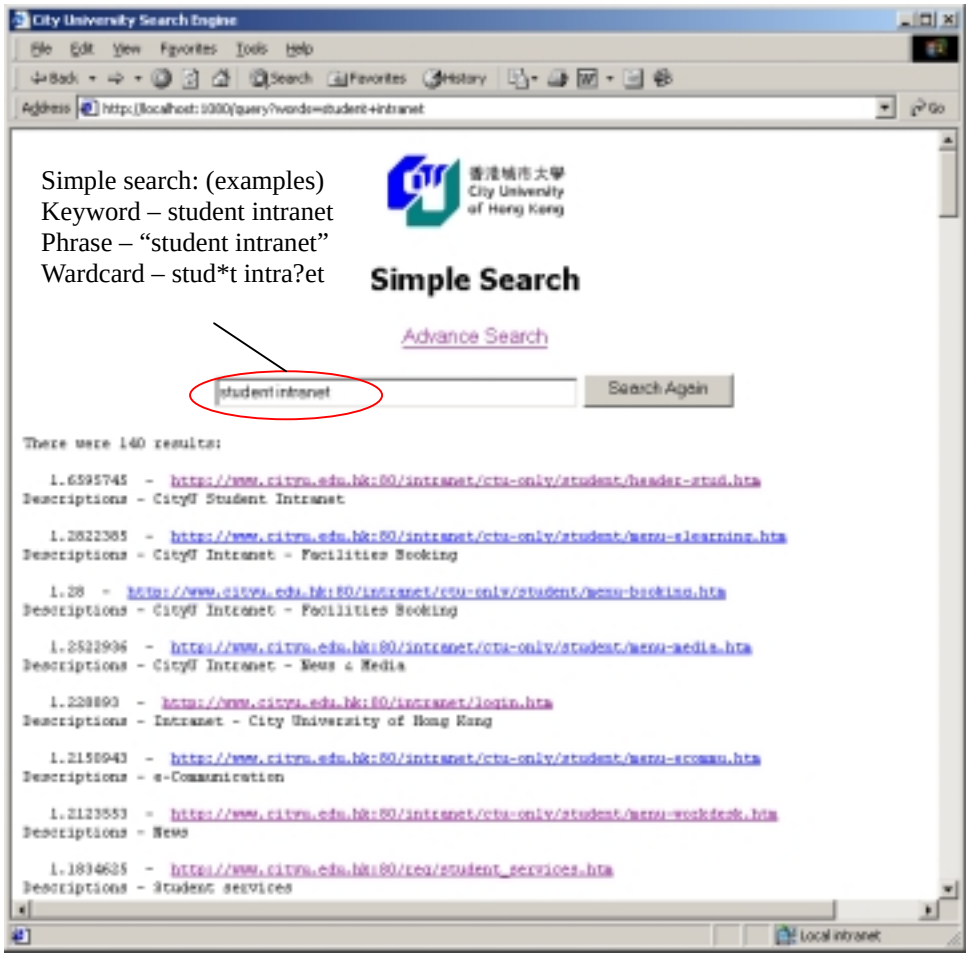


Figure 6.3 Simple Search Examples

- Phrase search is searching for the exact phrase from the table urlInfo. Its format is add two '"', one to the beginning of the query, one to the end of the query. As the example in figure 8.3. multi-phrase search is also supported in this system.
- Wildcard search use '*' or '?' to represent don't care characters. While '*' can represent one or more don't care characters, '?' can only represent one don't care character.
  e.g. c*y for cry, city, candy and century…etc. no matter the length of the word. c?y just for words with three characters long. Cry is included while city is not included.
- Boolean search has three input text fields, first is 'AND' search, then is 'OR' search, and 'Not' search in the third field example is shown in figure 8.4. It searches for information AND system OR staff NOT student

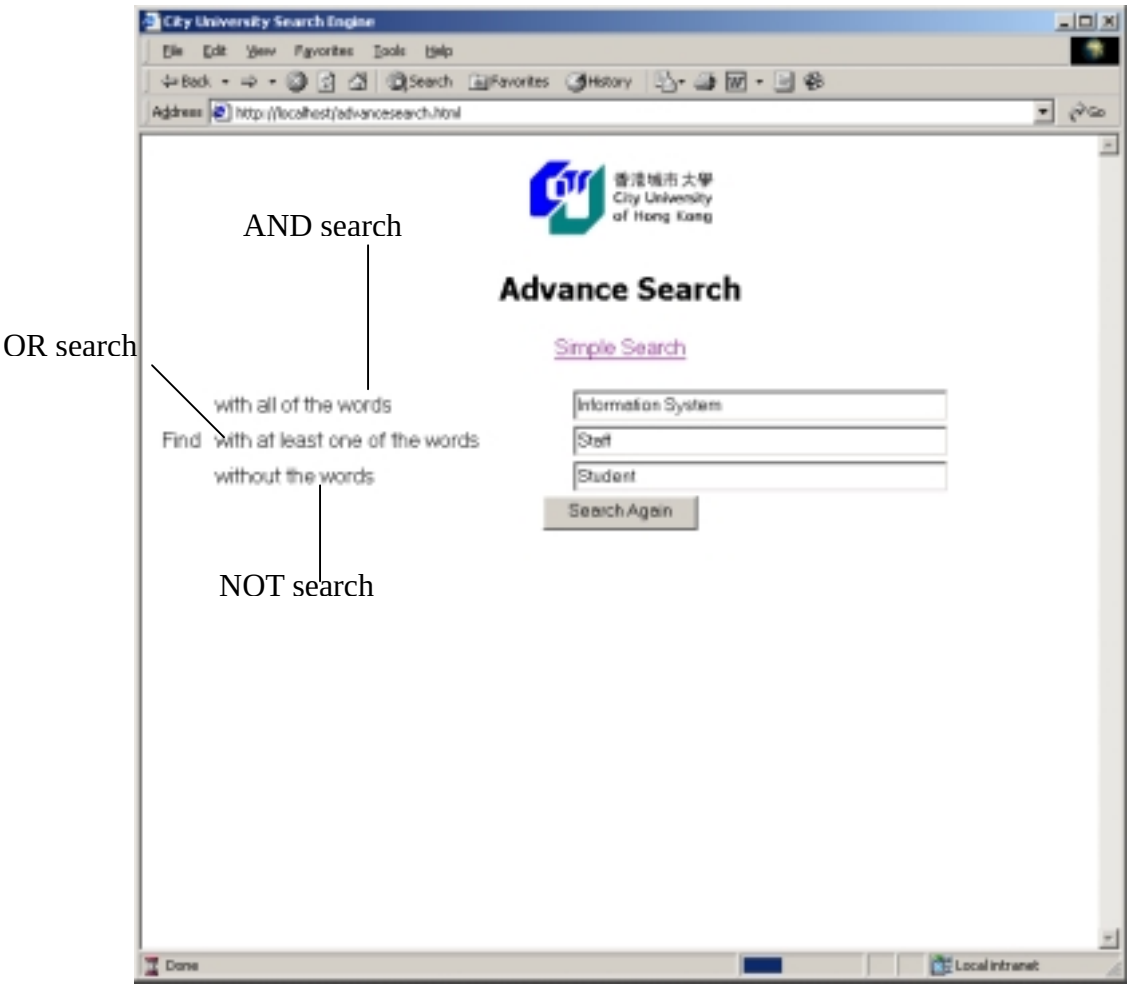


Figure 6.4 Advance Search Example

Reference

[HREF1] What is a “Web Crawler" ?
(http://research.compaq.com/SRC/mercator/faq.html)

[HERF2] inverted index
(http://burks.brighton.ac.uk/burks/foldoc/86/59.htm)

[MARC] Marckini, Fredrick. Secrets to making your Internet Web Pages Achieve Top Rankings (ResponseDirect.com, Inc., c1999)