

**TUGAS AKHIR MATA KULIAH EC7010**

**(Dosen Dr. Ir. Budi Raharjo)**

**DATA MINING EMAIL**



**OLEH**

**MOHAMAD HAIRYANOV**

**NIM 23203118**

**PROGRAM MAGISTER TEKNIK ELEKTRO  
BIDANG KHUSUS TEKNOLOGI INFORMASI – DIKMENJUR  
INSTITUT TEKNOLOGI BANDUNG**

## DAFTAR ISI

DAFTAR ISI .....	i
ABSTRAK .....	ii
BAB I : PENDAHULUAN	
I.1 Latar Belakang .....	1
I.2 Tujuan .....	2
I.3 Pembatasan Masalah .....	2
I.4 Sistematika Penulisan .....	2
BAB II : TINJAUAN PUSTAKA	
II.1 Data Mining .....	3
II.2 Data Mining Email .....	3
BAB III : ANALISA KEBUTUHAN DA DISAIN PROGRAM	
III.1 Analisa Masalah .....	5
III.2 Kemungkinan Pemecahan Masalah .....	5
III.3 Disain Program .....	6
BAB IV : IMPLEMENTASI DAN ANALISA	
IV.1 Melakukan <i>Search</i> Terhadap Mailbox .....	8
IV.2 Mempersiapkan Database .....	8
IV.3 Pembuatan Pengurai Email ( <i>Email Parser</i> ) .....	9
IV.4 <i>Query</i> Pada Database .....	10
BAB V : PENUTUP	
V.1 Kekurangan Dan Saran .....	15
V.2 Kesimpulan .....	16
LAMPIRAN-LAMPIRAN	
Lampiran 1: Skript Pembuatan tabel .....	17
Lampiran 2: Skript Pengurai Email ( <i>Mail Parsing</i> ) .....	18
Lampiran 3: Skript Mengembalikan Pengkodean MIME .....	20
Referensi .....	21

## ABSTRAK

Saat ini email mempunyai peranan yang sangat besar dalam berkomunikasi dengan lebih cepat dan lebih murah dibandingkan dengan metoda komunikasi yang lebih tradisional. Oleh sebab itu email merupakan media komunikasi yang paling banyak digunakan.

Untuk menggali informasi dari tumpukan email yang tidak sempat terbaca diperlukan suatu cara yang cepat yang disebut "*Data Mining Email*"

Secara sederhana dapat diartikan bahwa *Data mining* adalah suatu algoritma di dalam menggali informasi berharga yang terpendam atau tersembunyi pada suatu koleksi data (database) yang sangat besar sehingga ditemukan suatu pola yang menarik yang sebelumnya tidak diketahui

Yang dimaksudkan dengan *data mining email* adalah *Data mining* yang diterapkan pada koleksi email sehingga kita bisa mendapatkan informasi berharga yang tersembunyi didalam tumpukan email tersebut .

Kata kunci : *data mining, email, data mining email, database, main body, attachment, mailbox.*

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Saat ini email mempunyai peranan yang sangat besar dalam berkomunikasi dengan lebih cepat dan lebih murah dibandingkan dengan metoda komunikasi yang lebih tradisional. Oleh sebab itu email merupakan media komunikasi yang paling banyak digunakan, baik untuk kebutuhan perorangan maupun untuk kebutuhan organisasi, bahkan alamat email sudah merupakan atribut dari identitas seseorang ataupun organisasi.

Dengan relasi yang semakin luas memungkinkan si pemilik alamat email menerima banyak sekali email setiap hari dan terus bertambah tanpa kita sempat membacanya satu persatu. Padahal dari email-email tersebut memungkinkan banyak sekali informasi-informasi berharga yang harus kita ketahui dan banyak pula email-email yang harus hati-hati kita sikapi misalnya email-email yang berisi virus, email yang berisi ancaman atau serangan atau email-email yang berisi rahasia berharga yang tidak bisa diukur nilai kerugiannya dengan uang apabila informasi ini bocor. Bahkan mungkin saja kita pernah menerima email yang tidak perlu dibaca dan hanya memenuhi database email yang kita miliki karena email-email tersebut dikirimkan oleh *spammer*.

Untuk mendapatkan informasi-informasi berharga yang tersembunyi itu diperlukan suatu cara agar dapat menemukan informasi berharga yang tersembunyi dalam tumpukan email yang ada dalam database. Untuk alasan itulah mengapa tugas akhir ini dibuat dengan mengambil judul "*Data Mining Email*" dengan titik berat pembahasan terletak pada *main body* dan *attachment*.

Secara sederhana dapat diartikan bahwa *Data mining* adalah suatu cara dalam menggali informasi yang terpendam dari database yang besar sehingga menjadi informasi yang sangat berharga<sup>[1]</sup>. *Data mining* merupakan proses penemuan yang efisien sebuah pola terbaik yang dapat menghasilkan sesuatu yang bernilai dari suatu koleksi data yang sangat besar<sup>[2]</sup>. Dengan demikian *Data mining* adalah suatu pola yang menguntungkan dalam melakukan *search* pada sebuah database yang terdapat pada sebuah model. Proses ini dilakukan berulang-ulang hingga didapat satu set pola yang memuaskan yang dapat berfungsi sesuai yang diharapkan<sup>[3]</sup>.

Yang dimaksudkan dengan *data mining email* adalah *Data mining* yang diterapkan pada koleksi email yang tersimpan dalam database sehingga kita bisa mendapatkan informasi berharga yang tersembunyi didalam tumpukan email tersebut .

## 1.2 Tujuan

Berdasarkan uraian yang terdapat pada latar belakang, penulis tertarik untuk membahas *Data Mining Email* terutama penggalian informasi yang terdapat pada *main body* dan *attachment* dengan tujuan menghasilkan suatu software yang dapat :

- mencari email-email yang berisi pesan-pesan tertentu,
- mengklasifikasikan email-email berdasarkan isinya

## 1.3 Pembatasan Masalah

Agar pembahasan masalah tidak terlalu luas dan lebih terfokus pada tujuan maka beberapa pembatasan lingkup penelitian adalah sebagai berikut.

- Bagian email yang diteliti hanyalah bagian *from*, *main body* dan *attachment* sedangkan bagian lainnya diabaikan.
- Tugas *data mining* yang dilakukan adalah pencarian kata dan klasifikasi.
- Email-email yang dibahas adalah email-email yang mempunyai format *mbx* yang terdapat pada *Mozilla mailbox*.
- Bahasa pemrograman yang digunakan adalah PERL, khususnya fungsi-fungsi yang terdapat pada CPAN.
- Database yang digunakan adalah PostgreSQL
- *Console-base* yang digunakan klien adalah *psql*.
- *Command-line utilities* yang sering digunakan adalah *find*, *file* dan *antiword*.

## 1.4 Sistematika Penulisan

Format pelaporan tesis yang digunakan adalah sebagai berikut.

- BAB I PENDAHULUAN berisi latar belakang, tujuan, Pembatasan Masalah dan Sistematika Penulisan.
- BAB II TINJAUAN PUSTAKA berisi dasar-dasar teori tentang *data mining* dan *data mining email*.
- BAB III ANALISA KEBUTUHAN DAN DESAIN PROGRAM berisi analisa permasalahan, kemungkinan pemecahan masalah dan desain program.
- BAB IV IMPLEMENTASI DAN ANALISA berisi percobaan-percobaan untuk mendapatkan hasil yang baik serta analisa hasil percobaan.
- BAB V KESIMPULAN DAN SARAN

## **BAB II**

### **TINJAUAN PUSTAKA**

#### **II.1 Data Mining**

Beberapa pengertian *data mining* yang berhasil penulis himpun dari beberapa pendapat adalah sebagai berikut.

1. Secara sederhana dapat didefinisikan bahwa *Data mining* adalah ekstraksi informasi atau pola yang penting atau menarik dari data yang ada di database yang besar sehingga menjadi informasi yang sangat berharga<sup>[1]</sup>.
2. *Data mining* merupakan proses penemuan yang efisien sebuah pola terbaik yang dapat menghasilkan sesuatu yang bernilai dari suatu koleksi data yang sangat besar<sup>[2]</sup>.
3. *Data mining* adalah suatu pola yang menguntungkan dalam melakukan *search* pada sebuah database yang terdapat pada sebuah model. Proses ini dilakukan berulang-ulang (*iterasi*) hingga didapat satu set pola yang memuaskan yang dapat berfungsi sesuai yang diharapkan<sup>[3]</sup>.
4. *Data mining* adalah sebuah *class* dari suatu aplikasi database yang mencari pola-pola yang tersembunyi di dalam sebuah group data yang dapat digunakan untuk memprediksi perilaku yang akan datang<sup>[7]</sup>.

Berdasarkan beberapa pengertian diatas dapat ditarik kesimpulan bahwa *data mining* adalah suatu algoritma di dalam menggali informasi berharga yang terpendam atau tersembunyi pada suatu koleksi data (database) yang sangat besar sehingga ditemukan suatu pola yang menarik yang sebelumnya tidak diketahui. Oleh sebab itu istilah *data mining* sering disalahgunakan untuk menggambarkan perangkat lunak yang mengolah data dengan cara yang baru. Sebenarnya perangkat lunak *data mining* bukan hanya mengganti presentasi, tetapi benar-benar menemukan sesuatu yang sebelumnya belum diketahui menjadi muncul diantara sekumpulan data yang ada. Bahkan dengan menggunakan *data mining* dapat memprediksikan perilaku dan tren yang akan terjadi kemudian, sehingga bisa membuat para pengusaha menjadi lebih proaktif dan dapat mengambil keputusan dengan benar.

*Data mining* muncul setelah banyak dari pemilik data baik perorangan maupun organisasi mengalami penumpukan data yang telah terkumpul selama beberapa tahun, misalnya data pembelian, data penjualan, data nasabah, data transaksi, email dan sebagainya. Kemudian muncul pertanyaan dari pemilik data tersebut, apa yang harus dilakukan terhadap tumpukan data tersebut.

#### **II.2 Data Mining Email**

Sebagaimana pertumbuhan penggunaan internet yang sangat pesat akhir-akhir ini, maka email sebagai sarana komunikasi elektronik menjadi semakin luas penggunaannya. Setiap hari jutaan orang menggunakan email untuk saling bertukar pesan karena dapat dilakukan dengan cepat dan biaya yang cukup murah. Oleh sebab itu pertumbuhan pengguna email merupakan suatu deret ukur.

Menurut Phil Wolf dalam tulisannya tentang *Data Mining Email*, mengatakan bahwa “40% dari informasi-informasi penting yang dimiliki oleh perusahaan tersimpan di *email box*, tersembunyi dari *intranet search engines*, atau di kunci dalam desktop.”<sup>[4]</sup> Padahal kumpulan email tersebut kaya dengan informasi tentang :

- informasi sosial ( siapa yang bertanya tentang apa, siapa yang mendistribusikan informasi dan kepada siapa saja informasi itu diberikan ),
- pencatatan waktu ( pengiriman, penerimaan, pembacaan, melanjutkan pengiriman, pencetakan ),
- *thread* dan *propagasi* ( A ingin mengirim email ke B, tetapi melalui C yang oleh C dilanjutkan ke D dan seterusnya hingga akhirnya sampai ke B),
- alamat URLS di web,
- masa berlakunya surat, dan
- *entry points*, dari *mobile devices* ke robot ke perangkat lunak.

Sedangkan Orgnet.com menggunakan *Data Mining Email* untuk menemukan suatu pemetaan jaringan sosial kemasyarakatan dan pemunculan komunitas baru dalam masyarakat<sup>[5]</sup> pengguna email.

## **BAB III**

### **ANALISA KEBUTUHAN DAN DISAIN PROGRAM**

#### **III.1 Analisa Masalah**

Misalkan kita memiliki banyak jenis usaha dan semua email-email yang dikirimkan oleh para klien dari berbagai jenis usaha tersebut disimpan dalam satu tempat. Sedangkan kita tidak pernah membuat keputusan untuk setiap perusahaan tanpa sebelumnya memperhatikan email-email yang berkaitan dengan jenis usaha tersebut. Oleh sebab itu perlu dilakukan pengelolaan informasi-informasi yang terdapat di dalam email-email tersebut.

Untuk itu diperlukan suatu cara dalam memecahkan masalah tersebut, caranya kita harus melakukan *data mining* terhadap tumpukan email tersebut. Karena *data mining* adalah sebuah *class* dari aplikasi yang mencari pola-pola tersembunyi yang terdapat di dalam kelompok data.

Untuk dapat melakukan *data mining email* ini, kita memerlukan dua buah keterampilan teknis yang satu sama lain tidak saling berhubungan, yaitu pemrograman Perl dan DBA

#### **III.2 Kemungkinan Pemecahan masalah**

Salah satu alternatif pemecahan masalah yang akan dibahas adalah suatu *data mining email* dengan tujuan melakukan *data mining* terhadap email yang dihasilkan oleh *Mozilla mail box*. Dan untuk email yang memuat *attachment* maka kita khususkan dokumen yang di-*attach* adalah dokumen yang buat oleh MS Word. Alat inilah yang akan menguraikan email dan meng-*upload*-nya kedalam suatu database dimana kemudian kita bisa menganalisanya.

*Tool-tool* yang diperlukan untuk alat ini adalah :

- *Mozilla mailbox* yang mempunyai format *mbox*
- Perl, karena mempunyai modul-modul dengan kemampuan yang sempurna didalam menguraikan email yang tersedia di dalam CPAN.
- Database yang digunakan adalah PostgreSQL
- *Console-base* yang digunakan klien adalah *psql*.
- *Command-line utilities* yang sering digunakan adalah *find*, *file* dan *antiword*.

Sedangkan keterampilan khusus yang harus dimiliki oleh pengembang adalah:

- Memprogram dengan Perl, Anda memerlukan pengalaman pemula didalam memprogram dengan menggunakan Perl, yang terpenting anda harus memahami bagaimana caranya melaksanakan *command-line utilities* di dalam sebuah skript Perl. Andapun harus bisa menginstal dan menggunakan modul-modul Perl dari CPAN dan andapun harus sudah terbiasa menggunakan PostgreSQL dari Perl.

- PostgreSQL, anda perlu mengetahui bagaimana caranya menginstal PostgreSQL pada Linux juga penggunaan *precompiled binaries* seperti pada RPMs, *Debian packages* dan *tar balls* atau meng-*compile*-nya dari *source code*. Andapun harus terbiasa dengan standar SQL 92 dan SQL 99 dimana anda dapat menciptakan *tabel*, *view*, batasan (*constraints*), dan fungsi-fungsi *user-defined*.
- Untuk *Command-line*, anda perlu keterampilan bagaimana cara penggunaan *input* atau *output* dari proses satu ke proses selanjutnya

Sedangkan untuk memudahkan pembahasan materi ini asumsi-asumsi yang diberikan adalah :

- Perl 5.8.x telah diinstal dan dapat melakukan akses ke CPAN dengan benar. Juga diperlukan instalasi standar dari Postgres dengan pilihan konfigurasi sekuriti minimal. Sebuah contoh skript Perl akan terkoneksi ke database melalui koneksi socket, sehingga tidak akan ada komentar untuk baris perintah `tcPIP_socket = true` pada `postgreSQL.conf`.
- Koneksi memerlukan suatu kata sandi (*password*) :
 

```
# TYPE DATABASE USER IP-ADDRESS IP-MASK METHOD
local all all trust
host all all 127.0.0.1 255.255.255.255 password
```
- Menggunakan *PostgreSQL superuser account postgres*. Juga file `mbox` akan mereka proses **Sent** dan didalam direktori yang sama sebagaimana yang terdapat pada skript

### III.3 Disain Program

Ada empat tahapan penting yang harus dilakukan di dalam melakukan *data mining email* agar didapat suatu hasil yang diinginkan, yaitu :

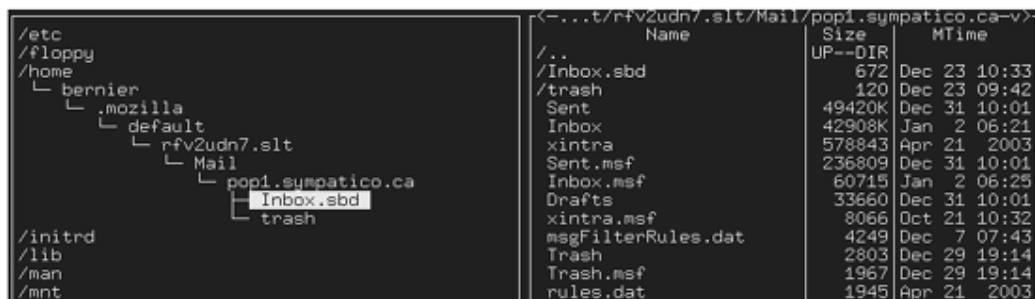
- Bagian pertama adalah melakukan *search* terhadap *mailbox* yang kita miliki, tujuannya adalah untuk mengetahui letak sub direktori (*path*) dimana *mbox* berada.
- Bagian kedua, mempersiapkan database untuk memudahkan di dalam mengekstrak informasi yang terdapat di dalam email. Database yang digunakan pada tahapan ini menggunakan PostgreSQL dan database yang dibuat kita beri nama `email` , sedangkan kolom-kolom penting pada tabel yang dibuat adalah kolom `messageId`.
- Bagian ketiga, adalah pembuatan Program pengurai email (*email parser*) yang terdiri dari beberapa langkah pengerjaan, yaitu :

- Menginstal modul-modul bawaan dari CPAN berkaitan , yaitu `Mail::MboxParser` dan `DBD::Pg`
- Menetapkan suatu koneksi database dengan server PostgreSQL
- Pada saat pembacaan email, dilakukan penguraian bagian-bagian email sehingga sesuai dengan struktur database
- Bagian terakhir adalah melakukan *query* terhadap database yang telah terbentuk. Beberapa trik yang dapat dilakukan adalah :
  - Meringkas semua pesan email yang memuat *main body* dan *attachment*
  - Menampilkan daftar semua pesan *attachment*
  - Melakukan konversi dari *attachment* MS Word menjadi bentuk text ASCII
  - Dan trik yang terakhir adalah melakukan pencarian atau penemuan pola yang diinginkan dengan cara *search* text terhadap kolom-kolom tertentu. Ada dua cara yang dapat digunakan, yaitu :
    - Ekspresi SQL `Like`
    - Ekspresi model PostgreSQL `POSIX` beraturan

## BAB IV IMPLEMENTASI DAN ANALISA

### IV.1 Melakukan *Search* terhadap Mailbox

*Mozilla mail client* mengirimkan *mailbox*-nya ke dalam *home directory* yang dimiliki *user*, biasanya didalam `~/.mozilla`. Gambar 1 memperlihatkan *mailbox* berbeda dari suatu profil yang ada. *Mbox* adalah file-file tanpa *extension*. Isyarat lainnya adalah memperhatikan ukuran file, secara normal *mailbox* adalah file yang ukurannya paling besar



Name	Size	MTime
UP--DIR		
/Inbox.sbd	672	Dec 23 10:33
/trash	120	Dec 23 09:42
Sent	49420K	Dec 31 10:01
Inbox	42908K	Jan 2 06:21
xintra	578843	Apr 21 2003
Sent.msf	236809	Dec 31 10:01
Inbox.msf	60715	Jan 2 06:25
Drafts	33660	Dec 31 10:01
xintra.msf	8066	Oct 21 10:32
msgFilterRules.dat	4249	Dec 7 07:43
Trash	2803	Dec 29 19:14
Trash.msf	1967	Dec 29 19:14
rules.dat	1945	Apr 21 2003

Gambar 1. Tampilan layar dari *Mozilla mailbox*.

Perintah berikut ini menampilkan semua *mailbox* yang terdapat pada salah satu contoh *Mozilla profile account*:

```
$ find ~/.Mozilla/default/rfv2udn7.slt/Mail/ -type f -not \  
-iname "*. *" -not -iname ".*" -print
```

### IV.2 Mempersiapkan Database

Sebelum kita dapat menciptakan skema database dengan benar, perlu diputuskan jenis informasi yang akan di-*extract* dari pesan-pesan tersebut. Ada banyak pilihan, tetapi diasumsikan bahwa yang berikut ini sudah mencukupi :

- Bagian email yang teliti adalah : from, To, CC, and Attachment.
- Email-email datang dari satu sumber.
- Sebuah email dapat dikirimkan ke lebih dari satu orang.
- Sebuah email dapat memiliki lebih dari satu buah CC.
- Sebuah email dapat memiliki lebih dari satu attachment.
- Hanya *attachment* dari dokumen MS Word yang diteliti

#### Database

Untuk mempersiapkan database, Pada console, buatlah database dengan nama *email* yang perintahnya sebagai berikut :

```
$ psql -U postgres -command "create database email" template1
```

Jika hal ini merupakan hal yang pertama kali kita menggunakan PostgreSQL, ubahlah *username* dengan perintah `postgres` dan beri password '123'. Dalam keadaan *default username* tidak memiliki password.

```
$ psql -U postgres --command "alter user postgres with password '123' " email
```

## Tabel

Kolom yang terpenting pada tabel adalah *messageId*, yang mana merupakan suatu kode *alphanumeric* yang unik yang menjadi bagian dari pesan email. Tabel yang telah didefinisikan tersebut tidak akan dapat memiliki populasi data kecuali telah ada *messageId* di dalam *mailId* tabel.

Untuk pesan bagian *body* dan *attachment* biasanya memiliki objek yang cukup lebar. Banyak tabel yang memiliki kolom dengan tipe *oid*, yang mengacu pada data yang nyata yang ditempatkan pada sistem *catalog pg\_largeobject*.

*Index* dibuat secara *default* oleh Postgres untuk masing-masing *primary key* yang didefinisikan di dalam tabel.

Skript pembuatan tabel akan membersihkan email database dan membuat tabel. Untuk melakukannya dapat kita eksekusi dari *console* dengan perintah :

```
$ psql -U postgres -f createTables.sql email
```

## IV.3 Pembuatan Pengurai Email (*Email Parser*)

Langkah pertama yang harus dilakukan adalah menginstal modul-modul dibawah ini dari CPAN beserta fungsi-fungsi yang terkait.

- `Mail::MboxParser`, adalah sebuah interface sederhana yang mempersiapkan akses dengan atribut *read-only* ke Unix-mailboxes
- `DBD::Pg`, adalah *interface* database antara Perl an database PostgreSQL

Skript *mbox parser* pertama-tama mendefinisikan variabel global kemudian menetapkan suatu koneksi database dengan server PostgreSQL:

```
my $dbh = DBI->connect($dsn,$user,$pass);  
die $DBI::errstr unless defined $dbh;  
$dbh->{PrintError} = 0;
```

Kemudian skript akan membaca *mailbox* dan dipecah menjadi bagian-bagian komponen untuk masing-masing pesan:

```
my $mb = Mail::MboxParser->new(\@myarray,  
    decode      => 'ALL',  
    parseropts => $parseropts);
```

Pada proses pengulangan (*loop*) yang dilakukan pada pesan dari *mailbox*, dilakukan pula identifikasi bagian-bagian menarik dari pesan tersebut :

```
my $id      = $msg->id;
my $from    = $msg->from->{email};
my $subject = $msg->header->{subject};
```

Modul *MboxParser* mengidentifikasi pesan bagian *main body* :

```
my $body = $msg->body($msg->find_body);
```

Bagian *main body* dan *attachment* dari pesan tersebut direferensikan ke dalam bagian *main* dan *attachment* yang terdapat pada tabel dengan atribut *oid*. Khusus untuk versi 7.1 PostgreSQL tidak dapat dilakukan *insert* data kedalam baris yang melebihi *default* ukuran halaman yaitu 8192 byte. Untuk memecahkan masalah ini sebagai pengganti dalam menyisipkan informasi dengan objek ukuran besar, maka lakukanlah *upload* objek ukuran besar tersebut dengan menggunakan *psql*:

```
my $myoid=`psql -U postgres --command "\\lo_import '$body_file'\n\n'$mymailbox\' " $db`;\nmy @oidarray = split(/ /,$myoid);
```

Metoda ini memerlukan lebih sedikit *coding* dan lebih sederhana untuk bisa dipahami dibanding dengan menggunakan *libpq's* dan lebih efisien untuk pemanggilan objek yang besar.

Untuk melakukan *insert* pesan email ke dalam tabel *main* :

```
$sql = "INSERT INTO main VALUES( ?, ?, ?, @oidarray[1])";\n$sth = $dbh->prepare($sql) or die $DBI::errstr;\n$sth->execute( $id, $from, $subject ) or die $DBI::errstr;
```

Potongan kode program dibawah ini akan menghasilkan daftar dari para penerima email:

```
#POPULATE TABLE 'mailto'\n\nfor my $msg_TO ($msg->to) {\n    $sql = "INSERT INTO mailto VALUES( ?, ? )";\n    $sth = $dbh->prepare($sql) or die $DBI::errstr;\n    $sth->execute( $id, $msg_TO->{email} ) or die\n    $DBI::errstr;\n}
```

Bagian yang paling penting dari program adalah mengidentifikasi *attachment* MS Word:

```
# DETERMINE ATTACHMENT MIME-TYPES\n\nmy $decodedattachment = `echo "$attachment" | ./decode.pl |\n    tee $attachment_file | file -i -`;\nmy @test0      = split(/ /, $decodedattachment);\nmy $mime_type = $test0[1];\nchomp $mime_type;\n\n# VALIDATING MSWORD DOCS
```

```

if ($mime_type eq "application/MS Word") {

    # CONVERT INTO READABLE TEXT
    `antiword $attachment_file > $MS Word_file;
    mv -f $MS Word_file $attachment_file`;

} else {

    # NOT MSWORD DOCS,
    # USE THE ENCODED MIME VERSION IN THE FILE
    `echo "$attachment" > $attachment_file`;

}

```

Program `decode.pl` melakukan *decode* pesan dari MIME Base64 kembali menjadi *encode* biner yang asli. Ada juga sebuah fungsi dari PostgreSQL yang dapat melakukan hal ini. Program kemudian akan menyalurkan keluaran *encode* ke dalam file *utility* untuk menentukan apakah benar-benar dokumen MS Word. Dan pada saat yang sama, *attachment* yang telah di-*decode* akan disimpan sebagai file temporer di `/tmp/attachment`.

Setiap *attachment* yang diidentifikasi sebagai dokumen MS Word dengan sebuah tipe MIME dari program aplikasi (MS Word) dijalankan melalui *utility antiword* untuk diubah ke dalam sebuah dokumen tipe text ASCII. Kemudian program akan meng-*upload*-nya sebagai objek berukuran besar.

Semua *attachment* disimpan di dalam database sebagai objek berukuran besar dalam bentuk pengkodean MIME yang asli

#### IV.4 Query Pada Database

Jika informasi tersebut sudah berada dalam bentuk database, maka trik selanjutnya adalah mendapatkan informasi yang diinginkan dengan cara-cara yang biasa berlaku pada database.

Pertama, meringkas semua pesan email yang memuat kedua-duanya baik *main body* ataupun *attachment*. (Mengacu pada tabel yang dibuat berdasarkan skript `Create tables`)

```

SELECT m.from_email AS From,
       m.subject     AS Subject,
       m.mailbody    AS Message,
       a.attachment  AS Attachment
FROM   main         m,
       attachment a
WHERE  m.messageid=a.message

```

Contoh hasilnya adalah sebagai berikut :

<b>From</b>	<b>Subject</b>	<b>Message</b>	<b>Attachment</b>
robert.bernier5@sympatico.ca	message 1	60227	60230
robert.bernier5@sympatico.ca	message 2	60233	60236
casestudy@postgresql.org	message 3	60239	60242
casestudy@postgresql.org	message 4	60245	60248

Pesan dan *Attachment* adalah oid atau direferensikan dengan bilangan, untuk objek berukuran besar disimpan didalam katalog `pg_largeobject`.

Untuk menampilkan daftar semua pesan dengan *attachment* dan mengidentifikasi *attachment* yang dari dokument MS Word.

```
SELECT m.messageid AS "Message ID",
       a.attachment AS "Attachment oid",
       CASE
         WHEN a.mime_type='application/MS Word'
         THEN 'true'
         ELSE 'false'
       END
       AS "MS Word attachment"
FROM   main m, attachment a
WHERE  m.messageid=a.messageid;
```

Hasilnya adalah sebagai berikut :

<b>Message ID</b>	<b>Attachment oid</b>	<b>MS Word attachment</b>
3FF79F08.6060208@sympatico.ca	60230	true
3FF79F29.2090101@sympatico.ca	60236	false
3FF7A511.2030104@postgresql.org	60242	false
3FF7A52A.8040203@postgresql.org	60248	true

Untuk mengembalikan menjadi text ASCII dilakukan pengkonversian dari *attachment MS Word*:

```
SELECT encode(lo.data, 'escape') AS "My Document"
FROM   pg_largeobject lo,
       attachmelt      a
WHERE  a.messageid = '3FF7A52A.8040203@postgresql.org'
AND    a.attachment = lo.loid;
```

Fungsi pengkodean mengarahkan ke `My Document` dari bentuk byte menjadi text dengan cara *me-remove* semua sekuen *escape* yang fungsi orisinal `lo_import` sisipkan.

## Pencarian Pola

Untuk mendapatkan hasil yang lebih detail, maka harus dilakukan proses pencarian (*search*) pada kolom-kolom khusus untuk penemuan pola yang diinginkan. Ada dua cara yang dilakukan pada *search text*, yaitu menggunakan ekspresi `SQL Like` atau dengan ekspresi model PostgreSQL POSIX beraturan.

## Penggunaan `LIKE`

Query ini akan melakukan *search* pada ungkapan “fungsi-fungsi *read* dan *write*” yang terdapat pada dokumen *Word* di bagian-bagian tertentu.

```
SELECT m.from_email AS "From",
       a.description AS "File name",
       CASE
         WHEN encode(lo.data, 'escape') LIKE '%These
functions read and write%'
         THEN 'true'
         ELSE 'false'
       END AS "Search results"
FROM pg_largeobject lo,
     attachment a,
     main m
WHERE a.messageid = '3FF7A52A.8040203@postgresql.org'
AND   a.attachment = lo.loid
AND   a.messageid = m.messageid;
```

Contoh hasilnya adalah sebagai berikut:

From	File name	Search results
casestudy@postgresql.org	doc2.doc	True

## Penggunaan Ekspresi Beraturan (*Regular Expressions*)

Query ini akan menghitung banyaknya email dengan *attachment* dan dari alamat mana dikirimkan dimulai dengan kata *ca* dan berakhir dengan *org*.

```
SELECT count(m.from_email)
FROM   main m,
       attachment a
WHERE  m.messageid = a.messageid
AND    m.from_email ~ '^ca'
AND    m.from_email ~ 'org$';
```

## BAB V PENUTUP

### V.1 Kekurangan Dan Saran

Dengan ketertarikan untuk tetap menggunakan skript Perl sesederhana mungkin, didapat asumsi bahwa semua email yang di-*attachment* datang dengan menggunakan pengkodean di dalam *Base64*. Dan juga pada saat penulisan aliran skript, pengkodean terasa sedikit kaku.

Database yang didefinisikan disini hanya sekedar demonstrasi sehingga anda bisa membuat bentuk-bentuk database yang lain sendiri.

Untuk menghemat waktu, menghindari masalah pada *query* SQL dan pendefinisian sebuah file yang nantinya akan diakses oleh klien *psql*, maka letakkanlah file-file tersebut di dalam sebuah versi pengendali

Buatlah perintah membersihkan database pada skript pembuatan tabel setiap kali skript Perl dijalankan.

*Antiword* agak sedikit kaku, oleh karena itu dia tidak dapat menguraikan file yang menurut pertimbangannya mempunyai ukuran terlalu kecil

Tampilan *mbox* pada *Mozilla* yang sampai pada klien dapat menipu, sebab dia tidak pernah benar-benar telah memindahkan email dari *box* sekalipun kita telah menghapusnya, maka sebaiknya telitilah dengan baik email-email tersebut di database yang kita pikir telah dihilangkan sebelumnya.

You can export large objects from the database by using the `lo_export` function in the `psql` client. For example `\lo_export 123 temp.txt` will save the large object with the oid number of 132 to a file named `temp.txt`.

Kita dapat mengirimkan objek berukuran besar dari database dengan menggunakan fungsi `lo_export` di dalam klien `psql`. Sebagai contoh perintah `\lo_export 123 temp.txt` akan menyimpan dengan tipe bilangan oid dari 123 ke file dengan nama `temp.txt`.

## **V.2 Kesimpulan**

Ada dua bagian penting untuk mendapatkan keberhasilan di dalam menambang data yaitu:

- Mekanisme yang mempersiapkan dokumen
- Algoritma yang digunakan dalam pencarian pola

Kita telah melihat bagaimana caranya memasukkan email ke dalam sebuah database dengan jelas, tetapi kita belum menangani kekuatan dan kesulitan dari ekspresi reguler di lingkungan database, walaupun demikian beberapa dari kalangan pengguna Perl telah memiliki gagasan-gagasan menarik dengan berbagai kemungkinan untuk menghadapi masalah tersebut.

## LAMPIRAN-LAMPIRAN

### Lampiran 1: Skript Pembuatan tabel

```
DELETE FROM pg_largeobject;
```

```
DROP TABLE mailid CASCADE;  
DROP TABLE main CASCADE;  
DROP TABLE mailto CASCADE;  
DROP TABLE mailcc CASCADE;  
DROP TABLE attachment CASCADE;
```

```
CREATE TABLE mailid(  
messageid text PRIMARY KEY  
);
```

```
CREATE TABLE main(  
messageid text,  
from_email text,  
subject text,  
mailbody oid,  
CONSTRAINT fk_main FOREIGN KEY (messageid) REFERENCES  
mailid,  
CONSTRAINT pk_main PRIMARY KEY (messageid,from_email)  
);
```

```
CREATE TABLE mailto(  
messageid text,  
mailto text,  
CONSTRAINT fk_mailto FOREIGN KEY (messageid) REFERENCES  
mailid,  
CONSTRAINT pk_mailto PRIMARY KEY (messageid,mailto)  
);
```

```
CREATE TABLE mailcc(  
messageid text,  
mailcc text,  
CONSTRAINT fk_mailcc FOREIGN KEY (messageid) REFERENCES  
mailid,  
CONSTRAINT pk_mailcc PRIMARY KEY (messageid,mailcc)  
);
```

```
CREATE TABLE attachment(  
messageid text,  
description text,  
mime_type text,
```

```

attachment oid,
CONSTRAINT fk_attachment FOREIGN KEY (messageid)
REFERENCES mailid,
CONSTRAINT pk_attachment PRIMARY KEY
(messageid,description)
);

```

## Lampiran 2: Skript Pengurai Email ( *Mail Parsing* )

```

#!/usr/bin/perl

use strict;
use Mail::MboxParser;
use DBI;

#####
# ACCESSING MBOX

my $mymailbox="Sent";

#DUMMY VARIABLE FOR PARSER OPTIONS
my $parseropts="";

open(FD, "$mymailbox");
my @myarray = <FD>;
close (FD);

#CREATE A NEW MboxParser OBJECT_
my $mb = Mail::MboxParser->new(\@myarray,
    decode      => 'ALL',
    parseropts => $parseropts);

my $num_messages = $mb->nmsgs;
my $message_counter = 0;
#
#####

#####
# DB CONNECTION VARIABLES
my $db = "temp";
my $dsn = "dbi:Pg:dbname=$db;host=127.0.0.1;port=5432";
my $user = "postgres";
my $pass = "123";

```

```

my $dbh = DBI->connect($dsn,$user,$pass);
unless (defined $dbh) {die $DBI::errstr;} $dbh->{PrintError} = 0;
#
#####

print "\nstarting upload of mail box \'$mymailbox\'\n";

#####
##
# LOOPING THROUGH MESSAGES IN MBOX
while (my $msg = $mb->next_message) {

    my $id = $msg->id;
    my $from = $msg->from->{email};
    my $subject = $msg->header->{subject};
    my $body = $msg->body($msg->find_body);
    my $body_file = "/tmp/body.txt";

    #POPULATE TABLE 'mailid'
    my $sql = "INSERT INTO mailid VALUES(\'$id\')";
    my $sth = $dbh->prepare($sql) or die $DBI::errstr;
    $sth->execute() or die $DBI::errstr;

    #POPULATE TABLE 'main'
    open(FD,">$body_file");
    print FD $body;
    close (FD);

    my $myoid = `psql -U postgres --command "\\lo_import
\'$body_file\' \'$mymailbox\'" $db`;
    my @oidarray = split(/ /,$myoid);
    $sql = "INSERT INTO main
VALUES(\'$id\',\'$from\',\'$subject\',@oidarray[1])";
    $sth = $dbh->prepare($sql) or die $DBI::errstr;
    $sth->execute() or die $DBI::errstr;

    #POPULATE TABLE 'mailto'
    for my $msg_TO ($msg->to) {
        $sql = "INSERT INTO mailto
VALUES(\'$id\',\'$msg_TO->{email}\')";
        $sth = $dbh->prepare($sql) or die $DBI::errstr;
        $sth->execute() or die $DBI::errstr;
    }
}

```

```

#POPULATE TABLE 'mailcc'
for my $msg_CC ($msg->cc) {
    $sql = "INSERT INTO mailcc
VALUES (\'$id\',\'$msg_CC->{email}\')";
    $sth = $dbh->prepare($sql) or die $DBI::errstr;
    $sth->execute() or die $DBI::errstr;
}

#####
##
#POPULATE TABLE 'attachment'
my $mapping = $msg->get_attachments;

for my $attachment_name(keys %$mapping)
{
    my $attachment = $msg->body($mapping-
>{$attachment_name});
    my $attachment_file = "/tmp/attachment";
    my $msword_file = "/tmp/msword.txt";

    # DETERMINE ATTACHMENT MIME-TYPES
    my $decodedattachment = `echo
"$attachment"|./decode.pl|tee $attachment_file|file -i
-`;
    my @test0 = split(/ /,$decodedattachment);
    my $mime_type = @test0[1];
    chomp $mime_type;

    #VALIDATING MSWORD DOCS
    if ($mime_type eq "application/msword") {
        #CONVERT INTO READABLE TEXT
        `antiword $attachment_file > $msword_file;mv -f
$msword_file $attachment_file`;
    } else {
        #NOT MSWORD DOCS,
        #USE THE ENCODED MIME VERSION IN THE FILE
        print "not msword document\n";
        `echo "$attachment" > $attachment_file`;
    }

    #ALL ATTACHMENTS ARE ARCHIVED IN MIME FORMAT
    #INCLUDING BINARY MSWORD DOCS

```

```

        $myoid = `psql -U postgres --command "\\lo_import
\'$attachment_file\' \'$attachment_name\'" $db`;
        @oidarray = split(/ /,$myoid);
        chomp $mime_type;

        #STILL NEED TO CONSIDER THE OID FOR THE READABLE
VERSION OF MSWORD DOCUMENT
        $sql = "INSERT INTO attachment
VALUES(\'$id\',\'$attachment_name\',\'$mime_type\',@oid
array[1])";
        $sth = $dbh->prepare($sql) or die $DBI::errstr;
        $sth->execute() or die $DBI::errstr;
    }
    # END OF ATTACHMENT PROCESSING

#####

    print "processed
.++$message_counter."/". $num_messages." messages\n";
}
# END OF WHILE LOOP
#####
##

print "DONE\n";

```

### **Lampiran 3 : Skript Mengembalikan pengkodean MIME ( *MIME* decoding)**

```

#!/usr/bin/perl

use MIME::Decoder;
$decoder = new MIME::Decoder 'base64' or die
"unsupported";
$decoder->decode(\*STDIN, \*STDOUT);

```

## Referensi

- [1] Yudho Giri Sucahyo “Data mining – Menggali Informasi Yang terpendam”  
<http://ikc.cbn.net.id/populer/yudho/yudho-datamining.zip>  
tanggal 21 Oktober 2004 pukul 15.47 WIB
- [2] A Data Mining Glossary  
<http://www.hearling.com/index.htm>  
tanggal 21 Oktober 2004 pukul 15.48 WIB.
- [3] Christina Chung “Applying Data Mining to Data Security”  
<http://sirius.cs.ucdavis.edu/teaching/289F/> tanggal 1 September 2004 pukul  
12.19 WIB.
- [4] Phil Wolf “DATA MINING EMAIL”  
<http://www.emergic.org/> tanggal 1 Desember 2004 pukul 06.05 WIB.
- [5] “DataMining Email to Discover Social Networks and Emergent  
Communities”, <http://www.orgnet.com/> tanggal 1 Desember 2004 pukul  
05.55 WIB.
- [6] Victor-Valeriu Patriciu, Liviu Rusu, Iustin Priescu, “Data Mining  
Approaches for Intrusion Detection in Email System Internet-Based” ,  
Military Technical Academy  
Tanggal 24 Nopember 2004 pukul 13.13 WIB
- [7] “What is data mining - A Word Definition From the Webopedia Computer  
Dictionary”  
<http://www.webopedia.com/TERM/D/> tanggal 1 Desember 2004 pukul  
06.02 WIB.
- [8] O’Reilly, OnLamp.Com, “Data Mining Email”, <http://www.onlamp.com/>  
tanggal 1 Desember 2004 pukul 05.59 WIB.