Analisa dan Perancangan Sistem Pencarian Menggunakan Metaphone Algorithm pada Sistem Informasi Perpustakaan

Dwi Agus Diartono
Fakultas Teknologi Informasi, Universitas Stikubank Semarang
email : dwiagus@unisbank.ac.id

Abstrak : Proses pencarian dan keakuratan data secara cepat dan mudah akan menjadi masalah dalam menentukan suatu keputusan. Sistem pencarian data secara manual sering menjadi penyebab utama dari kelambatan proses pencarian maupun ketidakakuratan data yang didapat. Hal ini disebabkan adanya "human error.." dalam melakukan pencatatan dan pencarian data, terutama ketika data yang tersedia berjumlah sangat besar. Masalahnya, pengguna sistem seringkali melakukan kesalahan dalam menuliskan keyword, terutama pada proses pencarian literatur berbahasa asing karena penulisan dan pengucapan sebuah kata seringkali berbeda. Kesalahan yang sering terjadi adalah pengguna sistem menuliskan kata yang dicari sesuai dengan bagaimana kata itu diucapkan dalam bahasa Indonesia. Metaphone Algorithm adalah algoritma yang dapat mengatasi perbedaan pengucapan dan penulisan sebuah kata. Algoritma ini diungkapkan pertama kali oleh Lawrence Philips pada bulan Desember, 1990. Algoritma ini dapat digunakan sebagai salah satu alternatif dalam menyelesaikan masalah kesalahan ketik pada proses pencarian literatur di perpustakaan.

Kata kunci : metaphone algorithm

PENDAHULUAN

Kecepatan proses pencarian dan keakuratan data adalah masalah berarti dalam menentukan suatu keputusan. Sistem pencarian data yang bersifat manual sering menjadi penyebab utama dari kelambatan proses pencarian maupun ketidakakuratan data yang didapat. Hal ini disebabkan oleh tingkat kemungkinan kesalahan manusia yang tinggi dalam melakukan pencatatan dan pencarian, terutama ketika data yang tersedia berjumlah sangat besar.

Di sebuah perpustakaan yang memiliki ribuan literatur, sistem komputerisasi sangat dibutuhkan dalam proses pencarian literatur yang diinginkan. Ketika sistem ini digunakan, literatur yang diinginkan akan didapat dengan waktu relatif cepat.

Online Public Access Catalogue (OPAC) adalah istilah yang digunakan dalam dunia perpustakaan untuk proses pencarian (literatur) menggunakan sistem informasi berbasiskan komputer. Penggunaan OPAC merupakan salah satu cara yang efektif dalam mendapatkan literatur yang diinginkan. Pada umumnya, cara menggunakannya cukup mudah, yaitu dengan mengetikkan kata kunci yang diminta *(keyword)*, misalnya judul literatur yang dicari, kemudian menekan tombol tertentu (tombol 'Search') untuk memulai proses pencarian. Hasil pencarian akan muncul sesaat setelah proses pencarian selesai yaitu berupa daftar literatur yang memiliki judul mirip dengan kata kunci yang diketikkan.

Masalahnya, pengguna sistem seringkali melakukan kesalahan dalam menuliskan keyword, terutama pada proses pencarian literatur berbahasa asing karena penulisan dan pengucapan sebuah kata seringkali berbeda. Kesalahan yang sering terjadi adalah pengguna sistem menuliskan kata yang dicari sesuai dengan bagaimana kata itu diucapkan dalam bahasa Indonesia.

Metaphone Algorithm adalah algoritma yang dapat mengatasi perbedaan pengucapan dan penulisan sebuah kata. Algoritma ini diungkapkan pertama kali oleh Lawrence Philips pada bulan Desember, 1990. Algoritma ini dapat digunakan sebagai salah satu alternatif dalam menyelesaikan masalah kesalahan ketik pada proses pencarian literatur di perpustakaan.

TINJAUAN PUSTAKA

Metaphone

Menurut Cesar Nicolas Pena Nunez, *HowToDoThings.com. Metaphone Algorithm* merupakan suatu algoritma yang mentransformasikan suatu kata ke kode berdasarkan bunyinya. Algoritma ini ditemukan oleh Lawrence Philips dan pertama kali dijelaskan di *Computer Language Magazine* pada bulan Desember 1990.

Sebenarnya, ide *Metaphone* berasal dari *Soundex*. *Soundex* adalah algoritma pertama yang cukup sederhana untuk mengatasi perbedaan pengucapan dan penulisan dalam proses pencarian. Pada prinsipnya, baik *Metaphone* maupun *Soundex* menghasilkan sebuah kode yang sama untuk semua kata yang pengucapannya mirip.

Metaphone lahir ketika Lawrence Philips menemukan bahwa *Soundex* tidak dapat sepenuhnya mengatasi perbedaan pengucapan dan penulisan dalam proses pencarian. Bahkan, ada pengguna sistem yang mengeluh bahwa *Soundex* tidak menghasilkan kata yang bunyinya sama. Contohnya pencarian kata "Stephen" tidak menghasilkan "Steven" dalam daftar hasil pencarian. Dan, sebaliknya, menghasilkan kata yang bunyinya sama sekali berbeda dengan kata yang diinputkan: contohnya pencarian kata "Cajun" menghasilkan "Cigna" dalam daftar hasil pencarian.

Perbedaan SoundMax dan Metaphone

Perbedaan pokok antara *Soundex* dan *Metaphone* adalah pada proses transformasi kata ke kode. Dalam mentransformasikan kata, *Soundex* menggunakan huruf dan angka untuk membentuk sebuah kode yang maksimal terdiri dari 4 karakter. Contoh kode hasil transformasi *Soundex* adalah "P202", "H345". Sedangkan pada algoritma *Metaphone*, proses transformasi dilakukan dengan menerapkan aturan yang biasa digunakan dalam pengucapan bahasa Inggris sehingga semua karakter dari kode hasil transformasi berupa huruf dan lebih mendekati kebenaran. Contoh kode hasil transformasi Metaphone adalah "STFN", "XBRT". Perbedaan yang lain adalah *Soundex* menganalisa kata dengan pola huruf per huruf. Contohnya, akhiran "-TIA" ditransformasi menjadi "300", karena T ditransformasi menjadi angka 3 dan semua huruf vokal ditransformasi menjadi angka nol "0". Sedangkan *Metaphone* selain menganalisa konsonan tunggal juga menganalisa kumpulan huruf yang disebut diftong. Contohnya, akhiran "-TIA" ditransformasi menjadi "X", karena jika diucapkan bunyinya seperti "sh".

Pada umumnya, kode yang dihasilkan dari transformasi *Metaphone* hanya terdiri dari maksimal 4 huruf. Contohnya, STFN adalah kode yang dihasilkan untuk kata "stephan", "stephen", "steven", dan XBRT adalah kode yang dihasilkan untuk kata "shubert".

Implementasi Metaphone dapat dimodifikasi sesuai dengan cara pengucapan dari bahasa yang diinginkan, misalnya bahasa Indonesia. Modifikasi tersebut dilakukan pada proses transformasi kata ke kode.

Aturan Metaphone

Ada beberapa aturan umum yang harus diperhatikan dalam implementasi *Metaphone.* Pertama, *Metaphone* mengurangi abjad menjadi 16 bunyi konsonan, yaitu:

B X S K J T F H L M N P R 0 W Y.

'0' di atas bukan huruf O, melainkan zero. Cara pengurangan tersebut akan dijelaskan pada subbab berikut.

Kedua, huruf pertama, baik berupa konsonan maupun vokal, tetap dipertahankan. Jika tidak muncul sebagai huruf pertama, vokal dihilangkan.

ANALISA DAN PERANCANGAN

Perancangan *Input*

Cara kerja sistem pencarian menggunakan *Metaphone Algorithm* adalah dengan membandingkan kode transformasi kata yang dicari dengan kode transformasi kata yang ada di dalam *database*. Kode transformasi kata yang dicari diperoleh dari hasil transformasi kata yang dimasukkan oleh pengguna sistem saat proses pencarian dilakukan. Sedangkan kode transformasi kata yang ada dalam *database* diperoleh dari hasil transformasi kata yang dimasukkan oleh pengguna sistem saat *setup* kata, atau *input* kata baru.

Input Kata Baru

Pada proses pencarian menggunakan *Metaphone Algorithm* berdasarkan kategori judul, sistem akan membandingkan kata yang dicari dengan kata yang terdapat dalam *database.* Banyaknya alternatif kata yang ditemukan oleh sistem tergantung pada banyaknya kata yang terdapat dalam tabel **KodeKata** di *database.*

Jenis masukan yang dibutuhkan adalah karakter berupa abjad dengan panjang maksimum adalah 15 huruf, sesuai dengan panjang *field* **Kata** pada tabel **KodeKata**. Akan tetapi, pengguna sistem dapat memberi masukan berupa karakter apa saja, karena sebelum ditransformasi menjadi kode dan disimpan, masukan tersebut akan dicek kebenarannya oleh sistem. Proses transformasi dan penyimpanan akan terjadi ketika hasil pengecekan yang dilakukan oleh sistem berupa minimal satu buah abjad.

Untuk menghindari perbedaan penulisan huruf besar dan huruf kecil, masukan berupa abjad akan diatur sebagai huruf besar. Masukan berupa spasi diantara dua abjad akan dianggap sebagai pemisah. Setiap kata yang dipisah akan dicek panjangnya hingga maksimal 15 huruf. Jika lebih dari 15 huruf, kata tersebut akan dipotong secara otomatis oleh sistem. Karakter tanda kutip (‘) diperlakukan khusus, karena penggunaannya dalam bahasa Inggris dapat berarti singkatan dari kata yang lain.

Input Keyword dalam Pencarian Literatur

Ada tiga jenis masukan yang akan berfungsi sebagai *keyword* pada proses pencarian. Jenis masukan yang pertama adalah kategori dasar pencarian, yaitu kategori judul dan kategori nama pengarang. Pengguna sistem dapat menggunakan salah satu dari kategori tersebut atau keduanya sekaligus. *Default* untuk masukan ini adalah kategori judul.

Jenis masukan yang kedua adalah logika **‘And’** atau **‘Or’**, dimana pengguna sistem harus memilih salah satu diantaranya. Masukan ini dibutuhkan ketika kedua kategori di atas digunakan secara bersamaan. *Default* untuk masukan ini adalah logika **‘And’**.

Setelah kategori pencarian dipilih, sistem membutuhkan masukan berupa karakter yang harus dicari ke *database.* Masukan dapat berupa karakter apapun yang dianggap sebagai kata kunci yang paling baik untuk menemukan literatur yang dicari. Akan tetapi, *keyword* yang paling baik adalah sebuah kata yang terdiri dari abjad. Hal ini disebabkan karena proses pencarian secara garis besar terdiri dari dua tahap yang secara otomatis akan dilakukan oleh sistem, yaitu proses pencarian tanpa *metaphone algorithm* dan proses pencarian menggunakan *metaphone algorithm.*

Pada proses pencarian tanpa *metaphone algorithm,* sistem akan mencari literatur dalam *database* yang memuat karakter sama persis dengan masukan yang diberikan pada bagian judul dan/atau nama pengarangnya. Sedangkan pada proses pencarian dengan *metaphone algorithm, keyword* yang dimasukkan oleh pengguna sistem akan dianggap sebagai sebuah kata, meskipun kenyataannya *keyword* tersebut terdiri lebih dari satu kata.

Relasi Basis Data

Relasi yang terjadi dalam basis data di sistem pencarian yang dibuat hanya memuat tabel yang digunakan dalam sistem pencarian.

Tabel yang digunakan dalam sistem pencarian adalah tabel **Books**, **Authors**, **BooksAuthors**, **KodeKata, KodeNama**, **BookKode**, dan **BooksNama.** Relasi *one to many* terjadi antara tabel **Books** dan **BooksAuthors**, tabel **Authors** dan **BooksAuthors**, tabel **Books** dan **BookKata**,

tabel **Books** dan **BooksNama**, tabel **KodeKata** dan **BookKata**, tabel **KodeNama** dan **BooksNama**.

Algoritma

Algoritma Pemasukan

Proses pencarian diawali dengan memilih kategori pencarian, yaitu pencarian berdasarkan judul, nama pengarang, atau judul dan/atau nama pengarang. Kemudian pengguna sistem harus mengetikkan kata yang akan dicari dalam *object* **text box** yang akan digunakan sebagai *keyword* oleh sistem.

Algoritma Pencarian

Penekanan tombol ***Search*** akan memulai proses pencarian. Pencarian di *field* **Title** di tabel **Books** akan terjadi jika pengguna sistem memilih proses pencarian berdasarkan judul, sedangkan pencarian di *field* **Authorname** di tabel **Authors** akan terjadi jika pengguna sistem memilih proses pencarian berdasarkan nama pengarang, dan pencarian di kedua *field* akan dilakukan jika pengguna sistem memilih kedua kategori dasar pencarian secara bersamaan.

Jika sistem berhasil menemukan literatur yang sesuai dengan *keyword,* maka akan ditampilkan judul literatur di *grid* **'List of Book'**. Akan tetapi, jika sistem tidak berhasil menemukan literatur yang sesuai dengan *keyword,* maka sistem akan mentransformasi *keyword* ke kode dan menampilkan daftar kata dan/atau daftar nama, yang bunyinya mirip dengan *keyword,* di *grid* **'List of Word/Name'**. Bersamaan dengan itu, secara otomatis, sistem juga akan mencari dan menampilkan literatur yang mengandung kata yang terdapat dalam *grid* **'List of Word/Name'**. Literatur yang ditampilkan pada *grid* **'List of Book'** dapat di-*filter* lagi dengan cara *double click* pada kata yang diinginkan, yang tertera dalam *grid* **'List of Word/Name'**.

Jumlah dan ragam alternatif kata/nama yang muncul di *grid* **'List of Word/Name'** sangat tergantung pada jumlah dan ragam kata/nama di tabel yang memuat kata/nama dan kode transformasinya, yaitu tabel **KodeKata** dan tabel **KodeNama**. Jumlah dan ragam kata/nama dapat diperkaya dengan menggunakan *form* **'Input New Word'** dan *form* **'Input New Name'.**

Berikut ini adalah gambaran kerja sistem pencarian yang telah dijelaskan di atas:

Input

1. *Keyword* dari literatur yang dicari berdasarkan judul atau nama pengarang.
2. Memilih salah satu dari kata hasil transformasi di *grid* **'List of Word/Name'**.

Process

1) Cari kata yang sesuai dengan *keyword* di *field* **Title** di tabel **Books** dan/atau *field* **Authorname** di tabel **Aauthors** dalam *database*.
2) Jika *found* maka *output* (tampilan daftar literatur di *grid* **'List of Book'**)
3) Jika *not found* maka:
    1. Transformasikan *keyword* ke kode menggunakan aturan *Metaphone*.
    2. Cari kode hasil transformasi itu di tabel **KodeKata** dan/atau **KodeNama**
    3. Tampilkan alternatif kata atau nama di *grid* **'List of Word/Name'** dan cari literatur yang mengandung alternatif kata yang ditampilkan di *grid* **'List of Word/Name'**.
    4. Cari kata yang dipilih pengguna sistem di *field* **Title** di tabel **Books** dan/atau *field* **Authorname** di tabel **Aauthors** dalam *database*.

Output

- Daftar literatur yang memuat kata, baik persis sama maupun mirip dengan *keyword* berdasarkan *Title* dan/atau *Author Name* di *grid* **'List of Book'**.

Implementasi Metaphone Algorithm

Berikut ini adalah implementasi dari *Metaphone Algorithm* yang akan digunakan dalam sistem pencarian:

Input

inStr : *string* yang diinputkan *(keyword)*

Process

1. Pre-Proses.

Langkah	Kondisi (*if*)	Aksi (*then*)
Pengecekan awal	• [illegible]	• [illegible]
Buang karakter yang bukan abjad	• [illegible]	• [illegible]
Pengecekan cepat	• [illegible]	• [illegible]
Pengecekan cepat	• [illegible]	• [illegible]
Uppercase		• [illegible]

Tabel 1. Pre-proses

2. Pengecualian huruf awal.

Langkah	Kondisi (*if*)	Aksi (*then*)
Pengecualian 1	[illegible] KN, GN, PN, AE, WR	• [illegible]
Pengecualian 2	• [illegible]	• [illegible] S
Pengecualian 3	• [illegible] WH	• [illegible] W

Tabel 2. Pengecualian huruf awal.

Langkah	Kondisi (*if*)	Aksi (*then*)
Huruf ganda	• [illegible]	• [illegible]
Vokal	• [illegible]	• [illegible]
B	• [illegible]	• B
C	• [illegible]	• X
	• [illegible]	• S
	• [illegible]	• K
D	• [illegible]	• J
	• [illegible]	• T
G	• [illegible]	• [illegible]
	• [illegible]	• J
	• [illegible]	• K
H	• [illegible]	• [illegible]
K	• [illegible]	• H
	• [illegible]	• K
P	• [illegible]	• F
Q	• [illegible]	• K
S	• [illegible]	• X
	• [illegible]	• S
	• [illegible]	• X
T	• [illegible]	• 0
	• [illegible]	• [illegible]
V	• [illegible]	• F
W	• [illegible]	• [illegible]
X	• [illegible]	• W
Y	• [illegible]	• KS
Z	• [illegible]	• Y
Default	• [illegible]	• S

3. Transformasi *keyword* menggunakan *looping*.

Keterangan:

– Vokal : A, I, U, E, O

– Konsonan : *not* Vokal

– DepanV : E, I, Y

– Khusus : C, G, P, S, T

Output

Kode hasil transformasi dalam bentuk huruf kapital.

IMPLEMENTASI

a. Proses

Proses penting yang terjadi adalah proses penyimpanan kata baru ke dalam *database.*

Ketika 'Save', ada dua hal penting yang dilakukan oleh sistem. Pertama adalah proses pengecekan *input.* Kedua adalah proses penyimpanan. Proses kedua terjadi jika proses pertama sukses, yaitu jika kata yang dimasukkan oleh pengguna sistem belum ada dalam *database.* Berikut ini adalah baris program yang melukiskan hal tersebut.

```
gckata=alltrim(thisform.txtckata.va
lue)
set exact on
locate for alltrim(ckata)=gckata
if found()
      set exact off
      messagebox ("This word,
"+gckata+", is on database!",;
                          48+0,"Infor
                    mation")
else
      cekkalimat()

endif
```

Proses penyimpanan diwakili oleh prosedur cekkalimat(). Hal pertama yang dilakukan oleh prosedur ini adalah memanggil prosedur pisahkata(). Kemudian prosedur ini akan melakukan transformasi terhadap kata yang dihasilkan oleh prosedur pisahkata(), dengan cara memanggil prosedur metaphone(). Pemanggilan prosedur metaphone() akan menghasilkan sebuah kode yang terdiri dari maksimal empat buah konsonan. Hal ketiga yang dilakukan oleh prosedur cekkalimat() adalah menyimpan kata tersebut bersama kode hasil transformasinya ke dalam tabel tKodeKata, jika kata tersebut belum terdapat dalam tabel. Berikut ini adalah baris program pada prosedur cekkalimat().

```
gncounter=1
gnlenkata=len(gckata)
glstatfinish=.F.

do while glstatfinish=.F.
      glstatkata=.T.
      * pisahkan kalimat menjadi
per kata
      pisahkata()
      * masukkan dalam tabel
      TKodeKata/TKodeNama
      set exact on
      if gckatabaru # ''
            locate for
alltrim(ckata)=gckatabaru
            set exact off
            if !found()

      gckeyword=gckatabaru
                  metaphone()

                  append blank

                  replace ckata
with gckatabaru
                  replace ckode
with gckode
            endif
      endif
enddo
```

Prosedur pisahkata() berfungsi untuk memotong masukan yang berupa gabungan kata menjadi kata per kata berdasarkan spasi atau tanda kutip (‘). Sebagai catatan, tanda kutip pada kata KID’S bukan merupakan pemisah kata, sehingga kata tersebut tetap merupakan satu kesatuan. Di bawah ini adalah baris program pada prosedur pisahkata().

```
lckatabaru=""
N=gncounter

do while glstatkata=.T.
      lchuruf=substr(gckata,N,1)

      * pengecekan huruf/abjad
      if isalpha(lchuruf)=.T.

      lckatabaru=lckatabaru+lchuruf
      endif

      if N=gnlenkata
            glstatfinish=.T.
            glstatkata=.F.
      else
            glstatfinish=.F.


      lcnext1=substr(gckata,N+1,1)

      lcnext2=substr(gckata,N+2,1)

            * ex: what 'metaphone'
is.
            * ex: something about
'mary'.
            * ex: chicken soup for
      the kid's soul
            if asc(lcnext1)=39
&& 39=tanda petik satu (')
                  if
asc(lcnext2)=32    && 32=spasi

      glstatkata=.F.

      gncounter=N+1
                  else

      glstatkata=.T.
                  endif
            else
                  if
isalpha(lcnext1)=.F.    and N+1 <=
gnlenkata

      glstatkata=.F.

      gncounter=N+1
else
      glstatkata=.T.
                  endif
            endif
      endif
      N=N+1
enddo
gckatabaru=alltrim(lckatabaru)
```

Input Keyword untuk Proses Pencarian

a. Proses

Proses penting yang terjadi adalah proses pencarian yang terdapat ada pada *‘database’*.

Proses pencarian secara garis besar terdiri dari dua tahap yang secara otomatis akan dilakukan oleh sistem, yaitu proses pencarian tanpa *metaphone algorithm* dan proses pencarian menggunakan *metaphone algorithm.* Pada proses pencarian tanpa *metaphone algorithm* sistem akan mencari *keyword* yang dimasukkan dalam text box ke *field* yang sesuai berdasarkan kategori yang dipilih (*field* yang memuat *title*

dan/atau *field* yang memuat *author*). Jika *keyword* yang dicari ditemukan, maka sistem akan menampilkan literatur yang memuat *keyword* tersebut dalam *grid* 'List of Book', dan proses pencarian selesai. Akan tetapi, jika *keyword* yang dicari tidak ditemukan, sistem akan melakukan proses pencarian menggunakan *metaphone algorithm* secara otomatis.

Dalam *proses* pencarian menggunakan *metaphone algorithm*, hal pertama yang dilakukan adalah memanggil prosedur metaphone(). Pemanggilan prosedur metaphone() akan menghasilkan kode transformasi yang terdiri dari maksimal empat buah konsonan. Kemudian, sistem akan mencari kode yang sama dengan kode hasil transformasi itu dalam tabel tKodeKata dan/atau tabel tKodeNama dan menampilkannya dalam *grid* 'List of Word/Name'. Setelah itu, secara otomatis, sistem juga akan mencari dan menampilkan literatur, dalam *grid* 'List of Book', yang memuat alternatif kata atau alternatif nama yang muncul dalam *grid* 'List of Word/Name'.

Ada dua kategori dasar pencarian yang dapat dipilih oleh pengguna sistem, yaitu kategori 'Title' dan kategori 'Author Name'. Pengguna sistem dapat menggunakan salah satu dari kategori tersebut atau keduanya sekaligus. Kedua kategori itu dapat digabung menggunakan logika 'And' atau 'Or'.

Pada penggabungan kategori menggunakan 'And', proses pencarian yang dilakukan hanya proses pencarian tanpa *Metaphone Algorithm.* Sistem akan mencari literatur yang memuat *keyword* 'Title' pada bagian judulnya dan juga memuat *keyword* 'Author Name' pada bagian nama pengarangnya. Jika literatur ditemukan, maka sistem akan menampilkan literatur yang memuat *keyword-keyword* tersebut dalam *grid* 'List of Book', dan proses pencarian selesai.

Pada penggabungan kategori menggunakan 'Or', sistem akan mencari literatur yang memuat *keyword* 'Title' pada bagian judul atau memuat *keyword* 'Author Name' pada bagian nama pengarang. Jika literatur tidak ditemukan, prosedur metaphone() akan dipanggil. Dengan kata lain, proses ini menggunakan kedua tahap proses pencarian, yaitu proses pencarian tanpa *metaphone algorithm* dan proses pencarian menggunakan *metaphone algorithm.* Kedua tahap proses pencarian juga diberlakukan pada proses pencarian menggunakan salah satu dari kategori dasar pencarian.

Berikut ini adalah baris program untuk 'Search', dimana proses pencarian yang dilakukan adalah proses pencarian berdasarkan kategori judul.

```
gckeyword=alltrim(upper(thisform.te
xt1.value))

* cari keyword di field cTitle
set safety off
select * from crsAll ;
where cTitle like '%'+gckeyword+'%'
;
into cursor crsBooks
* jika judul yg dicari tidak
ditemukan,
* cari kata yang bunyinya sama di
table kodekata
if _Tally = 0
      metaphone()
      * cari kode yang sama dengan
hasil transformasi
      select Tkodekata.ckata,
Tkodekata.lstat_kata ;
      from perpustakaan!tkodekata;
      where Tkodekata.ckode like
alltrim(gckode) ;
      order by Tkodekata.ckata ;
      into cursor crsKata
      * tampilan di grid kata
      thisform.grdTKodeKata.Records
ource = 'crsKata'
      go top
      thisform.grdTKodeKata.refresh
()
      * tampilkan judul yg bunyinya
mirip dgn keyword
      select distinct * from crsAll
;
      inner join
perpustakaan!tbookkata ;
      inner join
perpustakaan!tkodekata ;
      on Tbookkata.ckata =
Tkodekata.ckata ;
      on crsAll.cbookid =
Tbookkata.cbookid;
      where Tkodekata.ckode like
alltrim(gckode) ;
      order by crsAll.ctitle ;
      into cursor crsBooks
endif
```

```
set safety on
```

Berikut ini adalah baris program untuk 'Search', dimana proses pencarian yang dilakukan adalah proses pencarian berdasarkan dua kategori sekaligus.

```
* cari title di field cTitle,
author di field cAuthors

if
thisform.optiongroup1.option1.value
=1
      * operasi AND
      select * from crsAll ;
      where cTitle like
'%'+gccaritite+'%' ;
      and cauthors like
'%'+gccariauthor+'%' ;
      into cursor crsBooks
else
      * operasi OR
      select * from crsAll ;
      where cTitle like
'%'+gccaritite+'%' ;
      or cauthors like
'%'+gccariauthor+'%' ;
      into cursor crsBooks

      * cari kata yang bunyinya
sama di table kodekata
      if _Tally = 0
            * tampilan di grid kata
            thisform.gabungkode()


      thisform.grdTKodeKata.Records
ource = 'crsKata'
            go top

      thisform.grdTKodeKata.refresh
()

            * tampilkan buku yg
bunyi judul atau authornya
            * mirip dgn keyword
            hasilfilterkode()


            select * from crsAll ;
            inner join crsKode ;
            on crsAll.cbookid =
crsKode.cbookid ;
group by crsAll.cbookid ;
            order by crsAll.ctitle
;
            into cursor crsBooks
      endif

endif
```

KESIMPULAN

Dari sistem pencarian menggunakan *Metaphone Algorithm* yang telah dibuat, maka dapat diambil kesimpulan sebagai berikut:

1. Implementasi *Metaphone Algorithm* dalam *Online Public Acsess Catalogue* pada sistem informasi perpustakaan merupakan alternatif yang baik dalam menyelesaikan masalah kesalahan ketik pada proses pencarian literatur di perpustakaan karena dapat membantu pengguna sistem ketika mencari literatur berbahasa Inggris dimana terdapat perbedaan dalam pengucapan dan penulisan kata-katanya.
2. Cara kerja sistem pencarian menggunakan *Metaphone Algorithm* adalah dengan membandingkan kode transformasi kata yang dicari dengan kode transformasi kata yang ada di dalam *database*. Oleh karena itu, tingkat kebenaran persamaan bunyi dari kata yang dicari terletak pada persamaan kode transformasi kata yang dicari dengan kode transformasi kata yang ditemukan dalam *database.* Hal ini mengakibatkan pencarian tidak berjalan sempurna karena kata CARPENTER dan GROUPING yang mempunyai kode transformasi sama dianggap oleh sistem sebagai kata yang mempunyai bunyi sama ketika diucapkan.
3. Bahasa yang digunakan sebagai dasar evaluasi perbedaan pengucapan dan penulisan kata yang dicari pada sistem ini adalah bahasa Inggris.
4. *Keyword* yang dievaluasi menggunakan *Metaphone Algorithm* oleh sistem adalah *keyword* yang terdiri dari satu kata. *Keyword* yang dimasukkan lebih dari satu kata akan digabungkan dan dibaca oleh sistem sebagai satu kata.
5. Secara garis besar, proses pencarian dalam sistem ini terdiri dari dua tahap yang secara otomatis akan dilakukan oleh sistem, yaitu proses pencarian tanpa *metaphone algorithm* dan proses pencarian menggunakan *metaphone algorithm.*

6. Penghilangan huruf vokal untuk beberapa kasus justru tidak membantu.

DAFTAR PUSTAKA

1. Budi Sutedjo Dharma Oetomo, S.Kom, M.M (…), *Perencanaan dan Pembangunan Sistem Informasi,* Andi Offset, Yogyakarta
2. Edhy Sutanta (1996), *Sistem Basis Data "Konsep dan Peranannya dalam Sistem Informasi Manajemen",* Andi Offset, Yogyakarta
3. Fathansyah, Ir. (2001), *Basis Data,* CV. Informatika, Bandung
4. Harianto Kristanto, Ir. (1996), *Konsep dan Perancangan Database,* Andi Offset, Yogyakarta
5. Poerwadarminta,W.J.S. (1984), *Kamus Umum Bahasa Indonesia,* Balai Pustaka, Jakarta
6. Yourdon, E. (1989), *Modern Structured Analysis*, Prentice Hall International, Inc. New Jersey
7. http://www.entisoft.com/estools/StringWords_MetaPhone.HTML
8. Entisoft (1996-1999), Meta Phone Function String Words Class
9. CMP Media LLC (2002), The Double Metaphone Search Algorithm, C/C++ Users Journal
10. http://aspell.sourceforge.net/metaphone/
11. Kevin Atkinson, GNU Aspell (2002), Lawrence Philips' Metaphone Algorithm
12. http://foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?metaphone
13. Denis Howe, FOLDOC/Free On-Line Dictionary of Computing (1993), Metaphone
14. Cesar Nicolas Pena Nunez, HowToDoThings.com (2000-2003), The Metaphone Algorithm
15. Penton Media, Inc. (2003), Double Metaphone Sounds Great, Convert the C++ Double Metaphone algorithm to T-SQL