

5-2012

# Organizational Search in Email Systems

Sruthi Bhushan Pitla

Western Kentucky University, [sruthibhushan.pitla698@topper.wku.edu](mailto:sruthibhushan.pitla698@topper.wku.edu)

Follow this and additional works at: <http://digitalcommons.wku.edu/theses>



Part of the [Databases and Information Systems Commons](#)

---

## Recommended Citation

Pitla, Sruthi Bhushan, "Organizational Search in Email Systems" (2012). *Masters Theses & Specialist Projects*. Paper 1161.  
<http://digitalcommons.wku.edu/theses/1161>

This Thesis is brought to you for free and open access by TopSCHOLAR®. It has been accepted for inclusion in Masters Theses & Specialist Projects by an authorized administrator of TopSCHOLAR®. For more information, please contact [topscholar@wku.edu](mailto:topscholar@wku.edu).



ORGANIZATIONAL SEARCH IN EMAIL SYSTEMS

A Thesis  
Presented to  
The Faculty of the Department of Mathematics and Computer Science  
Western Kentucky University  
Bowling Green, Kentucky

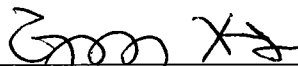
In Partial Fulfillment  
Of the Requirements for the Degree  
Master of Science

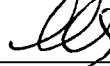
By  
Sruthi Bhushan Pitla

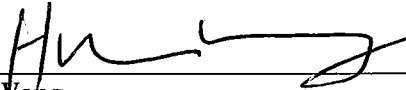
May 2012

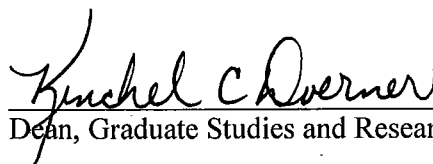
ORGANIZATIONAL SEARCH IN EMAIL SYSTEMS

Date Recommended 5/2/2012

  
Dr. Guangming Xing, Director of Thesis

  
Dr. Qi Li

  
Dr. Huanjing Wang

 21-may-2012  
Dean, Graduate Studies and Research      Date

## ACKNOWLEDGMENTS

It was a great pleasure working under my graduate advisor, Dr. Guangming Xing, who provided me with everything I need to succeed. His inspiration and guidance at each and every step made this Master of Science degree so rewarding and satisfactory. He always encouraged my work in every possible way and also gave me the freedom to express and implement my ideas without any restrictions. I feel very fortunate and proud to have been his student and really think the experience which I gained working under him is invaluable. I would like to whole heartedly thank Dr. Xing for the immense trust and patience he has over me. He constantly supported and directed me in each and every step. Without him this thesis would not have been so successful.

I would like to thank Dr. Qi Li and Dr. Huanjing Wang for their valuable time and suggestions that helped me improve this Thesis. I would like to thank all my friends at Western Kentucky University, my friends in India and my family for the never-ending support. I would like to specially thank my father Mr. Srihari Pitla and my mother Mrs. Prabhavathi Pitla for their everlasting love and encouragement for me to succeed.

## TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION .....	1
1.1 Background .....	1
1.2 Approach to the problem .....	2
1.3 Some real world scenarios .....	3
1.4 Organizational search.....	4
1.5 Mozilla Thunderbird framework.....	6
1.5.1 Extension / add-on development on Thunderbird framework .....	7
1.6 Thesis structure .....	9
CHAPTER 2: ARCHITECTURE OF THE SYSTEM .....	10
2.1 Modules in the TESO architecture.....	11
CHAPTER 3: QUERY PROCESSING MECHANISM.....	17
3.1 Different approaches used for the Query processing .....	17
3.2 Using the native file in extensible Markup Language (XML).....	24
3.3 Twig pattern matching .....	34
CHAPTER 4: IMPLEMENTATION DETAILS .....	37
4.1 Add-on for Mozilla Thunderbird .....	37
4.2 User Interface:.....	37
CHAPTER 5: CONCLUSIONS .....	49

BIBLIOGRAPHY ..... 50

## LIST OF FIGURES

Figure 1: Processing of Organizational information	6
Figure 2: Thunderbird Add-on Extension Framework	9
Figure 3: Architecture of TESO System	11
Figure 4 : RDBMS to Native XML file transformation	13
Figure 5: Integration of different tables with emails	21
Figure 6: Data represented as XML tree structure	27
Figure 7: XML Twig Query processing mechanism	28
Figure 8 : Structure of data in XML format	31
Figure 9 : DTD for XML data in the document	32
Figure 10 : XML tree structure with appropriate labeling of nodes	33
Figure 11 : Installation of add-on from a fil	37
Figure 12 : The developed add-on is shown after it is installed	38
Figure 13 : The user input is given which initiates query processing	39
Figure 14 : The relevant emails are retrieved and displayed after query processing	40
Figure 15 : Indexing mechanism in gloda database	42
Figure 16: Number of retrieved email results in case of normal and organizational searches	46
Figure 17: Representation of expected count of irrelevant email results in normal search	47
Figure 18: Representation of expected count of irrelevant email results in organizational search	47



## LIST OF TABLES

Table 1: Student table .....	22
Table 2: Section table.....	22
Table 3: Registration table .....	23
Table 4: Registration_section table.....	23
Table 5: Section_students table .....	23
Table 6: Expected results of normal text search .....	45
Table 7: Expected results of organizational search.....	45
Table 8: Expected performance results.....	48

## ORGANIZATIONAL SEARCH IN EMAIL SYSTEMS

Sruthi Bhushan Pitla

May 2012

52 Pages

Directed by: Guangming Xing, Qi Li, and Huanjing Wang

Department of Mathematics and Computer Science

Western Kentucky University

The storage space for emails has been increasing at a rapid pace day by day. Email systems still serve as very important data repositories for many users to store different kinds of information which they use in their daily activities. Due to the rapidly increasing volume of email data, there is a need to maintain the data in a most efficient way. It is also very important to provide intuitive and flexible search utilities to provide better access to the information in the email repositories, especially in an enterprise or organizational setting. In order to implement the functionality, we are presenting a tool name TESO. TESO is a tool for email searching using organizational information. This tool is designed to improve the relevancy of the email search by integrating the data from email servers and organizational information from directory services and other resources. We implement this functionality as an add-on for the Mozilla Thunderbird framework, which is an open source email client system developed by the Mozilla Foundation. The results are evaluated using the SQLite and the XML data. This work will serve as a handy tool in the area of existing information integration and keyword search on relational databases techniques and also helps in efficient access of XML information.

## CHAPTER 1: INTRODUCTION

### 1.1 Background

WWW (World Wide Web) has become an integral part of every person's life in performing their day-to-day activities, and many of us spend a significant amount of time browsing different data and on social networking applications (Facebook, Google Plus, Twitter, etc ). The users these days have plenty of options to communicate with each other. Even though there are many ways of communication, the email still serves as the most fundamental means for many of us to communicate [4]. We have known that social networking applications definitely serve as a handy medium to communicate, but in order to register or access the social networking service, we have to provide an email ID. This process of registering to the social networks in turn leads to the generation of more emails that make organizing the huge volume of emails a more complicated and a rather challenging task. The storage media have become dramatically affordable over the past years, which help in populating large scale archives of emails in the email systems. The large scale storage of emails demands a very good maintenance and efficient monitoring mechanism. Thus, it is very important that efficient data integration mechanism and search tools are available to handle such a large volume of email collection. Many web-based email services and email clients possess full text search support and many companies offer desktop applications that can support indexing, searching the file systems, emails and the browser caches and there are also many research prototypes [11,9,1,2] which perform the search operation. All these mechanisms have followed the

same genre as the traditional information retrieval schemes and they do not possess the capability of discovering and representing the entities and the relations among them. They lack the capability of acquiring the relationships among the different entities and the relationships from diverse and heterogeneous data sources. The work presented in this thesis is focused primarily on how to deal with the email data in the enterprise email systems or in the organizational setting and also on how to retrieve the related organizational information which is readily available as per the search criterion.

## **1.2 Approach to the problem**

As clearly stated in [1], the important goal in performing the information retrieval is to perform the integrated search of the user's personal information, coupled with the organizational and the web information from diverse heterogeneous sources. This kind of functionality especially plays a pivotal role in the enterprise setting. Although many of the social networking applications provide the users with the privilege to register to the circles or the different groups which serve as an abstract mechanism, organizing the emails based on the similar method has not yet been studied. In the case of email mechanism, it can be said that it has been successful in partially accomplishing the method. Email lists can be created to make it easy for a user to reach a predefined group of receivers rather than sending an email to each specific person which helps in reduction of hectic work and saves a lot of time. This mechanism is adopted in many enterprise systems to send emails to a predefined group of users very easily. However the problem is that this convenience only resides on the sender's side. This issue will be a particular focus in this research.

### **1.3 Some real world scenarios**

Scenario 1: Mailing list of all graduate students in a department (consider the Computer Science Department at Western Kentucky University)

At WKU we maintain emails of all the Computer Science graduate students in a separate mailing list `cs.grads@lists.wku.edu`. When there is a need to reach all the Computer Science graduate students, this mailing list will be used to send out an email. Thus this mailing list serves as a very handy thing and it also serves as an active discussion forum where a student does have a privilege to take a lead in the discussion. This allows very easy retrieval of the emails related to this well-defined group by searching the mailing list name over the “to” field in the email client. The vice versa may not work the same way, i.e.; when a student sends a well-related email to a specific individual recipient, finding such an email by the recipient will not be that easy if the recipient does not know anything about that specific email. This could be considered as a concern when it is from the recipient’s perspective.

Scenario 2: Consider a search operation performed by a faculty member:

Search information on a new course proposal “CS570 Security in Computing” at WKU

At most of the universities before setting up a course they have to go through a thorough review process and in this process it has to go through a review by several committees at different levels within the university. The members involved in the course proposal can be from different departments or disciplines. The Email communications from members in different committees often indicates different aspects for the course proposal according to their diversified perspectives. The following could be the different

possibilities:

1. Members from the same department are generally familiar with the course curriculum and the course complexity, so they are more likely talking about the content of the course.
2. Members from the other academic departments may talk about the interdisciplinary features of the course and the kind of impacts the new course can cause on their program.
3. The members of the library are most likely concerned about the required course material that is necessary in order to support curriculum of the course.

After the process of course approval and the course has been offered, the emails from different users in the organization are stored in different categories and so the emails from the students will also be in a totally different category. So when a keyword “CS570” is given as a search query it can result in emails from different categories based on the way the organizational information is setup. The organizational information is always not the same and it can evolve over time as there may be changes in the registration, student enrollment and also can be due to the other organizational issues.

#### **1.4 Organizational search**

Organizational search is a mechanism where the search operation is performed on the email system using information in the email system as well as the organizational information from the other heterogeneous data sources. Organizational information explains how organizations use information found within its environment to interpret and on how to adjust to the changes in the organizational setting. The organizational

information can be present in many forms depending upon the following factors:

1. Organizational functionality
2. Type of organization
3. Scalability of organization

#### **1.4.1 Types of organizational information**

In general, the organizational information can be broadly classified into the following two categories. The processing mechanism is illustrated in Figure 1.

1. Organizational information can be stored in different directory servers and other structured data sources. The structured data sources are the data repositories which store information in relational form or in the form of tables.

Examples:

- Active Directory
  - Student enrollment records in Banner systems at many universities.
  - Black board
2. Organizational information stored in unstructured data. The unstructured data sources can be the data which is stored in the xml format and it can also be the data stored as a plain text format.

The information can be in the form of

- signature text in emails
- Organizational information in word documents in shared directory.

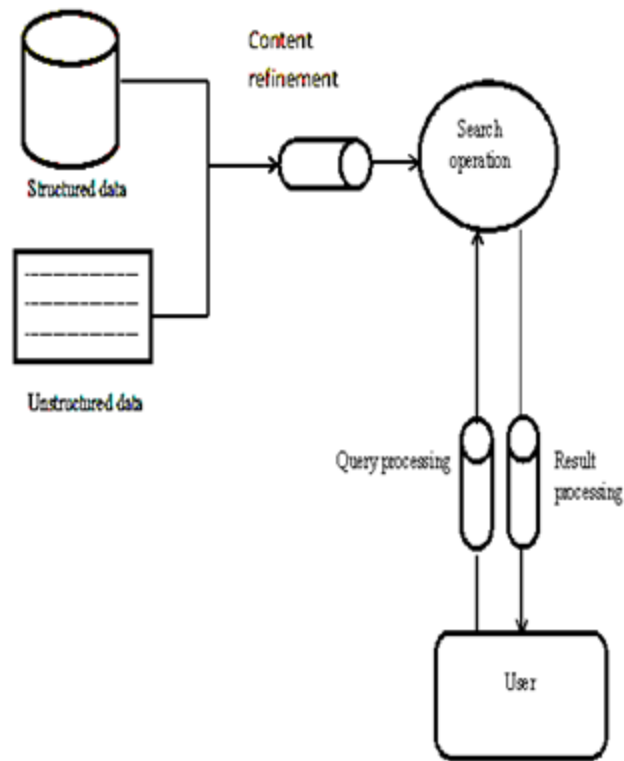


Figure 1: Processing of Organizational information

### 1.5 Mozilla Thunderbird framework

Thunderbird is an open source email client developed by Mozilla Foundation. It is a free open source program which is supervised and maintained by the Mozilla project. It can be compared to several other email clients available in the market such as Outlook Express, Eudora and Netscape communicator. It can run on a variety of operating system platforms like Windows 95, 98, Me, 2000, XP, Linux, Mac OS X, OS/2, Solaris.

Thunderbird can manage emails and information from a variety of sources.

It has features [22] like

- Multiple email and newsgroup accounts



- POP and IMAP protocols
- HTML mail formatting
- Import and export email accounts and messages
- Spell check as you type
- Deleting and detaching file attachments
- Advanced message filtering
- Folder retention rules
- Customizable
- Email stored as plain text

Features not included are as follows:

- Calendar, notes, and task management
- Support for HTML email (e.g. Hotmail)

It allows any organization to build their add-on which best fits their needs. We are focusing on the full text search, message filtering and add-on development features of the Thunderbird to accomplish the desired functionality.

### **1.5.1 Extension / add-on development on Thunderbird framework**

Extensions are the programs that help in addition of new features to the existing Thunderbird framework through the installation of XPIInstall modules which are known as “XPI” or zippy installation which is similar to the zip archive which we use to compressing/zipping the files for our daily use. They add new functionality [16] to Mozilla applications such as Firefox, Sea Monkey and Thunderbird. They can add

anything from a toolbar button to a completely new feature to get the desired functionality. They allow the application to be customized to fit the personal/organizational needs of each user if they need additional features. This customization feature serves as a very handy option in the implementation of our desired needs to get the desired functionality. Once the add-on is ready to use it can be updated in the future based on the contemporary version of the framework, this functionality is provided by the Mozilla Foundation.

### **Extension framework**

Extensions are packaged and distributed in ZIP files, with the XPI file extension.

The Hierarchy showing the contents in the typical XPI file is shown below.

Sample\_extension.xpi // This is equal to the name of the working folder,

Sample\_extension/

/install.rdf //It stores the general information about the extension;

/chrome.manifest // Registers the content in the files with the chrome engine;

/chrome/content // The contents of the extension such as the XUL and the JS files are included in this section;

/chrome/icons/default/ // Contains the default icons of the extension;

/chrome/locale/ // Contains the information about the localization.

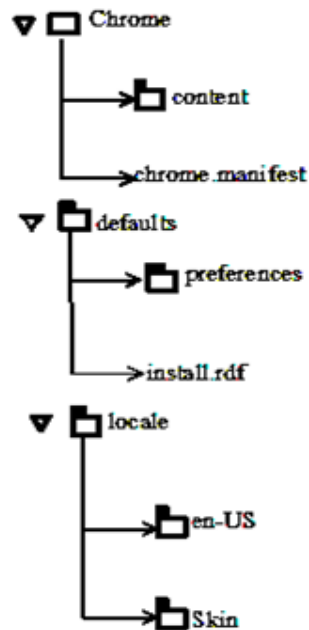


Figure 2: Thunderbird Add-on Extension Framework

## 1.6 Thesis structure

The remainder of the thesis is organized as follows. Chapter 2 presents the architecture of the proposed system. Chapter 3 describes the query processing mechanism. It reviews the recent developments in the area of the text search over relational data, which is the theoretical foundation of the work presented in the thesis. Implementation and basic experiments on the effectiveness of the system are discussed in the Chapter 4 and conclusions are given in Chapter 5.

## **CHAPTER 2: ARCHITECTURE OF THE SYSTEM**

TESO, a tool for email searching using organizational information is designed to improve the relevancy of the email searching mechanism by integrating the data from the email servers and the organizational information from directory services and other heterogeneous sources. The data processed in the organizational system consists of personal email data in the email system or the data which is stored in the local database and the organizational information which is obtained from diversified data sources. Typical architecture of the system in the case where a university is considered as an organization can be illustrated as shown in the Figure 3

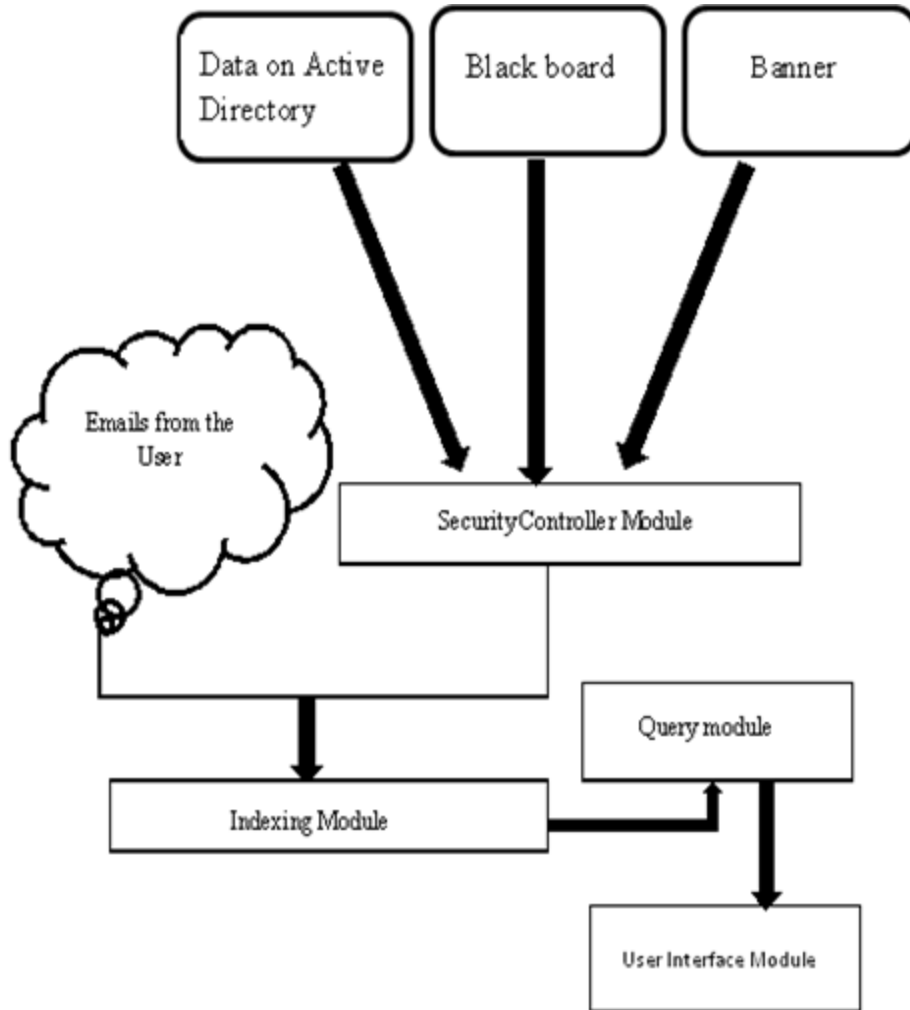


Figure 3: Architecture of TESO System

### Modules in the TESO architecture

The TESO architecture consists of five modules

#### 1. Data storage module:

This module deals with the aspect of how all forms of data (structured or unstructured) or the information in the organization is properly organized in the enterprise setting. In our case we need to consider user emails which are stored in the local archive/local database and organizational data stored on various diversified heterogeneous servers. For each of

the organizational data storage, a data adapter is needed to pull the data from a variety of data repositories into the email search utility to perform the desired functionality.

The data storage schemes which are supported are

- XML native format.[23]
- SQLite databases.[19]

**XML native format:**

The XML native format does not have the functionality of the databases and they do not really store the information as contrary to the databases. The native format can only be defined as a logical model or the logical architecture for an XML document. The documents are retrieved according to the way the logical model is designed.

The model includes the following segments

- Elements
- Attributes
- PCDATA
- Document order

The native format contains an XML document as its fundamental unit of the logical data storage which is just as the row in a table in the relational database is a fundamental unit in the relational database. The Native XML databases can be relational, object-oriented database structures and it can also be in the form of indexed files or the compressed files.

Most of the XML databases support many querying methods to perform the querying operation. But XPath is considered as the most commonly used mechanism to perform

the querying operation on the documents or collection of documents. XPath is used to filter and identify the nodes that match the criterion to get the most specific or the intended result. XSLT is another mechanism where it defines a set of XPath filters which can transform diversified documents using the XML grammar syntax and semantics. XQuery in association with XPath can also be used to filter the elements and the nodes. Figure 4 illustrates the RDBMS to Native XML file transformation mechanism.

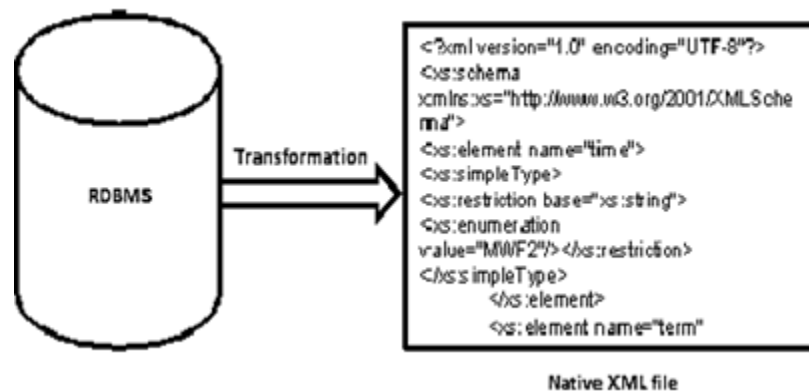


Figure 4 : RDBMS to Native XML file transformation

SQLite Data storage:

SQLite is a widely used storage mechanism for local/client storage on the web browsers [19] and it is considered most widely as a deployed database engine. It is used these days in browser application developments, in the embedded systems support and it has become an integral part in building many applications. Examples of the companies which use SQLite support are Mozilla, Apple, Adobe, Google and also in mobile application development. SQLite is compatible with the relational database management system and its main advantage is it uses small size for data storage. It is compatible with

most of the SQL querying syntaxes. It is embedded in the client application, whereas the other database management systems need a separate storage for the database.

### **Reasons why the SQLite is considered to be handy in the application development**

A complete database is stored in a single-cross-platform disk file [19] which is considered to be efficient as it does not require a separate back up mechanism to store the data.

- It contains a small code base which ranges in between 200KB to 350KB
- It has a very simple and easy to use Application Programming Interface (API)
- It provides the feature of cross platform support [19] which helps in porting other operating systems very easily.
- The SQLite database can be monitored and maintained using a Command Line Interface (CLI) and there are also GUI's available which can be used to monitor SQLite databases.
- It is faster when compared client/server database paradigms.

The SQLite Manager add-on which is an add-on that can be installed on the Mozilla Thunderbird framework is used in building the database structure for this implementation and it can be used to populate different tables in the database.

#### **2. Security controller module:**

The security controller module deals with the aspects of providing the overall security to the enterprise setting. The organizational information is comprised of different forms of data. The different forms of data are accessed by different users at different levels of



the organization. Thus data abstraction mechanism must be enforced among the different group of users i.e.; only the authorized users should be able to view the content related to them which inevitably enforces a good security mechanism. The organizational data storage may contain very sensitive information that should not be visible to a particular set of users; the data pulled from the storage server must perform the process of filtering before displaying the results to the end user. An ACL (access control list) which stores the information about the access permissions for different sets of users is used in the implementation to perform the process of filtering the data from the storage server before rendering the relevant results. Another very good thing about this implementation is that it also retrieves the results abiding to the security policy at the organization. This module has to be implemented by the hosting organization.

### 3. Indexing module:

The search indexing operation performs the operations of collecting, storing and parsing the data. This helps in faster and very efficient information retrieval as the frequently used data is indexed and it just needs to be properly parsed in order to get the most relevant search results. The organizational information pulled from the data storage will be stored in a local relational SQLite database or XML file that can be accessed in Mozilla. The emails and the organizational data are indexed. Email indexes in Thunderbird framework will be augmented with indexes for organizational data. The augmented indexes in the Thunderbird are stored in the Gloda (Global database) in the default profile folder in the specific SQLite database called gloda-messages-db.SQLite. The indexes in the Thunderbird framework are automatically generated which facilitates

an efficient and faster retrieval of the information from the email repository.

#### 4. Query module:

The query module deals with the aspect of performing the operation of information retrieval by accepting the search query from the user to satisfy his or her information needs. The querying mechanism considers the user issued keyword as a search query, interprets it and then performs the query processing to retrieve the most relevant emails based on the input. In our case we use the existing email search utility and search over the organizational data to carry out the query. The query would be considered as a refined integrated query to perform very efficient and relevant information retrieval.

#### 5. Clustering module:

Clustering module deals with the aspect of grouping the similar information into well-formed groups/clusters. The search results are analyzed and clustered based on the organizational information in the email i.e.; the emails from the different set of users can be stored separately in the form of different clusters. We can consider an example where if a search query is issued the results generated will be the aggregation of emails from different set of users here. There is a scope to enforce clustering mechanism which does the functionality of placing different set of users into different category. Thus combined with existing email classification/clustering techniques, this definitely offers many possibilities that can change how emails are organized in a more efficient manner in an organizational setting.

## CHAPTER 3: QUERY PROCESSING MECHANISM

### 3.1 Different approaches used for the Query processing

The integration of databases and the Information Retrieval schemes provide different ways for the users to query the information to get the desired results. The relational databases provide users with the privilege to query the well-structured data which are stored in the form of tables and the unstructured data does not require users to understand the information and it does not require users to understand or have a clear idea of the database schemas.

#### **Integration of data in different tables using the candidate network generation scheme**

**Candidate networks:** A candidate network [21] is the process of generation of joining expressions to get the joining network of tuples.

**Minimal joining networks:** A minimal joining network is the joining network of tuples [20] that satisfy the following conditions

- Total
- Minimal

**Total:** This means that the each keyword which is typed in as search query must be present in at least one of the tuple/record in the joining network of tuples.

**Minimal:** This means that it is not total it also inherits some properties. If any tuple/record is removed from the joining network of tuples then it is said to have only the minimal functionality.

The candidate networks can be of two types

- Complete
- Non redundant or no duplication

**Complete:** This means the collection of the candidate networks produces all the minimal joining network of tuples which are possible during the process of integration of data in different tables

**Non redundant or no duplication:** If any of the candidate networks are not considered there may be data which may well be contained in the minimal joining network [21] which was not discovered or which was not taken into consideration.

Different candidate network generation algorithms can be used to generate different efficient candidate networks and to evaluate it in a most efficient way.

### **Addressing the main problem**

The most important problem to address in the organizational system is to perform the integrated search operation on different relational tables in the database with the emails of different users to get the relevant results. A similar problem has been studied in the database communities by many researches in the area of keyword search on structured and semi-structured data [2], [3] where in the integration of data from different tables or relations is performed based on the user issued keyword. In the traditional Web searching, each page is viewed as an entity in the searching and ranking algorithms. However the structured data must be properly integrated as they are fragmented in the process of the normalization. The structured data must also be properly integrated keeping in view of all the primary and foreign key relationships among the different tables in order to retrieve the relevant entities of interest. The data used in this project can

be viewed as a relational database which is a SQLite database, where emails are stored in one relation. The “rows” in the email “table” are connected with data in other tables holding organizational information. When the search query is issued by the user the data in different tables is first integrated according to the search criterion and then the rows in the email table are referred and finally all the relevant emails are properly retrieved.

As shown in the Figure 5, let’s consider a case where a faculty table has to interact with a student table. This operation can be done in two different ways or by following two distinct paths as specified below

- A faculty can be related with a student through advising (Faculty↔Student)
- Through course enrollment (Faculty ↔ Section ↔ Enrollment ↔ Student)

Thus this forms a network with two distinct relations. In a relational database, this can be built using foreign-key relationships where in the foreign keys in different tables refer to the primary keys in related tables in order to establish the different relationships .

With the different indexing techniques for text data well developed, identifying the aspect of how the information in different chunks to get to relate to each other is the most important thing to consider. Let us consider the same example where the faculty table tries to relate to the student table. In this particular example we can clearly see that two networks can be generated out of all the possible networks to search for specific information. In a typical relational database, the number of tables and foreign-key relations are far more complicated than the example presented above. The process of exploring all the foreign key relationships in order to build different paths or networks to perform the search operation is far more expensive and it is considered to be cumbersome

in the practical sense. Thus in order to mitigate this complex possibility of routing the tables various techniques have been proposed lately. The candidate network based solution is considered to be the best possible approach in this kind of scenarios and they are well received and evaluated in these kinds of cases. Here in our implementation a candidate network based system provides an ideal dataset to perform and evaluate different experiments of several candidate network based algorithms. In order to improve the efficiency of the search utility the schema of the organizational information is also used to control the generation of different candidate networks in different possible ways.

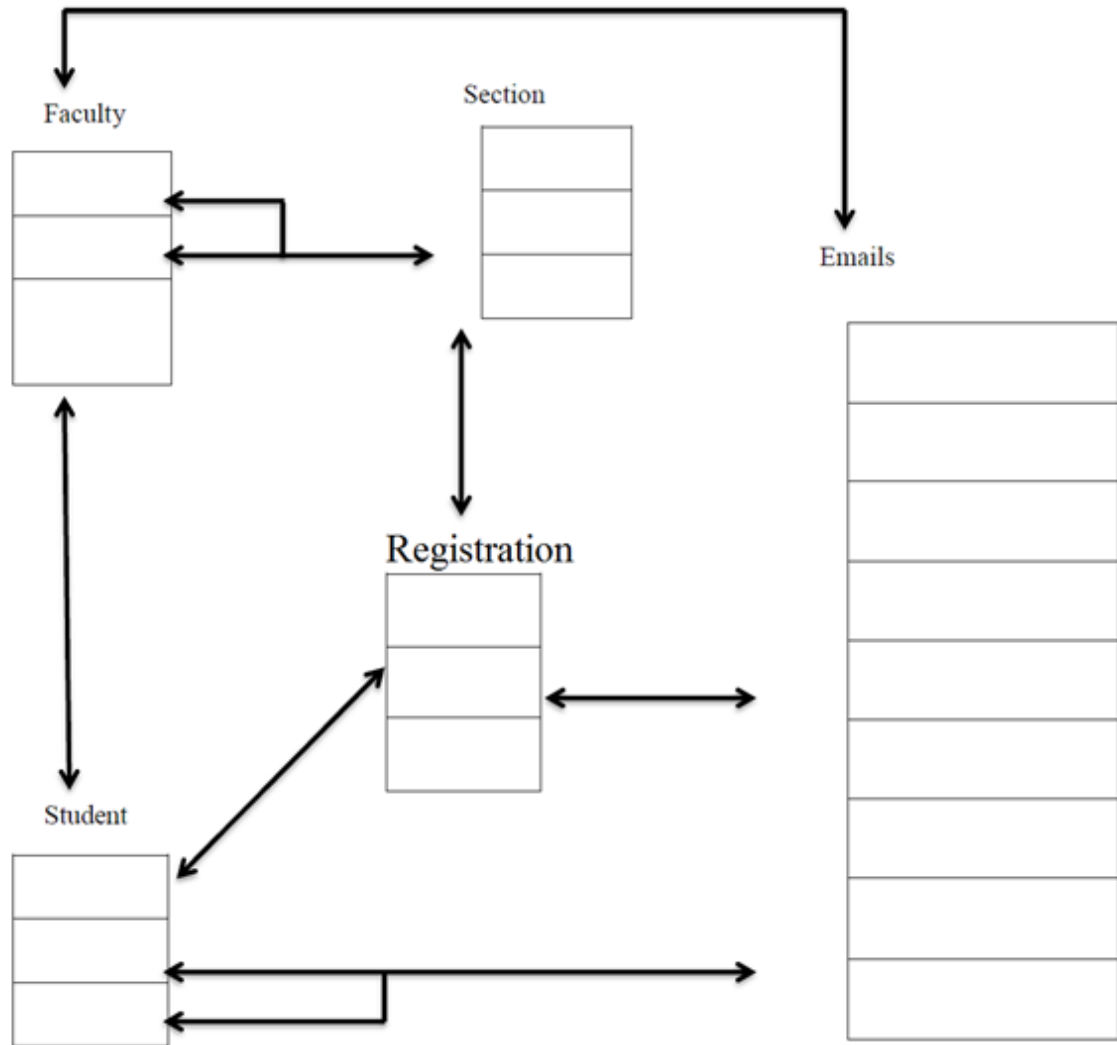


Figure 5: Integration of different tables with emails

**Database Design:** Five tables were used in the database design and they are listed below

- Student
- Section
- Registration
- Registration\_section
- Section\_students

**Student table:**

<b>Attribute</b>	<b>Type</b>	<b>Primary key</b>
name	Varchar	0
wkuid	Varchar	1
email	Varchar	0
major	Text	0
phone	Text	0

Table 1: Student table

**Section table:**

<b>Attribute</b>	<b>Type</b>	<b>Primary key</b>
Section_id	Integer	1
name	Text	0
course	Text	0
time	Text	0
location	Text	0

Table 2: Section table



**Registration table:**

<b>Attribute</b>	<b>Type</b>	<b>Primary key</b>
Registration_id	Integer	1
name	Text	0

Table 3: Registration table

**Registration\_section table:**

<b>Attribute</b>	<b>Type</b>	<b>Primary key</b>
Registration_id	Integer	0
Section_id	Integer	0

Table 4: Registration\_section table

**Section\_students table:**

<b>Attribute</b>	<b>Type</b>	<b>Primary key</b>
Section_ID	Integer	0
Student_ID	Integer	0

Table 5: Section\_students table

The registration\_section and Section\_students tables are not explicitly shown in the Figure 5 as their only functionality is to establish mapping between the registration, student and section tables.

## **Candidate network generation scheme**

The candidate networks share joint expressions between the data in different relations/tables. This results in a set of intermediate results [20] which in turn helps in using them in the computation of multiple candidate networks. The candidate network generator inputs the set of keywords  $k_1, k_2, k_3, \dots, k_m$ , the tuple sets which are not empty and the maximum candidate networks' size  $T$  and outputs a complete and non-redundant set of the candidate networks. The key challenge is to avoid the generation of redundant joining networks of tuple sets. For this the joining network of tuples must be non-minimal. The minimal total joining network of tuples is produced as an output by the candidate network algorithm [20]. The output does not contain any redundant candidate networks

## **3.2 Using the native file in extensible Markup Language (XML)**

### **Background details**

XML is widely accepted as the most efficient standard scheme for the data exchange in the Business to Business applications [11]. The data storage scheme is different from the traditional relational databases as the data in the XML is a self-describing and irregular and it is considered to be semi-structured data. The XML schema structure facilitates the process of query evaluation and it helps in querying the schema without the need to consider the original data. Most important feature of keyword search is that it enables easy access to search information without having to know the complex

query structures in the querying mechanism or there is no need to have a prior knowledge about the working of the queries or the querying mechanism. Since the data storage is not the main concern in the XML scheme the data can be of potentially large sizes [11]. The volume of data is not a major concern in the XML based processing.

### **Query languages and querying mechanism**

The query languages used to query and process the semi-structured format of the data for efficient XML processing are XQuery and XPATH [11]. In case of the structured perspective to select the structural information the data needs to be explored and the elements are identified which form the tree structure. The elements are selected as per the desired requirement in the form of tree pattern queries which are also known as twigs or twig queries. The twig queries select the elements very efficiently as per the specified tree structures which help in retrieving the most relevant information. The query processing mechanism is shown in Figure 7.

### **XML data storage mechanism**

The organizational information is stored using native file in eXtensible Markup Language(XML) format [15]. XML is the standard format for data exchange over the internet, providing interoperability between different business applications. XML data is represented in tree structured fashion, and allows sophisticated data in the relations being represented in XML format. In a business application there is definitely a need where one has to exchange data between a database and diversified systems which can be other applications or any other database, etc [23]. As the same information exchange mechanism may not be used by everyone, XML serves as a language to capture all the

semantics kept intact. The most important feature about the XML is it is easily understandable. A hierarchical format is perhaps the most structured format that is very easily readable by humans. XML helps in interoperability between different platforms and it is used in wide range of web applications as an exchange medium. In our application considering the scenario of the same course registration information as shown in Figure 5, it can be represented by the following XML segment as in Figure 6. In XML, the internal nodes are the markups, providing structure and semantic information in an application, and the real data are stored in the leaf node.

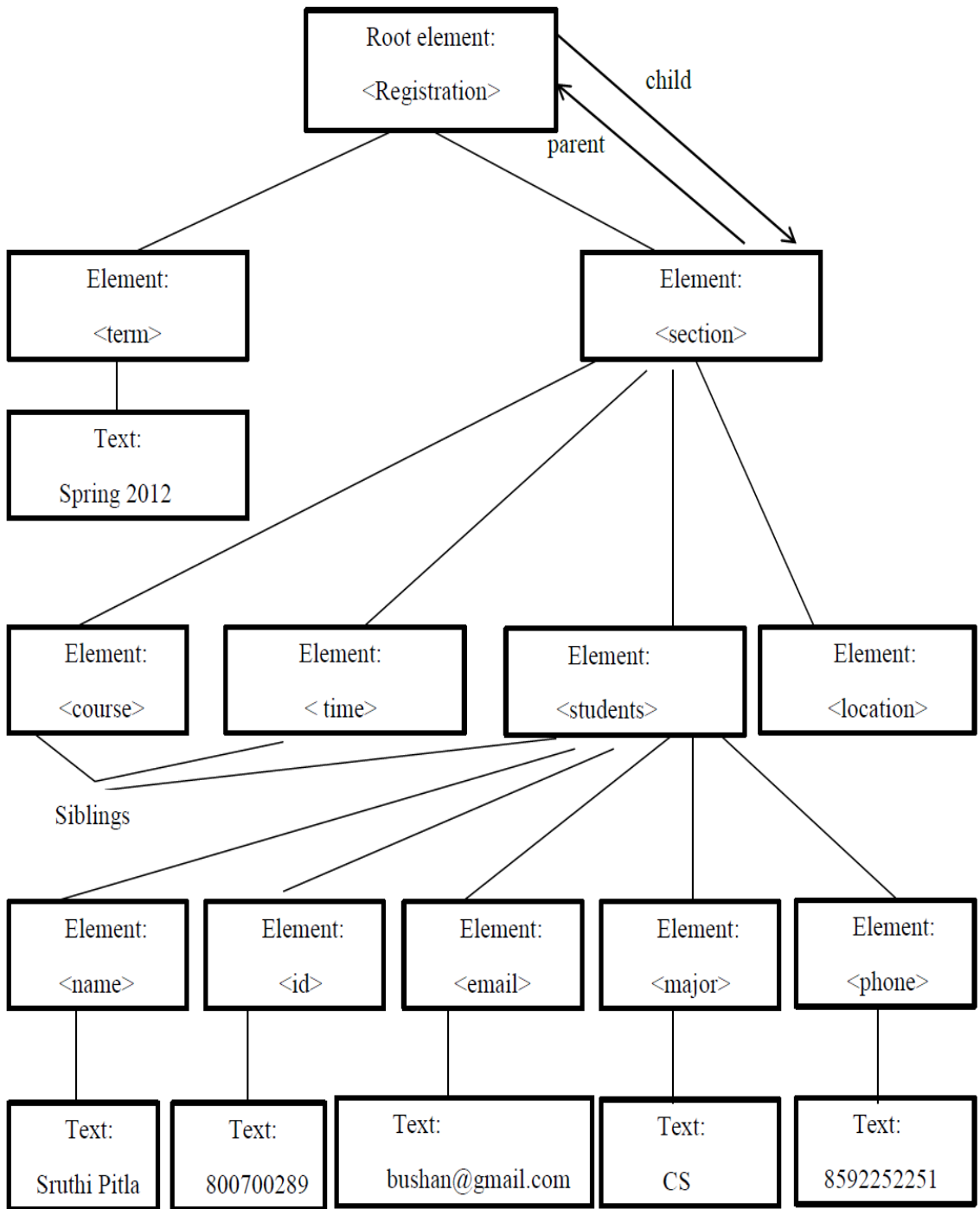


Figure 6: Data represented as XML tree structure

To perform a query “CS570”, the query mechanism is generated as shown in Figure 7

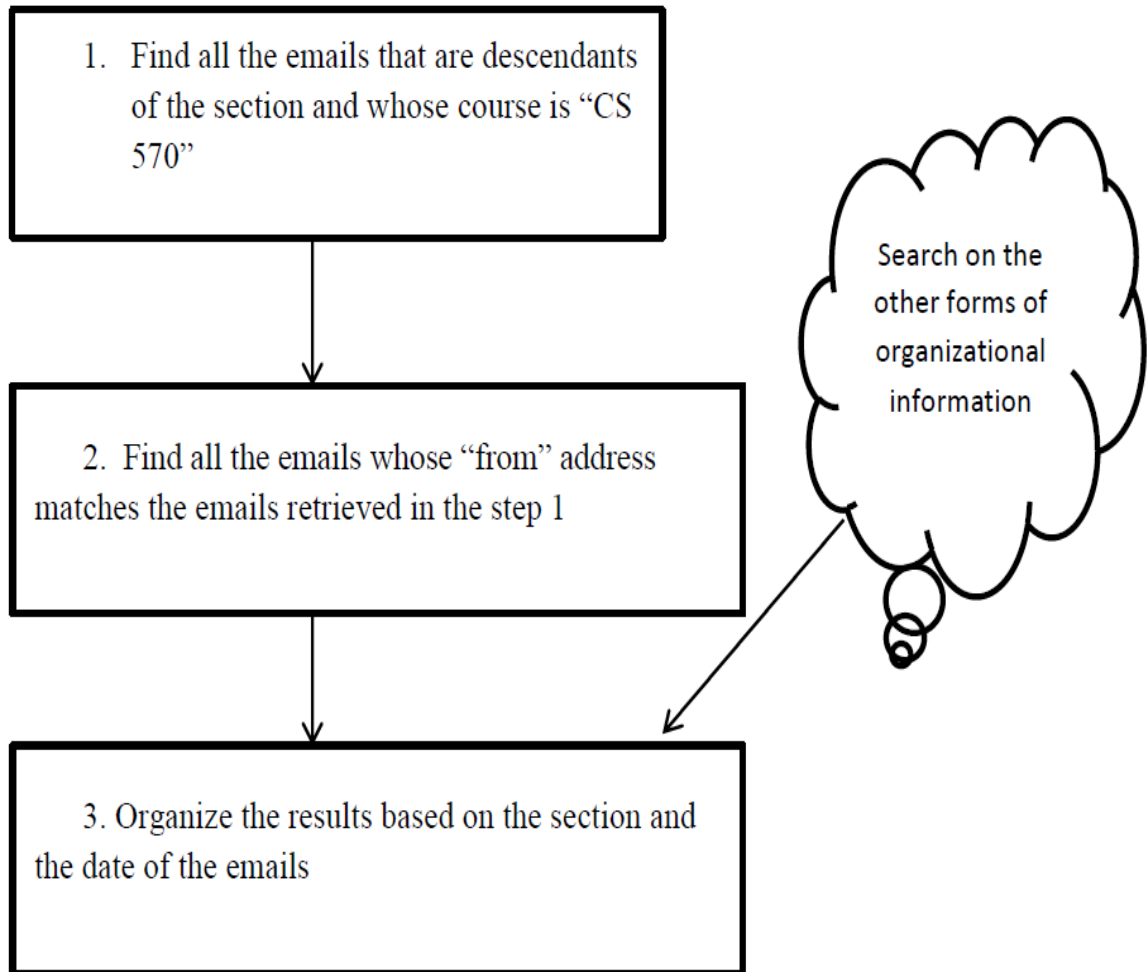


Figure 7: XML Twig Query processing mechanism

In this scenario, the query processing can also be thought of as the twig queries in XML data processing [11].

## **XML Twig Query processing**

There may be many twig patterns in an XML database. Finding all such occurrences of twig patterns in an XML database is considered as a major operation for the efficient extraction and the evaluation of XML queries. Many algorithms have been proposed to perform the process of finding different twig patterns. A Holistic Twig join labeling scheme [17] is considered to be appropriate with our implementation. According to this scheme only the labels of the lead nodes must be accessed which further yields to the efficient scanning of large number of elements[17] in the tree and it also improves the query processing and evaluation mechanisms. Below is the figure which illustrates the document which contains the information in XML form

```
<?xml version="1.0"?>
<!DOCTYPE registration SYSTEM "C:\Users\Bhushan\Desktop\xml.dtd">
<registration>
  <term>
    Spring2012
  </term>
  <section>
    <course>CS570</course>
    <time>MWF2</time>
    <location>SH1101</location>
    <students>
      <student>
        <name>Eric Smith</name>
        <id>800555333</id>
        <email>abc@email.com</email>
        <major>CS</major>
        <phone>8005554444</phone>
      </student>
      <student>
        <name>Sruthi Pitla</name>
        <id>800555444</id>
        <email>bushan@email.com</email>
        <major>CS</major>
        <phone>8005554555</phone>
      </student>
    </students>
  </section>
</registration>
```



```

</student>
    <student>
        <name>Harsha</name>
        <i d>800555222</i d>
        <email>harsha@email.com</email>
        <major>CS</major>
        <phone>8005554111</phone>
    </student>
    <student>
        <name>Ramu</name>
        <i d>800555888</i d>
        <email>ramu@email.com</email>
        <major>CS</major>
        <phone>8005554999</phone>
    </student>
    <student>
        <name>Rahul</name>
        <i d>800555222</i d>
        <email>rahul@email.com</email>
        <major>CS</major>
        <phone>8005554333</phone>
    </student>
</students>
</section>
</registration>

```

Figure 8 : Structure of data in XML format

The purpose of a DTD here is to define the structure of our XML document. It defines the structure with a list of legal elements which point to different data.

```
<?xml version="1.0" encoding="UTF-8"?>
<ELEMENT time (#PCDATA)>
<ELEMENT term (#PCDATA)>
<ELEMENT students ((student+))>
<ELEMENT student ((name, id, email, major, phone))>
<ELEMENT section ((course, time, location, students))>
<ELEMENT registration ((term, section))>
<ELEMENT phone (#PCDATA)>
<ELEMENT name (#PCDATA)>
<ELEMENT major (#PCDATA)>
<ELEMENT location (#PCDATA)>
<ELEMENT id (#PCDATA)>
<ELEMENT email (#PCDATA)>
<ELEMENT course (#PCDATA)>
```

Figure 9 : DTD for XML data in the document

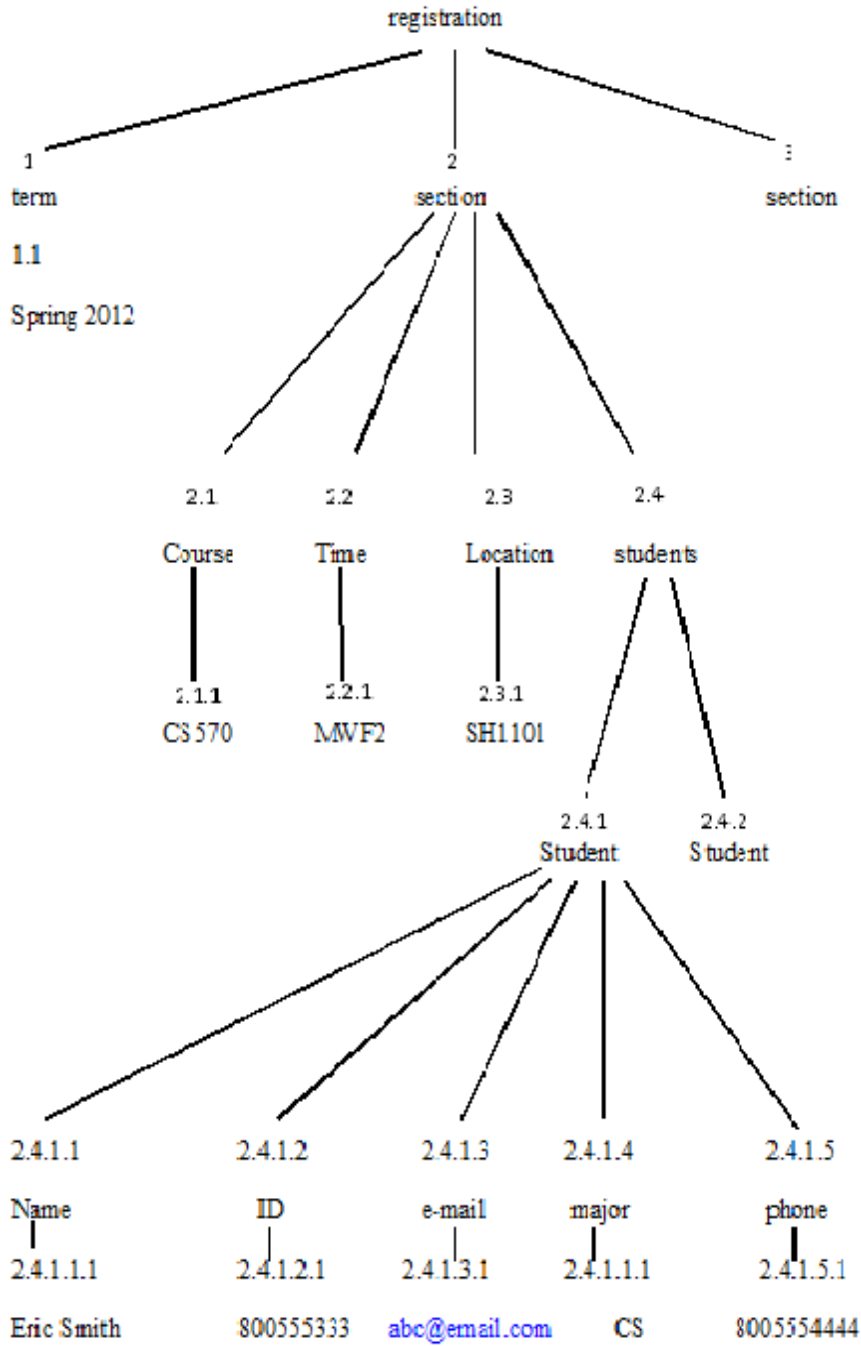


Figure 10 : XML tree structure with appropriate labeling of nodes

## **Twig Join Labeling Scheme**

A very good feature of this scheme is by using the label [17] value alone. The names of all the corresponding elements in that particular path can be easily retrieved in the top-down fashion which is the traversal from the root element to that corresponding element. Considering Figure 10 which shows an XML document with all the elements and the nodes properly labeled. For example, if label “2.4.1.3” is taken into consideration then with this label value we can examine that the path from the root node to retrieve the particular email is “registration/section/students/Student/email”. With this mechanism in order to evaluate the twig patterns, the only requirement is to access the labels of elements that only satisfy the leaf node condition in the query [17]. This also helps in matching the patterns in the intermediate path like “//section/students”. This helps in very good evaluation of the twig patterns, which are reliable and efficient.

### **3.3 Twig pattern matching**

For performing the pattern matching in the twigs there is a need to know the additional schema information from the proposed XML schema which will serve as a clue to parse the schema and better match the twig pattern. If we consider a particular tag from our XML document, the clue [17] is the all possible and distinct names of children which are descendants of that particular tag. These child elements can be easily derived from the DTD or the schema. Let us denote all the possible names of children of elements in a set CE. CE (t) denotes all the possible child elements which are direct descendants of parent node (t) [17]. For example, consider the DTD in the figure here. The tags of all possible children of a section are course, time, location and students are

author, title and chapter. So  $CE(\text{section}) = \{\text{course, time, location, students}\}$ . Using this idea we can easily map the elements. This approach is a straight forward mechanism here and we only need to scan the elements whose tags appear in leaf nodes of query [17]. If an element gets visited, we use Finite state machine mechanism to convert its label into element names along the path from the root to it and later perform the different string matching operations on it. If the path from the root to this element matches with our intended desired pattern, then the matching answers can be traced directly so this way we can find our desired output result by scanning the whole document which can be treated as an input list of strings, and sequentially the strings are matched till we find the exact match.

### **How the target data for searching in this project different from traditional relational data or XML data**

- The data is extracted from diversified heterogeneous sources which are distributed on the servers on the internet. The search operation here is to filter the emails and the main focus is on searching the email system. The organizational information is extracted from the other sources and as a result the format of data extracted is significantly different from each other and as the data extracted is not uniform or not of the similar format handling the query process is not very easy. There is no uniform query language which handles all the different data sources efficiently and which could be very effective in retrieving the relevant results.
- The other thing which must be effectively handled is the privacy and security issues of the data of the organization. The organization may contain different levels and

each level may be assigned to a different sensitivity level and there is definitely a need to protect the privacy of the organizational and personal data. Consider Figure 5 by taking the faculty member into perspective. The faculty member should be given appropriate privileges to access the data segments related to him/her i.e.; he/she should only be able to access the emails and email specific content which is private to him/her. The access to student private emails and email specific information should be denied, otherwise it is considered as a big privacy concern as the information is considered to be disseminated to the unauthorized users. To address these critical issues adapters can be used to pull the data from a variety of data sources to the system. After the data becomes available in our target system, the candidate networks can be generated in a similar way as the traditional search mechanism in the relational database system.

## CHAPTER 4: IMPLEMENTATION DETAILS

### 4.1 Add-on for Mozilla Thunderbird

Mozilla Thunderbird is an open source email client developed by the Mozilla Foundation [14]. Thunderbird can manage emails and information from a variety of sources. It has features like full text search, message filtering, message grouping and labeling that help in managing and finding the messages. Mozilla also provides an open mechanism to allow add-on development for the Mozilla family of applications like Firefox and Thunderbird. It allows any organization to build their own extension which best fits their needs.

### 4.2 User Interface:

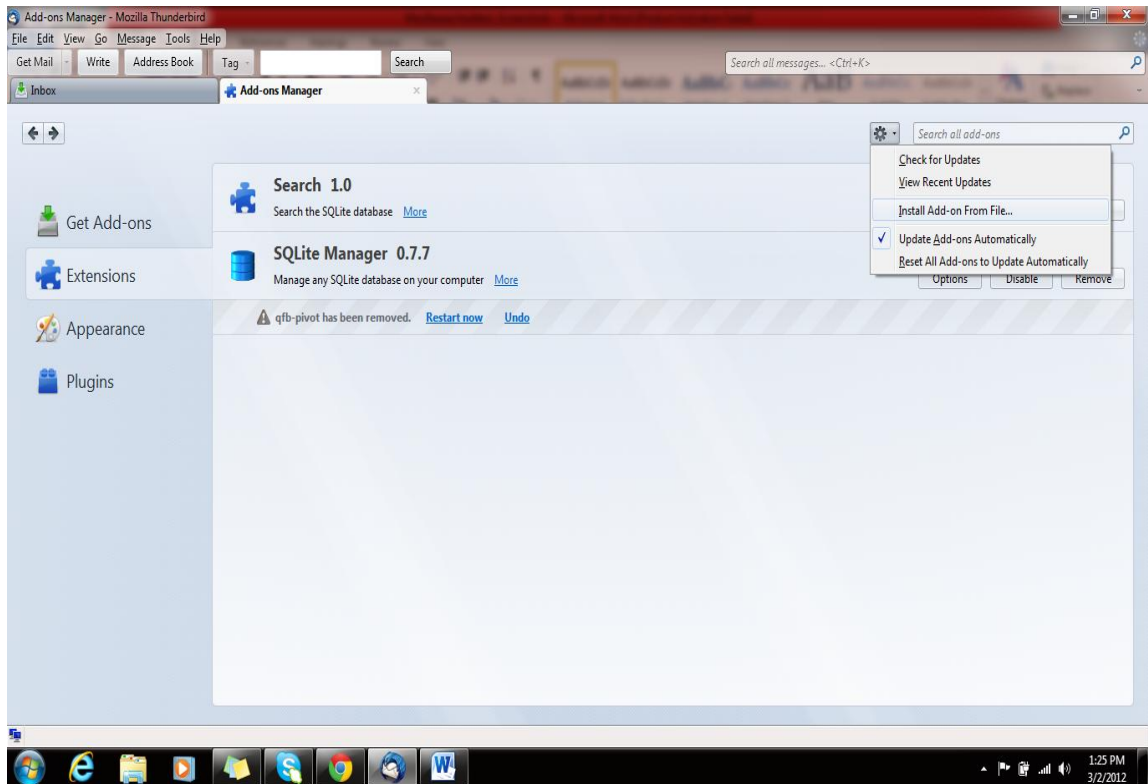


Figure 11 : Installation of add-on from a file

A new text box for query input is created in the add-on tool . The search results are returned in the same way as in a traditional search

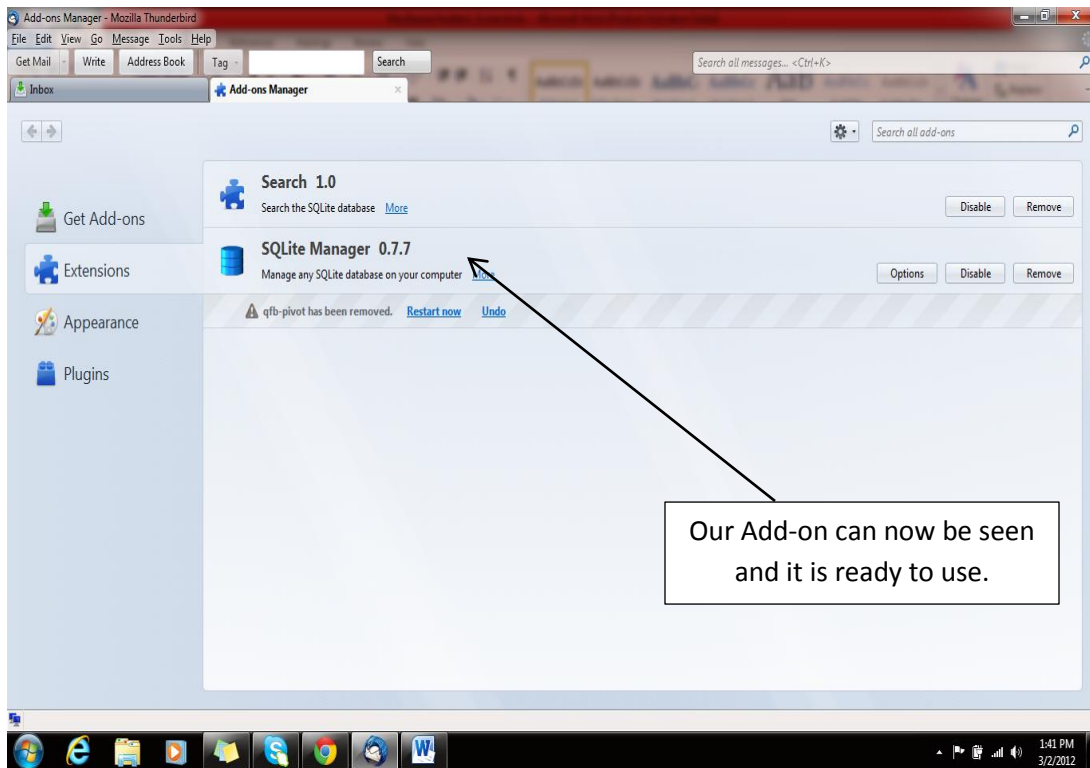


Figure 12 : The developed add-on is shown after it is installed



The user search query is given as input in the textbox as shown in the below screenshot. The query processing operation is then initiated to retrieve the relevant information.

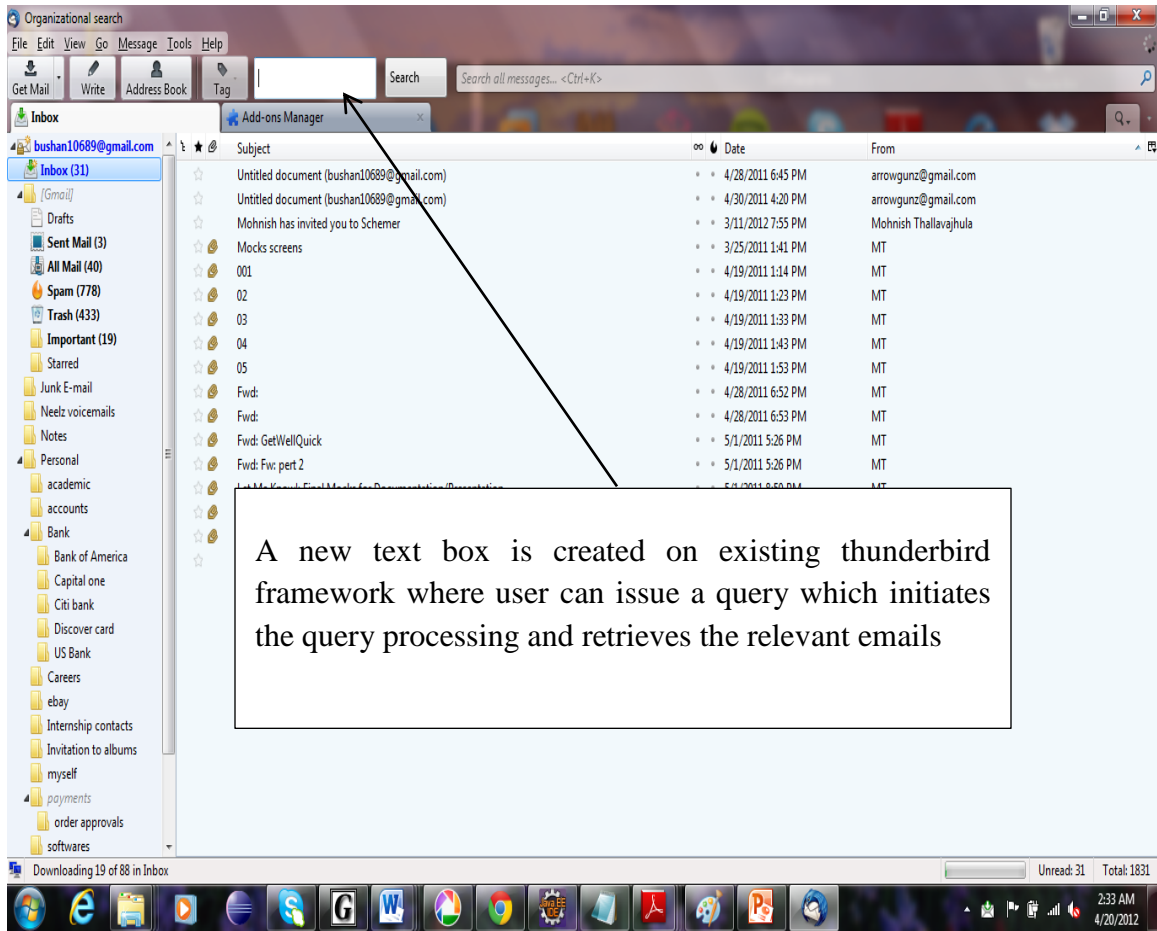


Figure 13 : The user input is given which initiates query processing

A particular Course number “CS530” is given as input and then the search button is clicked which then initiates the integrated search and retrieves the relevant emails with the content of the specific emails

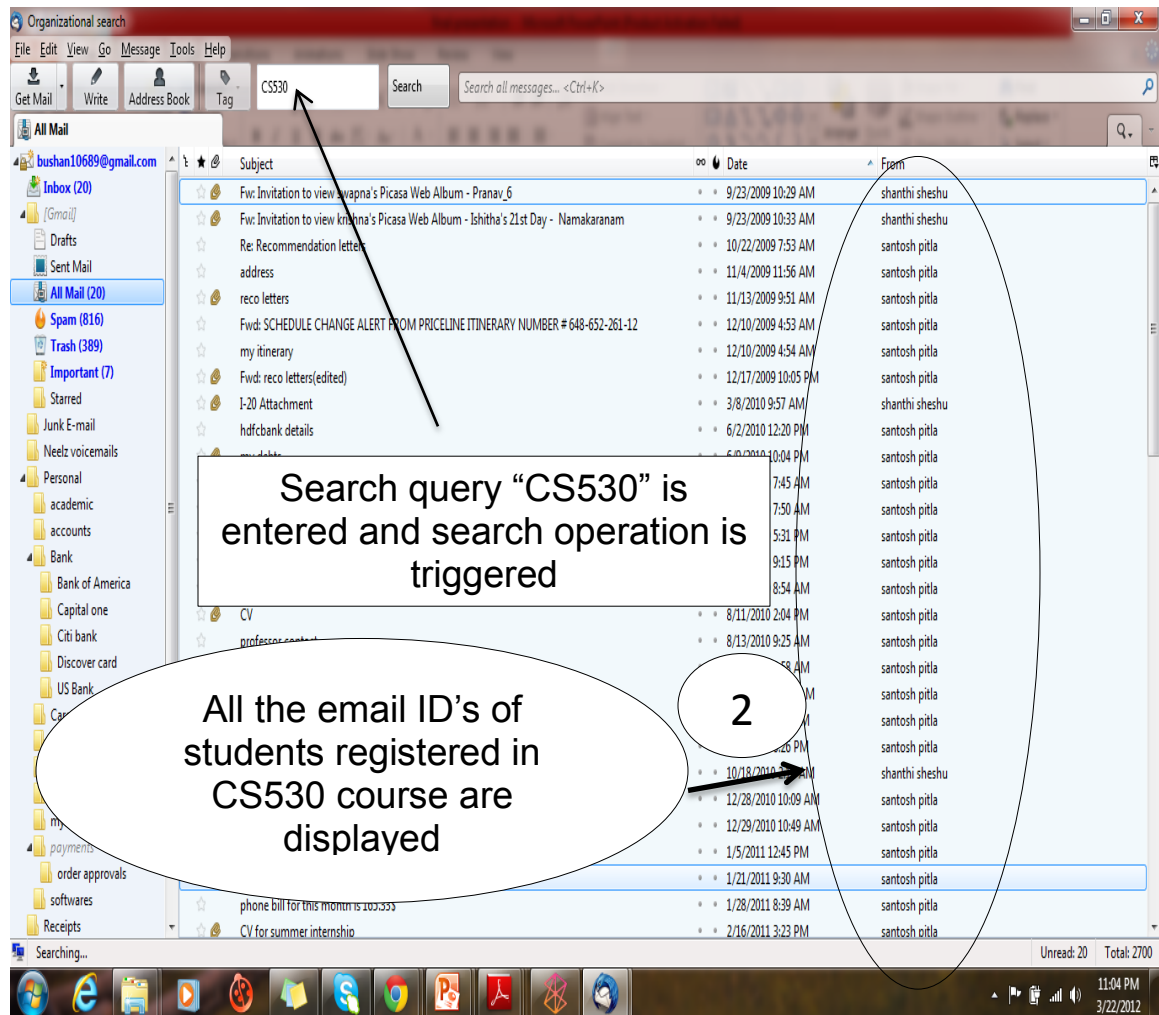


Figure 14 : The relevant emails are retrieved and displayed after query processing

## **Indexing and data storage**

### **Gloda (Global database)**

Gloda [13] is the indexing and search scheme in Thunderbird which serves as the centralized data storage repository for the thunderbird email search client. It provides us with the sophisticated full text search capabilities and also helps in the process of easy search results categorization. The gloda indexes are stored in the same SQLite database. The indexes are automatically generated and the auto indexes for each and every aspect of the email database are properly organized using the gloda. This helps in the faster and efficient access of the information that is stored in the email repository. The query processing operation takes care of the generation of the different candidate networks from the different entities in the SQLite database or from the data from the XML data sources.

### **Data storage:**

The gloda database is a SQLite database named "**global-messages-db.SQLite**"[13] and can currently be found in the user's Thunderbird profile directory. All the email related data which can be sent messages, received messages, message content, contact details, message headers and conversations are stored in this global repository. The advantage of this database file is that it can be moved to another location in the future so that it does not complicate backup procedures [13] and it is considered to be very efficient.

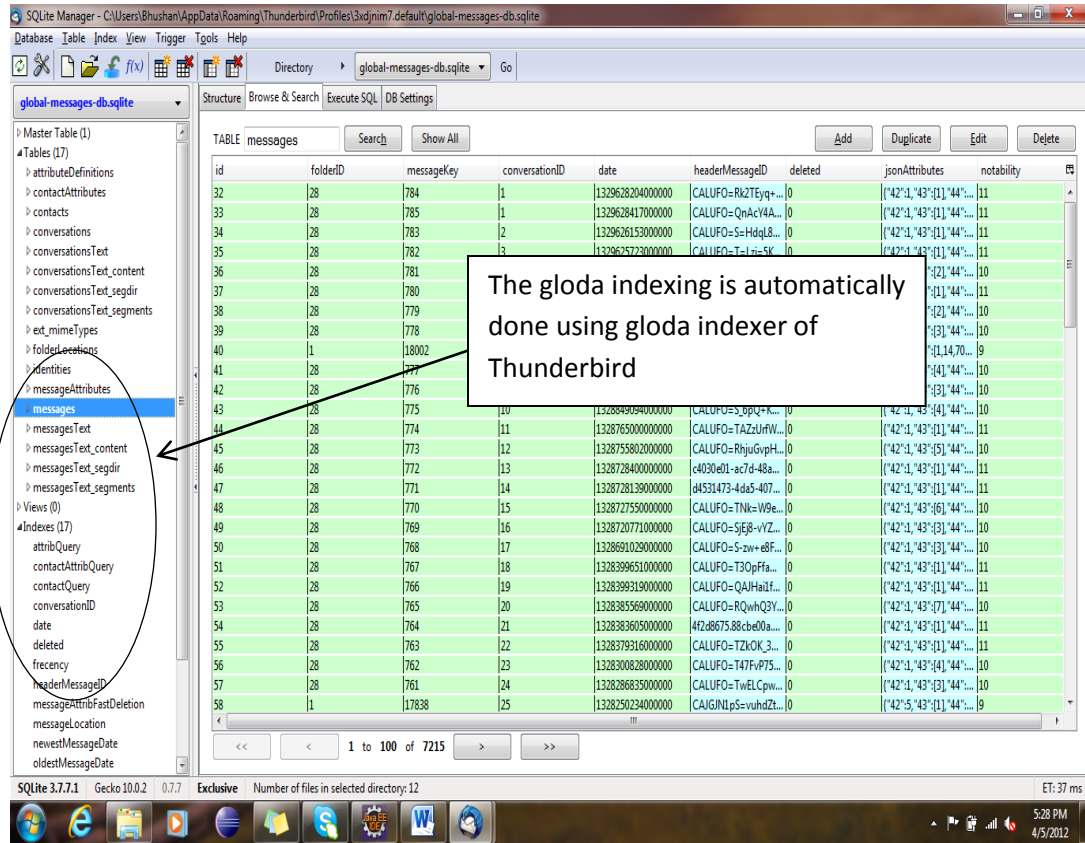


Figure 15 : Indexing mechanism in gloda database

### Gloda functionality as an index

Gloda functions as an index [13] to the stored data. It is important to carefully deal with this index as it can cause inconsistency in the granularity of files stored at different levels and it is not important to back it up and dealing with these indexes can cause problems in the backup mechanisms which are related to granularity of the files. All the indexes are also stored in the centralized global repository “global-messages-db.sqlite”

## **Placing the SQLite database at the proper location in the Thunderbird default profile folder**

For the proper functioning of the add-on the following must be done

- The SQLite database must be placed in the thunderbird profile folder.
- The SQLite database used in our add-on implementation is **western.sqlite**

### **Procedure to be followed in the different operating systems**

#### **In Windows environment:**

Profile folder [18] is located at the following location:

C:\Documents and Settings\<<Windows user name>\Application  
Data\Thunderbird\Profiles\<<Profile name>

#### **In Linux environment:**

Profile folder [18] is located at the following location:

~/.thunderbird/<Profile name>/

For third party build from Debian or Ubuntu, those builds store the profile folder here:

~/.mozilla-thunderbird<Profile name>

#### **In MAC OS X environment**

Profile folders[ 18] are in one of these locations:

~/Library/Thunderbird/Profiles/<Profile name>/

~/Library/Application Support/Thunderbird/Profiles/<Profile name>/

### **Experimental evaluations:**

The experimental results were taken based on the relevance of the query results. Quality of search is very difficult to verify, and can vary significantly from one application to another [11] depending on the different frameworks and methodologies. In order to quantify the search results retrieved on the web the precision and the recall are considered as important parameters to judge the efficiency of the retrieved search results. But relevance ranking definitely plays a major role when compared to the above mentioned parameters as most of the users expecting the relevant results to be retrieved and will only look at the first few results which were retrieved in order to satisfy his information needs. In the case of Email search, it is different when compared to web search. In this case, the user will try to go through a good number of emails to find the exact match or an almost correct match. In our experiment, we considered several measures such as the count of the emails that are retrieved, count of the emails that are relevant, count of the retrieved emails that are irrelevant, and the count of the relevant emails.

The count of the number of relevant emails is unknown, as we did not have any milestone set for the emails that are tagged already, however it is easier to find out the irrelevant emails through our experiment. We can infer that a good search utility or the search mechanism would be really handy in retrieving more relevant emails, which inevitably minimizes the count of the irrelevant emails. We did use common queries that conducted comparisons between normal text search and organizational search. The relevancy here is not a simple match and an email with a word that matches the keyword

doesn't mean that the email retrieved is relevant. Relevancy can only be determined by the user to mean that the user would see the exact or the specific email which he intends to see through the process of the information retrieval. Below is a table showing a rough estimation of the comparison between the organizational search and the normal text search

	<b>Normal Text Search</b>	
<b>Queries issued</b>	<b>Retrieved</b>	<b>Irrelevant</b>
CS 549	160	30
Graduate Committee	80	16

Table 6: Expected results of normal text search

	<b>Organizational Search</b>	
<b>Queries issued</b>	<b>Retrieved</b>	<b>Irrelevant</b>
CS 549	160	24
Graduate Committee	80	5

Table 7: Expected results of organizational search

Comparison of retrieved and irrelevant results in the case of normal search and the organizational search:

We have considered 160 email results as a bench mark in case of both the normal and organizational searches.

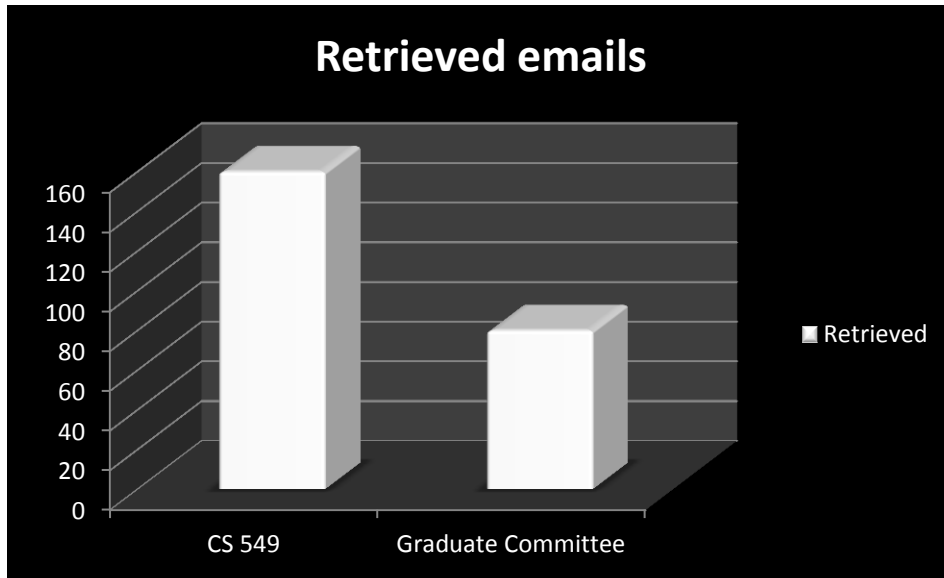


Figure 16: Number of retrieved email results in case of normal and organizational searches



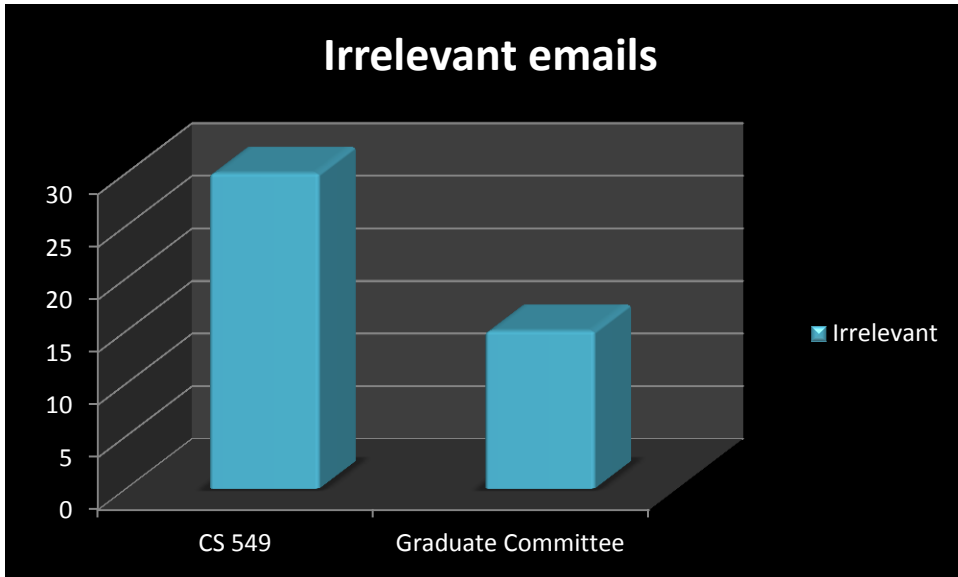


Figure 17: Representation of expected count of irrelevant email results in normal search

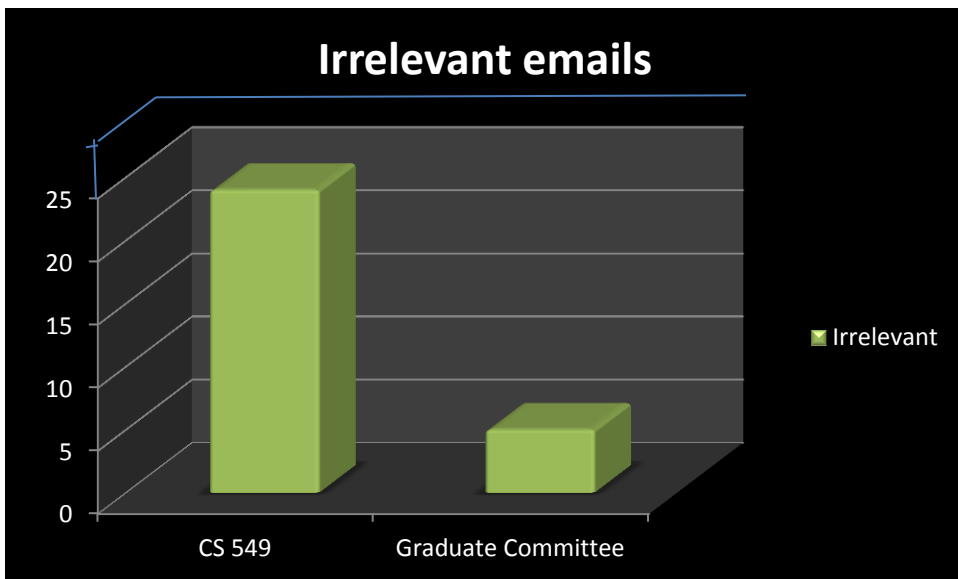


Figure 18: Representation of expected count of irrelevant email results in organizational search

Query issued	Normal search		Organizational search	
	Relevant emails	Irrelevant emails	Relevant emails	Irrelevant emails
CS 549	81.25%	18.75%	85%	15%
Graduate Committee	80%	20%	93.75%	6.25%

Table 8: Expected performance results

From the above comparison, it is clear that the organizational search is effective in retrieving many more relevant emails than using normal text search. We did not intend to use these limited experimental results that we have retrieved to show that the organizational search proposed in this research is definitely more effective than the normal text search in all the cases, but we believe that it definitely does offer an important facet in the email search system.

## CHAPTER 5: CONCLUSIONS

With the recent advancement in the area of storage mechanism and storage technologies, email collections or archives can definitely be considered as one of the most important data repositories to serve different purposes for many users. Social networking applications have been very successful by far on building online communities, however it lacks a feature of utilizing the networking information in email management, especially when the enterprise setting is considered. The information utilization is not well handled and has not been well studied. Our study and the experimental results have shown that using organizational information in the email search can significantly improve the search quality by retrieving the more relevant emails for the user. The organizational information thus provides a meaningful facet to present the efficient search results to the users, and thus provides better information navigation and inevitably provides a very good experience to the user. A common framework which could push the information from different sources into our tool for achieving better information integration would be the future prospect. The ranking and order of relevancy of emails can also be implemented in order to make the tool more user-friendly.

## BIBLIOGRAPHY

- [1] Soumen Chakrabarti, Jeetendra Mirchandani and Arnab Nandi. Spin: searching personal information networks. *SIGIR*, 674, 2005.
- [2] Yu Xu, Yannis Papakonstantinou, Efficient LCA based keyword search in XML data, *EDBT'08: Advances in database technology*, 535-546, 2008, Nantes, France.
- [3] Bhavana Bharat Dalvi, Meghana Kshirsagar, S. Sudarshan, Keyword search on external memory data graphs, In *Proceedings of the VLDB Endowment*, 1(1), 1189-1204, August 2008.
- [4] Luke McDowell, Oren Etzioni, Alon Y. Halevy: Semantic email: theory and applications. *J. Web Sem.* 2(2): 153-183 (2004).
- [5] CALO – Cognitive Assistant that Learns and Organizes. <http://www.ai.sri.com>
- [6] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *ICDL*, 85–94. ACM, 2000.
- [7] A. Cheyer, J. Park and R. Giuli. IRIS: Integrate. Relate. Infer. Share. In *ISWC 2005 Workshop on The Semantic Desktop*, 2005.
- [8] X. Dong, A. Halevy and J. Madhavan. Reference reconciliation in complex information spaces. In *ACM SIGMOD Conference*, 85–96, 2005.
- [9] D.R. Karger and D. Quan. Haystack: A user interface for creating, browsing and organizing arbitrary semi-structured information. In *Human Factors in Computing Systems (ACM CHI)*, 777–778, 2004
- [10] P. Domingos. Multi relational record linkage. In *Multi-Relational Data Mining*

- Workshop: collocated with ACM SIGKDD 2004*, 31 – 48, (2004).
- [11] Z Vagena, M. Moro and V.J. Tsotras. 2004. Twig query processing over graph-structured XML data. In *Proceedings of the 7th International Workshop on the Web and Databases: collocated with ACM SIGMOD/PODS 2004* (Web DB '04). ACM, New York, NY, USA, 43-48.
- [12] David Hawking, Nick Craswell, Peter Bailey, Kathleen Griffiths: Measuring Search Engine Quality. *Information Retrieval* 4(1): 33-59 (2001).
- [13] Gloda, The search utility for thunderbird,  
<https://developer.mozilla.org/en/Thunderbird/Gloda>
- [14] Thunderbird, <https://developer.mozilla.org/en/Thunderbird/>
- [15] C.M. Sperberg-McQueen and H. Thompson (2000). XML Schema, from  
<http://www.w3.org/XML/Schema.html>
- [16] Extension development <https://developer.mozilla.org/en/Extensions>
- [17] TJFast: effective processing of XML twig pattern matching at proceedings  
WWW '05 Special interest tracks and posters of the 14th international conference on  
World Wide Web ACM New York, NY, USA 2005
- [18] [http://kb.mozillazine.org/Profile\\_folder\\_-\\_Thunderbird](http://kb.mozillazine.org/Profile_folder_-_Thunderbird)
- [19] <http://www.sqlite.org/features.html>
- [20] Vagelis Hristidis, Yannis Papakonstantinou DISCOVER: Keyword Search in  
Relational Databases at Proceeding VLDB'02 Proceedings of 28<sup>th</sup> international  
conference on Very Large Data Bases

[21] Jeffrey Xu Yu, Lu Qin, Lijun Chang , Keyword search in relational databases: A survey

[22] Features of Thunderbird <http://www.mozilla.org/en-GB/thunderbird/features/>

[23] XML storage [http://www.web-enable.com/business/XML\\_beyond\\_hype.asp](http://www.web-enable.com/business/XML_beyond_hype.asp)

