# FASILITAS PENCARIAN TEKS CEPAT MENGGUNAKAN MICROSOFT INDEXING SERVICE PADA VISUAL FOXPRO

Iwan Tanto

Sekolah Tinggi Manajemen Informatika dan Komputer LIKMI
Jl. Ir. Juanda 96 Bandung 40132

E-mail: iwantanto@gmail.com

**ABSTRAK**

Fasilitas pencarian teks merupakan fasilitas yang umum dijumpai pada aplikasi *web*, karena tanpa fasilitas ini akan sulit sekali mencari sebuah informasi dari segudang informasi yang ada di *web*. Fasilitas pencarian teks dapat dibangun sendiri menggunakan berbagai bahasa pemrograman yang tersedia. Namun sistem operasi Windows telah menyediakan fasilitas pencarian yang bagus yaitu *Microsoft Indexing Service*, yang jarang dimanfaatkan.

**Kata-kata kunci**: pencarian, indeks, katalog, *query*

## 1. PENDAHULUAN

Untuk sebuah aplikasi *web* yang besar, fasilitas pencarian adalah sebuah fitur yang sangat membantu. Fasilitas ini dapat dibuat menggunakan berbagai bahasa pemrograman yang ada, namun membuat fasilitas pencarian yang handal tidaklah mudah. Fasilitas pencarian ini cukup mudah dibuat dengan menggunakan *Microsoft Indexing Service*.

Apabila mencoba membuat indeks pada *field* bertipe memo, Visual FoxPro akan menampilkan pesan kesalahan "*Operation is invalid for a Memo, General, or Picture field*." Kesalahan ini terjadi karena indeks pada Visual FoxPro memiliki lebar yang tetap, sedangkan *field* memo memiliki lebar yang bervariasi. Visual FoxPro dapat dipaksakan untuk melakukan indeks pada *field* memo dengan cara membatasi lebar memo:

INDEX ON PADR(CustNotes, 240) TAG CustNotes

Cara ini berhasil, namun hanya 240 karakter pertama yang akan diindeks oleh Visual FoxPro. Tetapi untuk angka yang lebih besar dari 240, cara tersebut tidak dapat digunakan lagi karena Visual FoxPro membatasi lebar ekspresi yang bisa dievaluasi sebagai indeks yaitu 240 karakter.

Walaupun cara ini umumnya berhasil, namun masih terdapat sedikit masalah. *Query* pada *field* memo dibatasi hanya untuk frasa yang tepat sama atau potongan frasa di awal memo. Sebagai contoh, *query* berikut hanya menemukan *record* di mana *field CustNotes* diawali dengan frasa "*My fox doesn't have fleas*":

```
SELECT * FROM Customer ;
WHERE CustNotes = ;
"My fox doesn't have fleas"
```

Jika di dalam basis data terdapat *field CustNotes* yang berisi teks "*My dog has fleas, but my fox doesn't have fleas*", maka *record* ini tidak termasuk dalam hasil pencarian karena frasa di dalam klausa *WHERE* tidak cocok dengan isi *field* memo.

Walaupun digunakan operator $ untuk melakukan pencarian menyeluruh di dalam *field* memo, cara ini tidak optimal dan terlalu lambat untuk sebagian besar aplikasi. Sebagai contoh, perintah berikut bekerja dengan baik namun tidak optimal:

```
SELECT * ;
FROM Customer ;
WHERE "My fox does not" + ;
  have fleas"$CustNotes
```

Operator $ bersifat *case-sensitive*, sehingga harus ditulis ulang *query* berikut:

```
SELECT * ;
FROM Customer ;
WHERE UPPER('MY FOX Does Not' + ;
  Have fleas')$UPPPER(CustNotes)
```

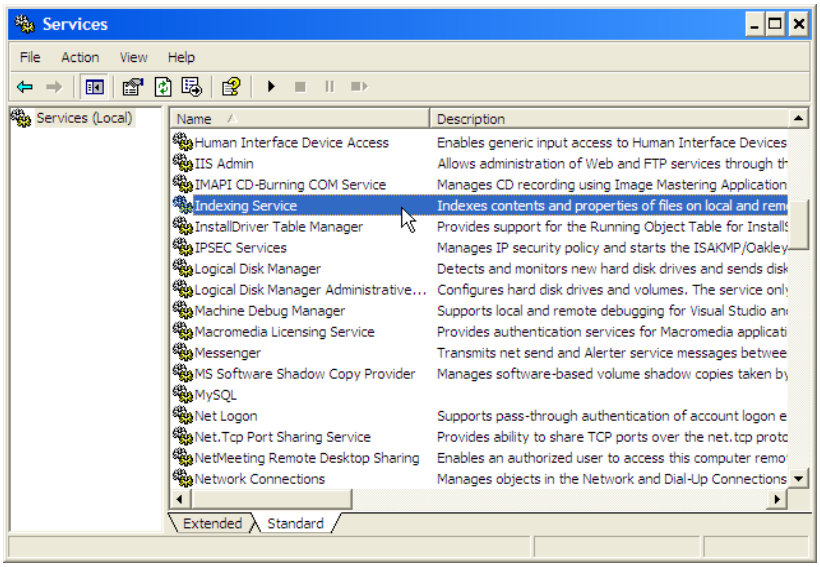
Operator $ tidak mendukung pencarian *fuzzy*. Pencarian *fuzzy*, seperti "*Does any fox have fleas*?", tidak akan menghasilkan apapun.

Ada beberapa cara untuk mengatasi ketiadaan fitur pencarian lanjut ini. Beberapa perusahaan perangkat lunak menyediakan fasilitas ini berupa aplikasi tambahan yang ditawarkan dengan harga yang cukup mahal. Karena itu lebih baik didayagunakan kemampuan yang telah tersedia untuk mendapatkan fasilitas ini, yaitu *Microsoft Indexing Service*. *Indexing Service* tersedia pada komputer yang berjalan di atas sistem operasi berbasis NT, misalnya Windows NT 4, Windows 2000, dan Windows XP.

## 2. MEMBUAT KATALOG

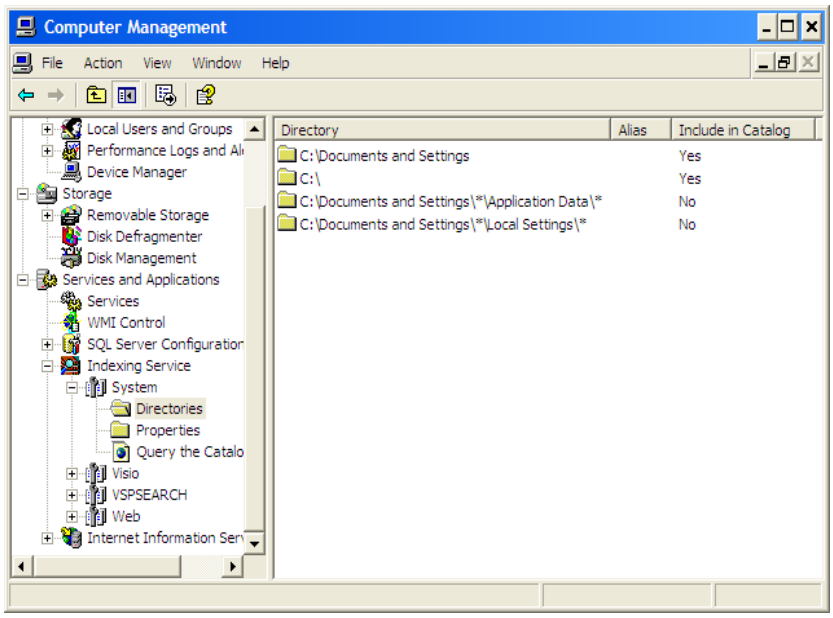
Tahap pertama adalah memeriksa apakah pada komputer telah terinstalasi *Indexing Service*. Bukalah *applet Service* melalui *Control Panel* ⇨ *Administrative Tools* ⇨ *Services*. *Applet Service* mencantumkan semua servis yang terinstalasi di dalam komputer dan dapat ditemukan servis berjudul *Indexing Service* (lihat Gambar 1).

Untuk mencobanya dapat dilakukan klik kanan pada *mouse* pada *Indexing Service* dan pilih menu *Properties* melalui menu konteks untuk membuka kotak dialog. Selanjutnya *Startup type* diubah menjadi *Automatic* agar servis ini dapat dijalankan secara otomatis saat komputer dijalankan. Jika bagian *service status* bertuliskan *stopped*, servis ini segera diaktifkan dengan cara mengklik tombol *Start*. Tombol OK digunakan untuk menutup kotak dialog *Properties*.




**Gambar 1. *Indexing Service***
**(*Applet* ini digunakan untuk menjalankan atau menghentikan *Indexing Service*)**

Setelah servis diaktifkan, dapat dilakukan konfigurasi melalui *Control Panel* ⇨ *Administrative Tools* ⇨ *Computer Management* (lihat Gambar 2). Di dalam *applet Computer Management* terdapat beberapa peralatan diagnostik yang bermanfaat seperti catatan performansi dan *event*, peralatan manajemen media penyimpanan, dan servis.




**Gambar 2. *Computer Management***
**(*Applet* ini digunakan untuk mengatur direktori yang termasuk dalam *Indexing Service*)**

Ketika melakukan instalasi sistem operasi, dibuat sebuah katalog pencarian teks *default* bernama *System*. Katalog ini tercantum di bawah *Indexing Service* di bagian kiri layar *Computer Management*. Jika *server web* terinstalasi di komputer, maka akan muncul juga sebuah katalog bernama *web*. Di dalam katalog ini terdapat *folder Directories*, *Properties*, dan *Query the Catalog*.

*Folder Directories* mencantumkan jalur yang termasuk atau tidak termasuk ke dalam indeks. File yang terdapat di dalam jalur tersebut akan ditambahkan ke dalam *Index Server*. *Query* terhadap file tersebut dapat dilakukan berdasarkan kata atau frasa tertentu. *Folder Properties* lebih kompleks. Untuk kebutuhan makalah ini, pengaturan *default* sudah mencukupi sehingga tidak perlu dilakukan perubahan apapun. Bagian *Query the Catalog* berguna untuk menguji katalog tersebut dan akan dibahas di bagian selanjutnya.

Berikut ini adalah contoh sederhana untuk menunjukkan bagaimana cara menggunakan *Indexing Service*. Yang harus dilakukan pertama kali adalah membuat sebuah *folder* yang berfungsi untuk menyimpan dokumen teks yang ingin di-*query* dan menambahkan sebuah katalog dengan jalur direktori menuju *folder* baru tersebut.

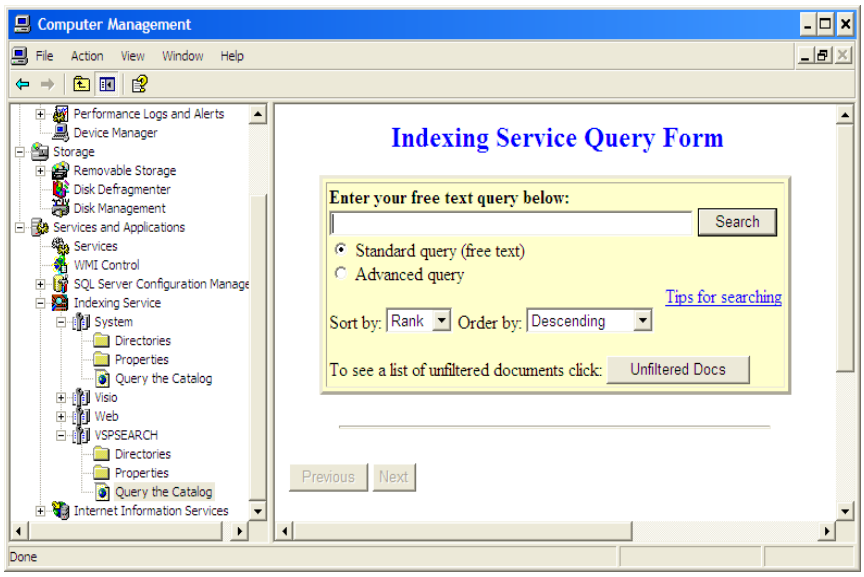
Asumsi DBC terdapat di C:\ApplicationName\Data dan dokumen teks disimpan di sebuah *subfolder* di dalamnya. Untuk itu dapat diikuti langkah berikut:

1. Membuat sebuah *folder* baru dengan nama C:\ApplicationName\Data\Documents.
2. Membuat sebuah katalog baru dengan mengklik kanan *node Indexing Service* yang terdapat di *applet Computer Management*. Pilih *New* ⇨ *Catalog* pada menu konteks. Kotak dialog *Add Catalog* akan terbuka.
3. Di dalam kotak dialog *Add Catalog*, VFPSEARCH diketikkan sebagai nama katalog yang baru dan ketikkan juga C:\ApplicationName\Data\Documents di bagian lokasi. Ini adalah lokasi untuk menyimpan indeks dan file sistem dari katalog baru. Untuk menutup kotak dialog dapat dilakukan dengan cara mengklik tombol OK. Akan ditampilkan pesan "*This catalog will remain off line until the service is restarted*."

   Katalog baru bernama VFPSEARCH telah ditambahkan ke dalam *Indexing Service.* Mungkin diperlukan menutup dan membuka kembali *applet Computer Management* untuk melihat katalog VFPSEARCH yang baru dibuat. Di dalam VFPSEARCH akan terdapat *folder Directories, Properties,* dan *Query the Catalog.*
4. Klik kanan pada *folder Directories* dan pilih *New* ⇨ *Directories*. Kotak dialog *Add Directories* akan terbuka.
5. Di dalam kotak dialog tersebut, ketikkan C:\ApplicationName\Data\Documents di kotak teks *Path* dan kotak teks Alias dikosongkan. Kemudian klik tombol OK.
6. Hingga saat ini belum ada file di dalam *folder* yang telah disiapkan. Untuk menguji *Indexing Service*, tempatkan beberapa file di dalam *folder Documents* tersebut. Sebuah file bernama SEARCHME.TXT perlu dibuat dengan menggunakan aplikasi pengolah kata atau *Notepad*, yang mengandung frasa "*My fox is too fast for fleas*".
7. Klik kanan *node Indexing Service* di *applet Computer Management* dan pilih *Stop*. Kemudian klik kanan dan pilih *Start* kembali untuk menjalankan ulang *Indexing Service*. Pilih *folder Directories* yang berada di dalam *node* VFPSEARCH. Kemudian di bagian kanan *applet Computer Management*, klik kanan direktori C:\ApplicationName\Data\Documents. Pilih *All Tasks* ⇨ *Rescan* (*Full*).

Sekarang dapat diuji *Indexing Service* untuk katalog dan direktori yang telah dibuat. Selanjutnya *Node Query the Catalog* yang terdapat di dalam katalog

VFPSEARCH diklik. Sebuah halaman *web* akan ditampilkan di bagian kanan jendela *Computer Management* (lihat Gambar 3). Halaman *web* ini berfungsi seperti fasilitas pencarian di internet, tetapi hanya melakukan pencarian di direktori yang tercakup di dalam katalog.




**Gambar 3. *Query the Catalog***
**(*Applet* ini digunakan untuk melakukan pencarian di dalam direktori)**

Untuk menguji apakah katalog telah diatur dengan baik, ketikkan sebuah kata atau frasa yang terdapat di dalam dokumen di kotak teks *free text query*, kemudian klik tombol Search. Hasil pencarian akan ditampilkan sebagai *hyperlink* ke dokumen tersebut. Dari contoh di atas, seharusnya akan muncul minimal sebuah *hyperlink* ke file SEARCHME.TXT yang telah disiapkan sebelumnya.

## 3. MENGGUNAKAN VISUAL FOXPRO DENGAN INDEXING SERVICE

Bagian terbaik dari *Index Server* adalah dapat dibuat program untuk mengendalikannya dan permintaan layanan. *Indexing Service* tidak memiliki *driver* ODBC, namun terdapat OLEDB sehingga dapat digunakan ADO untuk melakukan *query* ke *Index Server*. Berikut ini adalah contoh kode untuk melakukan *query* terhadap katalog VFPSEARCH untuk mencari file yang mengandung kata "*flea*":

```
LOCAL RS AS ADODB.Recordset
```

```
LOCAL CN AS ADODB.CONNECTION
LOCAL cCommandText AS String
LOCAL cnt AS Integer


** Create an OLEDB connection to the
** Indexing Service using the catalog named VFP
**
CN = CREATEOBJECT('ADODB.Connection')
cn.ConnectionString = ;
'Provider=MSIDXS;Data Source=VFPSEARCH;'
CN.Open()

** Build the query string and send the
** query to the Indexing Service.
**
cCommandText = ;
  [SELECT path, Filename FROM Scope() ] +;
  [WHERE CONTAINS('flea') > 0 ]
RS = CreateObject("ADODB.Recordset")
RS.Open(cCommandText, cn, 0)

** Scan through the result set and
** display to the screen
nCnt = 0
DO WHILE !RS.EOF
  nCnt = nCnt + 1
  ? nCnt, rs.Fields(0).Value, rs.Fields(1).Value
  RS.MoveNext
ENDDO
```

Perintah di atas akan menghasilkan dua objek ADO: sebuah objek koneksi dan sebuah objek *recordset*. Objek koneksi diatur untuk menggunakan OLEDB milik *Indexing Service*, sedangkan sumber data (*data source*) menggunakan katalog yang akan digunakan untuk *query*, yaitu VFPSEARCH. Untuk membuka objek *recordset* menggunakan objek koneksi dan sebuah *string query*.

Ketika Visual FoxPro menjalankan sebuah *query*, biasanya *query* tersebut dilakukan terhadap objek basis data seperti *tabel*, view, atau *stored procedure*. Menjalankan *query* terhadap *Indexing Service* sedikit berbeda karena di sini tidak ada objek basis data. Ini mengakibatkan *string query* akan terlihat sedikit aneh karena tidak digunakan tabel fisik tetapi digantikan dengan sebuah fungsi yang dinamakan SCOPE(). Dalam SQL standar, *string query* yang digunakan adalah:

```
SELECT path, filename
FROM SCOPE()
WHERE CONTAINS('fleas') > 0
```

*Query* ini akan menghasilkan dua kolom, yaitu lokasi dan nama file yang memenuhi kriteria pencarian tersebut. Fungsi SCOPE() adalah sebuah *pseudo-table* bagi semua file yang terdapat di dalam direktori yang tercantum di katalog. Pada contoh ini, SCOPE() berarti semua file di direktori C:\ApplicationName \Data\Documents.

Karena katalog bisa mengandung banyak direktori, fungsi SCOPE() dapat diatur untuk melakukan pencarian ke jalur tertentu. Sebagai contoh, SCOPE('C:\Application Name\OtherText') hanya melakukan pencarian ke dalam *folder OtherText* saja.

Fungsi CONTAINS() memerintahkan *Indexing Service* untuk mencari file yang mengandung kata atau frasa tertentu. Berikut adalah contoh lain:

```
SELECT path, filename, created
FROM SCOPE()
WHERE CONTAINS('dog NEAR fleas') > 0
ORDER BY size DESC
```

*Query* ini menghasilkan tiga kolom, yaitu lokasi, nama file dan tanggal dokumen dibuat, yang mengandung kata "*dog*" di dekat kata "*fleas*". Hasil pencarian diurutkan dari ukuran dokumen terbesar ke terkecil.

*Query* berikut ini untuk mencari dokumen yang memiliki jawaban atas pertanyaan: *Do I have lint in my belly button*?

```
SELECT path, filename, size
FROM SCOPE()
WHERE FREETEXT('Do I have lint in my belly button?') > 0
```

*Query* berikut ini untuk mencari dokumen yang mengandung kata FoxPro dan berukuran lebih besar dari 1024 byte.

```
SELECT path, filename
FROM SCOPE()
WHERE CONTAINS('FoxPro') > 0 AND size > 1024
```

Perintah SELECT yang dijalankan pada *Index Server* memiliki sintaks dan perilaku seperti perintah SELECT pada SQL standar, namun terdapat sedikit perbedaan sintaks dan keterbatasan. Menurut Microsoft, sintaks SQL untuk *Index Server* adalah:

```
SELECT Column_Name[, ...]
FROM Scope([cPathTransveral])
[WHERE FilterCondition [AND | OR FilterCondition ...]] [GROUP BY
GroupColumn [,GroupColumn ...]]
[ORDER BY Order_Item [ASC | DESC]
        [,Order_Item [ASC | DESC] ...]]
```

Penjelasan argumen berikut:

**Column_Name** – Lihat tabel 1 untuk daftar kolom yang tersedia dan fungsinya.

**Scope([cPathTransversal])** – Menunjukkan jalur pencarian. Jika cPathTransversal diabaikan, maka pencarian dilakukan terhadap seluruh direktori dalam katalog.

**FilterCondition** – Terdapat tujuh predikat yang dapat dikombinasikan (lihat tabel 2).

**GroupColumn** – Cara mengelompokkan hasil pencarian.

**Order_Item** – Cara mengurutkan hasil pencarian.

**Tabel 1. Daftar Column_Name yang paling sering digunakan**

| Column_Name | Deskripsi |
|---|---|
| Access | Kapan terakhir dokumen diakses |
| Contents | Isi dokumen |
| Created | Kapan dokumen dibuat |
| Directory | Lokasi fisik dokumen, tidak termasuk nama dokumen. |
| DocCreatedTm | Kapan dokumen dibuat |
| DocWordCount | Jumlah kata di dalam dokumen |
| FileName | Nama dokumen |
| HitCount | Banyaknya hasil pencarian di dalam dokumen |
| Path | Lokasi fisik dokumen termasuk nama dokumen. |
| Rank | Peringkat kecocokan hasil pencarian. Rentang nilai 0 sampai 1000. Angka yang besar menunjukkan kecocokan yang lebih baik. |
| ShortFileName | Nama dokumen dalam format 8.3 |
| Size | Ukuran dokumen dalam byte |
| Vpath | Lokasi virtual dokumen termasuk nama dokumen. |
| Write | Kapan terakhir dokumen diubah |

**Tabel 2. Daftar predikat yang dapat digunakan pada *FilterCondition***

| Tipe Predikat | Deskripsi |
|---|---|
| ARRAY | Membandingkan dua buah array menggunakan operator logika |
| Comparison | Menggunakan operator aritmetik untuk membandingkan data kolom dengan sebuah nilai literal |
| CONTAINS | Menentukan aturan Boolean atau kondisi pemenggalan untuk mencari kesamaan teks |
| FREETEXT | Pencarian untuk menemukan kesamaan terbaik untuk suatu frasa |
| LIKE | Pencarian berdasarkan kesamaan pola dengan karakter *wildcard* |
| MATCHES | Pencarian berdasarkan kesamaan pola dengan ekspresi reguler |
| NULL | Menentukan apakah hasil pencarian bernilai kosong |

Predikat yang paling sering digunakan adalah CONTAINS dan FREETEXT. Untuk penjelasan lebih lanjut mengenai sintaks *query*, fungsi, objek, dan semua aspek di dalam *Index Server*, dapat dibaca di Microsoft's MSDN *Library* di http://msdn.microsoft.com/library/en-us/indexsrv/html/indexingservicestartpage_6td1.asp

Sedangkan predikat FREETEXT() mencoba melakukan pencarian berdasarkan arti dari teks. Pencarian ini sama seperti mesin pencarian *Ask Jeeves* di internet yang cukup terkenal. Tentunya metode pencarian ini hanya efektif untuk teks dalam Bahasa Inggris. Sebagai contoh:

FREETEXT('Is Visual FoxPro Object-Oriented?')

Predikat CONTAIN() dapat melakukan pencarian dalam lima cara (lihat tabel 3). Perhatikan bahwa terdapat beberapa cara penggunaan predikat CONTAIN() yang hanya efektif untuk teks dalam Bahasa Inggris.

**Tabel 3. Lima cara penggunaan predikat `CONTAINS()`**

| Penggunaan | Deskripsi |
|---|---|
| Simple:<br>CONTAINS('word or phrase') | Mencari dokumen yang mengandung frasa yang tercantum di dalam petik tunggal |
| Prefix:<br>CONTAINS('"wordpart*"') | Mencari dokumen yang mengandung frasa yang dimulai dengan bagian kata yang tercantum di dalam petik tunggal, tanda * menandakan *wildcard.* Perhatikan bagian kata harus dibatasi dengan petik tunggal dan petik ganda.<br>Contoh : CONTAINS('"like*"') akan melakukan pencarian terhadap kata "*like*", "*liked*", dan "*likes*". |
| Proximity:<br>CONTAINS('word1 NEAR word2') | Mencari dokumen yang mengandung frasa di dekat frasa lain.<br>Contoh: CONTAINS('"Client" NEAR "Server"') |
| Linguistic Generation:<br>CONTAINS('FORMSOF (INFLECTIONAL,"Word")') | Mencari dokumen yang mengandung kata atau bentuk jamak, atau bentuk kata kerjanya.<br>Contoh: CONTAINS('FORMSOF (INFLECTIONAL,"fox")') akan menghasilkan pencarian dokumen yang mengandung kata "*fox*", "*foxes*", dan "*foxy*" |
| Weighted:<br>CONTAINS('ISABOUT("term 1"[WEIGHT(value)],"term2..., ")') | Mencari dokumen yang mengandung daftar kata dan mengurutkan hasilnya berdasarkan bobot yang ditentukan.<br>Contoh: CONTAINS('ISABOUT("FoxPro" WEIGHT(0.9), "dBase" WEIGHT(0.5), "xBase" WEIGHT(0.2))') |

*Index Server* hanya mendukung *query*. Tidak tersedia perintah INSERT, UPDATE, atau DELETE. Keterbatasan lainnya adalah *field* isi file boleh dipakai di klausa WHERE (biasanya sebagai bagian dari sintaks CONTAINS()), tetapi tidak boleh dipakai sebagai daftar kolom SELECT. Jadi dengan menggunakan *query Index Server*, isi file sebagai sebuah *string* hasil pencarian tidak bisa dihasilkan.

## 4. MENGGUNAKAN INDEKS PADA MEMO

*Index Server* tidak memiliki konsep tabel, baris, kolom, atau bahkan basis data. *Index Server* hanya mengenal file, artinya sebuah *field* memo harus diubah menjadi file. Cara terbaik untuk melakukan hal ini adalah dengan *trigger* INSERT dan UPDATE untuk menyalin isi *field* memo ke dalam sebuah file. Dengan cara ini, ketika sebuah memo ditambahkan atau di-*update*, sebuah dokumen teks baru disalin ke dalam direktori katalog *Index Server*.

Contoh, terdapat sebuah tabel ***Customer*** yang memiliki *field* memo ***custnotes*** yang akan ditambahkan ke katalog *Index Server* bernama VFPSEARCH. Tabel ***Customer*** memiliki *primary key* ***custid***. Berikut ini adalah kode untuk membuat *trigger* UPDATE dan INSERT:

```
CREATE TRIGGER ON Customer ;
  FOR UPDATE AS CustomerMemoIndex()
CREATE TRIGGER ON Customer ;
  FOR INSERT AS CustomerMemoIndex()
```

Berikut ini adalah kode *trigger*-nya:

```
FUNCTION CustomerMemoIndex()
LOCAL cFile

** Has something changed in the custnotes field?
**
IF !EMPTY(NVL(custnotes,’’)) AND ;
    (GETFLDSTATE(’custnotes’) = 2 ;
  OR GETFLDSTATE(’custnotes’) = 4)
  cFile=’C:\ApplicationName\Data\Documents\Customer_’ +;
        Customer.custid
  STRTOFILE(Customer.custnotes, cFile)
ENDIF
```

Kode ini dijalankan setiap kali *record* ***Customer*** bertambah atau di-*update*. Ketika *field* ***custnotes*** bertambah atau berubah, teks dari *field* ***custnotes*** ini ditulis menjadi sebuah file di dalam *folder* C:\ApplicationName\Data\Documents, yang terdaftar sebagai direktori yang diindeks oleh katalog VFPSEARCH.

Trik untuk memanfaatkan *Index Server* sebagai fasilitas pencarian pada *field* memo ini tampak sangat menarik, tetapi terdapat kekurangan besar yang harus dibayar. Dengan menggunakan skenario di atas, dalam sekejap akan diperoleh ribuan bahkan jutaan file teks di dalam direktori katalog VFPSEARCH. Walaupun media penyimpanan yang dibutuhkan untuk menyimpan file teks mungkin tidak menjadi

masalah (karena ukuran file teks umumnya kecil), namun jumlah file yang banyak akan menurunkan performansi indeksasi dan pencarian secara signifikan.

*Index Server* berjalan sebagai servis di latar belakang dan memindai setiap perubahan yang terjadi pada direktori katalog yang sedang aktif. Ketika terdeteksi sebuah file baru atau terjadi perubahan pada file, segera dibuat indeks baru. Pengaturan seberapa agresif Index Server melakukan pemeriksaan ini dapatlah dilakukan. Jika *Index Server* diatur terlalu pasif, maka dibutuhkan selang waktu yang cukup lama sejak perubahan file dilakukan sampai dilakukannya indeks ulang. Ini dapat meningkatkan performansi sistem, namun mengurangi akurasi indeks. Perlu dihitung dan ditemukan ukuran yang optimal sesuai dengan yang dibutuhkan sistem.

Teknik ini memungkinkan digunakannya nama file dari dokumen memo sebagai penghubung ke basis data. Contoh: misalnya akan dilakukan pencarian terhadap kata "*flea*" menggunakan predikat CONTAINS() di dalam katalog VFPSEARCH dan memperoleh hasil sebuah *record* dengan nama file Customer_L1V0S0M53 dimana nama file tersebut dapat dicari di tabel ***Customer*** dan melakukan *query* di Visual FoxPro dengan menggunakan *primary key* (L1V0S0M53) untuk mencari *record* yang berhubungan dengan tabel ***Customer***.

Sebuah fungsi *trigger* generik dapat dibuat dan diberi nama MemoIndex(). Fungsi ini dipanggil melalui *trigger* INSERT dan UPDATE dari tabel yang akan termasuk ke dalam katalog *Index Server*. Fungsi ini memiliki empat parameter:

**cTableName** – Nama tabel dari tabel yang menjalankan *trigger*

**cMemoField** – *Field* memo yang akan diindeks

**cKeyField** – Nama kolom *primary key*

**cCatalogPath** – Lokasi direktori katalog tempat menyimpan teks memo

Sekarang *trigger* UPDATE berubah menjadi:

```
CREATE TRIGGER ON Customer ;
  FOR UPDATE AS ;
    MemoIndex('Customer',;
    'custnotes',;
    'custid',;
    'C:\ApplicationName\Data\Documents')
```

Fungsi di atas harus dimodifikasi jika tabel memiliki lebih dari satu *field* memo yang akan diindeks atau jika tabel memiliki *primary key* majemuk.

## 5 . REMOTE INDEXING

Selain penggunaan *Index Server* pada aplikasi yang berdiri sendiri (*standalone*) atau *server web* dengan Visual FoxPro sebagai basis datanya, *Index Server* juga dapat membuat indeks dari *folder* yang terdapat di komputer lain di dalam sebuah jaringan. Sebuah katalog di komputer lokal dapat dibuat untuk mengindeks direktori yang berada di komputer lain.

Ketika membuat direktori baru di dalam katalog, alamat direktori harus dituliskan, nama *user* dan *password* untuk mengakses file yang di-*sharing* di komputer target. Kelemahan dari cara ini adalah semua komputer harus berjalan di sistem operasi Windows NT SP5, Windows 2000 atau Windows XP. Ini disebabkan *Index Server* berjalan sebagai NT Servis dan hanya dapat dijumpai di sistem operasi yang disebutkan di atas. Karena *Index Server* akan terus memonitor komputer di dalam jaringan dan melakukan *update* indeks, maka dapat timbul masalah performansi jaringan. Karena itu jeda waktu monitor dan performansi jaringan harus dihitung dengan hati-hati.

## 6. KESIMPULAN

Microsoft *Index Server* sangat membantu dalam membuat katalog dan fungsi pencarian. Namun fasilitas ini jarang sekali dimanfaatkan oleh *programmer*. Sayangnya sampai saat ini masih sedikit dokumentasi yang tersedia mengenai *Microsoft Index Server*.

## 7. DAFTAR PUSTAKA

[1]. Pinter, Les, 2004, *Visual FoxPro to Visual Basic.NET*, Sams Publishing.

[2]. http://msdn.microsoft.com/library/en-us/indexsrv/html/indexingservicestartpage_6t d1.asp