A Dimensional Category-based Email Management System

By
Peng Lu

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of
Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

June 20, 2007

© Copyright
2007, Peng Lu
NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

AVIS:
L’auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L’auteur conserve la propriété du droit d’auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.
Abstract

Due to the increasing usage of email for communications and other tasks, effective email management is becoming more and more important. People put a lot of efforts into organizing, storing, and searching to efficiently manage and maintain the useful information in their emails.

There are several kinds of email systems that people use extensively, which are “folder-based” (Microsoft Outlook Express), “search-based” (Yahoo! Bloomba), “category-based” (Apple Mail), and so on. This thesis will analyze these systems and also present a novel method to effectively and efficiently organize, manage, maintain, and search emails based on database technologies. We call the new system a “multi-dimensional categorization hierarchy-based” email management system, because it manages emails by using various categories, instead of the directories or folders used in traditional email systems.

As opposed to folder-based or category-based systems, our email system allows an email to be classified into more than one category, while a category can also have multiple parent categories in the same dimension. These features give users the flexibility to organize their emails within various categories and to make email management more effective than ever before.
Acknowledgements

First, I would like to sincerely thank my supervisor Dr. Mengchi Liu, for his guidance, encouragement, and patience that carried me on through difficult times. I am so lucky to have such a wonderful advisor. Without his direction and insights, this thesis would not have been possible.

I would also like to give a special thanks to my parents. We always feel so close, even though they live at the other end of the earth. Their well-being and encouragement are the strongest supports for me.

My gratitude also goes to Yifeng Gao, Zhihong Li, and Qunxiao Wang for sharing their ideas with me during my research.
Contents

Abstract iii

Acknowledgements iv

List of Tables ix

List of Figures xi

1 Introduction 1
  1.1 Motivation ................................................................. 1
  1.2 Objectives ................................................................. 5
  1.3 Outline of the Thesis .................................................. 6

2 Background 7
  2.1 Email Related Concepts and Standards ....................... 7
    2.1.1 Email Address ....................................................... 7
    2.1.2 Email Structure .................................................... 8
    2.1.3 Email Server ......................................................... 9
    2.1.4 Email Client ......................................................... 11
    2.1.5 Email Protocols .................................................... 11
  2.2 How Email Works ..................................................... 12

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
2.3 Email Security - SSL [9] ................................................................. 13
2.4 Email Management ....................................................................... 14
  2.4.1 Folder-based Management ...................................................... 15
  2.4.2 Category-based Management .................................................. 15
  2.4.3 Email Filtering ....................................................................... 16
  2.4.4 Summary .................................................................................. 16

3 Related Work ............................................................................. 17

  3.1 A Brief Survey of Email Client Applications ........................... 17
    3.1.1 Review of Microsoft Outlook Express ................................. 18
      3.1.1.1 Managing Multiple Email Accounts .............................. 19
      3.1.1.2 Composing and Sending Email Messages .................. 19
      3.1.1.3 Retrieving and Reading Email Messages ..................... 20
      3.1.1.4 Deleting Email Messages ........................................... 21
      3.1.1.5 Managing Email Messages ......................................... 22
      3.1.1.6 Searching Email Messages ......................................... 23
      3.1.1.7 Filtering Email Messages ............................................ 24
    3.1.2 Other Graphical Email Clients ............................................ 25
      3.1.2.1 KMail ......................................................................... 25
      3.1.2.2 Bloomba ................................................................. 26
    3.1.3 Web-based email Clients ..................................................... 28
    3.1.4 Comparison of Email Clients .............................................. 30

3.2 File Management Applications .................................................. 31
  3.2.1 Hierarchical File System ....................................................... 33
  3.2.2 Category-based Applications ................................................. 35
  3.2.3 Faceted Classification Applications ...................................... 37
  3.2.4 Formal Concept Analysis Applications ................................. 38

vi

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
3.3 Summary ................................................................. 40

4 Our Approach ......................................................... 41
4.1 Category ................................................................. 41
4.2 Multiple Dimensional Hierarchical Category Structure .............. 42
4.3 Definitions of the Structure ........................................... 45
4.4 Database Modeling and Design ........................................ 49
   4.4.1 Entity-Relationship Model and Database schema ............ 49
   4.4.2 Data Example ....................................................... 53

5 System Implementation ................................................... 56
5.1 System Architecture .................................................... 56
5.2 Default Values .......................................................... 62
5.3 User Access Mechanism ................................................. 62
5.4 General Email Client-side Operations .................................... 63
   5.4.1 Email Account Managing ........................................ 63
   5.4.2 Email Checking and Receiving .................................. 63
   5.4.3 Email Reading ....................................................... 65
   5.4.4 Email Composing and Sending .................................. 65
   5.4.5 Email Deleting ....................................................... 67
5.5 Category Managing Operations .......................................... 68
   5.5.1 Category Relation Retrieving Operations ....................... 68
   5.5.2 Category Detecting Operations ................................... 69
   5.5.3 Path Computing ..................................................... 73
   5.5.4 Relationship Creating ............................................. 78
   5.5.5 Relationship Removing ............................................ 79
   5.5.6 Category Creating .................................................. 79
   5.5.7 Category Deleting .................................................. 79
# List of Tables

3.1 Comparison of Email Clients ...................................................... 32
3.2 An Example of Formal Concept Analysis ........................................ 38

4.1 Data Sample of Email Table ......................................................... 53
4.2 Data Sample of Category Table ..................................................... 53
4.3 Data Sample of Email Content Table ............................................. 54
4.4 Data Sample of Email Classification Table ..................................... 54
4.5 Data Sample of Category Classification Table ................................. 54
List of Figures

2.1 Email Structure ................................................................. 10
2.2 How Email Works .............................................................. 12
2.3 How SSL Secures Email .................................................... 14

3.1 Outlook Express: Managing Multiple Email Accounts .......... 19
3.2 Outlook Express: Composing and Sending Email ............. 20
3.3 Outlook Express: Retrieving and Reading Email Messages . 21
3.4 Outlook Express: Deleting Email Messages ..................... 21
3.5 Outlook Express: Using Hierarchical Folders to Manage Email Messages . 22
3.6 Outlook Express: Searching Email Messages .................... 23
3.7 Outlook Express: Filtering Email Messages ....................... 24
3.8 KMail: Searching Email Messages .................................... 25
3.9 Bloomba: Searching Email Messages ............................... 27
3.10 Bloomba: Thread .............................................................. 27
3.11 The Macintosh Hierarchical File System ......................... 34
3.12 Multiple Hierarchical Category Structure ....................... 35
3.13 An Example of Formal Concept Analysis ......................... 39

4.1 Multiple Dimensional Hierarchical Category Structure ........ 44
4.2 Entity-Relationship Model ............................................... 50
4.3 Database Schema .......................................................... 51
5.1 System Architecture ............................................................... 58
5.2 Expanded System Architecture .............................................. 61
5.3 Email Account Managing ...................................................... 64
5.4 Algorithm of the Check New Email Operation ...................... 64
5.5 Email Checking and Receiving .............................................. 65
5.6 Algorithm of the Read Email Operation ............................... 66
5.7 Email Reading ................................................................. 66
5.8 Email Composing and Sending ............................................ 67
5.9 Algorithm of the Delete Email Operation ............................. 68
5.10 Example: Offsprings Retrieving ......................................... 69
5.11 Algorithm of the Offspring Retrieving Operation ................. 70
5.12 Algorithm of the Ancestor Retrieving Operation .................. 71
5.13 Algorithm of the Dimension Detecting Operation ............... 72
5.14 Example: Directed Cycle .................................................... 73
5.15 Algorithm of the Directed Cycle Detecting Operation .......... 74
5.16 Algorithm of the Top-down Path Computing Operation ....... 75
5.17 Example: Top-Down Path Computing .................................. 76
5.18 Algorithm of the Bottom-up Path Computing Operation ....... 77
5.19 Algorithm of the Relationship Creating Operation ............... 78
5.20 Algorithm of the Relationship Removing Operation ............. 80
5.21 Algorithm of the Dimensional Category-based Search .......... 82
Chapter 1

Introduction

1.1 Motivation

Electronic mail (email) is a computer-supported form of communication that allows the exchange of information via messages over the internet.

Before even the Internet appeared, email had already been invented. Email started in 1965 as a method for allowing multiple users of a time-sharing mainframe computer to communicate. There was no system that allowed electronic text messages to be transferred between users on different computers until the Automatic Digital Network (AUTODIN) system was developed in 1966. Apparently, the first email message was sent in 1971 by Ray Tomlinson. His breakthrough was the ability to send messages to other machines over the Internet, using the “@” sign to specify the receiving computer [39]. When the utility and advantages of email in the AUTODIN system becoming more widely known, the popularity of email increased. Email was quickly extended to be used over network systems, allowing users to send messages between various computers.

In the 1980s, with the advent of the widespread use of personal computers connected to the Internet, email became an important feature of collaboration and communication products. On the other hand, existing email systems were a crucial tool in building the
Internet.

Since 1991, the year that the World-Wide Web was introduced, the internet has been growing at an amazing rate. Technology has improved considerably, and email has been regarded as an essential tool for communication. There is no doubt that email will also be indispensable in future communications.

With the widespread use of the Internet (which is used by over one billion people today), email has been an increasingly popular and important method of communication in everyday life activity during the past 40 years.

Nowadays, email users may send and receive dozens or even hundreds of messages per day. They spend more and more time dealing with their emails. Email was originally developed for communication, but with the development of internet technology, today, email has evolved into a multipurposes tool [17]. In addition to simply exchanging information, email is now useful for other tasks: to deliver or archive documents, to store personal contact information, to schedule appointments, and to track the progress of works. However, every advantage has its disadvantages. People are now required to spend a lot of effort on organizing, storing, searching, and managing their emails.

Determining how to organize, manage, maintain, and retrieve useful information from emails has always been a challenge. For this reason, many email management systems (also known as email clients) have been developed to assist users in managing their email messages, including Microsoft Outlook Express, Apple Mail, Hotmail, and Yahoo! Mail. Typically, they handle the email messages via receiving, reading, composing, sending, organizing and searching facilities. With the increased volume and complicated content of email messages, finding ways to maintain and organize email messages is becoming more and more important.

Several methods are used by most email clients to organize email messages, such as using flat structured folders or hierarchical structured directories. The most commonly used method is to place email messages into flat structured folders, that are created and
defined by the users themselves. Users can group their email messages into various folders semantically, and maintain these messages at the same time. Placing an email message in a suitable folder helps users to find it easily. However, assigning every email message to only one folder is not the naturally accepted practice. Sometimes an email can be sent to more than one folder. Given an alternative method of organizing email messages, placing email messages in a hierarchical directory structure is cloned from file systems used by some email management systems (i.e. Microsoft Outlook Express). This type of structure means that people can manage their email messages via hierarchical folders. Besides managing email messages in a tree structure, one email message involves multiple knowledge fields (i.e. “friends”, “greetings”, “April”, “Ottawa”, and so on) and these belong in different parts of the tree. However, storing many copies of one email message in several different folders is inefficient and unnecessary. As an efficient alternative, the flat category structure is more natural and coincidental to human behaviors. It allows an email message to be assigned to many categories (also known as “virtual folders”), but not physical folders. Nevertheless, both the hierarchical folder structure and the flat category structure still have some problems.

Hierarchical folders The first problem with hierarchical folders is that they are not flexible for organizing a very large number of email messages. Each folder can have several sub-folders, but it can only have one parent folder, and each email message can be put into only one folder. The structure thus expands quickly in depth as the amount of email messages increases, and each folder may contain a large number of email messages. It is difficult to find a particular message in a specific folder that is in a deep and extensive tree. According to Balter [2], the extensive and deep filing of email messages into the hierarchical folders is not efficient because the time saved on searches is outweighed by the time spent on filing. In particular, it is hard to keep track of the deep folders when users want to find recent or frequently accessed information.
Secondly, a system of hierarchical folders is organized as a “static organization scheme” [13]. To store an email message in a folder, the user has to know the name, position, or the look of the folder, or even define a new folder. To retrieve information, the user needs to know the correctly ordered sequence of every folder concerned. In other words, users need to know how the structure was originally created to organize the email messages. As a result, they cannot find a specific email message using a different folder path during retrieval. It is hard to keep track of this kind of complex and fixed schemes.

Moreover, this structure is a tree structure, in which every folder has only one parent folder and there is only one path from the root for each folder. Also each folder may contain a number of email messages. People tend to forget the location of older email messages. To search for a message in a hierarchical folder structure, users are required to look over the whole tree. Every time they search, they must start from the root of the tree. The result may not be accurate and it may take too much searching time to go through the tree from root to target for every search task.

Flat categories Every time a user wants to find an email message, similar to the issue with the flat folder structure, the user has to face a long list of categories. If the user does not know or remember the position or name of the category, he or she could spend a lot of time scanning the categories one at a time until the target category is found. This progress is time-consuming. As a result, users would rather create new categories than look for existing ones. Accordingly, the redundancy of categories could be increased.

Furthermore, flat categories lack flexibility and efficiency. Users may define some categories that are related in some field. For example, “school of computer science”, “university center” and “graduate office” are all different departments of “Carleton University”. In a flat structure, users cannot organize or maintain the relationships of these categories. Therefore, users have difficulty with locating the set of categories that is actually related.

All of the drawbacks of the current structures for email management systems listed
above need to be improved, and these have motivated us to study a new structural model for the email management system.

1.2 Objectives

The main goal of the work presented in this thesis is to help email users to organize, manage, maintain, and search for email messages more efficiently. We researched the current email management systems and several widely used file management systems. Based on the comparison of these systems, we designed a category structure, which extended Yifeng Gao's work [10]. The extended category structure is called multiple dimensional hierarchical category structure. This structure overcomes the drawbacks and limitations of the existing flat and hierarchical structures. Based on this structure, we designed and implemented an email management system called EmailCat. The following is a list of the detailed objectives of this thesis:

First, study the current email management systems and file management systems, and analysis the structures used by these systems.

Second, propose the extended category structure, called a multiple dimensional hierarchical category structure. Then model this structure into objective entities and map it into relational database schema for practical usage.

Third, build an experimental email management system based on the multiple dimensional hierarchical category structure.

Fourth, compare our system with the flat structure in traditional email management systems to prove that our structure provides greater efficiency.
1.3 Outline of the Thesis

This thesis has four parts. The first part introduces the background knowledge concerned in this thesis. The second part investigates the related work of our research. The third part describes our approaches in detail. The fourth part describes the implementation of our experimental system and compares our system with other existing applications. The thesis ends with a summary, contributions, and questions for future research.

- Chapter 1 presents the motivation and objectives of this thesis.
- Chapter 2 introduces background knowledge that is important to understanding this thesis.
- Chapter 3 gives a survey of the related work of our research and existing solutions.
- Chapter 4 introduces the model of our extended category structure, gives the Entity-Relationship model, database schema, and operation rules of our approach.
- Chapter 5 presents our approach to the email management system, and discusses implementation issues. It introduces the basic rules of applying this system.
- Chapter 6 presents the conclusions of this thesis, lists the contributions of our work, and describes directions for future research.
Chapter 2

Background

This chapter provides a brief background that is important to understanding the motivation of this thesis. It includes some related aspects, such as email clients, the limitations and drawbacks caused by the current email management, and category-based applications.

Email, also known as Electronic mail or email, is a method of composing, sending, receiving, and managing messages over electronic communication systems. The term email applies both to the Internet email system and to Intranet systems that allow users within smaller groups, such as a single organization or company, to send messages to each other.

2.1 Email Related Concepts and Standards

The following Internet email related concepts and standards are used frequently and need to be understood clearly.

2.1.1 Email Address

An email address uniquely identifies a location where the email message can be delivered. It is a string of the form “user@domain”, such as “plu@carleton.ca”, which is read as
“plu at carleton dot ca”. The part before the “@” sign is the local part of the address, often the username of the recipient, and the part after the “@” sign is a domain name which can be looked up in the Domain Name System (DNS) to identify the mail servers accepting email messages for that address. There is also an alternate form of email address. According to Request for Comments (RFC) 2822, the standard for Internet message format, “an optional display name that indicates the name of the recipient (which could be a person or an organization)” [27] could be followed by the address specification surrounded by angled brackets, for instance, “Peng Lu (plu@carleton.ca)”.

2.1.2 Email Structure

Email messages consist of two major parts:

- **Header** - Structured fields such as sender (the email address from the person who sent this email), recipients that includes carbon copy (CC) and blind carbon copy (BCC), subject (a short line for the title of email), date (email related time), status flag (according to standard of Multipurpose Internet Mail Extensions - MIME [8], there are RECENT, SEEN, DELETED, ANSWERED, DRAFT, FLAGGED, and USER), the size (the size of email), and other basic information about the email.

- **Body** - Based on the simple original definition in RFC 2822, the message itself is unstructured text, sometimes containing a signature block at the end. The header is separated from the body by a blank line. However, most email messages today are more complex. In particular, the message body may have multiple nested parts, which means that a part may contain another part or parts inside. It may also include embedded HTML and any type of attachments such as images, sound clips, or PDF documents, etc. Each part of the body is separated by the same boundary string.

The structure of email messages is originally defined in RFC 2822 and MIME, which is a collection of a series of RFCs, RFC 2045 through RFC 2049.
RFC 2822, an updated version of RFC 822, specifies the basic structure of an email. This standard specifies the syntax only for text messages. In particular, it does not support the transmission of images, audio, video, or other sorts of structured data in an email message.

MIME, published as an extension of RFC 2822, supplements the email formatting rules to allow non-English text in both email headers and bodies, and defines a mechanism for including non-textual attachments in email bodies. Virtually all Internet email and a fairly large proportion of automated email is transmitted in MIME format.

Figure 2.1 shows an example of a normal email structure. In this example, the email body contains both text and attachment.

2.1.3 Email Server

An email server (also called a mail transfer agent, MTA, or a mail exchange server) is a computer program or software agent that transfers email messages from one computer to another. A mail submission agent (MSA) receives email messages from a mail user agent (MUA or email client) and contacts an email server for delivery of the email. An MTA receives messages from an MSA, or directly from an MUA. The delivery of email to a user’s mailbox on the email server usually takes place via a mail delivery agent (MDA). Most MTAs have MDA functionality built in. The incoming email messages are stored in each email user’s mailbox on the email server, such as “inbox” by default. Normally, there are several default folders in a user’s mailbox besides “inbox”. These are “sent”, “drafts”, and “trash”. The “sent” folder is for storing sent email messages, and the “drafts” folder is for holding email messages that have not been sent out. The “trash” folder is for storing deleted email messages temporarily.
This is a multi-part message in HIKE format.

--- Boundary(ID_b1Du14drXfb4o0AMWi4s) Content-type: multipart/alternative; boundary="Boundary(ID_b1Du14drXfb4o0AMWi4s)"

--- Boundary(ID_F+8il3RvGqHlkRPw0ikCIw) Content-type: text/plain; charset=gb2312 Content-transfer-encoding: 7BIT
Hello, this is a test.
Peng Lu

--- Boundary(ID_F+8il3RvGqHlkRPw0ikCIw) Content-type: text/html; charset=gb2312 Content-transfer-encoding: 7BIT

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
<meta http-equiv=Content-Type content='text/html; charset=gb2312'>
<meta content='MSHTML 6.00.6000.16414' name=GENERATOR>
<link rel='stylesheet'/>
</head>
<body background="#1f1f1f" style='font-family: Arial; font-size: 2em'>

Hello, this is a test. Peng Lu</body></html>

--- Boundary(ID_b1Du14drXfb4o0AMWi4s)---

Figure 2.1: Email Structure
2.1.4 Email Client

An email client is a computer program that is used to read and send email. Originally, the email client was intended to be a program only for reading the email messages, which would be received into a local mailbox from an email server. Today, email clients provide more functions, such as composing, searching, and managing email messages. We give a detailed introduction in the next chapter.

2.1.5 Email Protocols

Email functions through the use of the following basic set of internet standards, which are supported by virtually all mail servers and email clients.

**Simple Mail Transfer Protocol (SMTP or RFC 2821)** It has become the standard for email transfers on the Internet. The protocol transmits the email. Whenever users send an email message, the email client interacts with the specific email server via SMTP to do the sending. This email server may have conversations with other servers to actually deliver the email via SMTP.

**Post Office Protocol (POP[28]) and Internet Message Access Protocol (IMAP[7])** They specify the email retrieval protocols used by email clients. Generally, email clients using POP connect, retrieve all messages, save them on the user’s computer, delete them from the email server, and then disconnect. In contrast, IMAP supports both connected and disconnected modes of operation. Email clients using IMAP usually leave email messages on the email server until the user deletes them.
2.2 How Email Works

Email messages are exchanged between email servers using SMTP with software like Sendmail [29] or Microsoft Exchange [19]. Users use other email applications, such as Pine [37] or Microsoft Outlook [20], to access the email servers and download their email messages with one of several protocols, including the leading POP and IMAP protocols. Email messages can be stored either on the client or on the server side. The diagram 2.2 shows a typical sequence of events that happens when sender with username plu sends an email to recipient with username kevin.

- The sender "plu" creates an email message in his email client, such as Microsoft Outlook, and clicks "Send". His email client formats this message into standard
email format (i.e. MIME) and uses SMTP to send it to the specified email server at the domain “carleton.ca”.

• The sending email server at domain “carleton.ca” handles all the aspects of mail delivery until this message has been either accepted or rejected by the receiving email server at domain “uottawa.ca”. During this time, the email message is stored at the “sent” folder on the email server.

In detail, the sending email server at “carleton.ca” looks up the recipient’s domain name “uottawa.ca” in the Domain Name System (DNS) to find the mail servers accepting messages for that domain; then sends the message using SMTP to this mail server, which delivers it to the mailbox of the user “kevin”.

• The recipient “kevin” presses the “receive mail” button in his email client, which downloads the message using the Post Office Protocol (POP3) from his email server at the domain “uottawa.ca”.

2.3 Email Security - SSL [9]

Usually Emails are transferred across the internet as plain text. This leaves emails possible to be sniffed out the sensitive information. Encryption technologies enable us to secure our email servers and email clients to protect the email information. There are several methods of securing email, such as Secure Sockets Layer (SSL), Pretty Good Privacy (PGP [5]), certificates, and signatures. In these method, SSL is widely chosen by most email servers and email clients. It provides secure communications on the Internet for data transfers, including email. SSL works at the socket level. Socket is an end-point of a two-way communication link between two programs running on the network. In the email paradigm, one socket is used by the email server and the other by the email client. SSL can protect emails by automatically encrypting the data while it transfers between sockets.
SSL works at a lower level than the standard email protocols such as SMTP, POP, and IMAP. Therefore, these email protocols do not need to be modified to handle connections over an SSL protected channel. Figure 2.3 shows that SSL encrypts the email before it is sent out, and decrypts it after it is received at the other end.

2.4 Email Management

Above, we introduced the email related concepts and explained how email works. Since email has become one of the most important types of document, people developed several methods for email management. Since email is a special type of file, people imitate the methods of file management to organize email messages.
2.4.1 Folder-based Management

In computing, a folder is a Graphical User Interface metaphor for a directory in a file system. In email management systems, a folder is an entity which contains a group of email messages (and even other folders in a hierarchical structure). An email management system must provide the means to name and organize multiple email messages. Therefore, it introduces the concept of folders, which can be described as containers that organize a list of email messages logically. Hierarchical folder structure is commonly used by email management systems. However, this kind of tree structure does have some rigid restrictions. For example, every folder can only have one parent folder, except for the root folder, which has no parent, and every email message can only be placed within a certain folder.

2.4.2 Category-based Management

Category-based management is a new feature of several email clients such as the “Smart Mailboxes” of Apple Mail and the “Search Folders” (also known as “Virtual Folders”) of Microsoft Outlook. Categories allow email messages to be placed in one or more groups in the email management. It means that an email message can be located in one place but be put into multiple categories. For example, the user may want to create a category called “Purchases” that would include email receipts from online transactions. These email receipts may also be put into another category called “Banking”. Currently, category-based email management systems are flat structured. Hence, it suffers several drawbacks. For example, the categories are disorganized and loosely placed, even if there are relationships between some of them. Actually, in the computerized world, the definition of category is more reasonable and flexible. At least, it allows categories to have sub-categories. In other words, it allows categories to be further categorized.
2.4.3 Email Filtering

A mail filter is a piece of software that takes an input of an email message, and analyzes the header or body statements of this email. For the output, it might have several different actions depending on the filtering rules. Many email management systems support filtering rules that can complete various management tasks automatically: the automated sorting of incoming messages into different folders or categories, replying to or forwarding certain messages, and so on. To handle an email message, the rules are evaluated in sequence and the first rule that applies to the item triggers the email client to perform the specific action. In the research on email management [3], the difficulty with rules is that the process of composing a rule is cognitively rigorous and there is a hidden risk of misfiling email messages.

2.4.4 Summary

From the description above we can see that those structures used in managing email messages, such as the folder-based structure and category-based structure, are accepted within the email management systems. There are also some techniques, such as email filtering that has been developed for helping to manage email messages. However, the drawbacks of the current management systems restrain the efficiency and flexibility of email management systems. It is thus necessary to improve email management feature.
Chapter 3

Related Work

This chapter discusses related work. First, we introduce several popular email client applications. Since we concentrate more on how these email clients organize and search emails, we lay a strong emphasis on the study of email management. Next, we compare the features available in these email clients and their usability in dealing with large numbers of email messages. Because our research is focused on category-based email management, we also discuss related work on file management applications.

3.1 A Brief Survey of Email Client Applications

The emails to be sent or received by users would ordinarily be handed over to the email clients; however, most email clients today also include a number of features to better manage the ever-increasing number of email messages that users have to deal with. Many email clients allow for the sorting of messages into folders, searching based on a number of criteria, managing address books, and automatic filtering based on user-defined rules. The development of new features is ongoing. Recently, new features such as “virtual folders” (also known as “search folders”) and the easier creation of filters are making common tasks easier.
Normally, sorted by their characteristics, email clients can be grouped into text-based clients, graphical clients, and web-based clients. With new developments in programming technology, email users are welcoming graphical clients as an alternative to text-based clients because graphical email clients are much easier and more user-friendly to operate. Web-based email clients are a type of website that allows users to read and write email on the World Wide Web, as an alternative to using stand-alone email clients.

First, we introduce Microsoft Outlook Express [21], an outstanding graphical email client, in detail as a reference because of its popularity. We also give brief introductions to several other popular email clients focusing mainly on their management features, and then compare the features available in these applications and their usability in dealing with large numbers of messages.

Next, we introduce several web-based clients such as Yahoo! Mail, Hotmail, and IMP webmail (a webmail system used by the School of Computer Science, Carleton University), and compare their common features.

Finally, we compare and contrast the web-based clients with the graphical clients, then provide a comparison of the differences between them.

### 3.1.1 Review of Microsoft Outlook Express

Microsoft Outlook Express is an email client bundled with certain versions of Microsoft Windows. Outlook Express manages multiple email accounts, lets users compose and receive emails, generates various views to allow the customization of mailbox displays to ease the handling of big amounts of email, and automatically files incoming emails according to user-defined filtering rules.
3.1.1.1 Managing Multiple Email Accounts

Outlook Express manages multiple POP, IMAP, and SMTP email accounts to receive and send emails. This means that users can receive email messages from their various email accounts at different email servers, such as “plu@scs.carleton.ca” and “plu@connect.carleton.ca”. Outlook Express also allows users to send email messages from their various SMTP accounts on different email servers. As shown in Figure 3.1, user “plu” has two email accounts: “plu@scs.carleton.ca” on email server “daVinci.scs.carleton.ca” and “plu@connect.carleton.ca” on email server “connectmail.carleton.ca”. User “plu” can access both of these accounts via Outlook Express.

![Internet Accounts]

Figure 3.1: Outlook Express: Managing Multiple Email Accounts

3.1.1.2 Composing and Sending Email Messages

With Outlook Express, a user can send an email as soon as he/she has finished composing it, or he/she can compose a draft and temporarily save it into the Outlook Express default folder, “Outbox”, then review it later on, and finally send it to the email server. As we described in Section 2.2, Outlook Express connects to the specified email server, then sends the email to the user’s “Outbox” folder on the server. The email server will finish the sending operation. Reply and forward functions are also supported. In addition, users are allowed to include documents, pictures, and even music and videos in their email messages;
the only limitation is file size. As shown in Figure 3.2, many features, especially those related to fonts and graphics, are very useful for composing messages in HTML format.

From: plu@connect.carleton.ca (connectmail.carleton.ca)  
To: lu_peng@gmail.com  
Cc: plu.china@gmail.com, plu@scs.carleton.ca  
Subject: a test of sending email with Outlook Express  

Hello, this is a test.  
Peng Lu  

Figure 3.2: Outlook Express: Composing and Sending Email

3.1.1.3 Retrieving and Reading Email Messages

Each time the user starts up Outlook Express, the program automatically retrieves new email messages and delivers them to the “Inbox” folder. While Outlook Express is running, it checks for new emails automatically in the background by keeping connected to each specified email server and monitoring the user’s mailboxes on these servers every few minutes until the user exits the program.

To read email messages, the user can switch to the “Inbox” folder by clicking the inbox icon in the sidebar. As shown in Figure 3.3, Outlook Express displays a list of summaries of all email messages, which are parsed from the headers of the emails. By clicking a message header summary in the list, the user can read the detailed email contents in the message window below the list. Otherwise, the user can open a new separate window by double-clicking the selected message’s header summary in the list. If the message includes attachments, Outlook Express gives the user the option of opening the attachments either in their native applications (e.g., MS Word or Adobe PDF Reader) or saving the attachments.
Figure 3.3: Outlook Express: Retrieving and Reading Email Messages

to the local hard drive.

3.1.1.4 Deleting Email Messages

In Outlook Express, the user can select a message from the message header summary list and click the "Delete" button to delete the message. As shown in Figure 3.4,
a strikethrough line appears through the message header summary in the list to indicate that the message has been marked for deletion but not yet deleted permanently both in Outlook Express and on the related email server. To delete the message permanently, from both, the user needs to select “Purge”.

3.1.1.5 Managing Email Messages

Using a hierarchical folder structure to manage email messages is a remarkable feature of Outlook Express. A number of operations are available to users for managing email messages. These operations include creating folders and moving or copying email messages from the Inbox to these or other folders.

**Figure 3.5: Outlook Express: Using Hierarchical Folders to Manage Email Messages**

In Outlook Express, the default folders such as the Inbox, Outbox, Sent Items, Deleted...
Items, and Drafts are automatically created. Some folders reside locally on the hard drive and appear under the heading "Local Folders". Other folders reside on the email server where the user’s email account is located. These folders appear under the heading "the name of the mail account". In addition to these default folders, the user can also create new folders to help organize email messages. He/she can easily move a message from one folder to another by dragging the message into the appropriate folder. Figure 3.5 shows an example of how hierarchical folders can be used to manage email messages in Outlook Express.

3.1.1.6 Searching Email Messages

Searching is an important and useful feature of Outlook Express. Selecting "Find" from the tools menu shows a quick search bar above the message index. As shown in Figure 3.6, from here the user can search the current mail folder or all folders. Outlook Express searches the currently selected folder by default. To find the desired message, the user can provide one or more keywords in the four fields (From, To, Subject, Message Content), select the time range, or limit the status of the message (i.e. "message has attachment" or "message is flagged").

Figure 3.6: Outlook Express: Searching Email Messages
3.1.1.7 Filtering Email Messages

The user can employ filters (or rules) in Outlook Express to automatically sort incoming messages into different folders. The user can specify multiple conditions (such as keywords in the sender field or message body) for a single rule. He/She must then specify the detailed conditions or actions for the rule in the “Rule Description” section.

Figure 3.7: Outlook Express: Filtering Email Messages

Figure 3.7 shows an example. The user wants to move new email messages from “Carleton University” that contain the keyword “deadline” in the email body to the “deadline” folder automatically.
3.1.2 Other Graphical Email Clients

3.1.2.1 KMail

KMail [16] is a very functional email client that fits nicely into the K Desktop Environment (KDE) for UNIX and Linux workstations. It has features such as support for IMAP, POP3, multiple accounts, powerful filters, flexible searches, virtual folders, inline attachments, and so on.

Filters In KMail, most aspects of the email header can be selected while creating a filter. The filter actions are extensive and allow basic actions such as moving or copying to folders, as well as more complex operations like forwarding or replying, adding/removing an email header, or even rewriting it. This last action can be used to easily remove strings like

Figure 3.8: KMail: Searching Email Messages
“[mailing-list-name]” from the subject field of mailing lists. The user can choose if he/she want the filter to be applied to incoming or outgoing messages, or to use manual filtering instead. Regular expression matching is supported.

**Searching / Virtual Folders** KMail comes with greatly enhanced search capabilities. As shown in Figure 3.8, more than two search rules can be used, and the header operations are the same as for filters. While searching emails in a folder, the user can choose whether to search in its sub-folders as well. Entering a name for the search on the bottom and clicking “Open” creates a new virtual folder. These virtual folders have more functions than normal folders; for example, selecting ”Search” while viewing a virtual folder brings up the search used to create it with the option of modifying and/or renaming the virtual folder. However, there is no possibility of creating sub-folders within virtual folders.

### 3.1.2.2 Bloomba

In comparison to other major email clients, Bloomba [11] is often considered to be able to organize and search email efficiently. Additionally, Bloomba has other useful functions such as a strong anti-spam ability, smart group, portable datastore, and so on. It introduces the idea of categories and every email can be classified into multiple categories and displayed with different colors. It also automatically groups emails according to subject.

**Searching** The core strength of Bloomba is its search capability. As shown in Figure 3.9, this feature lets users quickly search the text in messages and attachments based on keywords, phrases, or even partial text, no matter where they are buried. Users can create “Saved Searches” that locate messages based on customized search criteria. With Bloomba, users no longer need to spend time maintaining and organizing their inboxes.
Managing Email Messages  There are several useful features in classifying email messages in Bloomba.

The user can create categories to organize their email messages. Messages that are associated with a category appear in the message list pane partially highlighted in the color the user has selected for that category. A message can be assigned to many categories including one of the default categories ("Business", "Community", "Personal") that come with Bloomba, as well as any custom categories created by the user.

Bloomba also uses a hierarchical structure to store the "Saved Searches" according to several predefined categories: "Address Mode", "Account", "Category", "Status", "Priority", and "Smart Group". The user can modify any of these groups, or create new ones.

One notable feature is that Bloomba can group email messages in a thread, a series of messages that are a "conversation" on one subject. Figure 3.10 shows an example of how Bloomba groups emails in a thread. A thread contains the original message and all the
replies to that message. If a reply contains the title of the original message, it is grouped in the thread. Bloomba lets users view their messages with threads turned either on or off. When turned on, the program groups the related messages together in a list that users can expand or collapse using the plus/minus sign next to the most recent message in the thread.

3.1.3 Web-based email Clients

Web-based email clients (also called webmail) are the implementation of an email client as a web application that allows users to access their email through a Web browser, as an alternative to using a desktop-based graphical client such as Microsoft Outlook, KMail, or Bloomba. A webmail client is usually offered by an email service to allow its users to access email stored on the server.

Yahoo! Mail [41] This is a famous web-based email system, one of the largest email providers on the Internet that serves millions of users. There are several ways in which Yahoo! Mail differs from a desktop client. Users can login at the Yahoo! Mail website (http://mail.yahoo.com/) to handle their email messages. Email messages are stored on the server-side. Users do not need to install anything but a web browser, which is normally pre-installed with operating systems. Therefore, it is not platform-dependent.

Since email has its own specific properties such as sender, message status, date, attachment, etc., Yahoo! Mail manages emails as multi-dimensional objects. For example, if a user searches a keyword “deadline” in his/her Yahoo! Mail, several results will be retrieved that are categorized according to different properties such as sender, message status, date, or attachments. If a user wants to find emails from a specific sender, it is easy to pick out the target email message from the results by selecting the sender category.

Windows Live Hotmail [22] Windows Live Hotmail (or Hotmail) is another popular web-based email service provided by Microsoft. It features two gigabytes of storage space,
automatic spell checking, custom filters, searching, and many other features. Users can either login to the Hotmail website (http://www.hotmail.com) by using their web browser or use Windows Live Mail or Microsoft Outlook with Outlook Connector [23] to view their mail outside of the web browser.

The indexed search capability of Hotmail includes searching through the address and subject headers, as well as the message contents for MSN Hotmail Plus accounts or higher. The search results will appear in a temporary new “folder” at the bottom of the folder list, and the user can currently search both “Mail” and “Contacts” as well as “Web” (Depending on which page the user is viewing, e.g. Inbox, Contacts, etc.).

**Gmail [12]** Gmail, officially Google Mail, is a web-based email provided by Google. Users can login at the Gmail website (http://www.gmail.com/) to access their email messages. Gmail offers its users a huge amount of storage space, which is one gigabyte initially. Until July, 2007, it provides almost three gigabytes of free space. There is a search-oriented interface. Gmail enables users to quickly search every email they’ve ever sent or received by using keywords or advanced search features. It also provides a conversation view which is similar to Bloomba’s thread view feature. Gmail is well-known for its use of the Ajax programming technique [18] in its design.

Compared to desktop-based email clients (graphical email clients), there are several advantages and disadvantages to webmail.

**Advantages**

- Email is stored remotely on a server, which means that users can check their email from any computer with a web browser and an Internet connection.

- Webmail is installation free and Platform independent. This means that users do not need to worry about the installation. With a web browser, users simply need to go to the URL of their webmail to access their email. This way, no matter what kind
of operating system is being used, users are able to access their webmail as long as there is a web browser for the system.

- Centralized management of webmail email is another advantage. There are several beneficial features and rules supplied by webmail providers, such as deleting trash email messages every few days, filtering spam, and detecting viruses.

Disadvantages

- Webmail is only offered through a website. As a result, users have to be online to access their webmail.

- Webmail always wraps the email messages in the website’s HTML. This can result in a decrease in the speed of use, especially on a slow network connection. Users may find it difficult to access larger email messages.

- Webmail usually has speed and functionality limitations. Since all the features of webmail are supplied on the server side, to perform an action, users send their requests to the server first and then wait for the server’s response. This process is not efficient on a slow network connection.

- Webmail usually has the limitation of email storage space. Users cannot keep the email messages on their local hard drives; instead, they have to download and save important email messages and delete other email messages.

3.1.4 Comparison of Email Clients

This review is a comparison of the features available in most popular mail clients and their usability in dealing with large numbers of messages. Table 3.1 gives a comparison of the general and technical features of these email client programs. It compares six different
email client. Three of them are graphical email client, and the others are web-based email client.

This comparison is focused on the operating system support, user accessing support, protocol support, features, storage, and management structure. From the table, it is easy to learn that these email clients have the similar functions, and also have their own special features. For example, we found that all web-based email clients in the table are platform independent. This is because that the user can access the webmail from any kind of system with a browser. However, unlike the graphical email client, the webmail has to be accessed remotely. Other than that, the web-based email client supports few protocols. Graphical email clients and web-based email clients also have many things in common. Filtering and searching functions are almost the fundamental functions to every email client. Thread view is a new concept but it has already existed in both graphical email client (i.e. KMail and Bloomba) and webmail (i.e. Gmail). Database technology is not popular in email clients. In the table, only Bloomba is using the database for its storage. Also, the management of email clients are various. There are three kinds of structures for email management: flat folder structure, hierarchical folder structure, and flat category structure.

3.2 File Management Applications

In computing, a file system is a method of storing and organizing files and data in a computer in order to make them easier to find and access. Because email is an important and special type of file, email management systems use file systems for reference and inherit many of their mechanisms. We introduce some of the related works from both the academe and the industry.
<table>
<thead>
<tr>
<th>Kind</th>
<th>Outlook Express</th>
<th>KMail</th>
<th>Bloomba</th>
<th>Yahoo! Mail</th>
<th>Hotmail</th>
<th>Gmail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
<td>Windows</td>
<td>Unix/Linux</td>
<td>Windows</td>
<td>any</td>
<td>any</td>
<td>any</td>
</tr>
<tr>
<td>Access</td>
<td>local</td>
<td>local</td>
<td>local</td>
<td>remote</td>
<td>remote</td>
<td>remote</td>
</tr>
<tr>
<td>Access Offline</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Multiple Accounts</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>POP3</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>IMAP4</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>SMTP</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Filters</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Search</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Thread view</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Database</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Flat Folder</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Hierarchical Folder</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Flat Category</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>hidden</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 3.1: Comparison of Email Clients
3.2.1 Hierarchical File System

Hierarchical File System (HFS) [1] is a file system originally developed by Apple Computer for use on computers running Mac OS. It is designed for use on storage devices such as floppy and hard disks that are partitioned into one or more volumes. When a volume is mounted, a volume reference number is assigned to the volume for identification. In HFS, files are grouped into folders (also called directories) and these folders are themselves grouped into other folders. The locations of files are stored in a single directory structure, called the Catalog File. The Catalog File is a $B^*$-tree that contains the catalog records and is analogous to a UNIX index node (inode). [1, 15] Each Catalog record contains the information of a single file or folder stored in the volume. This contrasts obviously with other file systems, where directory information is isolated to those files in a particular directory. The hierarchical arrangement of files and folders allows users to identify a file or folder uniquely by providing its volume reference number, its parent folder ID, and its name within that parent folder.

It should be noted that several catalog record fields are reserved for the exclusive use of the Finder, the default program used in the Mac OS operating systems. The Finder allows users to access the file system and automatically maintains associations between applications and data files. It acts like the shell in UNIX systems, but uses a graphical user interface. It works with the File Manager to maintain the organization of files and folders in a volume. The hierarchical relationship of folders within folders on the desktop corresponds directly to the hierarchical directory structure maintained in the volume.

However, there are some drawbacks to the hierarchical structure. Using the hierarchy, users are limited to a static organization, within which the knowledge domain narrows progressively from the top to the bottom. While retrieving information, the static organization becomes a restriction. In particular, users have to be familiar with the sequencing of the directories they use to store the information; otherwise, it is hard to reach the expected
Figure 3.11: The Macintosh Hierarchical File System

information when digging in the wrong path. As shown in Figure 3.11, for example, a user is concerned about location, and thus, groups contact information according to place, such as “Ottawa”, “Calgary”, and “Montreal”. This information is stored in the directory paths “/Canada/Ottawa”, “/Canada/Calgary”, and “/Canada/Montreal”. Alternatively, the user can change the topic to “where are the students from” and save the information to “/students/Ottawa”. In this case, there are two paths called “/Ottawa” in the hierarchy, each with different meanings. Therefore, the user cannot find the information about a student from Ottawa, if the user is looking in the directory “/Ottawa” under “/Canada” but not the correct entry “/student”. The situation can be worse, if the user cannot remember where the information about students who come from Ottawa has been stored. It will take a lot of time to scan the entire hierarchy to find the result. Obviously, there is too much redundant searching and the results may be rough. Another drawback results from the catalog file. This file stores both the directory and file records in a single data structure, which causes
performance problems when more than one program is trying to write to it. Any operation that creates or modifies a file has to lock the catalog file. Thus, it is not easy to browse or search the information, nor is it efficient to modify the information in HFS.

3.2.2 Category-based Applications

Figure 3.12: Multiple Hierarchical Category Structure
CASH2 [40] CASH2 is a multiple categorization file management tool for UNIX/Linux file systems designed in the TENEX C shell (TCSH). As a replacement for CASH [42], CASH2 is designed to be the primary means of organizing, grouping, managing, and retrieving files. It has been designed and implemented based on a multiple hierarchical category structure.

Similar to HFS, the lower level categories are more specific and the higher level ones more general in a multiple hierarchical category structure. All files can be classified into many categories. Specifically, a parent category may contain zero or many child-categories and files.

Compared to the hierarchical file structure, a multiple hierarchical category structure is more natural for organizing information, because it allows users not only to classify files into various categories, but also to classify these categories themselves into further categories. There are several advantages of this valuable behavior.

As shown in Figure 3.12, every category except the top category can have multiple paths from the top. This gives users more useful information when searching. Moreover, a category can have a different name in each of its different parent categories. This allows users to describe a certain category in different ways. Users may have different preferences for one category, and having multiple preferences can help them to find their target more easily.

The larger the number of categories, the more the complexity of the hierarchy will increase. As a result, problems can occur with a large number of categories or an excessively complicated hierarchy. To find certain files or categories, there may be several possible hierarchical structures. Another problem is that the multiple hierarchical category structure may be too flexible to construct, since users can classify both files and categories into several categories and even give one category multiple names.
3.2.3 Faceted Classification Applications

Traditionally, a hierarchical classification is one in which a single classification tree is used to represent the application's domain. To allow the assignment of multiple classifications to an object and enable the classifications to be ordered in multiple ways, a faceted classification has already been provided: “It does not assign fixed slots to subjects in sequence, but uses clearly defined, mutually exclusive, and collectively exhaustive aspects, properties, or characteristics of a class or specific subject. Such aspects, properties, or characteristics are called facets of a class or subject, a term introduced into classification theory and given this new meaning by the Indian librarian and classificationist S.R. Ranganathan and first used in his Colon Classification in the early 1930s.” [32] Faceted classification isn’t used only in library systems; it is also a formalization of a communication technique that is used in a wide variety of circumstances.

We can see faceted classification approaches in technology and in everyday life, from “personal information management systems” (e.g. ChurchInfo Project [6]), to customer-support technology (e.g. Primus' SolutionBuilder [25]), to “Metadata Search Engines” (e.g. Flamenco [34], Yahoo!). We will give a detailed example of metadata search engines to explain the usage of faceted classification. Metadata is structured information about files. Normally, metadata has several facets or attributes in various categories. This information is often stored in database record fields and tables. For instance, some characteristics of faceted classification can be seen in Yahoo!. The top-level categories are popularity-based groupings of topics, within which the “hot” subcategories (like “Games” and “Music”) are pushed to the top. Within “Music”, songs have such attributes as artist, title, length, genre, date, and so on.

According to the example given above, we can determine the advantages and disadvantages of faceted classification. A faceted classification begins with the objects being
classified, and is thus a bottom-up system. A faceted system focuses on the essential characteristics of the objects, which is effective for the system of storage and retrieval. It is easy to extend the range of categories rapidly by adding a new facet. Therefore, the system can be used to create deeper and more complex knowledge structures. A faceted classification does have its limitations, however. There is no formal standard for the construction of faceted structures because their classification starts from the objects themselves and the essential characteristics of objects can vary greatly. Also, when extending the faceted structure greatly, the faceted classification schemas do not respond well to the requirement for quick access to popular topics.

### 3.2.4 Formal Concept Analysis Applications

Formal Concept Analysis (FCA [26]) is a method of data analysis that takes a set of objects and their properties, and finds both all objects that share a common subset of properties and all properties shared by one of the natural object clusters. FCA was invented by Rudolf Wille in 1982. Now, it has successfully been applied to many fields in computing such as software engineering, AI and information retrieval. As an example, an on-line documents conceptual reorganizer using the conceptual diagrams of an automatically constructed class hierarchy has been implemented, which is called CREDO (Conceptual Reorganization of Documents [4]).
In a formal concept analysis system, the objects are organized in a lattice rather than a hierarchy. The lattice can automatically be constructed from objects and their attributes, without creating a hierarchy of objects by hand. As shown in Table 3.2, the elements on the left side are formal objects; the elements at the top are formal attributes; and the relationships between them are represented by the crosses. In this example, the formal objects are drinks and food: beer and ice wine are alcohol drinks, coke is soft drink, and chocolate is food. The attributes describe whether these drinks and food are alcohol or non-alcohol, whether they are sweet, or beverage.

![Concept Lattice Diagram](image)

**Figure 3.13: An Example of Formal Concept Analysis**

Figure 3.13 shows a line diagram of a concept lattice corresponding to the formal context in Table 3.2.

Since a formal concept analysis system always automatically generate the lattice of the objects and their attributes, there probably are more relationships in a FCA system
than a hierarchy of categories created manually. With a large amount of objects and their attributes, a FCA system will create all possible relationships. However, users may not need some of them. That generates the redundancies [14]. It also confuses users while retrieving the information because of a large amount of possible choices created by system or by users themselves.

3.3 Summary

From the related works above, we can see that the email clients described have the ability to manage email messages via various methods including filters, virtual folders, and categories based on the flat folder or category structure. However, with the ever-increasing amount of email messages, it is difficult to manage the vast quantity of messages with such fundamental structures. According to the file management systems that we study, we can also see that the category structure has been used in a wide variety of circumstances and that many related works have been done based on it. The hierarchy structure represents well the meaning of the document or information, while its other nice features are also likely to provide a more natural and flexible way to organize and search for information. Moreover, faceted classification shows another point of view that is based on the objects to be classified. We analyze the different structures in these file management programs. Based on our study, we will discuss the extended category structure for our email management system in the next chapter.
Chapter 4

Our Approach

EmailCat is a multiple dimensional hierarchical category-based email management system. Besides having all the features of a normal email management system, it also gives a more flexible category structure for managing emails.

4.1 Category

According to the Oxford English Dictionary, a category is a class or division of people or things having shared characteristics. Within the domain of computer file systems, a category is a user created object assigned to a group of items based on their similarity or certain conditions. In email management systems, a category is a named group to which emails are assigned based on common characteristics or certain conditions. For example, several emails saved in the email management system can be described by their shared properties, such as “greetings”, “friends”, “last month”, etc. Therefore, these properties can be used as category names to classify emails. A category name is usually defined by the user based on some aspect of a group of emails. On the other hand, it must also be easy for the user to remember. Users can also categorize emails using some of the common characteristics of emails shown in the following examples:
• The status of an email, such as "new", "seen", "sent", "deleted", etc.
• The related people of an email, such as the sender and the recipients.
• The received time of an email, such as "2006", "June", "11", etc.
• The subject of an email or what it is about, such as "Greetings", "IT News", "Schedule", etc.
• The keywords in the email content, such as: "deadline", "dollars", "mysql", etc.
• The file type of the attachments in an email, such as: "PDF", "JPG", "MPG", etc.

In Section 3.2, we introduced several file management structures based on category and analyzed the existing problems of these structures. To solve these problems, we will now introduce an extended category structure which is from Yifeng Gao’s work.

4.2 Multiple Dimensional Hierarchical Category Structure

As an improvement of the Multiple Hierarchical Category Structure [10, 40], the Multiple Dimensional Hierarchical Category Structure is introduced. So far, the concept of multiple dimensional data structures has been well known in the field of database technology. Multiple dimensional data structures can be in a data warehouse system using such techniques as the "star schema" [38], or they can be implemented in an OLAP system using data cubes and hypercubes to store and view the relational data efficiently. Finally, they can also be implemented in a multi-dimensional category model [36]. In a multi-dimensional category model, information can be classified into various categories basing on multiple dimensions. However, the categories within each dimension are unstructured and cannot be further classified. Therefore, we construct a Multiple Hierarchical Category Structure inside each dimension.

42

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Within a Multiple Dimensional Hierarchical Category Structure, information can be classified into multiple dimensions, and further classified into multiple categories hierarchically within each dimension. A category can be classified into a hierarchy of multiple categories within a dimension, but it cannot be classified into other dimensions. At this point, we keep the maximum flexibility for classifying the information, but set restrictions for the categories to make the category structure not as excessively flexible as the Multiple Hierarchical Category Structure.

As shown in Figure 4.1, there are three dimensions, “location”, “people”, and “time”, and three isolated hierarchies within each dimension. Each dimension has its own knowledge domain and does not intersect with any other. However, an email can carry information in several different domains. For example, user Lupeng received an email from his friend Kevin, a student of the University of Ottawa, in September, 2006. In this email, Kevin invited Lupeng to join his birthday party on September 14, 2006. The user can normally determine two dimensions, “Time” and “People”, from the email header. Furthermore, he/she may determine other dimensions such as “Location” and “Topic” from the email body, the content of the email. This email could be categorized into the following categories: “time”, “2006”, “September”, “14”, “people”, “Kevin”, “Lupeng”, “Location”, “Ottawa”, “University of Ottawa”, “topic”, “birthday party”, etc. These categories can belong to several possible hierarchies by using the Multiple Hierarchical Category Structure. For instance, “Ottawa” can be associated to any category such as “city” or “student” based on the user’s needs. Users can be confused or misled by this kind of ambiguity, since they may forget the details of how their category structure has been setup as the amount of information being stored increases. However, our extended dimensional category structure can overcome this shortcoming. It helps users make their category structure more natural and logical. Specifically, it gives users limited dimensions at the top of the structure to make the structure more easily understandable. Users are able to look for information from any dimension that covers a wide knowledge domain. Within a dimension, users are given a
Figure 4.1: Multiple Dimensional Hierarchical Category Structure
flexible multiple hierarchical category structure. Synchronously, other dimensions can be viewed at any time as references. Users can switch between different dimensions easily or even browse several dimensions at the same time. For example, to search an email containing information about “Kevin’s birthday party on September 14, 2006”, user Lupeng may begin the search from the “people” dimension to find “Kevin”, then browse in a set of the results in other dimensions - such as “time” and “topic” - to refine the results into a small range of emails dated September, 2006, from Kevin, and about a “birthday party”. Finally, user Lupeng can retrieve the exact information from the result. User Lupeng can also start the search from the “time” dimension or “topic” dimension. He will always get the same result set in the end.

In general, for categories, the dimensions are parallel and isolated to each other, and the sub-structure inside any dimension is hierarchical. While searching, users can always refine results by browsing the Multiple Hierarchical Category Structure within a dimension. The deeper the categories are, the more refined they are as well. Therefore, users can continue to refine the set of results by browsing another dimension related to the desired information. The more dimensions the users browse, the more accurate will be the information retrieved.

4.3 Definitions of the Structure

Referring to the definitions from the Multiple Hierarchical Category Structure [10, 40], we give the corresponding definitions of our category structure:

Email In general, email is a kind of file that is a stream (sequence) of bits stored as a single object. In our category structure, every email has a unique identity to be located in the category structure, and also has the properties we described before: sender, recipients (including To, Cc and Bcc), subject, size, time, content and attachment. It is represented as a node in a directed graph.
**Category** In our category structure, a category is an object that can have a group of child categories or emails. It also can also belong to more than one parent category. Therefore, a category can be represented as a node in a directed graph. As shown in Figure 4.1, all numbered nodes are categories. The category is the essential structure concept in our approach, and has several of the following main properties: unique category id, its level in the category structure, the number of child-categories and emails in it, and the time of its creation.

**Dimension** In our category structure, a dimension is a special category that can have a group of emails or child categories, but only one parent category, the top category. Also, the child categories of a dimension cannot be classified to another dimension, although emails can be classified to multiple dimensions. The dimension has the same properties as the category. However, unlike a normal category, it is always in the next level to the top category in the category structure.

**Category Classification** In our category structure, a category name is used to represent the classification of a category within its parent category. It is not meaningful or relevant with regards to any category except its parent category. A category also belongs to the same dimension of its parent category (except the top category). A category name is illustrated as a named directed edge in the directed graph. The head of this edge represents the child category, and the tail represents the parent category. The child category and the parent category belong to the same dimension (not including the top category). In Figure 4.1, “Student” is the name of category 6 with regard to category 2; therefore, category 6 is the child category, and category 2 is the parent category. Both of them belong to dimension 2 - people.

**Multiple Dimensional Hierarchical Category Structure** A multiple dimensional hierarchical category structure is a limited directed-acyclic-graph (DAG). According to the
definition of the Multiple Hierarchical Category Structure [10, 40], we give the definition
of the Multiple Dimensional Hierarchical Category Structure.

\[ G = (D \cup E, R, t) \]

in which

\[ D = (C, N, d, l, p, in, out), \]

and

\[ R = (N \cup U) \]

- \( D \) is a set of dimensions,
- \( E \) is a set of emails,
- \( R \) is a set of edges (relationships),
- \( N \) is a set of named edges for categories,
- \( U \) is a set of unnamed edges for emails,
- \( t \in C \) is the unique top of the graph,
- \( C \) is a set of categories,
- \( d \) is a function that maps a category to a dimension in the graph,
- \( l \) is a function that maps a category to a number that is the level of the category in the
  graph,
- \( p \) is a path function that maps the category to its paths from the top,
- \( in \) is a function that gives the number of incoming edges of a node,
- \( out \) is a function that gives the number of outgoing edges of a node,

such that the following hold.
1. For each $c \in C$ other than $t$, $\text{in}(c) \geq 1$ and $\text{out}(c) \geq 0$, i.e., a category other than top must have a parent category and can have 0 or more child categories and emails, therefore, for $t \in (C')$, $\text{in}(t) = 0$ and $\text{out}(t) \geq 0$;

2. For each $e \in E$, $\text{in}(e) \geq 1$ and $\text{out}(d) = 0$; i.e., an email must have a parent category;

3. $l(t) = 0$; i.e., the highest level has the lowest level value;

4. $d(t) = \emptyset$; i.e., the top has no dimension;

5. Let $c, c_1, c_2, \ldots, c_n \in C$, such that, for each pair $c_i$ and $c$, there is a named directed edge $n_i \in N$ from $c_i$ to $c$; i.e., $c$ is a child category of $c_i$, and then $d(c) = d(c_i)$ for $c_i \neq t$, and $l(c) > \max(l(c_1), l(c_2), \ldots, l(c_n))$ and $p(c_i) \subseteq p(c)$ for $1 \leq i \leq n$.

We use Figure 4.1 to show an example. Category 0 is the top of the structure and is at level 0. The circle nodes represent categories, and the rectangle nodes represent emails. A category has at least one parent category and zero or more child categories and emails. A category has exactly one dimension other than top category. An email has at least one parent category and can have one to many dimensions, or even no dimension. Category 13 is at level 3, which is lower than the level of both its parent categories 6 and 7 at level 2. The paths to category 13 include the paths of both its parent categories ("top/ people/ student/ Kevin" and "top/ people/ friend/ Kevin"). All of these categories, category 13 and its parent categories 6 and 7, are in the same dimension - dimension 2.

**Distance between categories** In our category structure, a category may have more than one possible paths to another category. Therefore, the distance between two categories is the number of edges of the shortest path between two categories. If there is no path from one category to another, the distance between the two categories is NULL.
4.4 Database Modeling and Design

As a multiple dimensional categorization email management system, the design goal of system is to realize the multiple dimensional hierarchical category structure in its application.

4.4.1 Entity-Relationship Model and Database schema

According to the definitions we describe in the previous section, our category structure contains two main kinds of data entities: email and category. In the structure graphic, they are represented by nodes that are connected by directed edges. We use the entity-relationship model to describe our structure.

As shown in Figure 4.2, there are two key entities (Category and Email) and three relationships (Category Classification, Email Classification and Contain) in the model.

Based on the database design, we convert this ER diagram into a relational database schema. Figure 4.3 shows the schema of the tables related to email management part in our system.

**Email** This is a kind of unstructured data. To store and manage it properly and efficiently, we divide email into two parts - email head (information) and email body (content) - and store these two parts in two isolated tables. They are related by a unique email identity string. The email head is a brief description of an email, and every email head part is related to an email body. The body is an unstructured data fragment containing text information or any other kind of information, such as images, audio, or even video files. Therefore, in our entity-relationship model, an email has a unique message identity (Eid), a subject, status (flag), contact information (i.e. the sender and recipient, the later of which also includes Cc - Carbon Copy, and Bcc - Blind Carbon Copy), the sent and received dates of this email, the size, the encoding charset, and the email body content (saved as raw data into a isolated
Figure 4.2: Entity-Relationship Model
Figure 4.3: Database Schema
table and related to email head information by a unique email identity string).

**Category**  It defines the category node in the multiple dimensional hierarchical category structure and describes the information of each node. It includes the following main attitudes: a unique category identity (Cid), the level information, which restricts the operations of the category, and other information such as the number of child categories, the number of classified emails in this category, etc.

**Contain**  It represents the relationship between the header and content of an email. We separate the email content from the email’s header information because the size of the email content always takes more space than all the other pieces of information, especially when there are attachments within the email. Therefore, we separate the email content from other information of an email and treat it as a data stream to improve the performance. As a result, an email always has one-to-one email content, and vice versa.

**Email Classification**  It describes the relationship between emails and categories. A category can classify zero to many emails, and an email can be classified into one or more categories in any dimension. Therefore we can retrieve the related categories of every email according to the categories' unique ids (Cids). Also, every email can be retrieved from one or more categories by its unique id (Eid).

**Category Classification**  It describes the relationship between categories. A category can classify one or more child categories, and at the same time be classified to zero to many parent categories (the top category has no parent category). We use multiple records to represent one or more relationships. Therefore, each category can have multiple parent categories by retrieving the records according to its unique category id (Cid). Also, every category except the top category has a dimension. Each category belongs to only one
<table>
<thead>
<tr>
<th>Email</th>
<th>subject</th>
<th>status</th>
<th>sender</th>
<th>recipient</th>
<th>cc</th>
<th>bcc</th>
<th>date</th>
<th>size</th>
<th>charset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Party on Sunday</td>
<td>ANSWERED</td>
<td><a href="mailto:kevin@uottawa.ca">kevin@uottawa.ca</a></td>
<td><a href="mailto:piu@carleton.ca">piu@carleton.ca</a></td>
<td>NULL</td>
<td>NULL</td>
<td>2006.09.14</td>
<td>23k</td>
<td>UTF-8</td>
<td></td>
</tr>
<tr>
<td>Survey reminder CGPSS</td>
<td>SEEN</td>
<td>Carieton <a href="mailto:universitygetColor@carleton.ca">universitygetColor@carleton.ca</a></td>
<td><a href="mailto:piu@carleton.ca">piu@carleton.ca</a></td>
<td>NULL</td>
<td>NULL</td>
<td>2007.01.26</td>
<td>2k</td>
<td>UTF-8</td>
<td></td>
</tr>
<tr>
<td>FW Order Confirmation</td>
<td>SENT</td>
<td><a href="mailto:piu@carleton.ca">piu@carleton.ca</a></td>
<td><a href="mailto:willis@hotmail.ca">willis@hotmail.ca</a></td>
<td>NULL</td>
<td>NULL</td>
<td>2007.03.24</td>
<td>71k</td>
<td>GB2312</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Data Sample of Email Table

<table>
<thead>
<tr>
<th>Category</th>
<th>cid</th>
<th>level</th>
<th>subcnum</th>
<th>subenum</th>
<th>description</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>56</td>
<td>168</td>
<td>root category</td>
<td>2006.07.11</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>168</td>
<td>people category, 1st level</td>
<td>2006.07.11</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>3</td>
<td>0</td>
<td>11</td>
<td>people Kevin, 3rd level</td>
<td>2006.09.03</td>
</tr>
</tbody>
</table>

Table 4.2: Data Sample of Category Table

dimension; however, at the same time, one dimension can have many categories. For every category, a dimension id is used to represent this relationship. Also, category name (Cname) is used to represent the mapping to that category from its parent category. Every category can have multiple category names since it can have many parent categories.

4.4.2 Data Example

Based on the schema of all tables we described above, Table 4.1 to Table 4.5 shows how the data is stored in the tables, based on the sample category structure displayed in Figure 4.1.

The entity Email is mapped to the table Email. “Eid” is the unique identifier that identifies the email, which is automatically generated when sending the email so that it becomes the primary key in the table.

The entity Category is mapped to the table Category. “Cid” is the unique identifier that identifies the category, and it thus becomes the primary key in the table.
### Email Content

<table>
<thead>
<tr>
<th>eid</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:00076ef32d4dccc1050201a8c0@IBM7DEB">00076ef32d4dccc1050201a8c0@IBM7DEB</a></td>
<td>... ...</td>
</tr>
<tr>
<td>&lt;01c76e745e02b151056c8228e01b9vwx&gt;</td>
<td>... ...</td>
</tr>
<tr>
<td><a href="mailto:8217c1cbef74577a9d4c826112b07d@e-academy.com">8217c1cbef74577a9d4c826112b07d@e-academy.com</a></td>
<td>... ...</td>
</tr>
</tbody>
</table>

Table 4.3: Data Sample of Email Content Table

### Email Classification

<table>
<thead>
<tr>
<th>cid</th>
<th>eid</th>
<th>description</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td><a href="mailto:00076ef32d4dccc1050201a8c0@IBM7DEB">00076ef32d4dccc1050201a8c0@IBM7DEB</a></td>
<td>... ...</td>
<td>2006.09.14</td>
</tr>
<tr>
<td>13</td>
<td><a href="mailto:00076ef32d4dccc1050201a8c0@IBM7DEB">00076ef32d4dccc1050201a8c0@IBM7DEB</a></td>
<td>... ...</td>
<td>2006.09.14</td>
</tr>
<tr>
<td>16</td>
<td><a href="mailto:00076ef32d4dccc1050201a8c0@IBM7DEB">00076ef32d4dccc1050201a8c0@IBM7DEB</a></td>
<td>... ...</td>
<td>2006.09.14</td>
</tr>
<tr>
<td>17</td>
<td><a href="mailto:00076ef32d4dccc1050201a8c0@IBM7DEB">00076ef32d4dccc1050201a8c0@IBM7DEB</a></td>
<td>... ...</td>
<td>2006.09.14</td>
</tr>
<tr>
<td>20</td>
<td><a href="mailto:00076ef32d4dccc1050201a8c0@IBM7DEB">00076ef32d4dccc1050201a8c0@IBM7DEB</a></td>
<td>... ...</td>
<td>2006.09.14</td>
</tr>
</tbody>
</table>

Table 4.4: Data Sample of Email Classification Table

### Category Classification

<table>
<thead>
<tr>
<th>parent_cid</th>
<th>cid</th>
<th>cname</th>
<th>dimension</th>
<th>description</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>top</td>
<td>NULL</td>
<td>root category</td>
<td>2006.07.11</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>people</td>
<td>2</td>
<td>people category</td>
<td>2006.07.11</td>
</tr>
<tr>
<td>7</td>
<td>13</td>
<td>Kevin</td>
<td>2</td>
<td>people: Kevin</td>
<td>2006.09.03</td>
</tr>
</tbody>
</table>

Table 4.5: Data Sample of Category Classification Table
The email content is mapped to the table EmailContent. The relationship Contain is mapped in this table as well. The primary key “Eid” in the table Email is mapped as the foreign key in the table EmailContent.

The relationship EmailClassification is mapped to the table EmailClassification. The primary keys in both Category and Email tables are mapped as the foreign keys in this table. The combination of “Cid” and “Eid” uniquely identifies an email classification relationship, and becomes the primary key in the table.

The relationship Category Classification is mapped to the table CategoryClassification. The primary key in the Category table is mapped as foreign keys in this table, and the combination of “Parentcid”, “Cid”, and “Cname” uniquely identifies a category classification record, and becomes the primary key in the table as well.

In the following chapter, we will explain how to design and implement our system based on this database model.
Chapter 5

System Implementation

In the previous chapter, we discussed the database schema of the categories. In this chapter, we describe in detail the design and implementation of our email system, EmailCat, based on the multiple dimensional hierarchical category structure. Our dimensional category-based email management system is designed to manage email in the category structure. This task is accomplished in several kinds of operations: general email client-side operations, category and email managing operations, dimensional category-based searching operations, and other operations. We discuss the details in the following sections.

5.1 System Architecture

We have implemented our system in JavaServer Faces (JSF [31]) framework and Ajax [18], which are Java-based Web application frameworks. By using JavaMail API [30], which can be combined with other packages of the Java application, we deal with mail servers, manipulate emails and our category structure in an embedded HSQLDB database [35], and represent this web application in an embedded Tomcat web application engine [33]. The whole application can be installed on the desktop as a standalone application or used as a normal web application. It is also platform independent. The browser becomes the interface
through which the user manages data and performs services either on the local machine or on the web. In addition, it is an extensible environment for which we can develop more functions later based on our category structure with the Java techniques described above. For example, these functions can include scheduling calendars, and managing the address book, news, photos, documents in the computer, etc. These applications are able to make the web an ever richer environment for getting tasks done. In our system, a local web server (Tomcat embedded version) handles the data delivery and content display from the local machine to the browser. It has been highly optimized for the task of email management, but is also capable of running locally installed web applications.

**MVC Framework** We have chosen to use Java technology because Java runs on almost desktop platforms, including Windows, Mac, Linux, and etc., as well as support servers, personal computers, and mobile devices. JSF and Ajax are popular frameworks, based on the Model-View-Controller (MVC) design pattern, which simplifies the development of user interfaces for Java applications. As a result, developers can separate data model and user interface view concerns. It allows developers to concentrate on either data model or user interface view at a time. It's fully supported developers to implement a fat-client application based on an abundant set of user interface components supplied by JSF and Ajax.

Figure 5.1 shows the diagram of our system architecture based on the MVC framework. In the diagram, the controller is the mediator between the view and the data model. In this part, a set of APIs is used for representing user interface components and managing their state, handling events and input validation, defining page navigation, communicating with email servers, and connecting with the database. The User Interface (the view part) is the presentation of the users' web browsers, which uses JavaServer Pages for its display technology. JSF and Ajax supply plenty of components to the user interface. They make the web application able to run as a standalone application on desktop.
Figure 5.1: System Architecture
**Tomcat embedded**  We use a local embedded web server to handle the data delivery and content display from the local machine to the browser. This local server will likely be highly optimized for its task, but will also be capable of running locally installed web applications. The Tomcat Embedded version can be embedded using the Embedded class, a convenient class for embedding a web server servlet container environment inside another java web application. Compared to other products such as Jetty and Resin, Tomcat Embedded has two advantages: it is light-weight and has outstanding performance. Also, Tomcat Embedded is platform independent since it is embeded inside the application.

**JavaMail API**  Since our design aims to incorporate mail facilities into our platform-independent Java application, we chose to use JavaMail API in our application to communicate with mail servers and manipulate emails on mail servers. JavaMail API provides a platform-independent and protocol-independent framework to build mail and messaging applications. It offers a protocol independent model for working with IMAP, POP, SMTP, MIME, and most Internet-related messaging protocols. It is also simple to use because it uses features of the Java programming language not available to other APIs. Additionally, it uses the Java programming language’s object model to avoid implementation complexity. Moreover, it fits with other packages for the Java application in order to facilitate its use with other Java APIs. We simply just include the JavaMail class package in our application to work with its APIs for implementing our mail functions.

**HSQL database Engine**  HSQLDB is a lightweight 100 percent Java SQL Database Engine, best known as a free, fast, relational database management system. This software is completely free under their BSD-style license, which allows users to use and distribute for free. According to the test done with PolePosition [24], a benchmark test suite for comparing database engines, the speed of HSQLDB is over 20 times faster than some other products, such as MySQL, Derby, etc. [35] It can be used as an embedded database engine.
by using the embedded class package inside any java application. It is much easier to use than most database systems, which need a separate installation with the application. Another reason for using HSQLDB is that HSQLDB is a good choice for the development, testing, and deployment of database applications in the Java language. We use it in our system because our system is a web interface application based on a web application framework, which offers flexible and resourceful APIs for HSQL database. It not only makes our system work more efficiently and reliably, but also simplifies our implementation.

Figure 5.2 shows the expanded system architecture, in which we can see that the system is generally divided into four parts: user interface, controllers, models, and data.

- **User Interface:** It is used to interact directly with users by both receiving user requests and displaying the corresponding results and responses to the users. The authorized users can manage their own information, their email accounts, the categories, and their emails. There are many functions of the user interface: user login, user registration, category creation, category edit, category deletion, email load, email composition, email deletion, email classification, email import, email search, and so on.

- **Controllers:** These are used to get user requests from the user interface, pick up the corresponding object models to do the processing, and respond with the results. There are four kinds of controllers in the system: user account controllers, category controllers, email controllers and search controllers.

- **Models:** These are used to handle directly database access, email servers access, and local file system access.

- **Data:** This layer is divided into two parts: the database and files on the disk. The persistent data stored in DBMS is the category structure and emails. In our system, we apply HSQLDB to store the persistent data of our category structure. EmailCat also stores email attachments as files on the disk. Additionally, it is able to import emails saved on the local drive to our system.
User Interface includes the functions such as user login, user register, category create, category edit, category delete, e-mail load, e-mail compose, e-mail delete, e-mail classify, e-mail import, e-mail search, and etc.

Figure 5.2: Expanded System Architecture
5.2 Default Values

To operate email and categories, we set up some default values in our system. According to the structure of email header fields, there are some corresponding default dimensions: time, people, status, and topic.

- **Time** - Every email is related to time. For example, there are two fields in an email header - the received time, the sent time, and even time sensitive content in the email message.

- **People** - This is a default dimension of every email. People use email to exchange information; therefore email must be related to people. In the email header, the sender and recipients fields are related to people.

- **Status** - This corresponds to the status flag of emails as we described in Section 2.2.2. There are seven default subcategories of Status: RECENT, SEEN, DELETED, ANSWERED, DRAFT, FLAGGED, and USER.

- **Topic** - Topic can also be a field of the email header - Subject. It is related to the email content. Topics can be organized by user defined sub-categories defined semantically.

There are several default categories as well, such as Inbox, Sent, and Trash, which are commonly used in most email clients.

5.3 User Access Mechanism

As we explained how email works in Section 2.3, users need to set up one or several email accounts on their email client to communicate with the email servers and handle their emails from the given accounts. We introduce a basic user access mechanism as a must
for our application. This can guarantee the management security of the users holding their own authorization to their account information and their emails. Also, our access mechanism is a must for the category structure. This can guarantee the management efficiency of the category structure. Individuals holding their own authorization will not mess up the whole category structure since it is organized by individual.

Every registered user holds permission for manipulations including email account managing, email operations, category and email managing, and other operations.

5.4 General Email Client-side Operations

Like other email applications, our system has all the basic operations of an email client, such as “email account managing”, “email checking and receiving”, “email reading”, “email composing and sending”, and “email deleting”.

5.4.1 Email Account Managing

To access multiple email accounts, we store the necessary information for connecting to the mail server to the database. Figure 5.3 shows a screenshot of the email account managing console.

5.4.2 Email Checking and Receiving

As we described in Section 2.2.5, there are several mail retrieval protocols for accessing an email server from a local email program. We handle the most common protocols (POP3, POP3 with SSL, IMAP and IMAP with SSL) in our system. On an email server, since emails are stored as a queue in the sequence of their arriving time, to check new emails we simply need to know the position of the oldest new email on the email server and the rest of email queue will all be new.
Algorithm: Check New Email

**Input:** Email Account Information

**Output:** New Email Message List: *new-list*

1. get all saved emails of this account from database to a list: *saved-list*
2. get the full email list of this account on the email server: *full-list*
3. **IF** email *e* in *full-list* is not in *saved-list** THEN
4. get sublist *new-list* from *e* to the end of *full-list*
5. save all emails in *new-list* to database
6. **END IF**

Figure 5.4: Algorithm of the Check New Email Operation
5.4.3 Email Reading

Based on the email structure we described in Section 2.2.2, we parse the email header and email body separately, and then read every field from the email header. For the email body, there are many different cases to consider. In particular, messages may have an arbitrary nesting of multiple parts and may also include embedded HTML and any type of attachments such as images, sound clips, or PDF documents. Therefore, to handle these cases in our system, we need to retrieve every email body part located, separate them by a special boundary string, and parse them to get the actual information contained.

Figure 5.7 shows a screenshot of the email reading console.

5.4.4 Email Composing and Sending

To compose, we make the email message follow the email structure described in Section 2.2.2, i.e. by filling in the necessary fields for sending the message, such as sender’s mail account (a user’s account), recipient (including To, CC and BCC), subject, content,
Algorithm: Read Email

Input: An Email Message
Output: Every Header Field and Body Part Content List: part-list

1. get the email header e-header and the body e-body
2. get the every field from e-header
3. Function parseBody(e-body)
   FOR each Part of the email body
   get the content of current-part
   IF current-part is not multiple-part THEN
     parse the part content and add the output to part-list
   ELSE
     call Function parseBody(current-part)
   END IF
END FOR
END Function

Figure 5.6: Algorithm of the Read Email Operation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
attachment and so on.

**To send,** we build the connection to the associated email server with the selected mail account. The connection is built via the SMTP protocol, which is introduced in Section 2.2.5. When an email is sent, it is also saved in the database and related categories such as **Status - sent** by default.

![Email Composing and Sending](image)

Figure 5.8: Email Composing and Sending

Figure 5.8 shows a screenshot of the email composing and sending console.

### 5.4.5 Email Deleting

**DELETED status** Every email in our system has a flag for marking the email status, which is one of the fields in the email header. This flag can be any value of the set: RECENT, SEEN, DELETED, ANSWERED, DRAFT, FLAGGED, and USER. For most email clients and servers, a delete operation only changes the email flag to DELETED the first time, which solves the problem of deleting useful emails by accident. An email message flagged as DELETED will be really deleted from server by performing a DELETE action or a PURGE command in some systems. Therefore, we deal with the delete operation in
the same way as other systems.

<table>
<thead>
<tr>
<th>Algorithm: Delete Emails</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A List of Emails To Be Deleted: <em>email-list</em></td>
</tr>
<tr>
<td>1 FOR each email account of the user</td>
</tr>
<tr>
<td>2 connect to the email server</td>
</tr>
<tr>
<td>3 get the full email list of this account on the mail server: <em>full-list</em></td>
</tr>
<tr>
<td>4 FOR each email <em>e</em> in <em>email-list</em> is in <em>full-list</em></td>
</tr>
<tr>
<td>5 IF the status flag of <em>e</em> is <em>DELETED</em> THEN</td>
</tr>
<tr>
<td>6 delete <em>e</em> from <em>full-list</em> on mail server</td>
</tr>
<tr>
<td>7 delete <em>e</em> from the database</td>
</tr>
<tr>
<td>8 ELSE</td>
</tr>
<tr>
<td>9 change the status flag of <em>e</em> to <em>DELETED</em> in the database</td>
</tr>
<tr>
<td>10 END IF</td>
</tr>
<tr>
<td>11 remove <em>e</em> from <em>email-list</em></td>
</tr>
<tr>
<td>12 END FOR</td>
</tr>
<tr>
<td>13 disconnect to the mail server</td>
</tr>
<tr>
<td>14 END FOR</td>
</tr>
</tbody>
</table>

Figure 5.9: Algorithm of the Delete Email Operation

### 5.5 Category Managing Operations

Generally, the core category operations include “add”, “delete”, “modify”, “move”, and “path trace”. Although there are differences between the operations, they do have some elements in common. Among the operations, we find that there are several basic operation rules that must be followed, which are Dimension Detecting, Directed Cycle Detecting, Path Computing, Relationship Creating, and Relationship Removing. Other operations will be executed based on these rules.

#### 5.5.1 Category Relation Retrieving Operations

Before performing an operation on a category, it is necessary to know the details of the category’s relationships. Starting from a category, we conduct the necessary operations to retrieve all of the offspring of the category from top to bottom, and all of the ancestors from
bottom to top. Based on these retrieving operations, we can perform further operations such as cycle detecting, relationship creating, and relationship removing.

**Offspring Retrieving**  To find all the offspring categories of a category node \( n \), we conduct a breadth first recursion. Figure 5.10 shows an example. We intend to find all offspring categories of the category 0 level by level, until no further nodes are found.

![Figure 5.10: Example: Offsprings Retrieving](image)

**Ancestor Retrieving**  To find all the ancestor categories of a category node \( n \), we also conduct a breadth first recursion as we did for **Offspring Retrieving** but in the opposite direction: bottom-to-top. We describe the detail algorithm as shown in Figure 5.12.

**5.5.2 Category Detecting Operations**

While performing an operation on a category, we need to keep our structure in the correct shape. As described in Section 4.6, every category (except top category) must be in exactly one dimension. Also, there is no directed cycle existing in the multiple dimensional hierarchical category structure.
Algorithm: Offsprings Retrieving

Input: A Category Node \( n \)
CategoryClassification database \( CC \)

Output: A list of all offsprings: \( o\text{-}list \)

1. IF \( n \) is the top category THEN
2. put all records in \( CC \) to \( o\text{-}list \)
3. remove \( n \) from \( o\text{-}list \)
4. ELSE IF \( n \) links to the top category directly THEN
5. put all records in the same dimension of \( n \) in \( CC \) to \( o\text{-}list \)
6. remove \( n \) from \( o\text{-}list \)
7. END IF
8. Initialized an empty queue: \( Q \) for category nodes
9. enqueue \( n \) into \( Q \)
10. WHILE \( Q \) is not empty
11. dequeue \( Q \) to HEAD
12. FOR each Child of HEAD from \( CC \)
13. IF Child is not in the \( o\text{-}list \) THEN
14. add Child to \( o\text{-}list \)
15. enqueue Child into \( Q \)
16. END IF
17. END FOR
18. END WHILE
19. return \( o\text{-}list \)

Figure 5.11: Algorithm of the Offspring Retrieving Operation
Algorithm: Ancestors Retrieving

Input: A Category Node n
CategoryClassification database CC

Output: A list of all ancestors: a-list

1. IF n is the top category THEN
2. set a-list to empty
4. ELSE IF n links to the top category directly THEN
5. add the top category to a-list
7. END IF
8. Initialized an empty queue: Q for category nodes
9. enqueue n into Q
10. WHILE Q is not empty
11. dequeue Q to HEAD
12. FOR each Parent of HEAD from CC
13. IF Parent is not in the a-list THEN
14. add Parent to a-list
16. enqueue Parent into Q
15. END IF
18. END FOR
19. END WHILE
20. return a-list

Figure 5.12: Algorithm of the Ancestor Retrieving Operation
**Dimension Detecting**  This operation is used to detect whether a category crosses over dimensions in the multiple dimensional hierarchical category structure to guarantee the correctness of the structure.

<table>
<thead>
<tr>
<th>Algorithm: Dimension Detecting</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A Category node n</td>
</tr>
<tr>
<td>CategoryClassification database CC</td>
</tr>
<tr>
<td><strong>Output:</strong> Boolean value Non-Crossed or Crossed</td>
</tr>
</tbody>
</table>

1. IF parent-cid of n is cid of Top category THEN
2. set dimension of n to cid to keep first level nodes on the top of each their dimension
3. ELSE
4. get the dimensions list d-list of n’s parents from CC
5. WHILE n’s dimension is not in the d-list
6. return Crossed
7. END WHILE
8. END IF
9. return Non-Crossed

Figure 5.13: Algorithm of the Dimension Detecting Operation

**Directed Cycle Detecting**  A directed cycle is a cycle with all of its edges oriented in the same direction. In our hierarchical category structure as represented by the directed graph, each node represents a unique category, and each directed edge between two nodes represents a unique category classification relationship. Therefore, a directed cycle in the category structure means that a category can be classified by its children, its offspring, or even by itself. This makes the possible classification infinite. As a result, our hierarchical categorization structure can be considered to be a DAG (Directed Acyclic Graph), which is a directed graph without directed cycles.

To give an example, as shown in Figure 5.14, we intend to classify category 2 to category 9 with the relationship K, i.e., to add a directed edge in our category directed graph from 9 to 2 with edge name K. If we do so, a directed cycle appears: 2 to 5 to 9 to 2. As a result, the edge K cannot be added to the graph, i.e., the category 2 cannot be classified to the category 9.
As shown in Figure 5.15, we check whether any directed cycles exist in any dimension to guarantee the correctness of the structure. To determine if there is a cycle between two category nodes \( m \) and \( n \), we conduct a breadth first recursion in the steps listed in Figure 5.15:

### 5.5.3 Path Computing

This operation is used to compute the paths related to a category. Its purpose is to supply users with detailed path information for a specified category in the structure. We conduct two different directions of path computing: top-down and bottom-up path computing. For certain activities such as browsing in the category structure, top-down path computing is the most efficient for assisting the user in getting the path information of the current browsing category. On the other hand, for selecting a category as a searched result, bottom-up path computing is the best choice for locating the selected category and getting the related path information.
Algorithm: Directed Cycle Detecting

Input: A Category Node m
      A Category Node n
      CategoryClassification database CC

Output: Boolean value Non-Cycled or Cycled

1  IF m = n THEN
2      return Cycled
3  END IF
4  Initialized an empty queue Q for category nodes
5  IF dimension of m ≠ dimension of n THEN
6      return Non-Cycled
7  ELSE
8      enqueue m into Q
9      WHILE Q is not empty
10     dequeue Q to HEAD
11     FOR each Parent of all the ancestor nodes of HEAD from CC
12     IF Parent = n THEN
13        return Cycled
14     ELSE
15        enqueue Parent into Q
16     END IF
17     END FOR
18     END WHILE
19  return Non-Cycled

Figure 5.15: Algorithm of the Directed Cycle Detecting Operation
**Top-Down Path Computing**  To display the path information of a category node, we render the category structure to a tree component of JSF. However, computing every possible full path of the start category node in a tree is unnecessary and inefficient. Instead of doing that, we give an optimized solution. While browsing in the structure (expanding the tree nodes downwards), we only compute the possible categories in the next level and load them into the tree. At the same time, we record the chosen nodes into the current path. A new node that is not a child category of any node in the current path is recorded as the last element in the list; otherwise, we create a new path from the top to its parent category, and put the new node following it.

**Algorithm: Top-Down Path Computing**

<table>
<thead>
<tr>
<th>Input:</th>
<th>A Category Node n</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CategoryClassification database CC</td>
</tr>
</tbody>
</table>

**Output: The Current Path which is a list of categories in sequence: path-list**

A list of child categories of n: next-list

1. IF path-list is empty THEN
2. add n into the path-list
3. ELSE
4. IF n is not in the path-list THEN
5. IF a element m in the path-list is related to n THEN
6. copy the elements from the first element in the path-list to m to the end of path-list
7. add n to the end of path-list
8. ELSE
9. add n in the path-list
10. END IF
11. END IF
12. END IF
13. get the child categories of n from CC and put into next-list

Figure 5.16: Algorithm of the Top-down Path Computing Operation

Based on this algorithm, shown in Figure 5.16, we receive optimized path information, which can record paths start from multiple categories in the same or different dimensions.

To give an example, as shown in Figure 5.17, we intend to record the path information starting at the top category 0. There are three child categories related to the category 0: category 1, category 2, and category 3. The users can choose the next node desired. In this

75
Figure 5.17: Example: Top-Down Path Computing
case, we assume that category 1 is chosen. This means that the edge a is saved in the path and, for now, the path is “/top/a”. As soon as category 1 is chosen, its child categories, category 4, category 5, and category 6, are found. If category 5 is taken, its edge e is added to the path, and its child categories, category 9 and category 10, are displayed to the user. The path now becomes “/top/a/e”. For instance, the user may jump back to select category 6 instead of choosing the next category from the children of category 5, and we know that category 6 is the child category of category 1, which has already been in the path. As a result, the directed edge f from category 1 to category 6 is added to the end of the path. The path becomes “/top/a/e/f”. The user is even able to switch to other dimensions. For instance, the next selected node can be category 3, and edge c will be added to the path following the “/top” because category 3 is the child of category 0. Therefore, we get the path “/top/a/e/f/top/c”. For the user, it is always possible to navigate the path on his or her own initiative.

**Bottom-Up Path Computing** While selecting a category in the expanded tree component, the path information can help users locate the current position in the whole structure. To find the path from a selected category node n upwards to the top category, we conduct a depth first search as shown in Figure 5.18.

```
Algorithm: Bottom-Up Path Computing

Input: A Category node n
CategoryClassification database CC
Output: The Shortest Path which is a list of categories in sequence:path-list
       The Distance between n to the top category:dis
1 set n as the End of path p and add to path-list
2 WHILE p is not the top category
3 get the parent categories of p from CC
4 get the highest level parent category p and add to path-list
5 END WHILE
1 set dis as the number of elements in path-list
```

Figure 5.18: Algorithm of the Bottom-up Path Computing Operation

77
### 5.5.4 Relationship Creating

To classify a category $n$ to another category $m$, there are several steps to create a category relationship between $m$ and $n$.

1. Check if it is possible to create the relationship: dimension detecting and directed cycle detecting;

2. Add the category classification relationship to database CategoryClassification; and
3. Update all of the offspring’s level of n if necessary.

5.5.5 Relationship Removing

To remove a relationship between category m and its child n, there are the following steps:

1. Remove the category classification relationship from database CategoryClassification;
2. Remove the category n if m is the only parent of n;
3. Remove the offspring and the relationships of n if they are only related to n; and
4. Update all of the offspring’s level of n if necessary.

5.5.6 Category Creating

This is a simple operation comprising two steps. The first step is to establish a category entity, which contains the general information of the category, and add the corresponding record to the table Category. The next step is to establish the category classification relationship and add it to the table CategoryClassification.

5.5.7 Category Deleting

To delete a category, the first step is to delete the category record from the table Category, then remove all related classifications from the table CategoryClassification. The next step is to find all of its offspring that are only related to this category and remove them from the table Category as well. The third step is to remove all related, expired relationships from the table CategoryClassification. Finally, the last step is to update the general information of the offspring of this category.
Algorithm: Relationship Removing

Input: A Category node m
       A Category node n
       Category database C
       CategoryClassification database CC

1. remove category classification relationship from m to n in CC
2. IF m is the only parent of n
3.   remove category n in C
4.   remove all relationships from n in CC
5.   remove all categories in C which is no relationship to any other category in CC
6. ELSE
7.   get the level difference from m to n:d
8.   IF d = 1 THEN
9.     update the level of n to the longest distance to the top in C
10.    Function updateLevels(n, C)
11.     FOR each child of n
12.        get the level of child: l
13.        IF l = (the level of n + 1) THEN
14.           update l to the longest distance to the top in C
15.           call Function updateLevels(child, C)
16.     END IF
17.    END FOR
18. END Function
19. END IF
20. END IF

Figure 5.20: Algorithm of the Relationship Removing Operation
5.6 Email Managing Operations

Besides the email client-side operations and category managing operations described above, we also supply more operations to manage emails. The basic operations are fundamental to all other operations. In our system, there are two basic operations: Email Classifying and Email Un-classifying.

5.6.1 Email Classifying

When classifying an email to the category structure, we need to pick up the target categories, and then the corresponding records are generated in the table EmailClassification. Unlike classifying categories, email can be classified across the different dimensions. However, there are restrictions for email classification:

- Emails cannot be classified to categories that are related and in the same path. For example, for category m and its offspring o, emails can be only classified to one of them.

- Emails are classified to the default dimensions and categories automatically by parsing their headers while loading them from email servers. For instance, emails are classified to a certain subcategory of the Status category, because every email has a status field in its header structure, such as SEEN, RECENT or FLAGGED. Therefore, emails have been classified whenever they are loaded and stored in our system. However, the user can also classify these emails further or re-classify them any time while using the system.

- There are some categories to which emails cannot be classified at the same time, such as SEEN and RECENT in the Status category, because, in our system, a RECENT email is an email that has never been read.
5.6.2 Email Declassifying

To declassify an email, we simply delete the corresponding records in the table EmailClassification. However, an email has at least one email classification relationship in the Time and Status dimensions, unless the email has been deleted.

5.7 Dimensional Category-based Search Operations

Searching is an important operation corresponding to category operations to generate the refined results for the user. Unlike a normal keyword search, in our system, we wrap the search result into corresponding categories, which are further classified within individual dimensions. Each dimension can be a subset of the rooted directed acyclic graph according to the multiple path information of the categories. Users can switch between dimensions and pick up the narrowed category structure to attain the desired targets.

**Algorithm: Dimensional Category-based Search**

| **Input:** keywords of the target |
| **Output:** a Result set: R |
| 1 Keyword search and get a result set of emails E |
| 2 Initialize a dimension d |
| 3 Initialize a category list C |
| 4 FOR every email e in E |
| 5 get e’s related category list Ce |
| 6 WHILE any c in Ce is in C |
| 7 wrap e into R |
| 8 END WHILE |
| 9 END FOR |

Figure 5.21: Algorithm of the Dimensional Category-based Search

5.8 Other Operations

Other operations are also implemented, such as import, recommended classifying, and blocking list.
**Import**  This operation is used to import emails from local drives. The user can save his or her emails from other email-clients like MS Outlook, Pine, and so on into their local directory structure. In our system, these emails can be imported along with their directory structure. The emails will be stored in the database Email and the directory structure will be converted to our category structure.

**Recommended Classifying**  This operation is used to help classify emails while loading them into our system. The user can define a keyword for any category stored in the table Category. While reading an email, we parse the content of the email and compare it to every keyword saved in the table Category, then display a recommended category list to the user to help the user manage his or her emails efficiently.

**Blocking List**  This operation is used to manage the subcategories of the default category People. Since we parse the people related fields from the email headers while loading email into our system, we create a list of email addresses that is related to people who are not welcome. Emails from those people are classified to the default category Trash directly.
Chapter 6

Contributions and Future Work

6.1 Contributions

EmailCat is designed to be an extended dimensional category-based email management system from Yifeng Gao's work. In order to organize and search information from emails in a more flexible and efficient way, we describe a category structure, called the multiple dimensional hierarchical category structure, which allows categories and emails to be multiply categorized within dimensions. Based on this structure, EmailCat overcomes some of the drawbacks of existing email management systems and provides a more flexible method for users to manage their emails. Below we summarize our contributions:

- We studied the existing solutions related to email management systems, such as desktop-based email clients and web-based email clients. We compared them for both their advantages and shortages in order to improve our research. We also studied file management, such as hierarchical file system, category-based system, and faceted classification applications. We analyzed them briefly to design our system in a better way.

- We analyzed Yifeng Gao's work, Web Category summarized the drawbacks of the category structure of Web Category. We extended the category structure of Web Category and gave our extended category structure.
• We explained our structure definitions, Entity Relationship Modeling, and our design work.

• We researched the system architectures of existing email systems and designed the system architecture of the EmailCat. Based on our extended category structure, we implemented the EmailCat operations with different algorithms.

These are the corresponding advantages of our work:

• It applies a database to manage categories and emails and takes advantage of the nice indexing and searching features of databases over conventional none-database applications;

• Is platform-independent;

• More natural and flexible to classify emails;

• More efficient and accurate to use a dimensional category-based search for information;

• Has a user-friendly interface; and

• Is easy to extend to a server-side web application based on our system architecture.

6.2 Future Work

Since EmailCat has some limitations, there is still much research and implementation work that needs to be done. We have a lot to accomplish in order to complete our system. Some of the future work is mentioned below:

EmailCat can build connections to mail servers via the most common protocols. However, there are more protocols that need to be covered.
Also, composing email in EmailCat has all basic functions to compose and send email. However, more features can be added in this aspect. For example, a flexible email sender field which allows the user to give any email address even it is different from the user's email accounts.

In addition, read receipt is a useful feature. It requests a replied email from the recipient to tell the user that the email has been read by the recipient. This feature can be added.

Furthermore, Internationalization is an important issue for email application, since email is an easy and popular way to communicate for people in different regions. More works need to be done on this aspect.

Besides the managing of categories and emails, more studies can be done to provide more features for the EmailCat system by taking advantage of database technology.

- It is possible to implement a ranking feature which is widely used by the recommendation systems. This feature can help the user easily located their concerned emails. For example, top ten emails of the week.

- A thread view is able to be generated in EmailCat. This feature gives all related emails to the user. It will assist the user find the related information from one email to another.
Bibliography


