HOME DAFTAR ISI

4

INFORMIX-4GL

4.1. Pendahuluan

Informix-4gl adalah bahasa pemrograman generasi ke-4 yang dikembangkan oleh Informix Software, Inc. dan didesign khusus untuk aplikasi database. Bahasa Generasi ke-4 seperti Informix-4gl merupakan kemajuan terakhir dalam pemrograman.

Apakah Bahasa Generasi ke-4 itu ?

Bahasa Pemrograman Generasi ke-4 didesign khusus untuk aplikasi. Bahasa tersebut sedikit lebih kompleks daripada bahasa pemrograman yang umum seperti COBOL atau C, dan bahasa generasi ke-4 tersebut mendekati definisi dari bahasa sehari-hari (Natural Language). Karena berfokus pada type yang spesifik dari aplikasi, maka bahasa generasi ke-4 dapat menangani apa yang kita inginkan dalam menyelesaikan program. Bahasa Generasi ke-4 juga mempunyai kemampuan yang penuh; seperti statement sederhana dapat melaksanakan beberapa pekerjaan yang ditransfer ke dalam bahasa mesin. Beberapa keuntungan dari Bahasa Pemrograman Generasi ke-4 adalah :

- Sangat sederhana dalam membentuk proses aplikasi
- Secara umum interaktif, sehingga sangat sederhana untuk proses debug
- Hasil dari aplikasinya sangat mudah untuk digunakan dan dapat memecahkan masalah secara efisien.

Bahasa Prosedur dan Non-Prosedur

Bahasa pemrograman dapat ditujukan sebagai bahasa prosedur atau non-prosedur. Jika menggunakan bahasa prosedur, kita harus menspesifikasikan dalam program kita bagaimana kita akan menyelesaikan sesuatu. Langkah-langkah tersebut mendekati pembuatan bahasa prosedur sangat flexibel, sehingga kita dapat menggunakannya untuk berbagai aplikasi. Contoh : Jika kita akan mendesign program menu menggunakan bahasa COBOL atau C kita harus menspesifikasikan langkah demi langkah bagaimana menampilkan menu dan mengatasi masukan dari pemakai.

Program tersebut juga harus terdapat statement untuk menampilkan judul menu dan menu pilihan dan untuk memindahkan kursor dari satu pilihan ke pilihan lain. Juga akan terdapat statement kondisi seperti IF atau CASE yang dapat melaksanakan berbagai pekerjaan tergantung dari masukan pemakai.

Menggunakan bahasa Non-prosedur kita harus menspesifikasikan hasil yang diinginkan. Contoh, dalam hal yang sama kita akan mendesign program menu, maka kita harus membentuk menu menggunakan suatu statement contohnya statement MENU dalam Informix-4gl. Kita tidak perlu untuk menggunakan statement PRINT untuk menampilkan judul menu dan menu pilihan sebab MENU mempunyai proses yang sudah ada untuk menampilkan menu tersebut. Kita juga tidak perlu menggunakan statement kondisi untuk mengatasi permintaan masukan dari pemakai, karena MENU akan membentuk statement seperti CASE.

Informix-4gl mengkombinasikan ciri-ciri dari bahasa prosedur dan non prosedur. Informix-4gl menyediakan statement-statement non-prosedur seperti statement MENU untuk membuat pembentukan aplikasi yang sederhana. Informix-4gl juga menyediakan statement-statement prosedur seperti IF, FOR, dan WHILE.

Implementasi/bentuk dari Aplikasi Informix-4gl

Informix Software, Inc. menawarkan 2 bentuk dari aplikasi Informix-4gl yaitu :

1. Informix-4gl C Compiler yang menggunakan preprosesor untuk membentuk source code Informix-ESQL/C. Kode ini yang diproses untuk menghasilkan C

source code, untuk kemudian dikompile dan dilink sebagai object code sehingga dapat dijalankan.

2. Informix-4gl RDS (Rapid Development System) yang menggunakan compiler untuk menghasilkan pseudo-code (p-code) dalam satu langkah. Kemudian kita meminta "runner" untuk menjalankan p-code dari aplikasi kita.

Kedua bentuk sama dalam hal penggunaan statement-statement 4GL, perbedaannya terletak dalam hal berikut :

- Perbedaan dalam perintah eksekusi program.

Compiler	RDS	Hasil
i4gl	4gl	Masuk dalam menu 4GL
c4gl file.4gl	fglpc file.4gl	Meng-kompile file.4gl
xfile.4ge	fglgo xfile.4gi	Menjalankan program xfile
i4gldemo	r4gldemo	Memasuki demo

- Perbedaan dalam ekstension nama file

Compiler	RDS	Keterangan
.o	.4go	File program hasil kompile
.4ge	.4gi	File program 4gl yang akan dieksekusi

Untuk masuk kedalam menu utama informix-4gl ketik r4gl dari promp unix kemudian akan tampil :

```
INFORMIX-4GL :   Module   Form   Program   Query-language   Exit
Create , modify , or run individual 4gl program modules
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -- - - - - - - - - - -- - Press CTRL-W for help
- - - -
```

Module	Bekerja pada program module Informix-4gl.
Form	Bekerja pada Screen Form.
Program	Spesifikasi komponen dari program multi-module.
Query language	Menggunakan SQL interaktif

Exit	Kembali ke sistem operasi

MODULE DESIGN MENU

Menu yang akan tampil dilayar bilai kita memilih module

```
Module ;  Modify  New  Compile  Program_Compile  Run  Debug  Exit
Change an existing 4gl program module
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -- - - - - - - - - - -- - Press CTRL-W for help
- - - -
```

Modify	Modifikasi module 4GL yang sudah ada
New	Membuat module 4GL
Compile	Meng-kompile module 4GL yang sudah ada
Program-compile	Meng-kompile program aplikasi
Run	Eksekusi module 4GL yang sudah dikompile atau multi module program aplikasi.
Exit	Kembali ke menu utama Informix-4GL

--Modify

Jika kita memilih modify, maka 4GL akan meminta nama dari module yang akan dimodifikasi, kemudian setelah selesai modifikasi module maka akan tampil menu modify module yang terdiri dari COMPILE, SAVE-AND-EXIT, dan DISCARD-AND-EXIT.

Jika kita memilih Compile maka akan tampil menu compile module yang terdiri dari : OBJECT, RUNABLE dan EXIT

OBJECT	akan membentuk file dengan ekstension .4go program yang Dikompile adalah multi-module program
RUNABLE	akan membentuk file dengan ekstension .4gi program yang dikompile adalah module yang stand-alone
EXIT	kembali ke menu modify module

Jika setelah di-kompile tidak ada kesalahan maka pada menu modify module akan tampil pada pilihan Save-and-Exit.

-- New

Jika pilihan ini dipilih maka akan dibentuk sebuah module 4GL yang baru,sebelum membuat kita harus memberikan nama modulenya.

-- Compile

Pilihan ini memungkinkan kita untuk meng-kompile module 4GL yang stand-alone tanpa melalui pilihan Modify.

Setelah itu akan tampil pula pilihan OBJECT, RUNABLE dan EXIT

-- Program-Compile

Pilihan ini sama seperti pilihan compile dari PROGRAM DESIGN MENU. Akan mengijinkan kita untuk meng-kompile dan menggabung module. Pilihan ini berguna jika kita mempunyai satu module dari sekelompok program yang kompleks dan ingin menguji dengan cara meng-kompilenya dengan module lain.

-- Run

Pilihan ini akan menjalankan module program dengan ekstension .4gi

-- Debug

Memilih pilihan ini berarti kita menggunakan Informix-4GL Interactive Debugger untuk menganalisa program.

-- Exit

Kembali ke menu Informix-4gl.

QUERY LANGUAGE MENU

Query language yang terdapat di 4GL sama seperti yang berada di Informix-SQL. Dengan pilihan ini kita tidak perlu keluar dari 4GL untuk bekerja menggunakan statement-statement SQL.

Type Statement

Terdapat 12 tipe statement yang disediakan oleh INFORMIX-4GL, yaitu :

1. Statement organisasi program :
   - FUNCTION
   - MAIN
   - REPORT

2. Statement definisi variabel :
   - DEFINE
   - GLOBALS

3. Statement tugas :
   - INITIALIZE
   - LET

4. Statement aliran program :

- CALL	- GOTO	- FOR
- CASE	- IF	- SLEEP
- CONTINUE	- LABEL	- FOREACH
- EXIT	- RETURN	- WHILE

5. Statement interaksi screen :

- CLEAR	- DISPLAY ARRAY	- MESSAGE
- CLOSE FORM	- DISPLAY FORM	- OPEN FORM
- CLOSE WINDOW	- INPUT	- OPEN WINDOW

- DISPLAY - MENU - PROMPT

6. Statement report :
   - START REPORT
   - OUTPUT TO REPORT
   - FINISH REPORT

7. Statement definisi data :
   - ALTER INDEX - CREATE TABLE - DROP TABLE
   - CLOSE DATABASE - DATABASE - RENAME COLUMN
   - CREATE DATABASE - DROP DATABASE - RENAME TABLE
   - CREATE INDEX - DROP INDEX - DROP VIEW

8. Statement manipulasi data :
   - DELETE - SELECT
   - INSERT - UNLOAD
   - LOAD - UPDATE

9. Statement manipulasi kursor :
   - CLOSE - OPEN
   - DECLARE - PUT
   - FETCH - FLUSH

10. Statement manajemen dinamik
   - EXECUTE
   - FREE
   - PREPARE

11. Statement akses data :
   - GRANT - UNLOCK TABLE
   - LOCK TABLE - REVOKE

12. Statement keutuhan data :

- BEGIN WORK - ROLLBACK WORK
- COMMIT WORK - ROLLFORWARD DATABASE
- CREATE AUDIT - START DATABASE
- DROP AUDIT - VALIDATE
- RECOVER TABLE

4.2. Perintah dasar, loop dan kondisi pada Informix-4GL

Dalam setiap modul program 4GL harus terdapat **main program.**

Dalam main program harus ada statement **MAIN**

```
Main
   ……..
   …..statement……
   ……..……
End main
```

MAIN → menyatakan awal blok program .

END MAIN → menyatakan akhir blok program.

Contoh :

```
Main
    CREATE DATABASE filkom
    CREATE TABLE t_matakuliah
        (kd_mk       char(8),
         nm_mk       char(30),
         sks         smallint,
         jenis       char(1))
    INSERT INTO t_matakuliah
        VALUES (“DU-14201”,”Agama”,2,”W”)
End main
```

```
Main
   display ‘Universitas Gunadarma’ at 6,7
   display ‘Jl. Akses UI Kelapa Dua’ at 7,7
   attribute (reverse,blink)
end main
```

Output :

Definisi Variabel program

Variabel program disimpan sementara dalam memory. Variabel yang akan dipakai harus didefinisikan terlebih dahulu.

Syntak :

DEFINE variabel-name data-type[,...]

Program Interaktif

- Input dari keyborad

Syntak :

PROMPT display_list for variabel

- Menampilkan ke layar

Syntak :

```
DISPLAY  {BY NAME variable-list | variabel-list TO { field-list |
    screen-record [[n]].*} [,...] [USING format]
   | AT row,column ] }
    [ ATTRIBUTE (attribute-list)]
```

Contoh :

```
Main
   Define
       nm    char(25),
       ket   char(30)

     Let ket = 'mahasiswa Universitas Gunadarma'
     Prompt  'Nama anda   : ' for  nm
        Display 'Mhs. yang bernama ',nm,ket at 10,5
        Display 'Mhs. yang bernama ',nm,' adalah',ket at 12,5
End main
```

```
Main
   Define  nm   char(25),
           kls   char(5)

     prompt 'Masukkan nama anda  : ' for nm
     prompt 'Masukkan kelas anda  : ' for kls

     display 'Saya yang bernama : ',nm
     display 'Sekarang duduk di kelas ',kls
     display 'Sedang belajar Informix'
```

```
End main
```

Program Flow

Program flow adalah statemen untuk mengkontrol jalannya program. Statemen ini tidak dapat dipakai dalam SQL.

Statemen tersebut adalah :

- FOR – END FOR

Contoh penggunaan FOR ….. END FOR

```
-- membuat program dengan menggunakan FOR … END FOR
-- kegunaan tanda '--' sama dengan tanda '#'

Database filkom
Main
    define np char(8), nama char(25), kls char(5),i smallint

FOR i = 1 to 5
     prompt 'ketik npm anda    = ' for np
     prompt 'ketik nama anda  = ' for nama
     prompt 'ketik kelas anda  = ' for kls

     INSERT INTO t_master (npm,nama_mhs,kelas)
         VALUES (np,nama,kls)

    Display 'N P M     : ',np at 6,5
    Display 'Nama      : ',nama at 7,5 attribute(blink)
    Display 'Kelas     : ',kls at 8,5
     Sleep 2
     Clear screen
END FOR

display 'selesai dech ……by…….' At 15,10
end main
```

- WHLE – END WHILE

Untuk mengerjakan perintah selama kondisi benar

Syntak :

WHILE boolean –expression

Statement

...

[CONTINUE WHILE]

...

[EXIT WHILE]

END WHILE

```
{contoh penggunaan  WHILE …..END WHILE }

database filkom
main
      define  np char(8), nama char(25), kls char(5), jawab char(1)

let jawab='y'
WHILE jawab='y' or jawab='Y'   #boleh ditulis jawab matches'[Yy]'
          Prompt 'ketik npm anda     = ' for np
          Prompt 'ketik nama anda   = ' for nama
          Prompt 'ketik kelas anda   = ' for kls
   Insert into t_master (npm,nama_mhs,kelas) values (np,nama,kls)
   Prompt 'ingin tambah data lagi [y/t] / ' for jawab
   Clear screen
END WHILE
End main
```

- IF – END IF

  Untuk memilih diantara dua kemungkinan

  Syntak :

  IF boolean-expression THEN

  Statement

  ...

  [ELSE

  Statement

  ...]

  END IF

- CASE – END CASE

  Untuk melakukan sekumpulan perintah sesuai dengan nilai ekspresinya.

  Syntak :

```
CASE [ ( expresion ) ]
        WHEN {expression | boolean-expresion}
                Statement
                ...
                [EXIT CASE]
        WHEN {expression | boolean-expresion}
                Statement
                ...
                [EXIT CASE]
        [OTHERWISE]
                statement
END CASE
```

4.3. Menggunakan database pada Informix-4GL

Program dengan menggunakan DATABASE

```
Database filkom                 #contoh program cari data
    Main
       Define
            nm    char(25), kls  char(5)

      SELECT nama,kelas INTO nm,kls
      FROM t_master
      WHERE npm='10197477'

      Display 'Nama  : ',nm at 5,9
      Display 'Kelas  : ',kls at 6,9
  End main
```

```
Database filkom
Main
Define
     nm   like t_master.nama,
     kls    like t_master.kelas,
     m_npm like t_master.npm

prompt 'Masukkan NPM yang dicari : ' for m_npm

select nama,kelas into nm,kls from t_master where npm=m_npm

display nm clipped,'/',m_npm,'/',kls at 10,12
end main
```

Contoh program INSERT

```
# statement yg diawali tanda '#' tidak diakses
# Ini adalah contoh untuk memasukkan data kedalam table
# Program ini dibuat oleh : Rizky
# Tanggal pembuatan : 24 /11/2000
#-----------------------------------------------------------------

Database filkom      # kalau pindah fakultas,ganti database
Main
     INSERT INTO t_matakuliah (kd_mk,nm_mk,sks,jenis)
               VALUES ('DK-14202','Fisika',2,'W')
End main
```

```
{ contoh kedua program insert data
 dibuat oleh ……
selain menggunakan tanda '#' boleh juga menggunakan tanda'{ }'
}

Database filkom
Main
    Define  kode_mk   char(8),
              nama_mk   char(25),
              jsks     smallint,
              jns     char(1)

let kode_mk = 'DK-14202'       #contoh penggunaan LET
let nama-mk = 'fisika'
let jsks = 2
let jns ='W'

INSERT INTO t_matakuliah(kd_mk,nm_mk,sks,jenis)
        VALUES (kode_mk,nama_mk,jsks,jns)

End main
```

```
#contoh ketiga program insert data
# bandingkan format penulisan define

database filkom

main
   define kode_mk char(8),nama_mk char(25),jsks smallint,
             jns char(1)
```

```
prompt ‘masukkan kode matakuliah = ‘ for kode_mk
prompt ‘masukkan nama matakuliah = ‘ for nama_mk
prompt ‘masukkan jumlah sks          = ‘ for jsks
prompt ‘masukkan jenis matakuliah = ‘ for jns

#dibawah ini adalah statement ‘sql’
# penulisan statement sql-nya sama dengan contoh kedua

INSERT INTO t_matakuliah (kd_mk,nm_mk,sks,jenis)
   VALUES (kode_mk,nama_mk,jsks,jns)

end main
```

Contoh Program DELETE

```
Database ekonomi
  Main
        Define np like t_master.npm

       Prompt ‘masukkan npm anda : ‘ for np

    DELETE FROM t_master WHERE npm=np
 End main
```

```
Database filkom
Main
     Delete from t_master where kelas[1]=’4’ and ket_lulus=’W’
End main
```

Contoh Program UPDATE

```
Database sastra
Main
   Define
     np  like t_master.npm,
     almt like t_master.alamat

   prompt ‘masukkan npm anda     : ‘ for np
   prompt ‘masukkan alamat baru : ‘ for almt
```

```
#perintah sql
    UPDATE t_master set alamat = almt WHERE npm = np

End main
```

```
Database sastra
Main
   UPDATE t_master set kota='Depok' WHERE kota='Cimanggis'
End main
```

RECORD

Record merupakan kumpulan dari program variable yang kemungkinannya berbeda tipe data yang satu dengan lainnya.

Sebelum memakai suatu record , kita harus mendefinisikan terlebih dahulu pada DEFINE.

```
record_name
            RECORD
                  Var_list data_type [,…….]
            END RECORD
```

```
Define
      rec_mk RECORD
              kd_mk    char(8),
              nm_mk    char(25),
              sks    smallint,
              jenis char(1)
     END RECORD
Main
   Select kd_mk,nm_mk,sks,jenis INTO rec_mk.*
    from t_matakuliah where kd_mk = 'TI-17203'

#-----------------------------------------------------------
#sql diatas bisa ditulis seperti dibawah ini
{ Select kd_mk,nm_mk,sks into rec_mk.kd_mk THRU rec_mk.sks
From t_matakuliah where kd_mk='TI-17203' }
#-----------------------------------------------------------

    display 'Nama Matakuliah   : ',rec_mk.nm_mk
```

```
    display 'Kode Matakuliah    : ',rec_mk.kd_mk
end main
```

Jika kita mendefinisikan record sama seperti pada suatu table (seperti contoh diatas), maka penulisannya dapat disederhanakan menjadi :

```
record_name RECORD LIKE table_name.*
```

```
Define
     rec_mk RECORD LIKE t_mk.*
main
  ……
end main
```

PERINTAH SELECT DENGAN CURSOR

- Mendeklarasikan cursor untuk mewakili perintah **SELECT**
- Mendapatkan row dengan **FOREACH**
- Mendapatkan row dengan **OPEN** , **FETCH** dan **CLOSE**

Mendeklarasikan cursor untuk select

sintaks : **DECLARE nama_cursor [scroll] CURSOR FOR**

SELECT statement

```
DECLARE k1 CURSOR FOR
Select * from t_master
Where kelas[1,2] = '4K'
```

```
DECLARE kur CURSOR FOR
Select npm,nama from t_master
Where nama[1]='R'
```

Mendapatkan row dengan FOREACH

Statement 'foreach' menyebabkan urut-urutan statement dikerjakan satu persatu.

sintaks : **FOREACH nama_kursor [INTO var]**
……statement……
…….

[CONTINUE FOREACH]
…….
[EXIT FOREACH]
…….
END FOREACH

- Nama kursor di FOREACH harus sudah pernah didefinisikan pada statement DECLARE
- INTO digunakan untuk menampung hasil query setiap row
- CONTINUE FOREACH atau EXIT FOREACH digunakan untuk interupsi pada kondisi tertentu
- Diakhiri dengan END FOREACH

```
# contoh penggunaan FOREACH

database filkom
main
    define rec_mk RECORD LIKE t_matakuliah.*

declare k1 cursor for
select * from t_matakuliah where kd_mk[1,2]='DU"

foreach k1 INTO rec_mk.*
       display 'Kode Matakuliah   : ',rec_mk.kd_mk
       display 'Nama Matakuliah  : ',rec_mk.nm_mk
       display 'Sks                      : ',rec_mk.sks
end foreach
end main
```

CONTINUE FOREACH

Statement IF dapat digunakan bersama statement CONTINUE FOREACH untuk menghentikan urutan statement yang ada pada FOREACH loop, dan melanjutkan pembacaan baris berikutnya.

```
#contoh penggunaan statement 'continue foreach'
database filkom
main
   define rec_mhs RECORD LIKE t_master.*

declare k2 cursor for
    select * from t_master
```

```
foreach k2 into rec_mhs.*
    if rec_mhs.kota = 'Jakarta' then
          continue foreach
    end if
        display 'N P M   : ',rec_mhs.npm
       display 'Nama    : ',rec_mhs.nama
       display 'Kota     : ',rec_mhs.kota
       display '  '
end foreach

end main
```

EXIT FOREACH

Statement IF dapat digunakan bersama statement EXIT FOREACH untuk menghentikan urutan statement yang ada pada 'foreach loop' , serta keluar dari 'foreach loop'

```
# contoh penggunaan 'exit foreach'
database filkom
main
   define rec_mhs RECORD LIKE t_master.*,jw char(1)

declare k3 cursor for
select * from t_master

foreach k3 into rec_mhs.*
        display 'N P M    : ',rec_mhs.npm
       display 'Nama      : ',rec_mhs.nama
       display 'Alamat    : ',rec_mhs.alamat

      prompt 'Ingin lihat data berikutnya ?[y/t] : ' for jw
      if jw matches'[Tt]' then
         exit foreach
      end if
end foreach

end main
```

Mendapatkan row dengan OPEN , FETCH dan CLOSE

Statement **foreach** diatas, sebenarnya mengkombinasikan beberapa fungsi statement tunggal :

- melakukan OPEN cursor yang terkait dengan statement SELECT

- Secara berulang melakukan FETCH (mengambil) baris, dan menjalankan sejumlah statement yang ada pada FOREACH loop.
- Melakukan CLOSE cursor, setelah seluruh baris terbaca seluruhnya.

OPEN

=====

Untuk melakukan aktif set dari statement SELECT yang dikaitkan dengan suatu cursor.

```
DECLARE kur CURSOR FOR
     SELECT * FROM  t_matakuliah
        WHERE jenis = 'P'
OPEN kur
```

Informix-4GL akan mencari baris sesuai statement diatas dan meletakkan cursor pada posisi awal, sebelum baris pertama.

DK-14301	Pemrog.Gene.4	3	P
TI-17204	Sistem Pakar	2	P
TI-18201	Automata	2	P

FETCH

======

Untuk menggerakkan cursor pada baris berikutnya dalam suatu aktif set, dan mengambil informasi pada baris tersebut.

```
#contoh penggunaan statement fetch

FETCH kur INTO rec_mk.*
```

Output dari statement diatas, sbb.

◄	DK-14301	Pemrog.Gene.4	3	P	cursor
	TI-17204	Sistem Pakar	2	P	
	TI-18201	Automata	2	P	

Jika FETCH dijalankan sekali lagi, maka cursor akan berpindah ke baris kedua.

Bila cursor berada setelah baris terakhir, maka tidak ada lagi informasi yang akan diambil. Pada saat yang sama Informix akan mengisi variabel status dengan informasi NOT FOUND, sebagai tanda bahwa cursor sudah berada di 'luar batas'.

DK-14301	Pemrog.Gene.4	3	P
TI-17204	Sistem Pakar	2	P
TI-18201	Automata	2	P

Status = NOT FOUND cursor◄——

Karena kita tidak dapat memperkirakan berapa jumlah baris dalam aktif set, maka kita harus selalu memeriksa isi variabel status, segera setelah statement FETCH.

Contoh :

```
FETCH kur INTO rec_mk.*
IF status = notfound then
    Message 'Tidak ada data'
ELSE
    display 'Kode Matakuliah   :,rec_mk.kd_mk
    Display 'Nama Matakuliah :,rec_mk.nm_mk
END IF
```

CLOSE

======

Jika sudah tidak membutuhkan informasi atas baris-baris yang diperoleh dari statement SELECT, maka gunakan statement CLOSE atas kursor yang terkait.

```
#contoh penggunaan statement Close

  CLOSE kur
```

Dapat menggunakan WHILE loop untuk pengulangan pembacaan (FETCH) suatu baris dan memeriksa variabel status selama seluruh baris yang ada dalam aktif set diproses.

Sintaks :

WHILE ekspresi Boolean

……
……
[CONTINUE WHILE]
……
……
[EXIT WHILE]
END WHILE

```
#contoh 1
Database filkom
        Define r_mk RECORD LIKE t_mk.*,jawab char(1)
Main
   DECLARE kurs CURSOR FOR
   Select * from t_mk where nm_mk[1]='A'

    OPEN kurs
    Let jawab='y'
```

```
    WHILE jawab matches'[Yy]'
          FETCH kurs INTO r_mk.*
             IF status = notfound then
                Message 'Data tidak ada'
                EXIT WHILE
             END IF
             Display 'Nama Matakuliah : ',r_mk.nm_mk
             Display 'Kode Matakuliah  : ',r_mk.kd_mk
             Prompt 'Lihat data selanjutnya ? [y/t]' for jawab
      ENDWHILE
      CLOSE kurs
End main
```

```
#contoh 2, untuk memastikan suatu aktif set itu kosong
Database filkom
        Define r_mk RECORD LIKE t_mk.*,jawab char(1)
Main
    DECLARE kurs SCROLL CURSOR FOR
    Select * from t_mk where nm_mk[1]='B'

    OPEN kurs
    FETCH FIRST kurs INTO r_mk.*
        If status = notfound then
             Message 'Tidak ada data pada aktif set'
```

```
        Else
              Message 'Minimal ada satu data pada aktif set'
        End if
     CLOSE kurs
End main
```

NEXT : menandakan baris berikutnya setelah baris aktif (active set)

PREVIOUS } : menandakan baris sebelumnya dari baris aktif
PRIOR }

FIRST : menandakan baris pertama pada baris aktif

LAST : menandakan baris terakhir pada baris aktif

CURRENT : menandakan pada baris yg aktif pada active list

RELATIVE m : menandakan baris ke-m relatif pada kursor yg aktif

ABSOLUTE n : menandakan baris ke-n, n bias integer atau variabel dari program

ARRAY

Array merupakan kumpulan record dengan tipe yang sama

Sintaks : **array_name ARRAY[n] OF RECORD**
Var_list data_type[..,…..]
END RECORD

n = bilangan integer, merupakan jumlah record yg dapat ditampung oleh array tersebut.

```
#contoh
ar_krs ARRAY[12] OF RECORD
       npm1     like t_nilai.npm,
       nama1    like t_nilai.nama,
       kd_mk1   like t_nilai.kd_mk,
       nm_mk1   like t_nilai.nm_mk
END RECORD
```

```
#contoh
ar_mhs ARRAY[150] OF RECORD LIKE t_master.*
```

#contoh program lengkap tentang ARRAY

```
Database filkom
Main
Define ar_mhs ARRAY[150] OF RECORD LIKE t_master.*,
          i,j smallint
DECLARE  k1  CURSOR FOR
SELECT * from t_master where kelas[1]='4'

Let i=1
FOREACH  k1  INTO ar_mhs[i].*
       Let i = i + 1
END FOREACH

Let j = i – 1
FOR i = 1 to j
    Display 'Data ke-',i,' ',ar_mhs[i].*
END FOR
End main
```