

# **DASAR PEMROGRAMAN DATABASE MENGUNAKAN DELPHI 7**

**Oleh :  
Hieronimus Edhi Nugroho, M.Kom**

**Fakultas Teknologi Informasi  
Universitas Stikubank  
Semarang  
2004**

## **KATA PENGANTAR**

Saat ini menulis sebuah program bukan lagi merupakan hal yang sulit, ada banyak sistem yang menawarkan berbagai fasilitas untuk memudahkan pemrogram menulis programnya. Umumnya sistem semacam itu menggunakan pendekatan visual serta pendekatan event, dimana pemrogram cukup menggambarkan layar yang akan digunakan dan kemudian menentukan apa yang terjadi apabila pemakai menggunakan salah satu atau beberapa komponen yang ada di layar. Sistem pemrograman semacam ini disebut sebagai lingkungan pengembangan terpadu (Integrated Development Environment / IDE).

Salah satu IDE yang diproduksi oleh Borland adalah Delphi. Delphi menggunakan pendekatan visual maupun event serta menyediakan sejumlah komponen siap pakai. Pemrogram cukup menggunakan komponen-komponen yang sudah disediakan, atau kalau perlu menyesuaikan perilaku komponen tersebut, dalam membuat program.

Banyak aspek yang dapat dibahas dari Delphi, mengingat Delphi memang ditujukan untuk spektrum sistem yang sangat luas, mulai dari sistem bisnis sampai dengan sistem yang lebih teknis. Buku ini hanya membahas bagaimana Delphi digunakan untuk membuat sistem yang berorientasi ke bisnis.

Buku ini mencoba menawarkan pendekatan yang tidak hanya terpaku pada kebutuhan praktis tetapi juga menjelaskan mengapa dan bagaimana sesuatu terjadi di Delphi. Diharapkan melalui buku ini, pembaca tidak hanya dapat membuat program menggunakan Delphi tetapi juga memahami apa yang sebenarnya terjadi.

Semarang, Juni 2005

Edhi Nugroho, M.Kom

## **DAFTAR ISI**

KATA PENGANTAR.....	i
DAFTAR ISI.....	ii
BAB 1 DELPHI.....	1
Apa itu Delphi ?.....	1
Lingkungan Delphi.....	1
Code Editor.....	2
Form.....	3
Component Pallete.....	4
Object Inspector.....	5
Watch List.....	5
Tombol-tombol ShortCut.....	6
Bekerja dengan Delphi.....	6
Membuat proyek aplikasi baru.....	6
Menyimpan proyek aplikasi.....	6
Memasukkan komponen ke dalam form.....	7
Mengubah isi properti komponen di Form.....	8
Mengubah lokasi komponen di Form.....	11
Mengubah ukuran komponen di Form.....	11
Membuat event handler.....	11
MEMBUAT APLIKASI KASIR TOKO BUAH.....	13
BAB 2 ARSITEKTUR APLIKASI DATABASE .....	18
Database.....	19
Database local versus database remote.....	21
Transactions.....	21
Arsitektur Aplikasi Pengolah Database.....	22
Koneksi database.....	22
BDE (Borland Database Engine).....	22
dbExpress.....	23
ADO.....	23

Dataset.....	23
Data Source.....	25
User Interface.....	25
BAB 3 MEMBUAT APLIKASI DATABASE MENGGUNAKAN BDE.....	27
MENGGUNAKAN BDE.....	28
Membuat AplikasiBDE1.....	28
Membuat AplikasiBDE2.....	33
BAB 4 MEMBUAT ALIAS DAN MENGATUR KONFIGURASI BDE.....	37
Alias.....	38
Membuat Alias.....	38
BAB 5 STUDI KASUS : SISTEM INFORMASI PERSONALIA SEDERHANA.....	42
Diagram UseCase.....	42
Dokumen Skenario.....	42
Diagram Kelas.....	44
Pembahasan.....	44
BAB 6 MEMBUAT TABEL DAN MENGISI TABEL.....	46
Membuat tabel (Create).....	47
Membuat tabel secara programming.....	49
Mengisi Tabel (Insert).....	57
Menambahkan sub menu berkas.....	57
Mengolah Tabel Golongan.....	58
Menguji input .....	62
Menguji input secara program.....	62
Exceptions.....	63
Menguji input menggunakan event OnExit.....	64
Menguji input menggunakan BeforePost.....	64
Mengolah data Bagian.....	66
Mengolah data Pegawai.....	69
Menguji input menggunakan field dari tabel lain.....	72
BAB 7 Mencari RECORD DAN MENYARING RECORD.....	75
Mencari record.....	76
FindKey.....	76
FindNearest.....	76
Locate.....	77

Lookup.....	77
Membuat aplikasi Presensi.....	78
Menyaring record.....	80
Relasi Master-Detail.....	81
Menyaring record menggunakan properti Filter.....	86
BAB 8 CALCULATED FIELD dan LOOKUP FIELD.....	92
Calculated Field.....	93
Lookup Field.....	93
BAB 9 POSTGRESQL.....	104
Sejarah PostgreSQL.....	105
Memperoleh PostgreSQL.....	106
Instalasi PostgreSQL.....	106
Memasang PostgreSQL .....	107
BAB 10 Structured Query Language (SQL).....	113
SQL.....	114
Komponen SQL.....	114
Data Definition Language (DDL).....	114
CREATE TABLE.....	114
CREATE VIEW.....	115
ALTER TABLE.....	116
DROP TABLE dan DROP VIEW.....	117
Data Manipulation Language (DML).....	117
INSERT.....	117
SELECT.....	117
UPDATE.....	118
DELETE.....	119
BAB 11 MEMBUAT KONEKSI KE REMOTE DATABASE.....	120
Membuat koneksi ke server.....	121
Menggunakan BDE.....	121
Menggunakan ODBC.....	123
Menggunakan dbExpress.....	125
Komponen database sesuai dengan metoda koneksi.....	126
Mengirim query ke server.....	127
BAB 12 MENGGUNAKAN TQUERY.....	128

<u>TDatabase.....</u>	<u>129</u>
<u>TQuery.....</u>	<u>131</u>
<u>Memasukkan data menggunakan TQuery.....</u>	<u>132</u>
<u>Query dengan parameter.....</u>	<u>134</u>
<u>Mengambil record menggunakan TQuery.....</u>	<u>136</u>
<u>Menggunakan konstanta sebagai query.....</u>	<u>140</u>
<u>Mengubah isi record.....</u>	<u>140</u>
<u>Menghapus record.....</u>	<u>143</u>

## **BAB 1 DELPHI**

### **APA YANG AKAN KITA PELAJARI ?**

- Mengenal lingkungan Delphi
- Membuat proyek aplikasi baru
- Menyimpan dan mengambil proyek
- Memasukkan komponen ke dalam form
- Mengatur tampilan komponen visual
- Mengkompilasi dan menjalankan program

**WAKTU LATIHAN : 3 JAM**

## ***Apa itu Delphi ?***

Delphi merupakan Lingkungan Pemrograman Terintegrasi (Integrate Development Environment / IDE). **Delphi bukan bahasa pemrograman**, tetapi perangkat lunak yang menyediakan seperangkat alat (tools) untuk membantu pemrogram dalam menulis program komputer. Lalu, bahasa apa yang digunakan oleh Delphi? Delphi menggunakan Object Pascal sebagai bahasa pemrogramannya. Object Pascal merupakan bahasa Pascal yang diberi tambahan kemampuan untuk menerapkan konsep-konsep OOP (Object Oriented Programming). Seluruh sintak Object Pascal menggunakan aturan yang ada di dalam Pascal, termasuk perintah-perintah dasar seperti *control structures*, *variables*, *array*, dan sebagainya.

Peralatan yang disediakan oleh Delphi memberikan kemudahan bagi pemrogram untuk membuat program secara visual (***visual programming***), *visual programming* adalah metoda dimana sebagian atau keseluruhan program dibuat dengan cara 'menggambar'-kan tampilan / hasil akhir dan kemudian meminta beberapa perangkat untuk membuat kode-kode program berdasarkan gambaran hasil akhir tersebut.

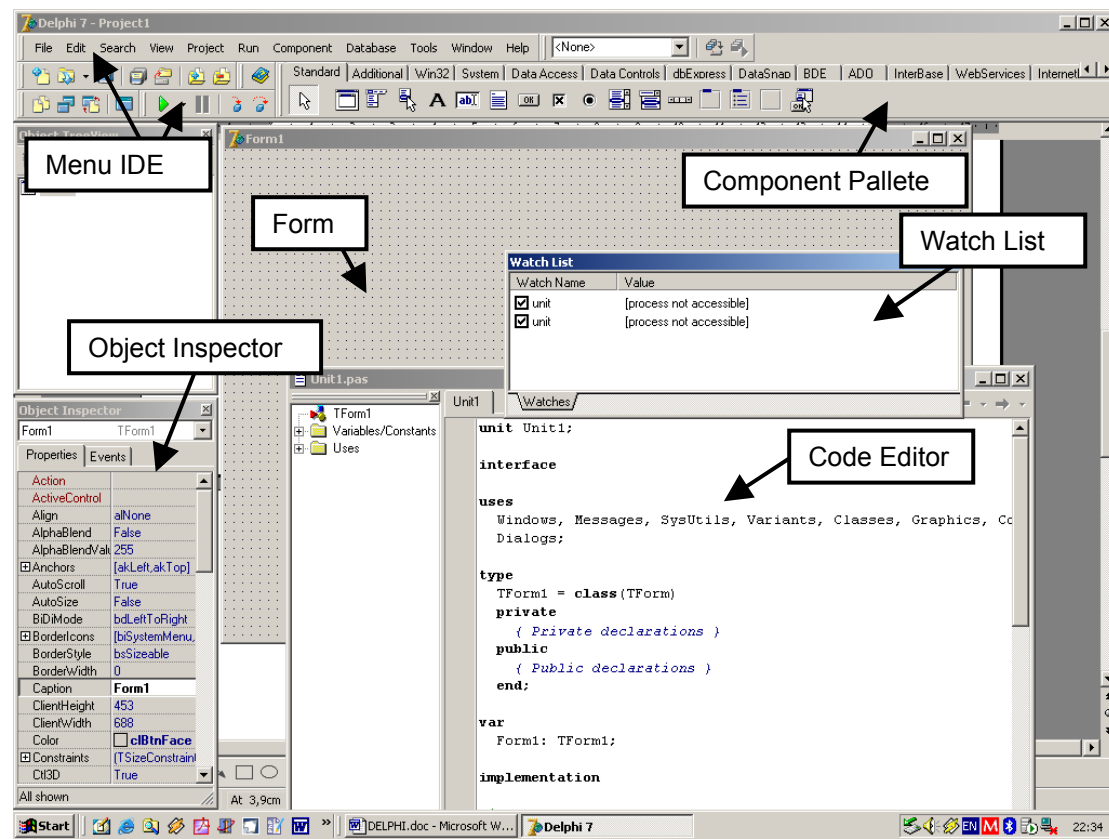
Karena program yang dibuat di dalam Delphi berjalan di dalam sistem operasi Windows maka kegiatan program dilakukan berdasarkan metoda ***event-driven programs***. Event-driven programming adalah metoda mengeksekusi kode program berdasarkan pesan (*messages / events*) yang diberikan oleh pemakai ataupun oleh sistem operasi atau program lainnya. Sebagai contoh : apabila pemakai menekan tombol kiri mouse dan kemudian melepaskannya dengan cepat (kita mengenal itu sebagai 'klik') maka tindakan tersebut akan membuat aplikasi menerima pesan 'MOUSE DOWN' yang disertai dengan informasi tombol mana yang ditekan dan lokasi kursor saat 'klik' dilakukan, tetapi apabila pemakai menekan tombol kiri mouse dan kemudian menggeser mouse tanpa melepaskan tombol kiri maka aplikasi akan menerima pesan 'MOUSE MOVE'.

## ***Lingkungan Delphi.***

Beberapa peralatan yang disediakan oleh Delphi dan cukup diketahui antara lain:

- Code Editor
- Form
- Object Inspector
- Component Pallette
- Project Manager
- Watch List
- Compiler dan Linker
- Debugger
- dan sebagainya

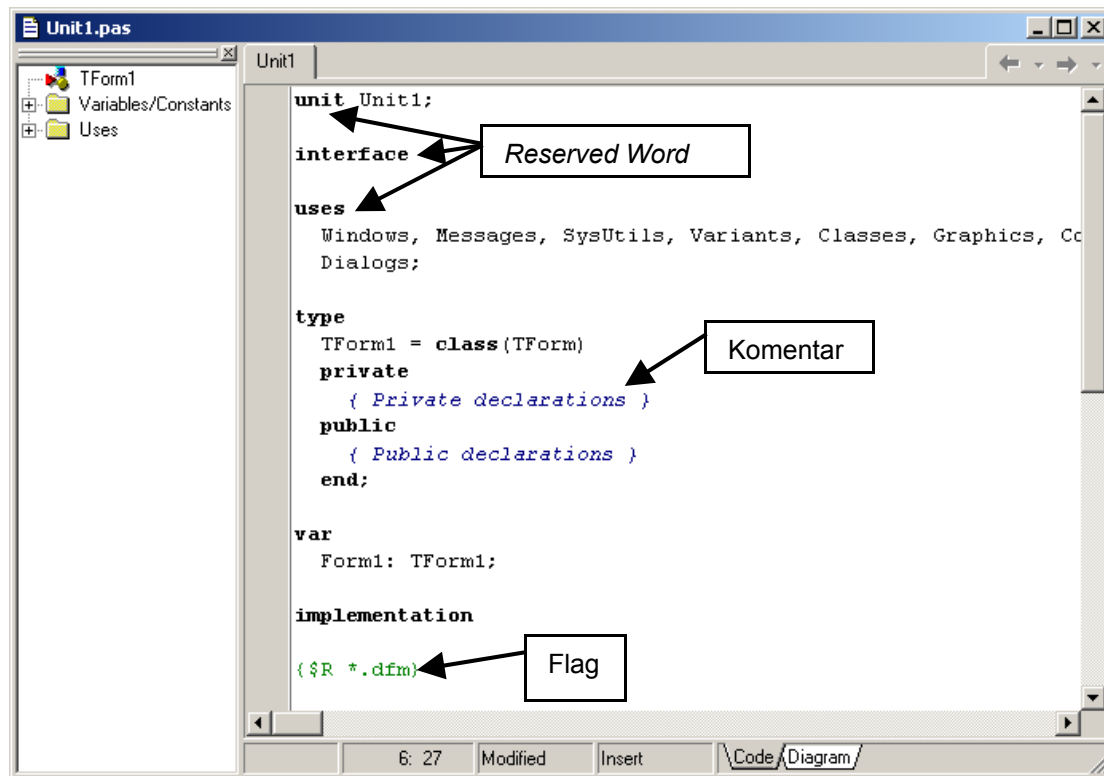
Gambar 1.1 menunjukkan Lingkungan Delphi dan beberapa peralatan yang disediakan oleh Delphi.



Gambar 2.1 Lingkungan Pemrograman Delphi

## Code Editor

Code Editor merupakan peralatan yang digunakan untuk menuliskan kode-kode program. Code Editor menyediakan sejumlah fasilitas penyuntingan (editing) seperti : copy, cut, paste, find, replace, dan sebagainya. Code Editor mengetahui apakah yang ditulis merupakan perintah Object Pascal atau bukan dan menampilkan tulisan sesuai dengan tipe / kelompok tulisan tersebut. Gambar 1.2 memberikan contoh bagaimana Code Editor membedakan tampilan tulisan sesuai dengan kelompok tulisan. File yang berisi kode-kode program disimpan dengan nama akhiran .pas.



**Gambar 2.2 Code Editor**

## Form

Form merupakan area dimana pemrogram meletakkan komponen-komponen input dan output. Delphi akan secara otomatis membuat kode-kode program untuk membuat dan mengatur komponen-komponen tersebut. Umumnya pada setiap aplikasi ada paling tidak satu buah form dan form tersebut dijadikan sebagai form utama (Main Form).

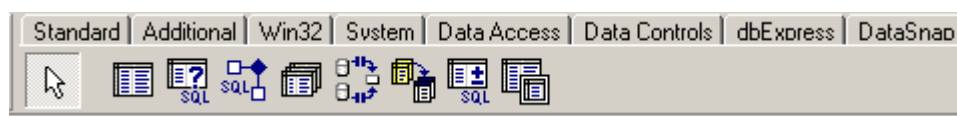
Setiap form selalu disimpan dalam 2 buah file, yaitu : 1) file dengan akhiran .dfm ; 2) file dengan akhiran .pas. File berakhiran .dfm menyimpan informasi mengenai komponen-komponen yang ada di dalam form sedangkan file berakhiran .pas menyimpan informasi mengenai kode-kode program yang berhubungan dengan form tersebut. Tetapi, tidak setiap file .pas selalu mempunyai pasangan .dfm. Gambar 1.3 memberi contoh pemakaian Form.



**Gambar 2.3. Form dan Komponen**

## Component Palette

Component Palette adalah peralatan yang menyediakan daftar komponen yang dapat digunakan oleh pemrogram. Gambar 1.4 memberi ilustrasi dari Component Palette

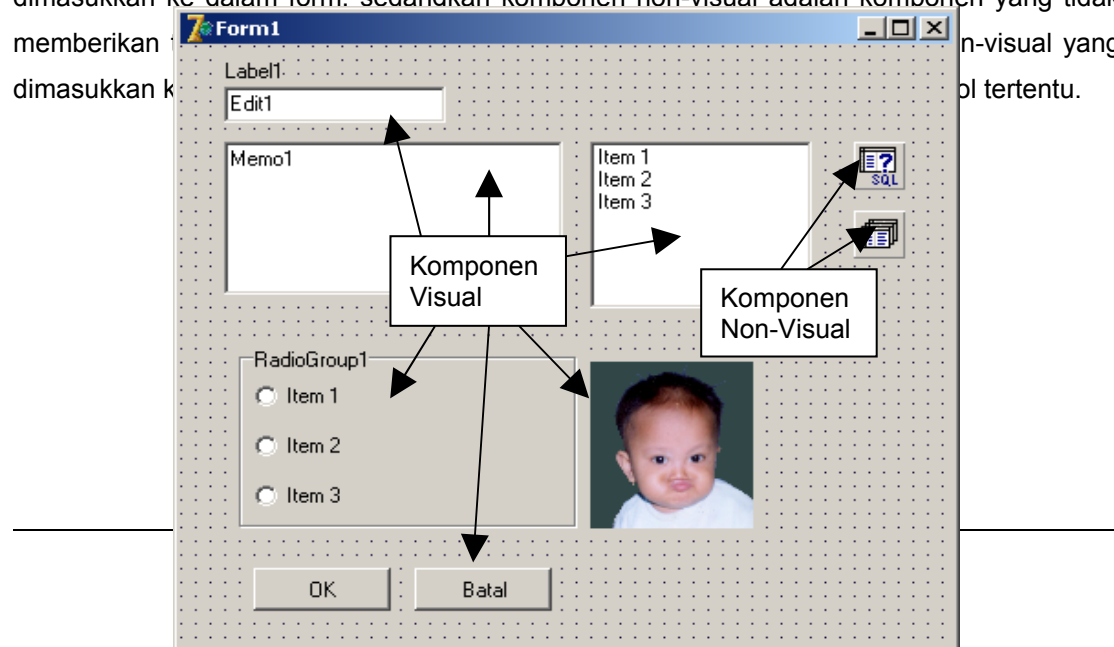


**Gambar 2.4. Component Palette**

Komponen di dalam Delphi dibedakan menjadi dua macam, yaitu :

1. Komponen Visual
2. Komponen Non Visual

Komponen Visual adalah komponen yang memberikan tampilan tertentu pada saat dimasukkan ke dalam form. sedangkan komponen non-visual adalah komponen yang tidak memberikan tampilan tertentu pada saat dimasukkan ke dalam form.



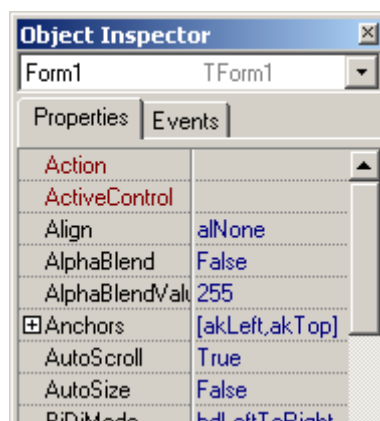
**Gambar 2.7. Komponen Visual dan Non-Visual di Form**

### **Object Inspector**

Object Inspector adalah peralatan yang digunakan untuk mengatur properti dari komponen yang ada di form termasuk properti form. Object Inspector memberi dua macam peralatan, yaitu :

1. Properties
2. Events

Peralatan Properties adalah peralatan yang digunakan untuk mengubah atau mengatur nilai-nilai dari properti komponen sedangkan Peralatan Events digunakan untuk membuat event-handler. Event handler adalah prosedur yang digunakan khusus untuk menanggapi satu event / message tertentu.

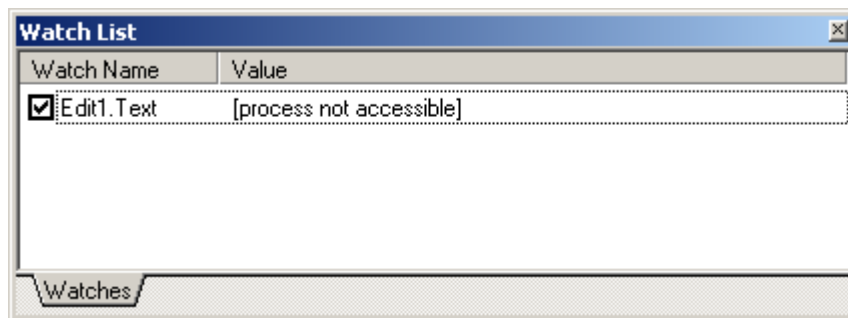


**Gambar 2.8. Object Inspector**

### **Watch List**

Watch List merupakan peralatan yang digunakan untuk memeriksa isi satu variabel atau properti tertentu saat program sedang dieksekusi. Watch List biasanya digunakan bersamaan

dengan Break Points dan Step-by-Step execution. Kita akan mempelajari ini pada akhir pelatihan untuk melihat bagaimana mencari kesalahan di dalam program dengan cepat.



**Gambar 2.9. Watch List**

### Tombol-tombol ShortCut

Untuk berpindah dari satu peralatan ke peralatan lain anda dapat melakukannya dengan mengklik jendela dari peralatan tersebut, tetapi ada cara yang lebih cepat untuk berpindah dari satu peralatan ke peralatan lain. Tabel 1-1 menunjukkan tombol keyboard yang dapat digunakan untuk keperluan tersebut.

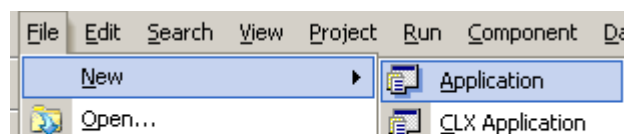
**Tabel 2-1. Tombol Shortcut**

Tombol Keyboard	Keterangan
F12	Pindah dari Form ke Code Editor dan sebaliknya
F11	Pindah ke Object Inspector
F10	Pindah ke menu utama
Shift-F5	Menampilkan Watch List

## Bekerja dengan Delphi

### Membuat proyek aplikasi baru

1. Jalankan Delphi
2. Pilih menu *File | New | Application*



**Gambar 2.10. Menu membuat aplikasi baru**

3. Anda akan memperoleh sebuah form kosong dan sebuah file dengan nama Unit1.pas

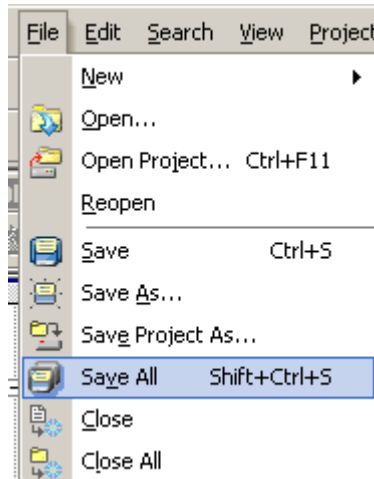
### Menyimpan proyek aplikasi

Delphi menyimpan proyek aplikasi ke dalam beberapa file seperti ditunjukkan pada Tabel 1-1.

**Tabel 2-2. File-file proyek**

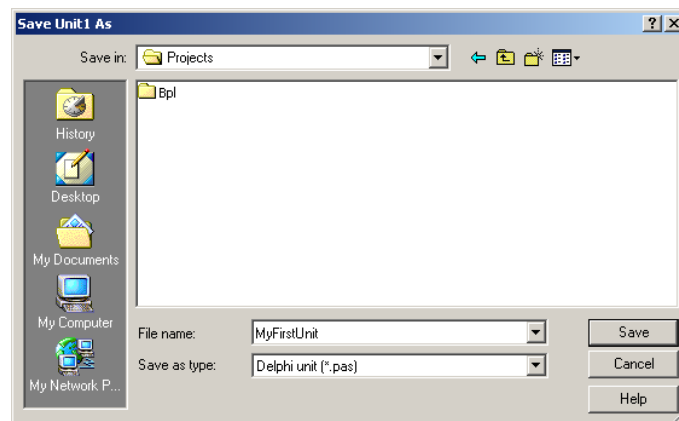
File	Keterangan
.dpr	File berisi keterangan tentang proyek aplikasi
.pas	File berisi kode-kode program yang digunakan dalam aplikasi
.dfm	File berisi informasi tentang form dan komponen-komponen di dalam form tersebut
.res	File berisi data-data resource seperti icon, bitmap dan sebagainya

1. Gunakan menu *File | Save All*.



**Gambar 2.11. Menu Save All**

2. Delphi akan menanyakan nama file .pas untuk menyimpan kode-kode program. Isi dengan MyFirstUnit.pas dan kemudian klik tombol Save.



**Gambar 2.12. Menyimpan file unit**

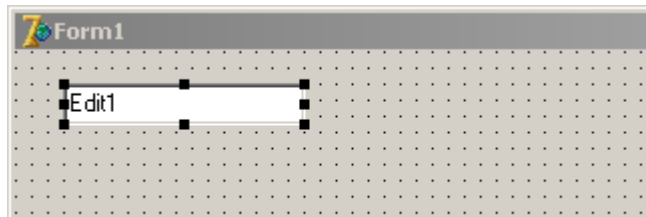
3. Delphi akan menanyakan nama file .dpr yang digunakan untuk menyimpan informasi tentang proyek aplikasi, isi dengan MyFirstProject.dpr dan kemudian klik tombol Save.

### Memasukkan komponen ke dalam form

1. Pindah ke Form dengan mengklik jendela Form atau menekan F12 beberapa kali sampai anda berada di Form.

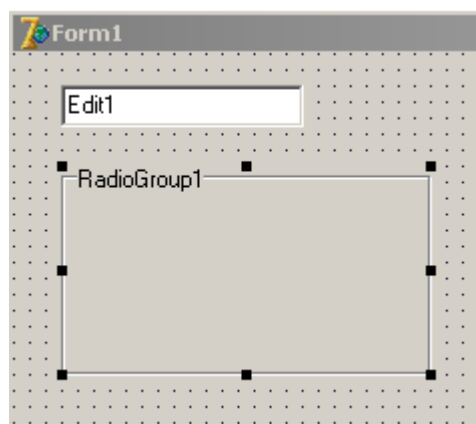


2. Klik komponen TEdit di **Component Palette**.
3. Klik salah satu lokasi di Form untuk meletakkan komponen.



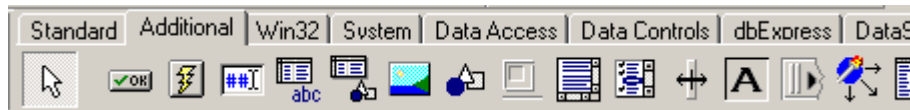
**Gambar 2.13** Memasukkan komponen di Form

4. Klik komponen TRadioGroup di **Component Palette**.
5. Klik salah satu lokasi di Form untuk meletakkan komponen.



**Gambar 2.14** Komponen TRadioGroup

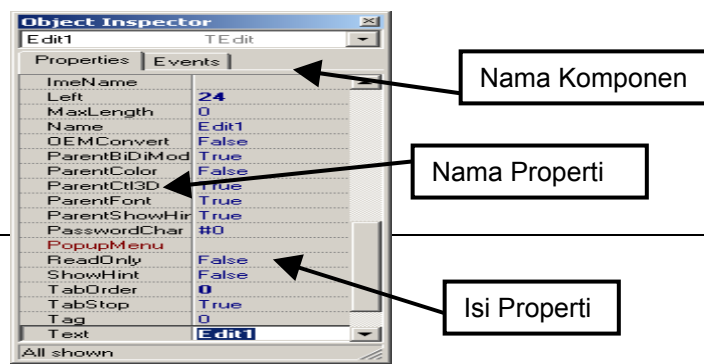
6. Klik tab *Additional* dan anda akan memperoleh komponen-komponen yang berbeda.



**Gambar 2.15** Komponen lain di tab Additional

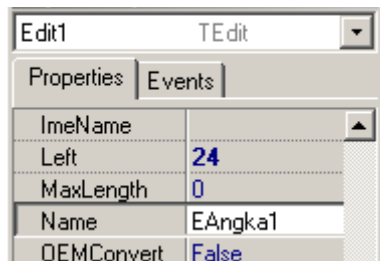
## Mengubah isi properti komponen di Form

1. Klik komponen Edit1 di Form dan kemudian tekan tombol F11. Delphi akan menampilkan Object Inspector dengan isian properti dari Edit1.
2. Bagian properti dari Object Inspector dibagi menjadi dua bagian (Gambar 1.16), yaitu :  
1) Nama Properti ; 2) Isi Properti.



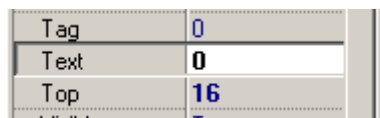
**Gambar 2.16 Object Inspector**

3. Klik bagian isi dari properti Name dan kemudian tulis nama properti EAngka1.





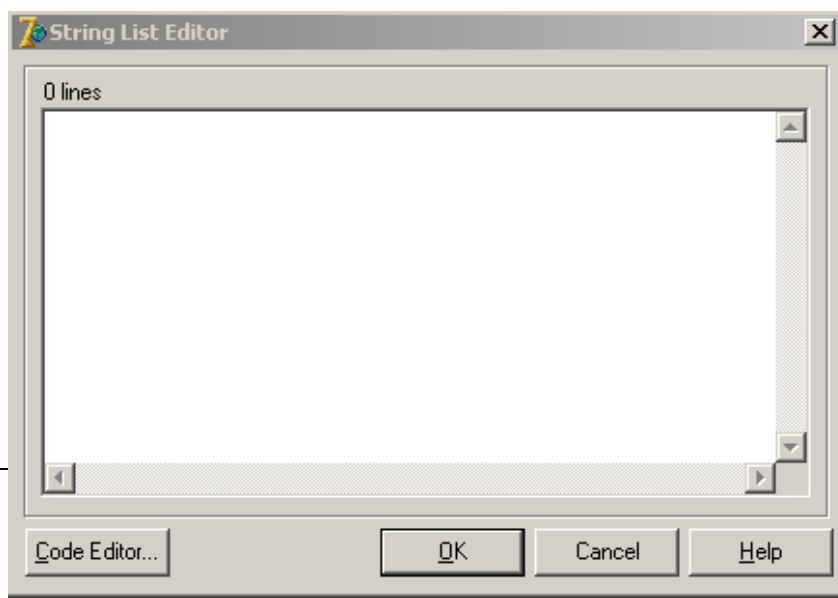
**Gambar 2.17. Mengubah isi properti Name**

4. Geser scrollbar di Object Inspector sampai anda menemukan properti Text. Klik properti Text dan isikan angka 0.



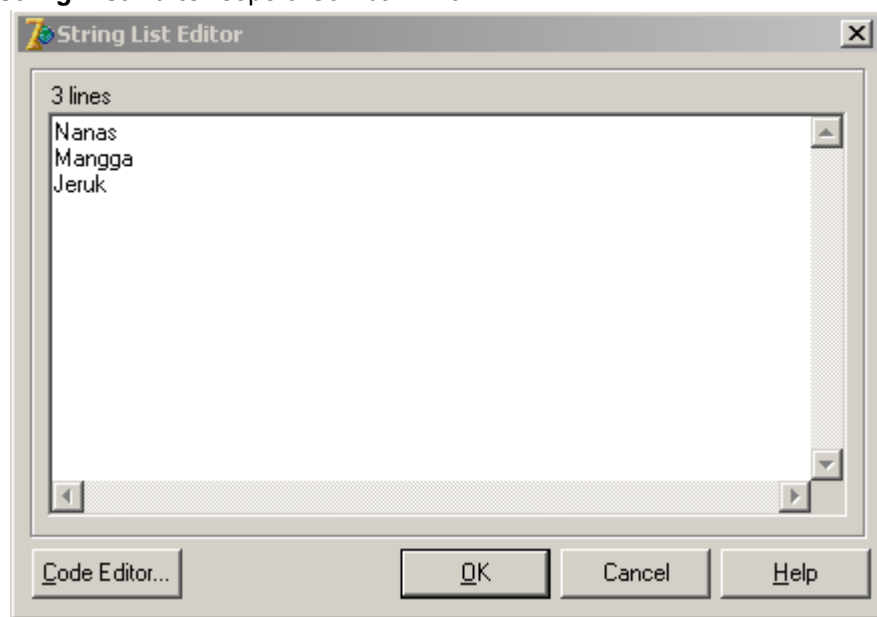
**Gambar 2.18 Isi properti Text**

5. Mengubah isi properti tidak selalu dilakukan langsung di Object Inspector, tergantung dari tipe dari properti tersebut Object Inspector akan menampilkan editor properti yang sesuai.
6. Klik komponen RadioGroup1 dan kemudian tekan tombol keyboard F11.
7. Klik properti Items. Anda tidak dapat mengubah isi properti Items di Object Inspector karena Items bertipe TStrings. Untuk mengubah isi properti Items klik tombol  .. Delphi akan  menampilkan String List Editor seperti Gambar 1.19.



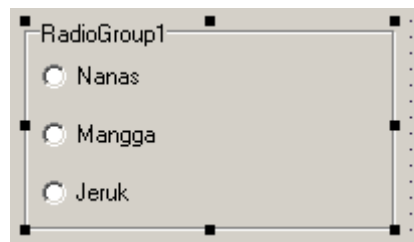
**Gambar 2.19 String List Editor untuk mengisi properti Items**

8. Isi **String List Editor** seperti Gambar 1.20.



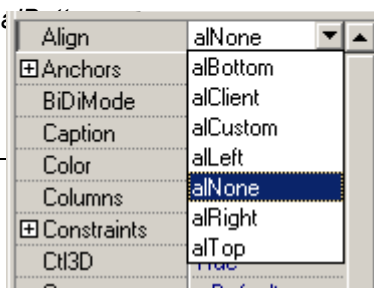
**Gambar 2.20 Isi properti Items**

9. Klik tombol **OK** di String List Editor dan Delphi akan menampilkan RadioGroup1 seperti Gambar 1.21.



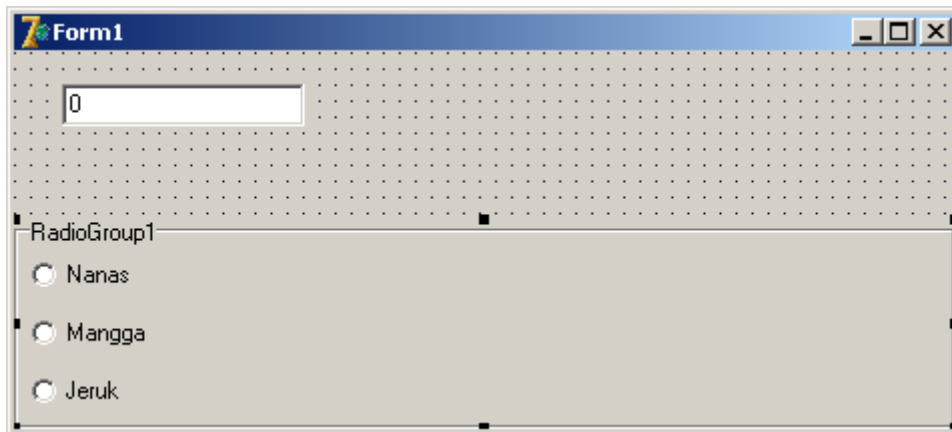
**Gambar 2.21 Isi RadioGroup1**

10. Geser **scrollbar** di **Object Inspector** sampai anda menemukan properti Align. Klik isi properti **Align** dan pilih **alNone**.



**Gambar 2.22 Properti Editor dari properti Align**

11. Perhatikan sekarang lokasi RadioGroup1 berubah ke bawah.



**Gambar 2.23 Lokasi RadioGroup1**

### **Mengubah lokasi komponen di Form**

1. Klik komponen Edit1 di Form.
2. Tanpa melepas tombol kiri mouse, geser mouse anda ke sembarang arah. Komponen akan mengikuti arah mouse.
3. Lepaskan tombol kiri mouse dan komponen akan berpindah tempat ke lokasi baru.
4. Selain cara tersebut, anda dapat mengubah isi properti Left dan Top untuk mengubah lokasi komponen di Form

### **Mengubah ukuran komponen di Form**

1. Masih menggunakan Edit1 di Form, klik kotak hitam yang ada di pinggir Edit1 dan kemudian tanpa melepas tombol kiri mouse, geser mouse sehingga ukuran Edit1 berubah.
2. Selain menggunakan cara tersebut, anda juga dapat mengubah isi properti Height dan Width untuk mengubah ukuran komponen.

### **Membuat event handler**

Pada latihan ini kita akan membuat event handler untuk menangani pemilihan items pada RadioGroup1. Setiap kali Item di RadioGroup1 dipilih maka kita akan mengisi properti Text dari EAngka1 dengan kata 'Nanas', 'Mangga' atau 'Jeruk' sesuai dengan item yang dipilih. Event yang berkaitan dengan pemilihan items di RadioGroup1 adalah event OnClick. Item

yang diklik oleh pemakai dapat dilihat di properti RadioGroup1.ItemIndex, dengan 0 (nol) menunjukkan item pertama. Algorithma event handler OnClick ditentukan seperti Listing 1.1.

#### Listing 2-1 Algorithma event handler OnClick

```
Jika RadioGroup1.ItemIndex = 0 maka  
  Isi EAngka1.Text dengan 'Nanas'  
tetapi Jika RadioGroup1.ItemIndex = 0 maka  
  Isi EAngka1.Text dengan 'Mangga'  
Jika RadioGroup1.ItemIndex = 0 maka  
  Isi EAngka1.Text dengan 'Jeruk'
```

1. Klik komponen RadioGroup1 dan tekan F11 untuk menampilkan ObjectInspector.
2. Klik tab Events
3. Double click isian dari event OnClick, Delphi akan membawa anda ke Code Editor dan membuat sebuah prosedur seperti Listing 1.2.

#### Listing 2-2. Event handler onClick yang dibuat oleh Delphi

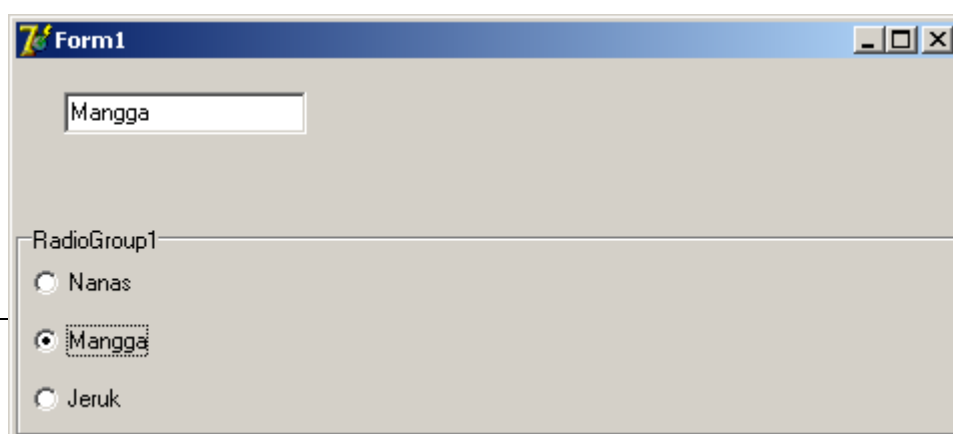
```
procedure TForm1.RadioGroup1Click(Sender: TObject);  
begin  
  
end;
```

4. Isi event handler OnClick pada Listing 1.2 menjadi seperti Listing 1.3.



#### Listing 2-3. Isi event handler OnClick dari RadioGroup1

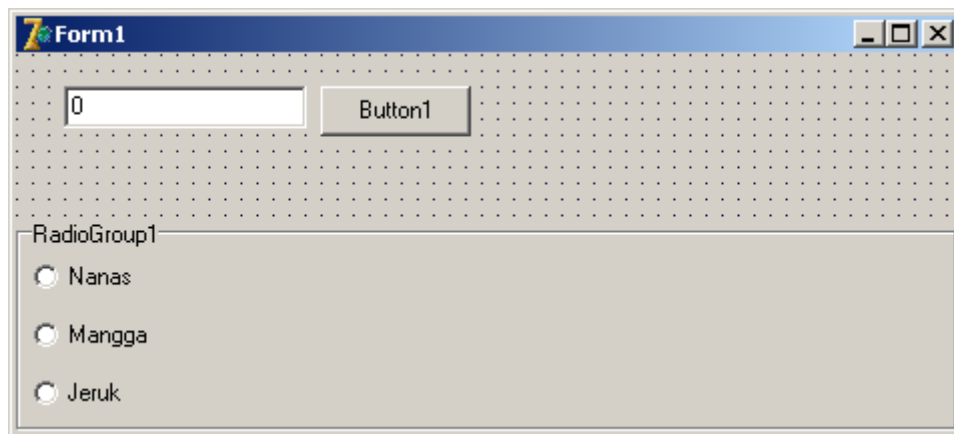
```
procedure TForm1.RadioGroup1Click(Sender: TObject);  
begin  
  if RadioGroup1.ItemIndex = 0 then  
    EAngka1.Text := 'Nanas'  
  else if RadioGroup1.ItemIndex = 1 then  
    EAngka1.Text := 'Mangga'  
  else if RadioGroup1.ItemIndex = 2 then  
    EAngka1.Text := 'Jeruk';  
end;
```

5. Jalankan program dengan menekan tombol F9 atau mengklik tombol 



**Gambar 2.24 Hasil program**

6. Klik tombol  untuk menutup program.
7. Tambahkan komponen Button  ke dalam Form1 sehingga anda memperoleh Gambar 1.25. dan atur agar properti dari tombol tersebut seperti Tabel 1.



**Gambar 2.25 Penambahan komponen Button**

**Tabel 2-3 Properti Button1**

Properti	Isi
Name	KeluarBtn
Caption	Keluar

8. Buat event handler OnClick untuk tombol KeluarBtn seperti pada Listing 1.4.

#### **Kasus**

Tulis program untuk menangani pembayaran pembelian buah di Toko 'Fruit'. Setiap kali pembeli membeli buah maka kasir akan memasukkan :

- Berat buah yang dibeli dalam kilogram
- Jenis buah

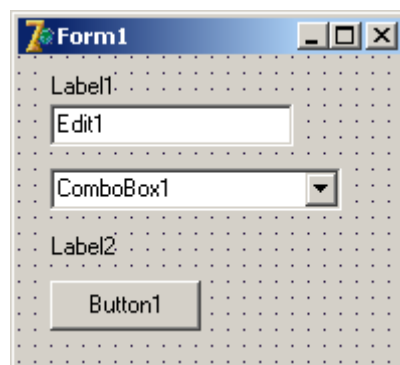
Program kemudian menghitung harga yang harus dibayar oleh pembeli berdasarkan berat dan jenis buah yang dibeli. Harga satuan per kg untuk tiap jenis buah seperti pada Tabel 1.

**Tabel 2-4 Harga satuan per kg buah**

Buah	Harga per kg
Jeruk	5.000
Mangga	15.000
Durian	20.000
Pepaya	3.000
Anggur	25.000
Apel	15.000

Jawab :

1. Buat aplikasi baru.
2. Masukkan komponen TEdit, TComboBox dan TButton seperti pada Gambar 1.26. Atur agar properti dari masing-masing komponen seperti pada Tabel 1.5.



**Gambar 2.26 . Tata letak program kasir**

**Tabel 2-5 Properti dari komponen-komponen program kasir**

Komponen : Label1

Properti	Isi
Caption	Berat (kg)

Komponen : Edit1

Properti	Isi
Text	0

Komponen : ComboBox1

Properti	Isi
Items	Jeruk Mangga Durian Pepaya Anggur Apel
Style	csDropDownList

Komponen : Button1

Properti	Isi
Caption	Hitung

3. Buat event handler OnClick dari Hitung seperti pada Listing 1.5

#### Listing 2-5 Event handler OnClick dari Button1

```
procedure TForm1.Button1Click(Sender: TObject);
var
  berat,harga,bayar : integer;
  jenis : integer;
begin
  //ubah string ke integer
  berat := StrToInt(Edit1.Text);
  //ambil jenis buah yang dibeli
  jenis := ComboBox1.ItemIndex;
  case jenis of
    0 : harga :=5000;
    1 : harga :=15000;
    2 : harga :=20000;
    3 : harga :=3000;
    4 : harga :=25000;
    5 : harga :=15000;
  end;
  bayar:= berat * harga;
  //tampilkan di Label2
  Label2.Caption := IntToStr(bayar);
end;
```

4. Kompilasi dan kemudian jalankan program, cobalah mengisi berat dan memilih salah satu jenis buah. Hasil program diperlihatkan di Gambar 1.27.

The screenshot shows a simple Windows application window with a title bar that says 'Form1'. Inside the window, there are three main components: a text box at the top labeled 'Berat (kg)' containing the number '20', a dropdown menu below it with 'Jeruk' selected, and a label at the bottom left showing the calculated value '100000'. A button labeled 'Hitung' is positioned at the bottom center of the window.

### Gambar 2.27 Hasil program Kasir

Tampilan hasil perhitungan pembayaran dari program Kasir yang kita buat tidak terlalu bagus, kita akan mencoba memformat tampilan pembayaran agar muncul : "Pembayaran : Rp 100.000".

5. Ubah event handler OnClick dari Button1 sehingga seperti Listing 1.6

### Listing 2-6 Memformat tampilan pembayaran

```
procedure TForm1.Button1Click(Sender: TObject);
var
  berat,harga: integer;
  bayar:real;
  jenis : integer;
begin
  //ubah string ke integer
  berat := StrToInt(Edit1.Text);
  //ambil jenis buah yang dibeli
  jenis := ComboBox1.ItemIndex;
  case jenis of
    0 : harga :=5000;
    1 : harga :=15000;
    2 : harga :=20000;
    3 : harga :=3000;
    4 : harga :=25000;
    5 : harga :=15000;
  end;
  bayar:= berat * harga;
  //tampilkan di Label2
  Label2.Caption := Format('Pembayaran : %4.0m', [bayar]);
end;
```

6. Hasil program ditunjukkan pada Gambar 1.28.



The screenshot shows a simple Windows application window titled "Form1". Inside the window, there is a label "Berat (kg)" above a text input field containing the number "20". Below this is a dropdown menu currently displaying "Jeruk". Underneath the dropdown is a label showing the calculated result: "Pembayaran : Rp100.000". At the bottom of the window is a button labeled "Hitung".

**Gambar 2.28 Hasil program setelah dilakukan format**

## **BAB 2 ARSITEKTUR APLIKASI DATABASE**

### **APA YANG AKAN KITA PELAJARI ?**

- Konsep database, record, field, relasi tabel, database lokal, database remote
- Arsitektur aplikasi pengolah database
- Metoda-metoda koneksi bekerja

**WAKTU LATIHAN : 2 JAM**

## Database

Database adalah sekumpulan **file** atau **tabel** yang saling berhubungan satu sama lain. Tiap file / tabel terdiri atas sekumpulan data yang mempunyai karakteristik berbeda. Tiap kumpulan data dengan karakteristik berbeda tetapi mempunyai kesatuan arti disebut **record** sedangkan data terkecil dalam record disebut sebagai **field**. Tiap field selalu mempunyai informasi tentang :

- Nama field
- Tipe
- Panjang

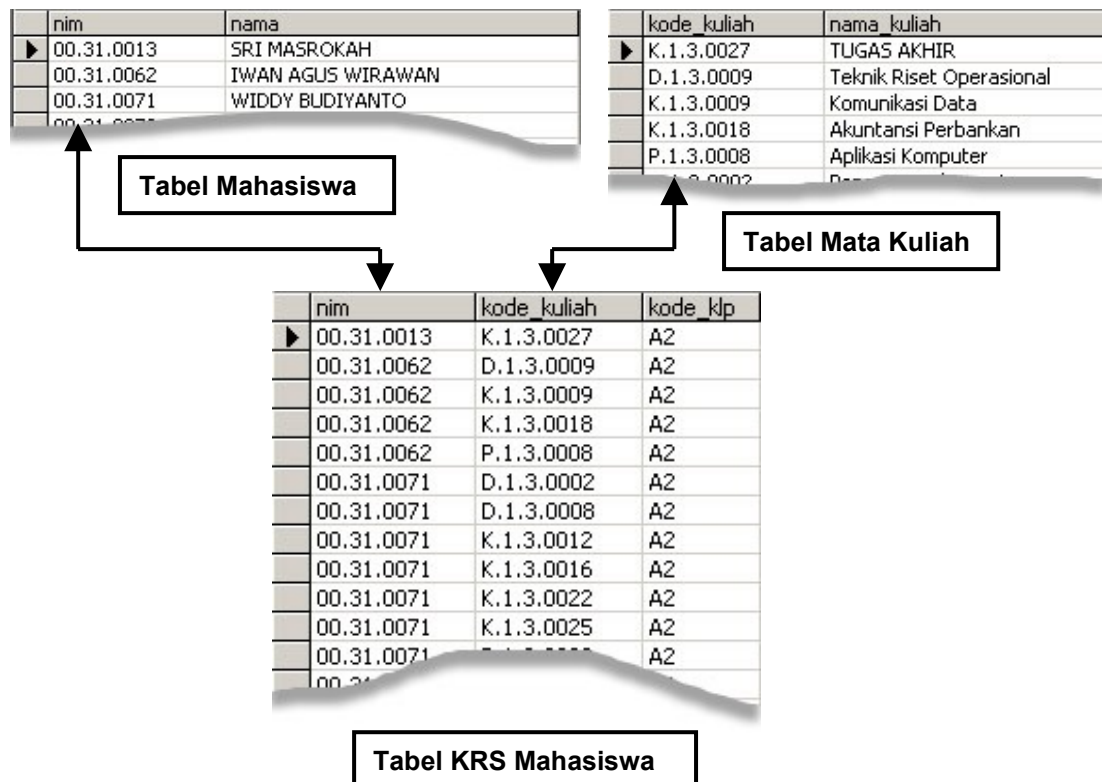
Bergantung kepada database yang anda gunakan, tiap field juga dapat mempunyai informasi lain seperti : nilai baku (default value), batasan / aturan nilai yang dapat disimpan (constraint / rule) dan sebagainya. Field yang menjadi pembeda antara satu record dengan record lain dalam satu tabel disebut sebagai field kunci (**primary key**). Gambar 2.1 memperlihatkan data krs mahasiswa serta bagian-bagian yang disebut record dan field.

nim	nama	kode_kuliah	nama_kuliah
00.31.0013	SRI MASROKAH	K.1.3.0027	TUGAS AKHIR
00.31.0062	IWAN AGUS WIRAWAN	D.1.3.0009	Teknik Riset Operasional
00.31.0062	IWAN AGUS WIRAWAN	K.1.3.0009	Komunikasi Data
00.31.0062	IWAN AGUS WIRAWAN	K.1.3.0018	Akuntansi Perbankan
00.31.0062	IWAN AGUS WIRAWAN	P.1.3.0008	Aplikasi Komputer
00.31.0071	WIDDY BUDIYANTO	D.1.3.0002	Pengantar Ekonomi
00.31.0071	WIDDY BUDIYANTO	D.1.3.0008	Perilaku Dalam Organisasi
00.31.0071	WIDDY BUDIYANTO	K.1.3.0012	Pemrograman Terstruktur
00.31.0071	WIDDY BUDIYANTO	K.1.3.0016	Statistik Deskriptif
00.31.0071	WIDDY BUDIYANTO	K.1.3.0022	Bahasa Inggris 2
00.31.0071	WIDDY BUDIYANTO	K.1.3.0025	Pengantar Bisnis
00.31.0071	WIDDY BUDIYANTO	K.1.3.0027	Paket Program Niaga
00.31.0071	WIDDY BUDIYANTO	K.1.3.0028	Teknik Riset Operasional

The diagram includes labels and arrows: 'Nama Field' points to the header row; 'record' points to a single row; 'field' points to a single column.

Gambar 3.29. Tabel Pengambilan Kuliah

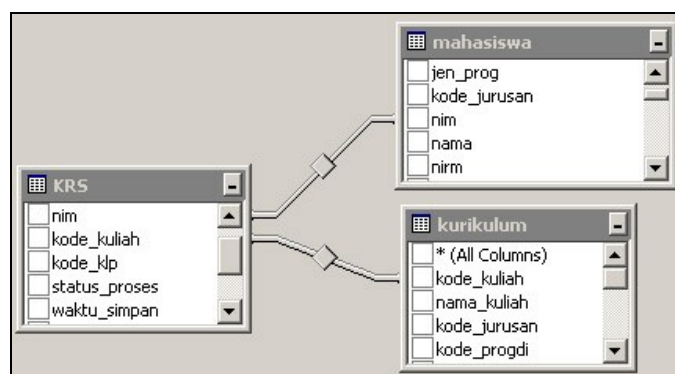
Perhatikan bahwa setiap mahasiswa dapat mengambil lebih dari satu mata kuliah dan satu mata kuliah dapat diambil oleh beberapa mahasiswa, apabila struktur tabel disimpan dengan bentuk seperti Gambar 2.1 maka akan ada beberapa record yang mempunyai informasi yang sama, misalnya informasi mengenai pengambilan mata kuliah dari mahasiswa **Iwan Agus Irawan**. Untuk lebih mengefisienkan tempat penyimpanan maka data pengambilan mata kuliah tersebut disimpan ke dalam tiga buah tabel, yaitu tabel mahasiswa, tabel krs dan tabel mata kuliah, seperti ditunjukkan pada Gambar 2.1.



**Gambar 3.30 Tabel untuk menyimpan KRS Mahasiswa**

Dengan menggunakan tabel-tabel seperti Gambar 2.2 maka penyimpanan data dapat dilakukan dengan lebih efisien karena data tentang mata kuliah dan mahasiswa tidak perlu diulang-ulang dan hanya data nim, kode kuliah serta kelompok yang diulang-ulang. Tindakan semacam itu disebut sebagai **Normalisasi**, saya tidak akan membicarakan tentang bagaimana melakukan normalisasi, anda yang tidak tahu tentang normalisasi silahkan membaca dan mempelajari buku-buku tentang perancangan *database relational*.

Dari Gambar 2.2 terlihat bahwa antara satu tabel dengan tabel lain saling mempunyai relasi, relasi tersebut dapat digunakan untuk memperoleh informasi di tabel lain berdasarkan informasi yang ada di tabel tertentu. Database yang disusun dari relasi tabel-tabel didalamnya disebut sebagai **Relational Database**. Secara bagan, relasi antar tabel dari pengambilan mata kuliah dapat digambarkan seperti Gambar 2.3.

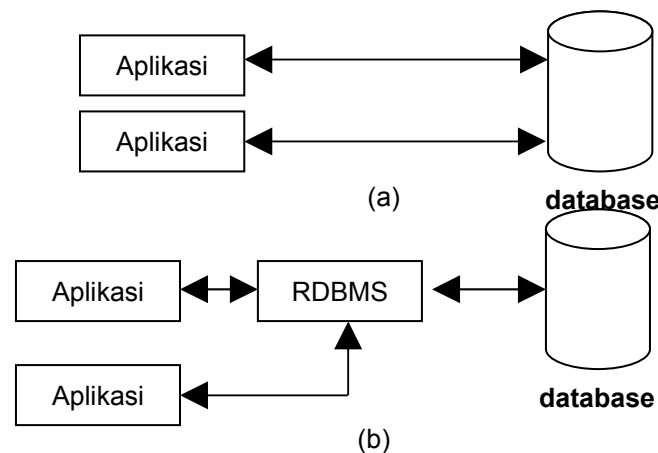


**Gambar 3.31 Relasi Tabel**

## Database local versus database remote

*Database local* adalah database yang disimpan di drive komputer lokal atau di drive jaringan lokal, umumnya aplikasi juga dijalankan di komputer yang sama meskipun tidak selalu terjadi. Karena aplikasi, terlepas dijalankan di komputer yang sama atau di komputer lain, mengakses langsung file data (tabel) yang digunakan maka *database local* juga sering disebut sebagai **file-based databases**. Aplikasi yang menggunakan database lokal sering disebut sebagai **single-tiered applications** karena aplikasi dan database menggunakan bersama-sama sistem pengelolaan file.

*Database remote* adalah database yang disimpan di komputer yang terpisah dengan aplikasi, seringkali bahkan data yang ada di *remote database server* tidak berada di satu mesin tetapi disebarkan pada beberapa mesin lain. Di komputer yang menyimpan database dijalankan satu aplikasi yang menangani secara khusus pengelolaan database dan disebut sebagai *remote database management system* (RDBMS), aplikasi yang membutuhkan data akan berhubungan dengan RDBMS dan tidak mempunyai akses langsung ke data.



Gambar 3.32 (a) Database local ; (b) Database remote

Aplikasi yang menggunakan database remote sering disebut sebagai **two-tiered applications** atau **multi-tiered applications** karena aplikasi dan database berada di mesin yang terpisah. Aplikasi berkomunikasi dengan RDBMS melalui bahasa SQL (*Structured Query Language*).

## Transactions

*Transactions* merupakan sejumlah kegiatan yang harus dilakukan dengan sukses sebelum akibat kegiatan tersebut menjadi permanen (*committed*). Apabila salah satu dari kegiatan tersebut gagal maka seluruh transaksi akan dibatalkan (*roll back*).

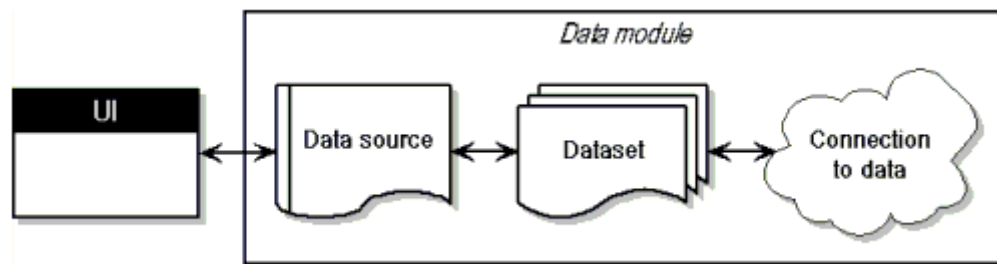
Tidak semua database mendukung *transactions* meskipun database tersebut termasuk *remote database server*, sebagai contoh database MySQL tidak mendukung pemakaian *transactions* sementara MSSQL mendukung pemakaian *transactions*.

## **Arsitektur Aplikasi Pengolah Database**

Aplikasi yang mengolah database disusun dari sejumlah komponen, yaitu :

- Koneksi database
- Data Module
- Dataset
- Data Source
- User Interface (UI)

Secara umum, hubungan antar masing-masing komponen diperlihatkan pada Gambar 2.5.



**Gambar 3.33 Hubungan antar elemen aplikasi pengolah database**

## **Koneksi database**

Delphi menyediakan berbagai cara untuk menghubungkan aplikasi anda ke database yang digunakan oleh aplikasi, metoda-metoda yang dapat digunakan oleh aplikasi untuk berhubungan dengan database antara lain :

- BDE (Borland Database Engine)
- dbExpress
- ADO

## **BDE (Borland Database Engine)**

BDE merupakan metoda yang didukung dan dikembangkan oleh Borland sendiri. BDE menyediakan sejumlah API untuk berhubungan dengan database. Koneksi database menggunakan BDE bersifat transparan, dimana aplikasi menggunakan satu prosedur yang sama untuk mengelola database lokal maupun database remote, bahkan aplikasi tidak perlu diubah / disesuaikan apabila terjadi perubahan terhadap model database yang digunakan.

BDE menyediakan sejumlah driver untuk berhubungan dengan berbagai macam database baik database lokal seperti Paradox, Access, DBase dan sebagainya, maupun database remote seperti MSSQL, Interbase, maupun Oracle. BDE juga dapat menggunakan driver ketiga melalui ODBC (*Open Database Connectivity*) dalam mengakses database, ini berarti BDE dapat menggunakan database apa saja asalkan paling tidak database tersebut menyediakan driver ODBC. Pada umumnya setiap komputer yang menggunakan sistem

operasi Windows sudah mempunyai driver ODBC untuk sistem database yang umum digunakan.

Hubungan antara aplikasi terhadap database dilakukan melalui pemakaian **Alias**. Alias adalah nama sebutan yang digunakan untuk menyatakan konfigurasi koneksi ke database tertentu, dengan mengubah konfigurasi pada alias tersebut maka aplikasi dapat diarahkan ke database lain atau bahkan berubah dari pemakaian database lokal menjadi database remote. Aplikasi database yang menggunakan BDE hanya dapat dijalankan di komputer yang mempunyai BDE, itu berarti anda harus melakukan pemasangan (*instalasi*) BDE pada tiap komputer.

### **dbExpress**

dbExpress merupakan sejumlah driver ringan (*lightweight*) berupa file .dll yang dapat digunakan untuk menghubungkan aplikasi ke database. Aplikasi yang menggunakan dbExpress cukup diinstall bersama dengan driver dari database yang digunakan, dengan demikian ukuran keseluruhan aplikasi serta metoda pemasangan menjadi lebih sederhana. dbExpress mempunyai beberapa keterbatasan, yaitu :

- Hanya dapat digunakan untuk remote database server.
- Hanya mendukung unidirectional datasets. Ini berarti fungsi penelusuran record sangat dibatasi, anda hanya dapat maju ke record berikutnya dan tidak dapat mundur ke record sebelumnya.
- Tidak dapat digunakan untuk mengubah isi database secara langsung, meskipun anda dapat mengubah isi database melalui perintah SQL UPDATE atau menggunakan dataset khusus untuk dbExpress.
- Tidak ada fasilitas untuk menyaring record.
- Tidak ada fasilitas untuk melakukan *lookup field*.

Tetapi, diluar semua keterbatasan tersebut, dbExpress menyediakan implementasi yang ringan dan tidak membutuhkan sumber daya besar serta akses database yang cepat.

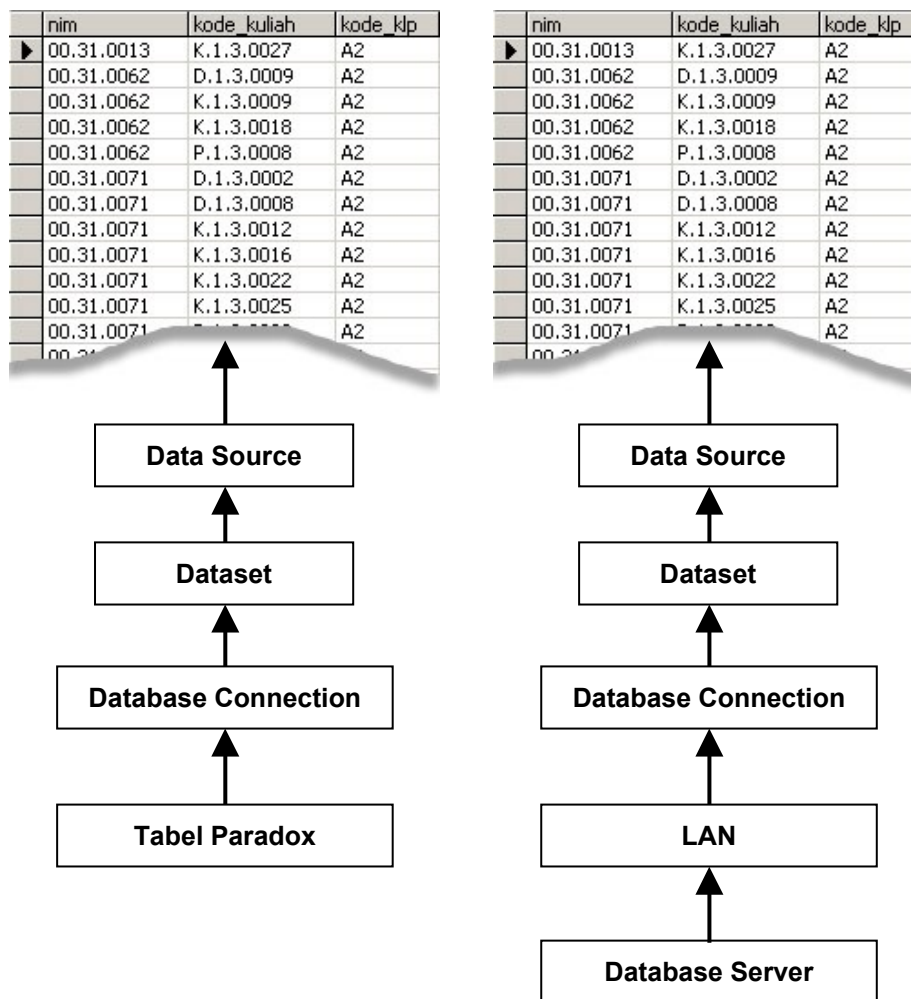
### **ADO**

ADO (*ActiveX Data Objects*) merupakan sejumlah komponen COM berupa file .dll yang memungkinkan aplikasi berhubungan dengan database melalui provider OLE DB. ADO merupakan bawaan dari sistem operasi Microsoft dan semua sistem operasi Windows sudah mempunyai ADO kecuali Windows 95 dan Windows NT. Agar kedua sistem operasi tersebut mempunyai komponen ADO maka anda secara terpisah harus memasang MDAC (*Microsoft Database Access Components*).

### **Dataset**

Dataset merupakan jantung dari aplikasi database. Dataset merepresentasikan record dari database yang sedang diakses. Melalui komponen dataset maka aplikasi selalu melihat

record sebagai sesuatu yang nampak sama meskipun secara fisik mereka sebenarnya berbeda. Gambar 2.6 memberikan ilustrasi bagaimana aplikasi melihat record.



**Gambar 3.34. Dataset**

Dari Gambar 2.6 terlihat bahwa record yang diolah oleh aplikasi tidak tampak berbeda meskipun sebenarnya secara fisik maupun konsep Paradox dan Database Server menggunakan metoda yang berbeda dalam menyimpan data.

Melalui komponen dataset, aplikasi dapat melakukan navigasi terhadap record-record yang sedang diolah, mengubah isi record (bergantung kepada metoda koneksi yang digunakan), membatasi apa yang ditampilkan oleh record, menghapus record dan sebagainya. Komponen dataset yang digunakan bergantung kepada metoda koneksi, Tabel 2.1 memberi petunjuk pemakaian dataset sesuai dengan metoda koneksi.

**Tabel 3-6 Komponen Dataset dan Metoda Koneksi**

Koneksi	Dataset
BDE	TTable
	TQuery
	TStoredProc

	TNestedTable
ADO	TADOTable
	TADOQuery
	TADOStoredProc
dbExpress	TSQLDataSet
	TSQLTable
	TSQLQuery
	TSQLStoredProc

### **Data Source**







Data source berfungsi sebagai perantara bagi user interface (antar muka) dan dataset yang merupakan representasi dari informasi di database. Satu data source dapat digunakan bersama-sama oleh beberapa komponen data control, dengan isian masing-masing data control akan selalu sinkron.







### **User Interface**

Merupakan hal yang baik apabila kita memisahkan antarmuka di form dengan bagian dari aplikasi yang berhubungan dengan database. Pemisahan ini akan memberikan beberapa keuntungan, antara lain : kelenturan perancangan, perubahan cara mengakses database tidak akan mengubah antarmuka bagi pemakai, demikian halnya, mengubah antarmuka tidak perlu mengubah cara mengakses database.

Komponen-komponen yang digunakan sebagai antarmuka disebut sebagai **Data Controls**. Tabel 2.2 memberikan penjelasan beberapa fungsi dari komponen data controls yang disediakan oleh Delphi.

**Tabel 3-7 Komponen Data Controls**

Komponen	Icon	Keterangan
DBGrid		Menampilkan isi tabel dalam bentuk grid (tabel).
DBNavigator		Tombol Navigasi
DBText		Menampilkan isi field bertipe string
DBEdit		Menampilkan isi field dalam bentuk Edit Box.
DBMemo		Menampilkan isi field dalam bentuk Memo
DBImage		Menampilkan isi field berupa gambar / image

DBListBox		Menampilkan record untuk field tertentu dalam bentuk daftar / list box
DBComboBox		Menampilkan record untuk field tertentu dalam bentuk daftar tarik / combo box
DBCheckBox		Menampilkan record untuk field tertentu dalam bentuk check box
DBRadioGroup		Menampilkan record untuk field tertentu dalam bentuk daftar tombol radio / radio button
DBLookupListBox		Menampilkan record untuk field tertentu dalam bentuk daftar / listbox berdasarkan field lain (lookup).
DBLookupComboBox		Menampilkan record untuk field tertentu dalam bentuk daftar tarik / combo box berdasarkan field lain.
DBRichEdit		Menampilkan isi field dalam bentuk rich edit.
DBCtrlGrid		Komponen seperti TPanel tetapi untuk mengatur tampilan masing-masing record dengan berbeda.
DBChar		Menampilkan isi tabel dalam bentuk grafik.

## **BAB 3 MEMBUAT APLIKASI DATABASE MENGUNAKAN BDE**

### **APA YANG AKAN KITA PELAJARI ?**



- Menggunakan BDE untuk koneksi database lokal.
- Menggunakan Data Controls bersama BDE.
- Menggunakan DBNavigator.

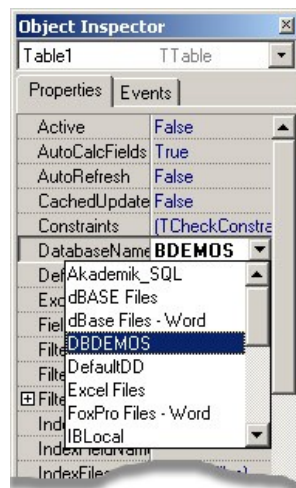
**WAKTU LATIHAN : 2 JAM**

## MENGGUNAKAN BDE

Anda mungkin sudah tidak sabar untuk mencoba membuat aplikasi database. Bab ini akan menunjukkan kepada anda bagaimana membuat aplikasi database dalam bentuk yang paling sederhana menggunakan BDE. Latihan berikut ini mensyaratkan anda sudah menginstall Delphi 7 secara lengkap termasuk demo program bawaan dari Delphi 7.

### Membuat AplikasiBDE1

1. Buat aplikasi baru.
2. Pilih tab Data Access di Component Palette dan kemudian masukkan komponen DataSource (DB)  ke form.
3. Pilih tab BDE dan kemudian masukkan komponen Table (DBTables)  ke form.
4. Pilih Table1 yang ada di form, kemudian aktifkan Object Inspector (F11) dan klik properti editor dari DatabaseName, pilih DBDemos (Gambar 3.1). Properti DatabaseName digunakan untuk menghubungkan komponen DataSet ke database yang akan digunakan oleh aplikasi. DatabaseName dapat berisi nama alias atau direktori dimana tabel-tabel disimpan.



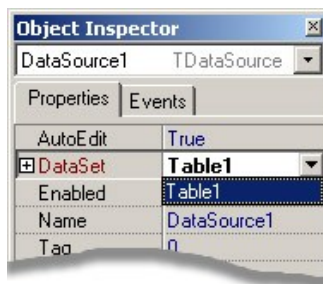
**Gambar 4.35. Properti DatabaseName dari Table1**

5. Atur agar properti TableName berisi Animals.dbf (Gambar 3.2). Properti TableName digunakan untuk memilih tabel yang akan dibuka.



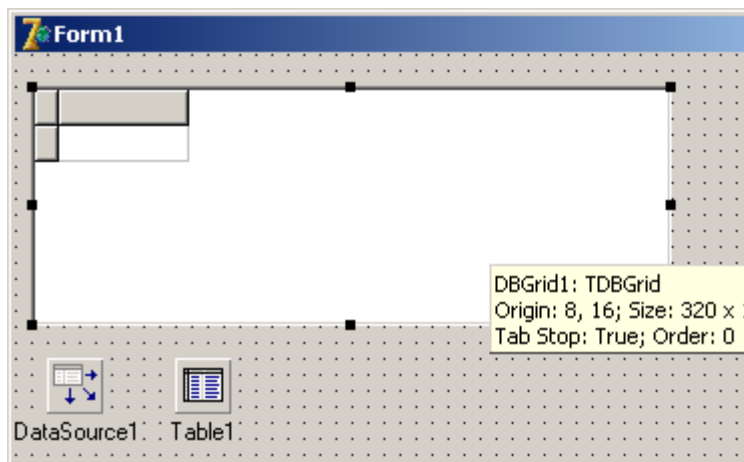
**Gambar 4.36. Memilih tabel yang akan dibuka**

6. Pilih DataSource1 dan kemudian atur agar properti DataSet berisi Table1 (Gambar 3.3).



**Gambar 4.37. Menghubungkan Datasource1 ke Table1**

7. Pilih tab Data Controls dan kemudian masukkan komponen DBGrid ke Form (Gambar 3.4).



**Gambar 4.38. DBGrid**

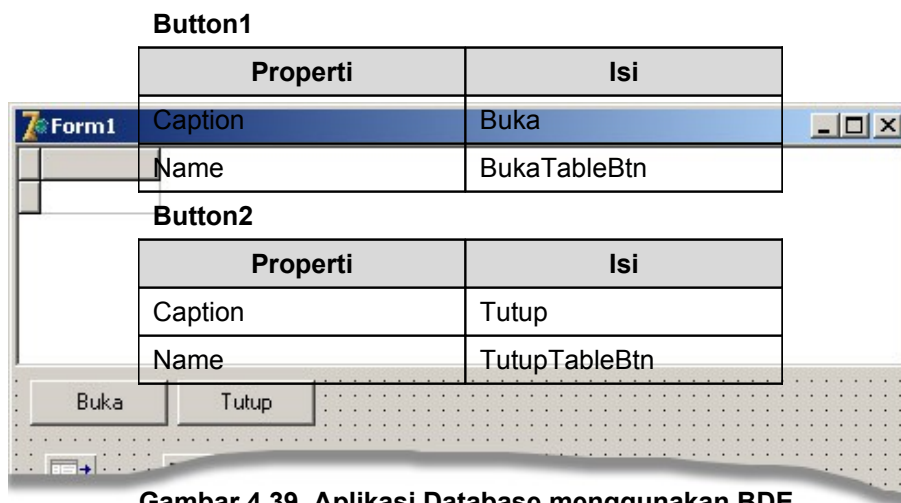
8. Atur agar properti-properti DBGrid1 berisi seperti pada Tabel 3-1.

**Tabel 4-8 Isi properti dari DBGrid1**

Properti	Isi
Align	alTop
DataSource	DataSource1

9. Tambahkan 2 buah komponen TButton ke dalam Form1 dan kemudian atur agar properti dari Button1 serta Button2 berisi seperti Tabel 3-2. (Gambar 3.5)

**Tabel 4-9. Properti Button1 dan Button2**



**Gambar 4.39. Aplikasi Database menggunakan BDE**

10. Pilih Button1, aktifkan Object Inspector dan pilih tab Events.
11. Double click events OnClick dan kemudian isikan Listing 3-1 sebagai event handler OnClick untuk Button1.

**Listing 4-7. Event handler OnClick dari Button1**

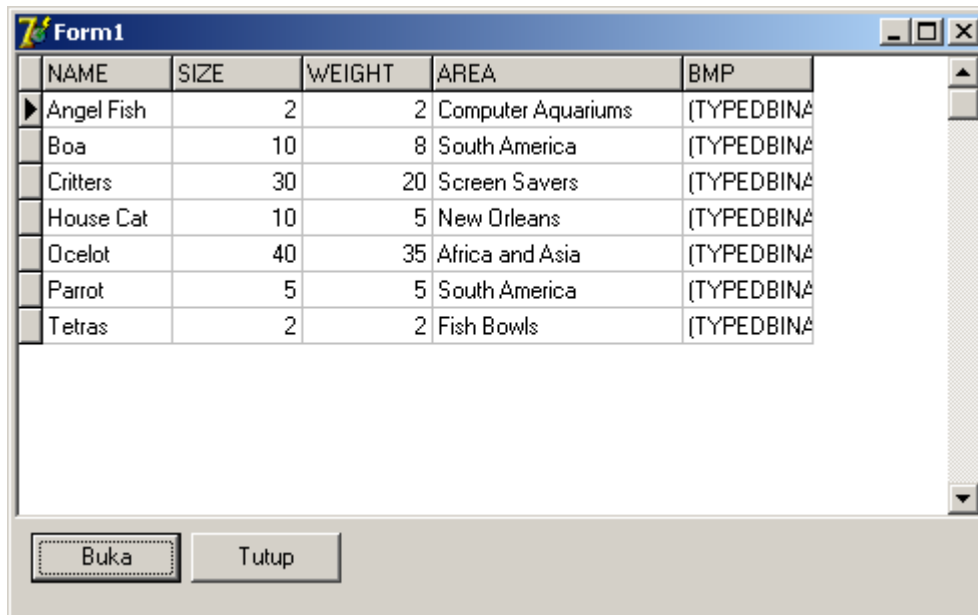
```
procedure TForm1.BukaTableBtnClick(Sender: TObject);
begin
  Table1.Open;
end;
```

12. Dengan cara yang sama buat event handler OnClick dari Button2 seperti Listing 3.2.

**Listing 4-8. Event handler OnClick dari Button2**

```
procedure TForm1.TutupTableBtnClick(Sender: TObject);
begin
  Table1.Close;
end;
```

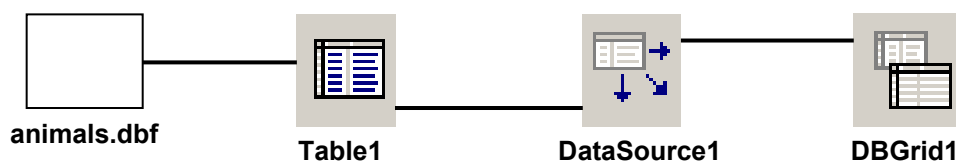
13. Simpan program dengan nama AplikasiBDE1.dpr dan file unit sebagai Unit1.pas.
14. Kompilasi program dan kemudian jalankan, apabila anda mengklik tombol Buka maka program akan membuka Table1 dan isi Table1 (file Animals.dbf) akan ditampilkan di DBGrid1. Apabila anda mengklik tombol Tutup maka Table1 akan ditutup atau sama dengan menutup pemakaian file Animals.dbf. Hasil program saat tombol Buka diklik ditunjukkan pada Gambar 3.6 (*catatan : saya menyesuaikan ukuran DBGrid1 dan memindah lokasi Button1 dan Button2 untuk memperoleh tampilan yang agak lebih nyaman*)



**Gambar 4.40. Hasil program menampilkan isi file Animals.dbf**

Wow... ternyata mudah sekali membuat aplikasi database!!!! Memang, apabila anda hanya ingin menampilkan isi sebuah tabel!!! Sebelum kita melanjutkan membuat aplikasi yang lebih rumit, kita pelajari terlebih dahulu apa yang terjadi di aplikasi pertama kita ini.

File Animals.dbf yang disimpan di alias DBDEMOS dihubungkan ke aplikasi melalui Table1, DBGrid1 mengambil isi file Animals1 melalui komponen DataSource1 yang dihubungkan ke Table1. Relasi yang terjadi antara masing-masing komponen di AplikasiBDE1 digambarkan melalui Gambar 3.7.



**Gambar 4.41. Hubungan antar komponen di koneksi BDE**

Pada saat kita memberi perintah :

### Table1.Open

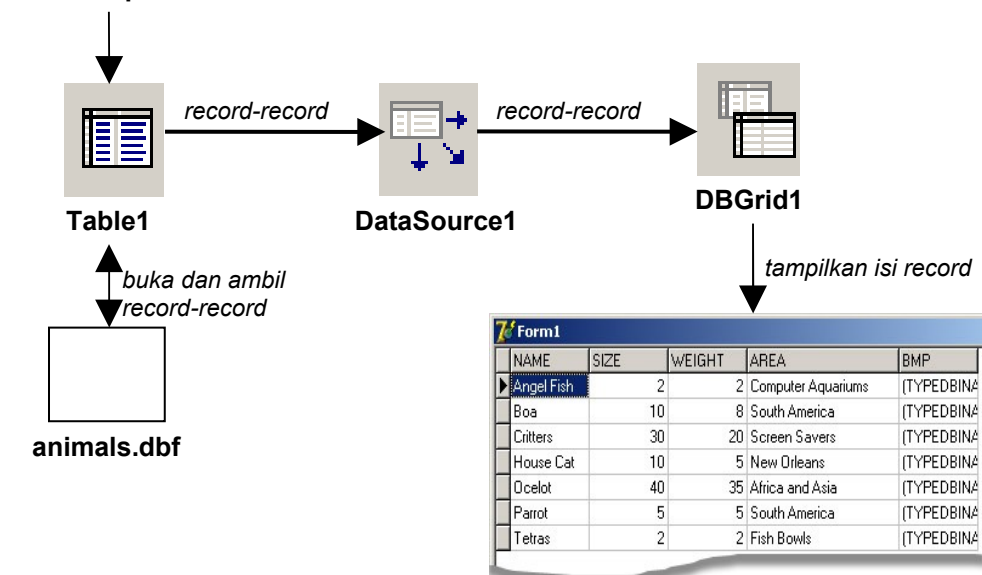
maka kita memerintahkan kepada Table1 untuk membuka file yang terhubung ke dirinya, mengambil record-record di dalamnya dan kemudian menyerahkan isi record-record tersebut ke DataSource1, selanjutnya DataSource1 menyerahkan isi record-record tersebut (termasuk semua informasi yang berkaitan dengan record tersebut) ke DBGrid1. DBGrid1 segera setelah menerima isi record akan menampilkan isi record tersebut.

Dan, saat kita memberi perintah :

### Table1.Close

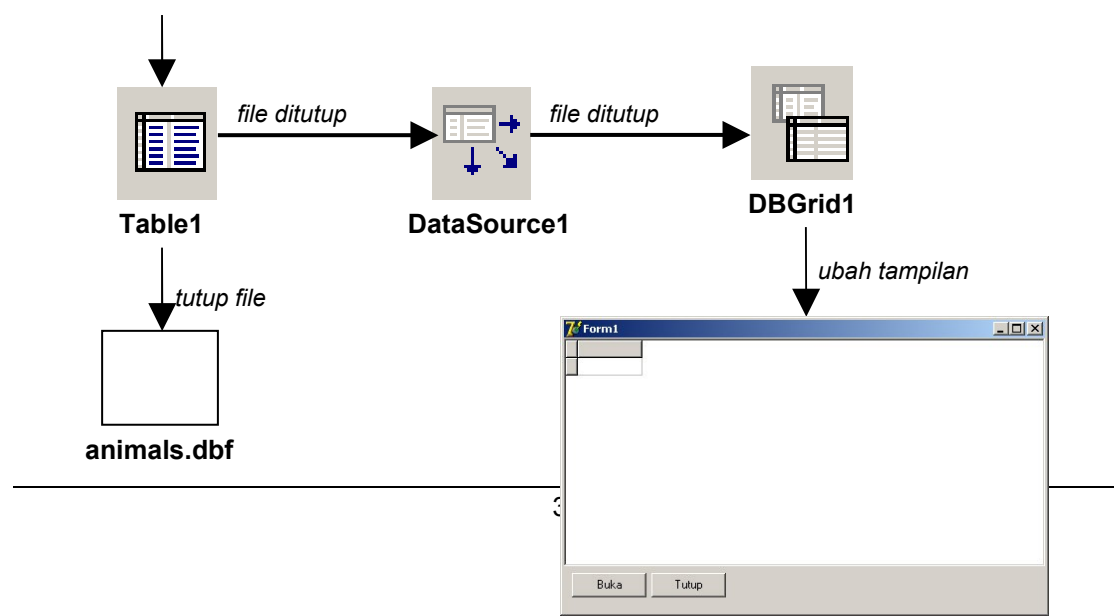
maka Table1 akan menutup file Animals.dbf serta memberitahu ke DataSource1 bahwa file Animals.dbf telah ditutup, DataSource1 memberitahu DBGrid1 bahwa tidak ada record yang dapat ditampilkan dan pesan ini menyebabkan DBGrid1 mengubah tampilannya. Gambar 3.8 dan Gambar 3.9 menunjukkan apa yang terjadi saat kita memberi perintah Open dan Close.

### Table1.Open



Gambar 4.42. Perintah Table1.Open

### Table1.Close



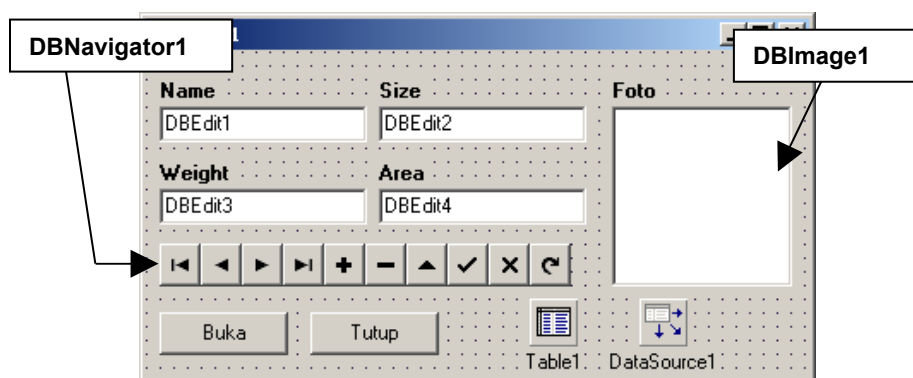
**Gambar 4.43. Perintah Table1.Close**

### Membuat AplikasiBDE2.

Apabila di AplikasiBDE1 kita menggunakan DBGrid sebagai data controls maka di AplikasiBDE2 kita akan menggunakan DBText, DBEdit dan DBImage serta DBNavigator sebagai data controls. Kita akan mempelajari bagaimana menampilkan isi field serta melakukan navigasi.

Saya menanggapi bahwa pada saat ini anda sudah cukup mahir menggunakan komponen-komponen sehingga saya tidak akan lagi menunjukkan secara rinci komponen apa yang perlu anda gunakan, saya hanya akan menunjukkan tata letak form dan sedikit petunjuk tentang properti dari masing-masing komponen yang harus anda atur.

1. Buat aplikasi baru dan simpan dengan nama AplikasiBDE2.dpr
2. Gunakan Gambar 3.10 dan Tabel 3-3 untuk mengatur Form1.



**Gambar 4.44. Tata letak komponen di Form1**

**Tabel 4-10. Properti komponen-komponen AplikasiBDE2**

**Table1**

Properti	Isi
DatabaseName	DBDEMOS
TableName	Animals.dbf

**DataSource1**

Properti	Isi
DataSet	Table1

#### DBNavigator

Properti	Isi
DataSource	DataSource1

#### DBEdit1

Properti	Isi
DataSource	DataSource1
DataField	NAME

#### DBEdit2

Properti	Isi
DataSource	DataSource1
DataField	SIZE

#### DBEdit3

Properti	Isi
DataSource	DataSource1
DataField	WEIGHT

#### DBEdit4

Properti	Isi
DataSource	DataSource1
DataField	AREA

#### DBImage1

Properti	Isi
DataSource	DataSource1
DataField	BMP
Stretch	True

#### Button1

Properti	Isi
Caption	Buka

#### Button2

Properti	Isi
Caption	Tutup

3. Buat event handler OnClick dari Button1 dan Button2 seperti pada Listing 3.3

#### Listing 4-9. Event handler OnClick dari Button1 dan Button2








```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Table1.Open;
end;

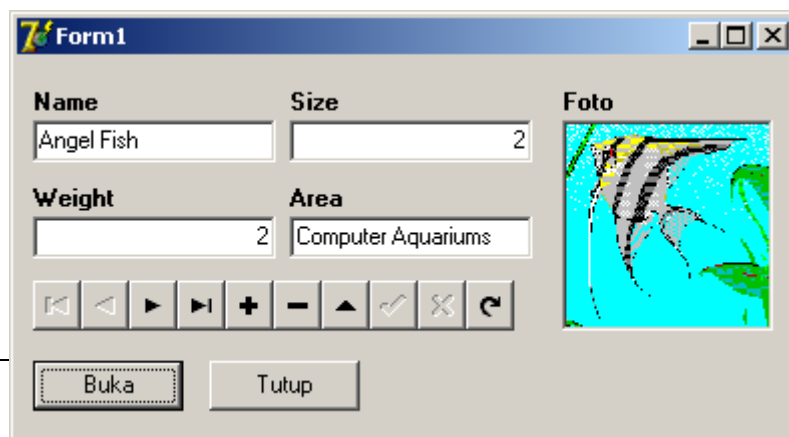
procedure TForm1.Button2Click(Sender: TObject);
```

```
begin
  Table1.Close;
end;
```

4. Kompilasi program dan kemudian jalankan. Klik tombol Buka dan kemudian klik tombol-tombol di DBNavigator. Fungsi dari masing-masing tombol diperlihatkan di Tabel 3-4. Hasil program diperlihatkan pada Gambar 3.11.

**Tabel 4-11. Fungsi tombol-tombol di DBNavigator**

Icon	Konstanta	Fungsi
	nbFirst	Menuju record pertama
	nbPrior	Menuju record sebelumnya
	nbNext	Menuju ke record berikutnya
	nbLast	Menuju ke record terakhir
	nbInsert	Menyisipkan record baru sebelum record saat ini
	nbDelete	Menghapus record saat ini dan membuat record berikutnya aktif
	nbEdit	Mengatur dataset ke modus Edit sehingga record dapat diedit.
	nbPost	Menyimpan perubahan di record saat ini secara permanen
	nbCancel	Membatalkan perubahan di record saat ini
	nbRefresh	Memperbarui record / mengambil ulang record



The screenshot shows a Windows application window titled "Form1". It contains a data entry form with the following fields and controls:

- Name:** A text box containing "Angel Fish".
- Size:** A text box containing the number "2".
- Weight:** A text box containing the number "2".
- Area:** A text box containing "Computer Aquariums".
- Foto:** A small image box showing a colorful fish (likely an Angel Fish) swimming in water.
- Navigation Buttons:** A row of buttons with icons for navigating between records: first, previous, next, last, insert, delete, edit, post, cancel, and refresh.
- Action Buttons:** Two buttons at the bottom: "Buka" (Open) and "Tutup" (Close).

**Gambar 4.45. AplikasiBDE2**

## **BAB 4 MEMBUAT ALIAS DAN MENGATUR KONFIGURASI BDE**

### **APA YANG AKAN KITA PELAJARI ?**

- Apa itu Alias.
- Menggunakan BDE Configuration untuk membuat dan mengatur Alias

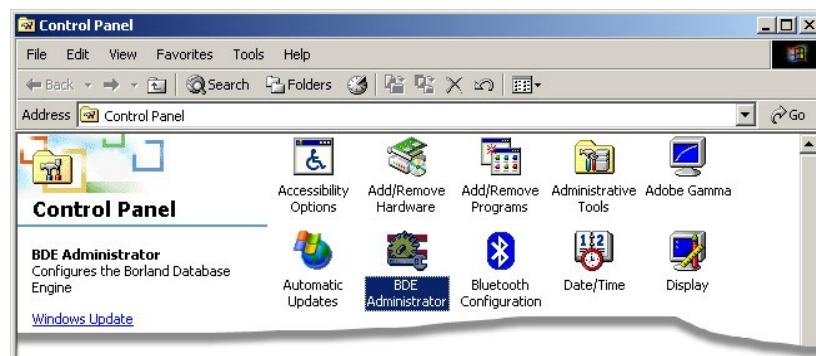
**WAKTU LATIHAN : 1 JAM**

## Alias

Alias adalah nama yang digunakan untuk menyatakan sejumlah pengaturan yang terkait dengan database tertentu. Pengaturan Alias umumnya dilakukan di luar aplikasi sehingga aplikasi terbebas dari proses-proses pengaturan dan implementasi aplikasi menjadi lebih sederhana. Sebagai contoh : apabila saat mengembangkan program di rumah anda menggunakan database lokal dan anda menggunakan nama alias MYPROJECT maka MYPROJECT dapat diatur agar menunjuk ke direktori dimana file-file tabel disimpan, tetapi di kantor anda mengatur agar MYPROJECT menunjuk ke server MySQL. Meskipun database yang digunakan berbeda tetapi anda tidak perlu setiap kali mengatur pemakaian database di aplikasi yang sedang anda tulis tersebut, asalkan aplikasi menggunakan alias MYPROJECT sebagai nama database maka kemana MYPROJECT mengarah sudah tidak relevan bagi aplikasi.

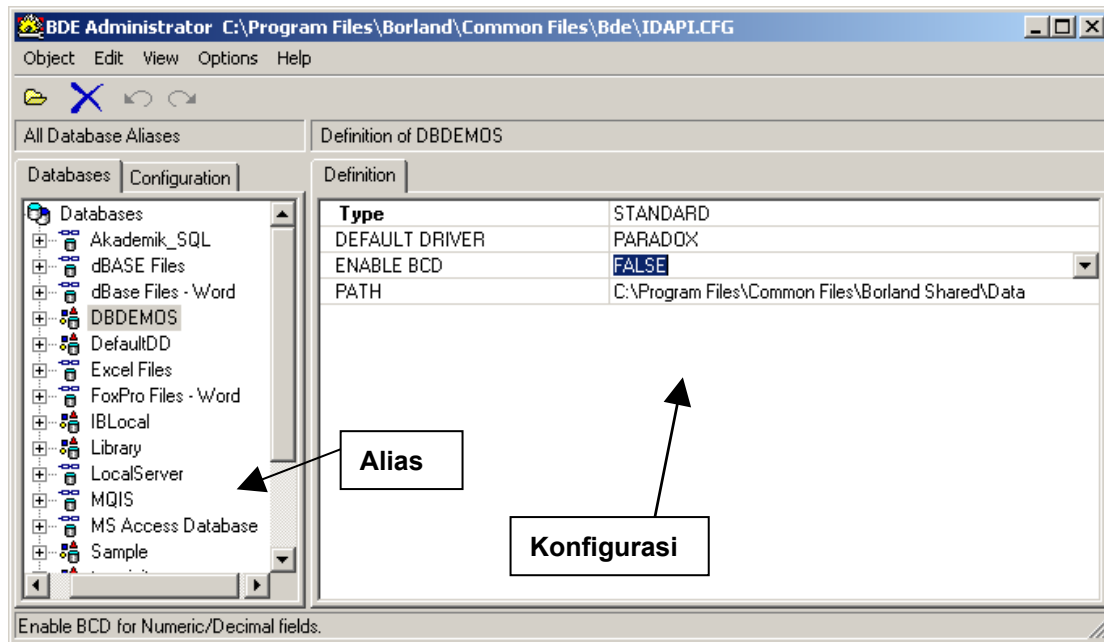
## Membuat Alias

Bagaimana caranya membuat Alias ? Anda dapat menggunakan BDE Administrator yang terletak di Control Panel untuk mengelola Alias. Gambar 4.1 memperlihatkan lokasi dari BDE Administrator di Control Panel.



**Gambar 5.46. BDE Administrator**

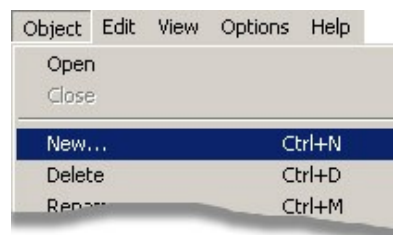
Gambar 4.2 memperlihatkan tampilan dari BDE Administrator saat dijalankan. Nama-nama alias yang dapat digunakan oleh aplikasi ditunjukkan di bawah tab Databases di sebelah kiri tampilan. Ada satu hal yang perlu anda ketahui bahwa BDE Administrator tidak hanya memperlihatkan nama alias tetapi juga nama-nama DSN (Data Source Name) yang diatur melalui ODBC Data Source Administrator. Saat ini kita belum menggunakan ODBC sehingga kita tidak akan membahas bagaimana menghubungkan ODBC ke BDE.



**Gambar 5.47. BDE Administrator**

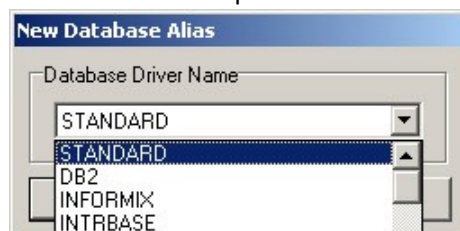
Dalam contoh ini, kita akan membuat alias dengan nama Aquarium yang menunjuk database lokal di C:\Program Files\Common Files\Borland Shared\Data. Langkah-langkah yang perlu dilakukan untuk membuat alias Aquarium adalah sebagai berikut :

1. Tutup Borland Delphi anda. Sebaiknya anda membuat alias pada saat Delphi sedang tidak aktif.
2. Jalankan BDE Administrator di Control Panel sehingga anda memperoleh tampilan seperti Gambar 4.2.
3. Pilih menu Object | New (Gambar 4.3)



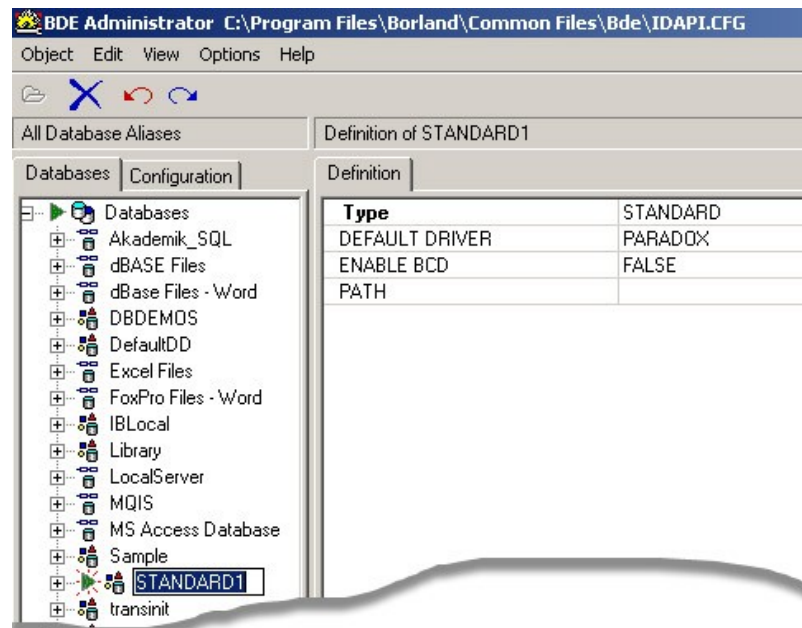
**Gambar 5.48. Menu membuat Alias baru**

4. BDE Administrator akan menampilkan kotak dialog *New Database Alias*. Anda dapat memilih driver yang sesuai dengan tipe database anda melalui combobox. Karena kita akan menggunakan file Paradox maka pilih driver STANDARD. (Gambar 4.4)



**Gambar 5.49. Memilih driver yang sesuai dengan tipe database**


- Setelah anda memilih driver yang sesuai, klik tombol OK maka BDE Administrator akan membuat sebuah alias dengan nama STANDARD1 seperti Gambar 4.5

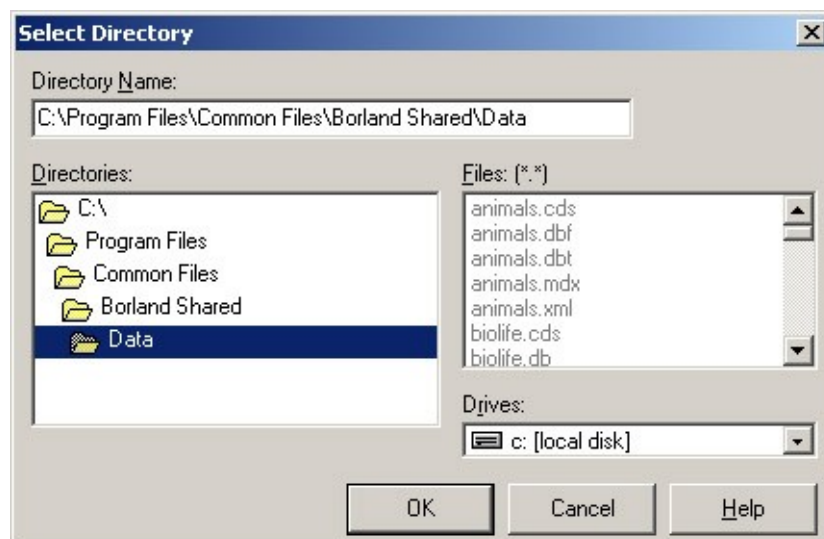


**Gambar 5.50. Alias STANDARD1**

- Ubahlah nama alias STANDARD1 menjadi AQUARIUM dengan mengklik kanan pada nama STANDARD1 dan kemudian memilih Rename.

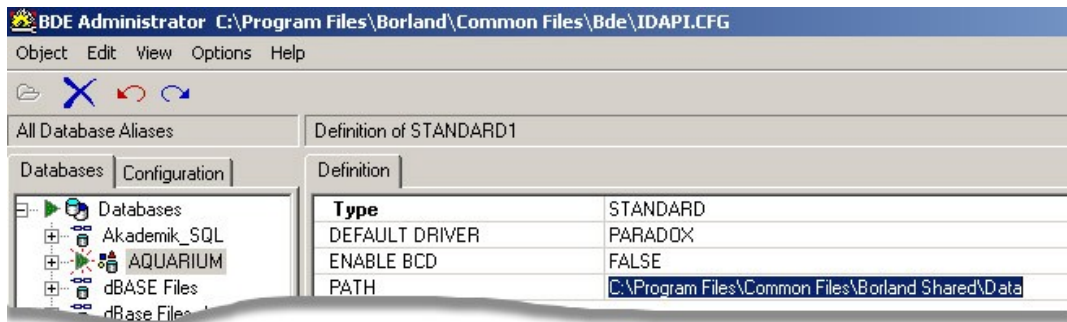
Kita akan mengatur agar alias AQUARIUM menunjuk ke lokasi C:\Program Files\Common Files\Borland Shared\Data dengan cara sebagai berikut :

- Klik tombol  yang terletak di sebelah kanan PATH.
- BDE Administrator akan menampilkan kotak dialog *Select Directory*. Pilih direktori C:\Program Files\Common Files\Borland Shared\Data. Klik OK untuk kembali ke BDE Administrator (Gambar 4.6).



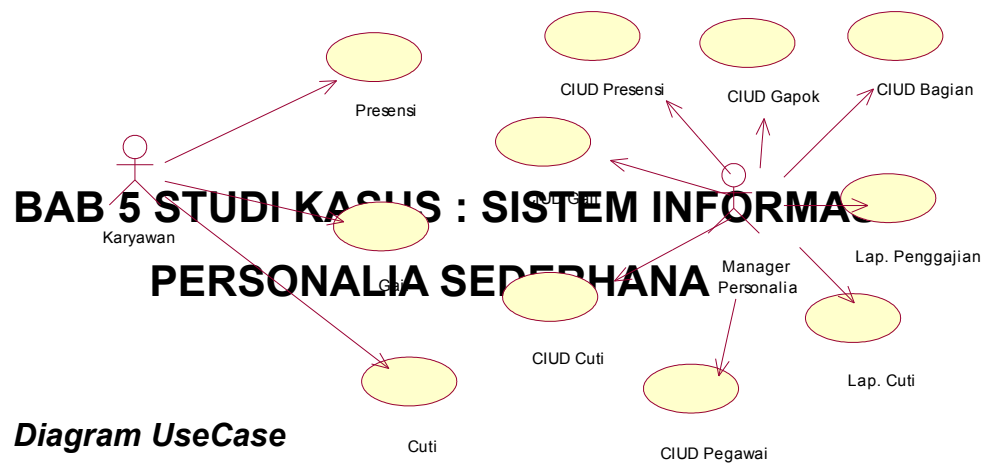
**Gambar 5.51. Memilih lokasi database**

9. Gambar 4.7 memperlihatkan alias AQUARIUM dan konfigurasi dari alias AQUARIUM.



**Gambar 5.52. Alias AQUARIUM**

10. Pilih menu *Object | Apply* dan jawab Yes untuk menyimpan konfigurasi dari alias AQUARIUM.
11. Jalankan Delphi dan kemudian ambil proyek AplikasiBDE2.dpr. Gunakan menu *File | Open* untuk memilih proyek AplikasiBDE2.
12. Pilih Table1 dan kemudian di Object Inspector ubah isi DatabaseName menjadi AQUARIUM.
13. Kompilasi dan jalankan program.



**Gambar 6.53. Diagram UseCase Sistem Personalia**

### ***Dokumen Skenario***

#### **UseCase : *Presensi***

##### **Skenario :**

1. Sistem meminta karyawan memasukkan NIP.
2. Sistem mencari NIP di tabel Pegawai

##### **Kemungkinan 1 : *NIP tidak ditemukan***

1. Munculkan pesan NIP salah.
2. Sistem kembali ke Langkah 1 pada Skenario Utama.

##### **Kemungkinan 2 : *NIP ditemukan***

1. Sistem menampilkan data karyawan berupa NIP, Nama, Nama Bagian serta Tanggal Sekarang.
2. Sistem menyimpan informasi tentang NIP di Tanggal Sekarang tabel Presensi.
3. Sistem kembali ke Langkah 1 pada Skenario Utama

#### **UseCase : *Gaji***

##### **Skenario :**

1. Sistem meminta Manager memasukkan bulan gaji

2. Sistem mengambil data pegawai
3. Untuk setiap data pegawai sistem melakukan :
  1. Sistem menghitung jumlah hari masuk (JHM) berdasarkan NIP di Presensi
  2. Sistem menghitung Uang Transport (UT) dengan rumus :  $UT = JHM * 5000$ .
  3. Sistem menghitung Gaji Pokok (GP) berdasarkan golongan pegawai.
  4. Sistem menghitung Gaji Kotor (GK) sebagai :  $GK = GP + UT$ .
  5. Sistem menghitung Pajak (PJK) sebagai :  $PJK = GK * 0.10$ .
  6. Sistem menghitung Gaji Bersih (GB) sebagai :  $GB = GK - PJK$ .
  7. Sistem menyimpan UT, GK, PJK dan GB.
  8. Sistem mencetak Slip Penggajian.
4. Sistem diakhiri.

**UseCase : Cuti**

**Skenario :**

1. Sistem meminta NIP karyawan
2. Sistem mencari NIP Karyawan.

**Kemungkinan 1 : NIP tidak ditemukan**

1. Munculkan pesan NIP salah
2. Sistem diakhiri

**Kemungkinan 2 : NIP ditemukan**

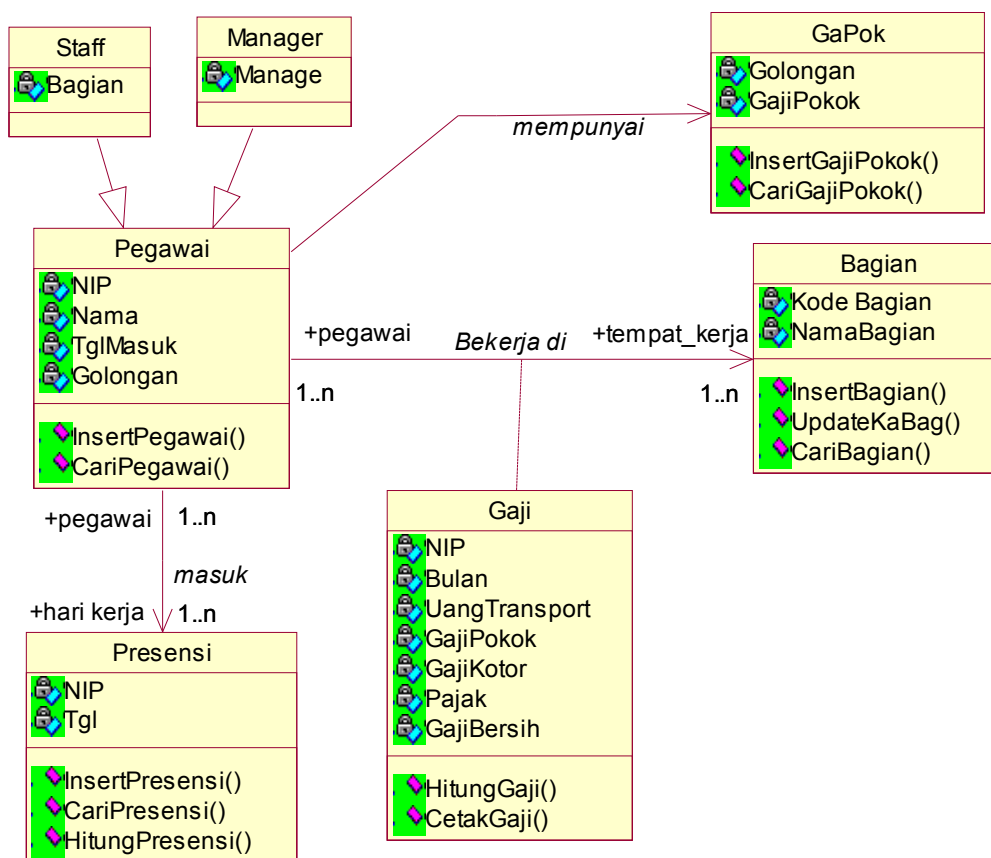
1. Sistem menghitung Hari Cuti (HC) sebagai :  $HC = \text{Tgl Akhir Cuti} - \text{Tgl Mulai Cuti}$
2. Sistem mencari data jumlah hari cuti (JHC) yang sudah diambil berdasarkan NIP
3. Sistem menghitung sisa cuti (SC) sebagai  $SC = 12 - JHC - HC$ .
4. Apabila  $SC > 0$  maka sistem akan menyimpan NIP, Tgl Mulai Cuti dan Tgl Akhir Cuti, tetapi apabila  $SC \leq 0$  maka sistem menampilkan pesan 'Cuti ditolak'.
5. Sistem diakhiri.

**UseCase : Laporan Penggajian**

**Skenario :**

1. Sistem meminta bulan laporan
2. Untuk setiap pegawai yang ditemukan di penggajian
  1. Cetak detail penggajian untuk pegawai tersebut.
  2. Hitung Total UT (TUT) , Total GP (TGP), Total GK (TGK), Total PJK (TPJK), dan Total GB (TGB).
3. Cetak TUT, TGP, TGK, TPJK, dan TGB.

## Diagram Kelas



Gambar 6.54. Diagram Kelas Sistem Personalia

## Pembahasan

Gambar 5.1 dikenal sebagai Diagram UseCase, yaitu diagram yang menggambarkan fungsi-fungsi utama dari sistem yang akan dibuat. Dari Gambar 5.1 terlihat bahwa sistem personalia PT XYZ mempunyai fungsi utama yaitu :

1. Fungsi Presensi
2. Fungsi Penggajian
3. Fungsi Cuti
4. Fungsi Pelaporan Penggajian

Fungsi-fungsi tersebut nantinya harus disediakan oleh perangkat lunak personalia PT XYZ, dari Gambar 5.1 juga terlihat bahwa fungsi 1 sampai dengan fungsi 3 terkait dengan kegiatan karyawan sedangkan fungsi 4 dibutuhkan oleh Manager Personalia. Rincian apa yang dilakukan oleh masing-masing fungsi diperlihatkan melalui Dokumen Skenario.

Kelas-kelas seperti digambarkan oleh Gambar 5.2 menggambarkan informasi apa saja yang akan disimpan oleh sistem. Kelas-kelas tersebut kemudian diimplementasikan menjadi struktur tabel serta program.

Struktur tabel hasil konversi kelas sangat dipengaruhi oleh relasi antar kelas yang ada. Mengingat buku ini tidak membicarakan tentang Analisa dan Perancangan berorientasi objek maka saya tidak akan membahas lebih lanjut, anda yang ingin tahu lebih lanjut tentang cara mengkonversi kelas menjadi tabel silahkan membaca buku karangan William Blaha atau mendownload artikel saya tentang hal ini di situs pribadi saya. Secara ringkas, struktur tabel hasil konversi dari kelas diperlihatkan pada Gambar 5.3.

Tabel : Pegawai		
Field	Tipe	Panjang
NIP	C	10
Nama	C	50
TglMasuk	D	8
Golongan	N	1
KodeBagian	C	3
Posisi	N	1

Tabel : Presensi		
Field	Tipe	Panjang
NIP	C	10
Tgl	D	8

Tabel : Bagian		
Field	Tipe	Panjang
KodeBagian	C	3
Nama	C	15

Tabel : GaPok		
Field	Tipe	Panjang
Golongan	N	1
GajiPokok	N	6

Tabel : Gaji		
Field	Tipe	Panjang
NIP	C	10
Bulan	N	1
GajiPokok	N	6
GajiKotor	N	6
UangTransport	N	4
Pajak	N	4,2

**Gambar 6.55. Struktur Tabel Sistem Personalia**

Sistem Personalia PT XYZ di atas akan kita jadikan contoh nyata membangun aplikasi database. Bab-bab berikut akan mengajak anda secara bertahap mengimplementasikan Sistem Personalia PT XYZ. Saya akan menunjukkan bagaimana mengimplementasi rancangan tersebut menggunakan database lokal maupun database remote. Implementasi dengan database remote menggunakan server PostgreSQL, meskipun demikian anda dapat menggunakan sembarang server asalkan server tersebut menyediakan driver ODBC atau dbExpress.

## **BAB 6 MEMBUAT TABEL DAN MENGISI TABEL.**

### **APA YANG AKAN KITA PELAJARI ?**

- Membuat tabel dari dalam program.
- Membuat program mengolah data
- Menguji keabsahan input

**WAKTU LATIHAN : 3 JAM**

## Membuat tabel (Create)

Membuat tabel di database lokal dapat dilakukan dengan dua cara, yaitu :

1. Menggunakan perangkat lunak pembuat database
2. Melalui perintah-perintah di dalam program aplikasi.

Di dalam buku ini saya tidak akan membahas mengenai bagaimana membuat database melalui perangkat lunak pembuat database tetapi akan membahas bagaimana memberikan fasilitas membuat tabel dari dalam aplikasi. Mengapa diperlukan fasilitas semacam itu ? Seringkali ketika kita menerapkan aplikasi belum tersedia tabel-tabel data. Ini berarti aplikasi anda harus menyiapkan tabel-tabel dari awal (kosong). Kita mulai membangun aplikasi database kita dengan membuat menu yang dapat digunakan untuk membuat tabel-tabel Sistem Personalia. Perintah yang digunakan untuk membuat tabel dari dalam program adalah: *Table1.CreateTable*, dengan Table1 merupakan nama komponen dataset TTable. Urutan langkah yang dilakukan dalam membuat tabel adalah sebagai berikut :

1. Tentukan DatabaseName.
2. Tentukan tipe tabel dengan mengisi *Table1.TableType := <tipe\_table>*. Dengan *<tipe\_table>* berisi salah satu dari konstanta di Tabel 6.1.
3. Non aktifkan tabel dengan memberi perintah *Table1.Active := False*.
4. Untuk tiap field yang ada di dalam tabel ini, definisikan field sebagai :
  1. with *Table1.FieldDefs* do begin
  2.     *Clear; //bersihkan definisi sebelumnya*
  3.     with *AddFieldDef* do begin *//mendefinisikan tipe field*
  4.         *Name := <nama field>; //nama field*
  5.         *DataType := <tipe field>; //tipe field seperti tabel 6.2*
  6.         *Required := <diisi True jika field ini tidak boleh kosong>*
  7.     end;
5. Berikan perintah *Table1.CreateTable*.

**Tabel 7-12. Konstanta tipe\_table**

Tipe Tabel	Keterangan
ttDefault	Menentukan tipe tabel berdasarkan nama ekstensi
ttParadox	Tabel bertipe Paradox
ttDbase	Tabel bertipe dBase
ttFoxPro	Tabel bertipe FoxPro
ttASCII	Tabel merupakan file ASCII

**Tabel 7-13. Konstanta tipe\_field**

Tipe Field	Keterangan
------------	------------

ftUnknown	Tipe field tidak diketahui
ftString	Kumpulan Karakter
ftSmallint	Integer 16 bit
ftInteger	Integer 32 bit
ftWord	Integer 32 bit unsigned
ftBoolean	Boolean
ftFloat	Pecahan
ftCurrency	Mata Uang
ftDate	Tanggal
ftTime	Jam
ftDateTime	Tanggal dan Jam
ftAutoInc	Auto increment, isi otomatis +1 dari record terakhir
ftBlob	File binary
ftMemo	Memo (text)
ftGraphic	Bitmap
ftFmtMemo	Memo (rich text)

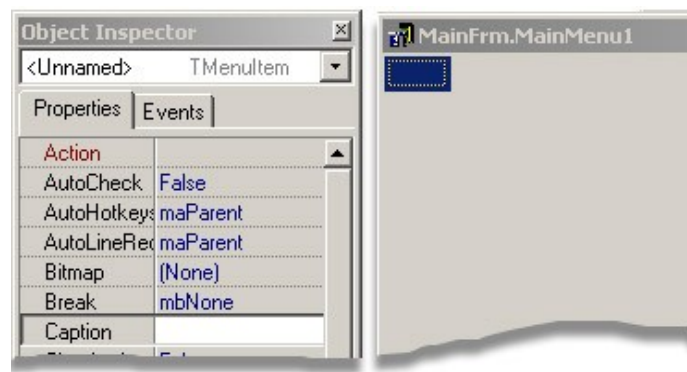
### Membuat tabel secara programming

1. Menggunakan BDE Administrator, buat sebuah alias dengan konfigurasi seperti pada Tabel 6.3.

**Tabel 7-14. Konfigurasi Alias untuk Sistem Personalia**

Komponen	Isi
Nama	DBPersonalia
Tipe	STANDARD
Path	C:\Personalia\Database

2. Buat aplikasi baru dan simpan dengan nama SistemPersonalia.dpr dan Form sebagai Main.pas.
3. Ubah nama form sebagai MainFrm dan kemudian tambahkan komponen MainMenu ke MainFrm.
4. Kita akan menambahkan satu menu utama dan dua buah menu anak.
5. Double-click MainMenu1 di MainFrm sehingga anda memperoleh menu editor seperti Gambar 6.1



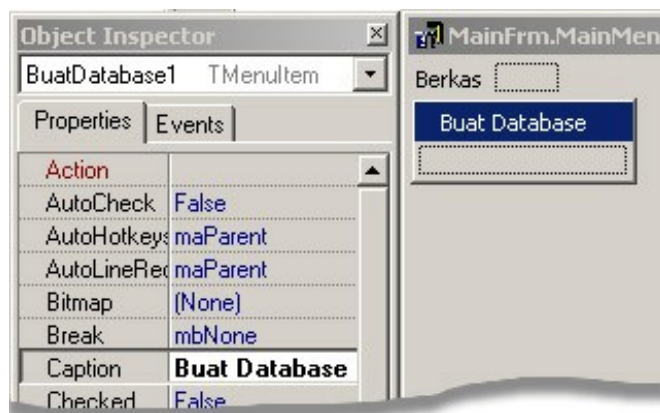
**Gambar 7.56. Menu Editor**

6. Aktifkan Object Inspector (F11) dan isi properti Caption dengan 'Berkas'. Perhatikan perubahan yang terjadi di Menu Editor. (Gambar 6.2).



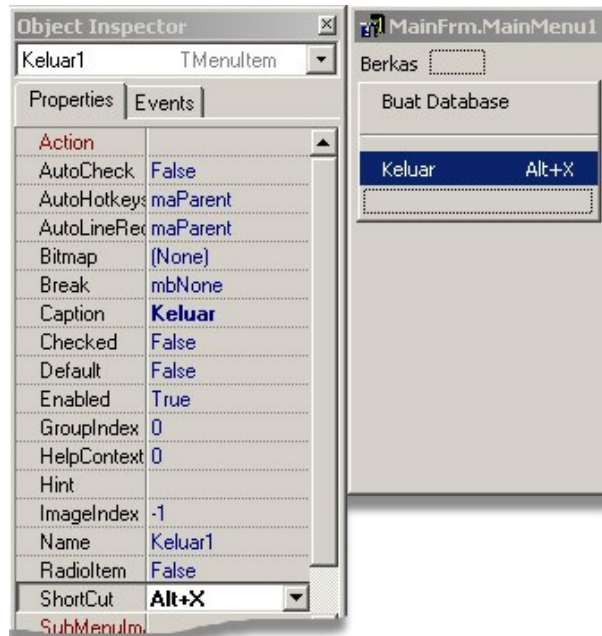
**Gambar 7.57. Menu Berkas**

7. Klik menu kosong di bawah menu Berkas. Pindah ke Object Inspector dan tulis 'Buat Database' di properti Caption. Perhatikan perubahan yang terjadi di Menu Editor (Gambar 6.3). Apa yang kita lakukan adalah mendefinisikan sub menu di bawah menu utama Berkas.



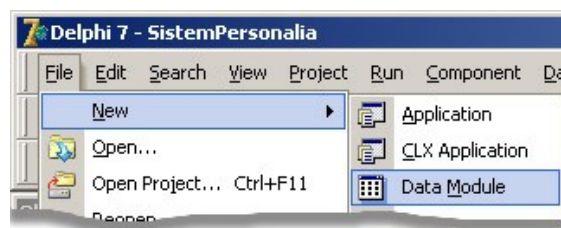
**Gambar 7.58. Sub Menu Buat Database**

8. Klik menu kosong di bawah sub menu Buat Database dan kemudian properti Caption anda isi dengan '-' (minus). Delphi akan menampilkan garis pemisah menu. Klik menu kosong di bawah pemisah menu dan isi properti Caption dari sub menu tersebut dengan 'Keluar'. Isi properti ShortCut dari sub menu Keluar dengan 'Alt+X', apabila shortcut yang anda inginkan tidak tersedia anda dapat mengetik Alt+X atau jika shortcut tersedia anda dapat memilih dari daftar.(Gambar 6.4).



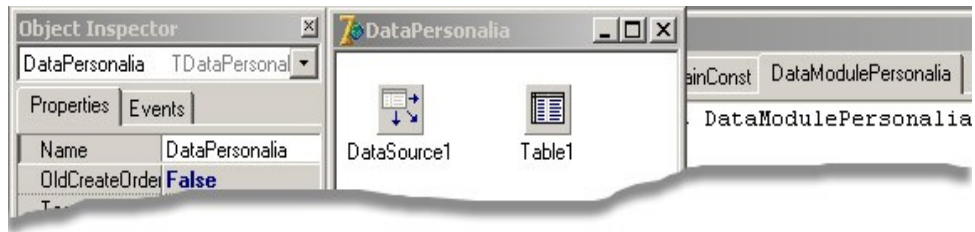
**Gambar 7.59. Menu Berkas (lengkap)**

9. Tutup Menu Editor sehingga anda kembali ke MainFrm.
10. Tambahkan sebuah form dengan tipe Data Module melalui menu File | New | Data Module (Gambar 6.5). Form Data Module akan kita gunakan sebagai tempat untuk menyimpan komponen dataset dan datasource yang digunakan oleh aplikasi ini.



**Gambar 7.60. Menu Data Module**

11. Simpan form Data Module sebagai DataModulePersonalia.pas dan ubah properti Name dari form DataModule1 menjadi DataPersonalia. Tambahkan satu datasource dan satu buah komponen dataset TTable (Gambar 6.6). Atur agar properti DatabaseName dari Table1 berisi 'DBPersonalia'.



**Gambar 7.61. Form DataPersonalia**

12. Buat sebuah unit baru melalui menu File | New | Unit (Gambar 6.7) dan simpan sebagai MainConst.pas. Unit MainConst.pas akan kita gunakan sebagai tempat mendefinisikan konstanta dan variabel global yang digunakan di MainForm maupun form lain. Anda mungkin heran, mengapa tidak cukup menuliskan konstanta dan variabel global di MainForm.pas saja ? Alasannya sederhana saja, pemisahan unit tersebut akan memudahkan kita mencari dan menentukan lokasi konstanta maupun variabel saat kita menemui kesalahan. Percayalah, kalau suatu hari anda berhubungan dengan lebih dari 15 form maka anda akan mengetahui bahwa kebiasaan semacam ini akan sangat membantu.



**Gambar 7.62. Menu Unit**

13. Pilih MainForm di Code Editor dan kemudian hubungkan MainForm.pas dengan MainConst.pas melalui menu File | Use Unit (Gambar 6.8). Pilih MenuConst dari kotak dialog dan kemudian klik tombol OK (Gambar 6.9)



**Gambar 7.63. Menu File | Use Unit**



**Gambar 7.64. Memilih unit yang akan digunakan**

14. Definisikan sebuah metoda private BuatTabel dari MainForm (Listing 16-3) dan kemudian **tulis** (saya beri huruf tebal karena memang anda harus menulis sendiri!!) implementasi metoda BuatTable seperti di Listing 16-4.

**Listing 7-10. Mendefinisikan metoda private**

```
type
  TMainFrm = class(TForm)
    <dipotong>
  private
    { Private declarations }
    procedure BuatTabel; <-- tulis ini
  public
    { Public declarations }
  end;
```

**Listing 7-11. Implementasi metoda BuatTabel**

```
implementation

uses MainConst, DataModulePersonalia;

{$R *.dfm}

procedure TMainFrm.BuatTabel;
begin
  with DataPersonalia do
  begin
    //masukkan tipe table, kita gunakan paradox (standard delphi)
    Table1.TableType := ttParadox;

    with Table1 do
    begin
      //tabel Pegawai
      TableName := 'Pegawai';
      //non aktifkan tabel
      Active:=False;
      // Tabel yang sudah ada tidak boleh dihapus
      if not Exists then begin
        //Pertama, kita buat rincian field
        with FieldDefs do begin
          Clear;
          with AddFieldDef do begin
            Name := 'NIP';
            DataType := ftString;
            Size:=10;
            Required := True;
          end;
          with AddFieldDef do begin
```

```

        Name := 'Nama';
        DataType := ftString;
        Size := 50;
    end;
    with AddFieldDef do begin
        Name := 'TglMasuk';
        DataType := ftDate;
    end;
    with AddFieldDef do begin
        Name := 'Golongan';
        DataType := ftInteger;
    end;
    with AddFieldDef do begin
        Name := 'KodeBagian';
        DataType := ftString;
        Size := 3;
    end;
    with AddFieldDef do begin
        Name := 'Posisi';
        DataType := ftInteger;
    end;
end; //with FieldDefs
// Berikutnya, definisikan index untuk
with IndexDefs do begin
    Clear;
    with AddIndexDef do begin
        Name := '';
        Fields := 'NIP';
        Options := [ixPrimary];
    end; // with AddIndexDef
end; //with IndexDefs
// panggil metoda CreateTable untuk membuat tabel
CreateTable;
end;
TableName := 'Presensi'; //tabel Presensi
Active:=False; //non aktifkan tabel
// Tabel yang sudah ada tidak boleh dihapus
if not Exists then begin
    //Pertama, kita buat rincian field
    with FieldDefs do begin
        Clear; //bersihkan definisi sebelumnya
        with AddFieldDef do begin
            Name := 'NIP';
            DataType := ftString;
            Size:=10;
            Required := True;
        end;
        with AddFieldDef do begin
            Name := 'Tgl';
            DataType := ftDate;
        end;
    end; //with FieldDefs
    //Berikutnya, definisikan index untuk
    with IndexDefs do begin
        Clear;
        with AddIndexDef do begin
            Name := '';
            Fields := 'NIP';
            Options := [ixPrimary];
        end; // with AddIndexDef
    end; //with IndexDefs
end; //with IndexDefs

```

```

    // panggil metoda CreateTable untuk membuat tabel
    CreateTable;
end;
TableName := 'GaPok'; //tabel GaPok
Active:=False; //non aktifkan tabel
// Tabel yang sudah ada tidak boleh dihapus
if not Exists then begin
    //Pertama, kita buat rincian field
    with FieldDefs do begin
        Clear;
        with AddFieldDef do begin
            Name := 'Golongan';
            DataType := ftInteger;
            Required := True;
        end;
        with AddFieldDef do begin
            Name := 'GajiPokok';
            DataType := ftInteger;
        end;
        IndexDefs.Clear;
    end; //with FieldDefs
    // panggil metoda CreateTable untuk membuat tabel
    CreateTable;
end;
TableName := 'Gaji'; //tabel Gaji
Active:=False; //non aktifkan tabel
// Tabel yang sudah ada tidak boleh dihapus
if not Exists then begin
    //Pertama, kita buat rincian field}
    with FieldDefs do begin
        Clear;
        with AddFieldDef do begin
            Name := 'NIP';
            DataType := ftString;
            Size:=10;
            Required := True;
        end;
        with AddFieldDef do begin
            Name := 'Bulan';
            DataType := ftInteger;
        end;
        with AddFieldDef do begin
            Name := 'GajiPokok';
            DataType := ftInteger;
        end;
        with AddFieldDef do begin
            Name := 'GajiKotor';
            DataType := ftInteger;
        end;
        with AddFieldDef do begin
            Name := 'UangTransport';
            DataType := ftInteger;
        end;
        with AddFieldDef do begin
            Name := 'Pajak';
            DataType := ftFloat;
        end;
    end; //with FieldDefs
    // Berikutnya, definisikan index untuk
    with IndexDefs do begin
        Clear;

```

```

        with AddIndexDef do begin
            Name := '';
            Fields := 'NIP';
            Options := [ixPrimary];
        end; // with AddIndexDef
    end; //with IndexDefs
    // panggil metoda CreateTable untuk membuat tabel
    CreateTable;
end;
end; //with Table1
end; //with DataPersonalia do
end;

```

15. Klik sub menu Buat Database di MainFrm, Delphi akan secara otomatis mengartikan klik tersebut sebagai perintah untuk membuat event handler OnClick dari sub menu Buat Database. Tulis Listing 6-5 sebagai implementasi dari Buat Database.

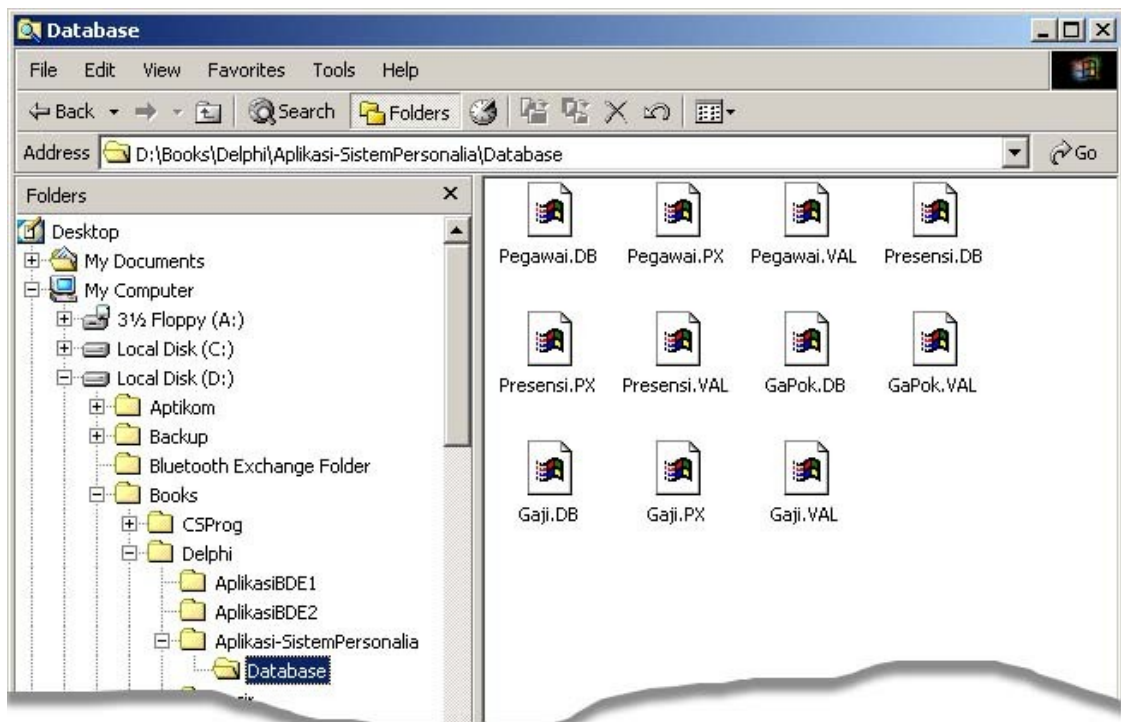
**Listing 7-12. Event handler OnClick dari menu Buat Database**

```

procedure TMainFrm.BuatDatabase1Click(Sender: TObject);
begin
    BuatTabel;
end;

```

16. Kompilasi program dan kemudian jalankan. Pilih menu Buat Database. Anda memang tidak melihat apapun terjadi tetapi coba jalankan Windows Explorer dan lihat di direktori <app\_path>\database akan ada beberapa file (tabel) seperti Gambar 6.10.



**Gambar 7.65. Hasil perintah CreateTable**

### **Mengisi Tabel (Insert)**

Ada 5 tabel yang digunakan di dalam sistem personalia PT XYZ, tetapi hanya 4 tabel yang diisi secara manual, yaitu : Tabel Pegawai, Tabel Bagian, Tabel GaPok dan Tabel Presensi, sedangkan Tabel Gaji diisi melalui proses Penggajian. Dalam sub bab ini saya akan menunjukkan bagaimana mengisi tabel-tabel tersebut dengan menggunakan komponen-komponen data controls. Kita juga akan mempelajari kapan dan bagaimana meyakinkan bahwa isian yang diisi oleh operator sudah benar dan boleh disimpan. Dalam implementasi ini, saya memilih untuk membuat satu form untuk tiap tabel, dengan demikian ada 5 buah form. Alasan saya memilih satu form untuk satu tabel (tepatnya, satu kesatuan isian) karena inilah implementasi paling mudah.

### **Menambahkan sub menu berkas.**

1. Kita akan menambahkan tiga buah sub menu di bawah sub menu Buat Database.
2. Double-click komponen MainMenu1 sehingga anda memperoleh Menu Editor.
3. Pilih sub menu '-----' (separator) di bawah sub menu Buat Database, dan kemudian tekan tombol keyboard Insert sehingga anda memperoleh sub menu kosong di bawah sub menu Buat Database. Isi properti Caption dari sub menu kosong tersebut dengan Pegawai (Gambar 6.11).



**Gambar 7.66. Sub menu Pegawai**

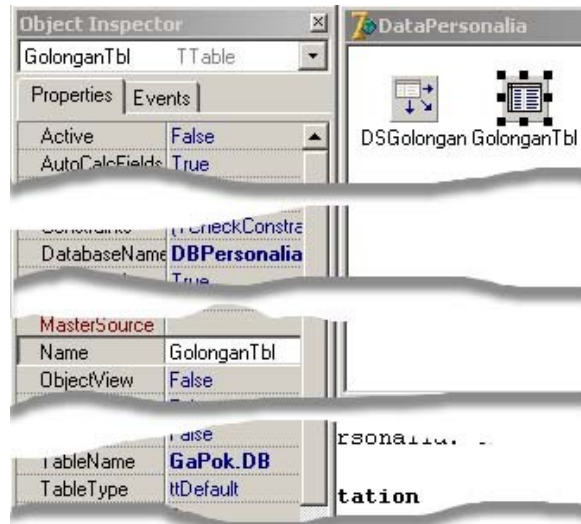
4. Lakukan hal yang sama untuk membuat sub menu Bagian, GaPok dan Presensi (Gambar 6.12).



**Gambar 7.67. Sub menu pengolahan tabel**

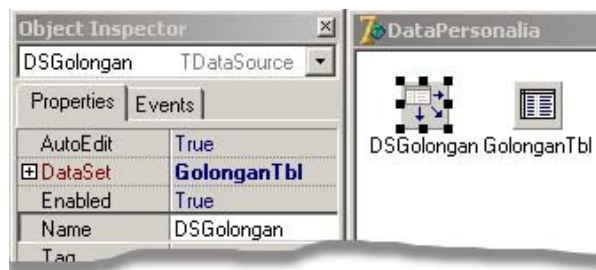
## Mengolah Tabel Golongan

1. Tambahkan satu buah dataset Table ke data module DataPersonalia dan kemudian ubah nama komponen menjadi GolonganTbl, atur agar properti DatabaseName berisi DBPersonalia dan TableName berisi GaPok.db (Gambar 6.13)



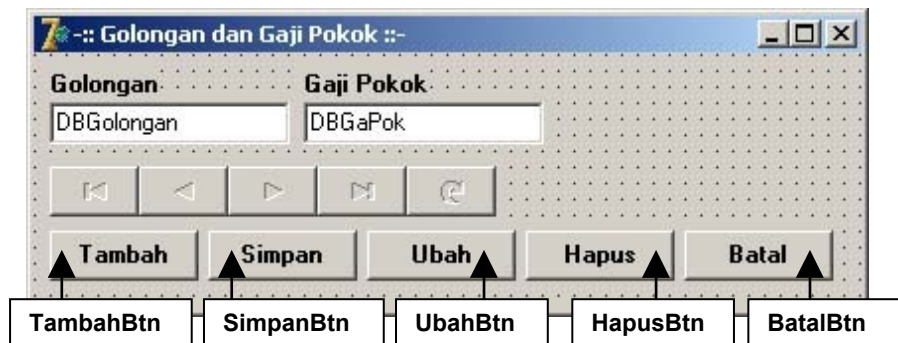
Gambar 7.68. Komponen GolonganTbl

2. Tambahkan satu data source ke data module DataPersonalia. Ubah nama data source menjadi DSGolongan. Atur agar properti DataSet berisi GolonganTbl. (Gambar 6.14)



Gambar 7.69. Menghubungkan datasource ke GolonganTbl

3. Buat sebuah form baru melalui menu *File | New | Form*. Atur agar properti name dari form baru berisi GolonganFrm dan kemudian simpan form ini sebagai GolonganForm.pas. (Catatan : merupakan kebiasaan baik apabila anda memberi nama form dengan bentuk <xxx>Frm dan unit menggunakan nama <xxx>Form, dengan <xxx> sesuai dengan fungsi / tujuan utama dari form tersebut. Penamaan seperti ini dinamakan Hungarian Style dan bertujuan memudahkan kita dalam mengelola file aplikasi).
4. Tambahkan beberapa komponen data controls DBEdit dan DBNavigator ke dalam GolonganFrm sehingga anda memperoleh tata letak seperti Gambar 6.15. Atur agar properti masing-masing data control seperti pada Tabel 6.4.



**Gambar 7.70. Tata letak form GolonganFrm.**

**Tabel 7-15. Properti data controls form GolonganFrm.**

Komponen : DBGolongan		Komponen : DBGaPok	
Properti	Isi	Properti	Isi
DataSource	DataPersonalia.DSPegawai	DataSource	DataPersonalia.DSPegawai
DataField	Golongan	DataField	GaPok

- Aktifkan form GolonganFrm, pilih DBNavigator1 dan atur melalui properti VisibleButtons dari sehingga hanya muncul tombol nbFirst, nbPrior, nbNext, nbLast dan nbRefresh. (Gambar 6.16)



**Gambar 7.71. Mengatur tombol DBNavigator1**

- Sekarang, kita akan menghubungkan MainFrm ke form GolonganFrm. Pindah atau aktifkan MainFrm dan kemudian gunakan menu File | Use Unit | GolonganFrm untuk menghubungkan form MainFrm ke GolonganFrm.
- Klik menu Berkas di MainFrm dan kemudian klik sub menu Golongan. Isi event handler OnClick dari Golongan1 seperti di Listing 6.x. Perintah GolonganFrm.ShowModal akan membuat form GolonganFrm ditampilkan, tetapi

sebelum itu tabel Golongan dibuka terlebih dahulu melalui perintah `DataPersonalia.GolonganTbl.Open`.

**Listing 7-13. Event handler OnClick dari Golongan1**

```
procedure TMainFrm.Golongan1Click(Sender: TObject);
begin
    //cek apakah GolonganTbl sudah dibuka, jika belum buka pegawaitbl
    if DataPersonalia.GolonganTbl.Active = False then
        DataPersonalia.GolonganTbl.Open;

    //tampilkan form pegawai
    GolonganFrm.ShowModal;
end;
```

Latihan berikut ini akan menunjukkan kepada anda perintah-perintah yang digunakan oleh Delphi untuk memasukkan data, menghapus data dan mengubah data. Perintah-perintah yang berhubungan dengan Sisip, Ubah dan Hapus diperlihatkan di Tabel 6.5.

**Tabel 7-16. Perintah-perintah IUD (insert, update, delete)**

Perintah	Keterangan	Events
Insert	Menyisipkan record pada posisi record saat ini	OnNewRecord, BeforeInsert, AfterInsert
Append	Menambah record di urutan terakhir	OnNewRecord
Post	Menyimpan record	OnPostError, BeforePost, AfterPost
Edit	Mengubah record	BeforeEdit, AfterEdit, OnEditError
Cancel	Membatalkan edit atau insert atau append	BeforeCancel, AfterCancel
Delete	Menghapus record saat ini	BeforeDelete, AfterDelete, OnDeleteError

- Aktifkan `GolonganFrm` Hubungkan `DataModulePersonalia` ke `GolonganFrm` melalui menu `File | Use Unit | DataModulePersonalia`.
- Double-click tombol `TambahBtn` untuk membuat event handler `OnClick` dari `TambahBtn` dan tulis seperti Listing 6-5.

**Listing 7-14. Event handler OnClick dari TambahBtn**

```
procedure TPegawaiFrm.TambahBtnClick(Sender: TObject);
begin
    //tambah satu record baru
    DataPersonalia.PegawaiTbl.Append;

    //ubah tampilan tombol, hanya tombol simpan yang aktif
    TambahBtn.Enabled:=False;
    SimpanBtn.Enabled:=True;
    UbahBtn.Enabled:=False;
    HapusBtn.Enabled:=False;
    BatalBtn.Enabled:=True;
end;
```

10. Lakukan hal yang sama untuk membuat event handler OnClick dari SimpanBtn, HapusBtn, UbahBtn dan BatalBtn seperti Listing 6-6.

**Listing 7-15. Event handler OnClick dari SimpanBtn, HapusBtn, UbahBtn dan BatalBtn**

```
procedure TGolonganFrm.SimpanBtnClick(Sender: TObject);
begin
    //simpan permanen
    DataPersonalia.PegawaiTbl.Post;

    //ubah status tombol
    SimpanBtn.Enabled:=False;
    UbahBtn.Enabled:=True;
    HapusBtn.Enabled:=True;
    TambahBtn.Enabled:=True;
    BatalBtn.Enabled:=False;
end;

procedure TGolonganFrm.UbahBtnClick(Sender: TObject);
begin
    //masuk ke mode Edit
    DataPersonalia.PegawaiTbl.Edit;

    //ubah status tombol
    TambahBtn.Enabled:=False;
    SimpanBtn.Enabled:=True;
    UbahBtn.Enabled:=False;
    HapusBtn.Enabled:=False;
    BatalBtn.Enabled:=True;
end;

procedure TGolonganFrm.HapusBtnClick(Sender: TObject);
begin
    //hapus record saat ini
    DataPersonalia.PegawaiTbl.Delete;
end;

procedure TGolonganFrm.BatalBtnClick(Sender: TObject);
begin
    //Batal perintah sebelumnya, kecuali Delete
    DataPersonalia.PegawaiTbl.Cancel;

    //ubah status tombol
    TambahBtn.Enabled:=True;
    SimpanBtn.Enabled:=False;
    UbahBtn.Enabled:=True;
    HapusBtn.Enabled:=True;
    BatalBtn.Enabled:=False;
end;
```

11. Buat pula event handler OnActivate dari GolonganFrm untuk mengatur status tombol saat form diaktifkan seperti pada Listing 6-7.

#### Listing 7-16. Event handler OnActivate dari PegawaiFrm

```
procedure TGolonganFrm.FormActivate(Sender: TObject);
begin
    TambahBtn.Enabled:=True;
    SimpanBtn.Enabled:=False;
    UbahBtn.Enabled:=True;
    HapusBtn.Enabled:=True;
    BatalBtn.Enabled:=False;
end;
```

### Menguji input

Secara fungsional form GolonganFrm yang baru saja kita buat sudah dapat berfungsi dengan baik, dimana kita dapat melakukan fungsi Insert (Menambah), Delete (Hapus) dan Update (Edit), tetapi ada satu kelemahan dari form GolonganFrm tersebut, yaitu : **tidak ada pengujian terhadap input yang dimasukkan**. Anda dapat memasukkan sembarang angka untuk golongan dan program tidak menguji apakah angka golongan yang anda masukkan merupakan angka yang sah atau tidak.

Ada dua metoda yang dapat anda gunakan untuk menguji input yaitu :

1. Menggunakan program dan perintah pengujian (if atau case).
2. Menggunakan isi field dari tabel lain (*lookup*) sebagai contoh input yang sah.

Kita akan menggunakan metoda menguji keabsahan input menggunakan program untuk menguji keabsahan golongan.

### Menguji input secara program

Ada dua cara yang dapat anda gunakan untuk menguji keabsahan input secara program, yaitu :

1. Melalui event OnExit dari masing-masing data control.
2. Melalui event BeforePost dari komponen dataset.

Masing-masing cara mempunyai keuntungan dan kerugiannya sendiri. Menggunakan event OnExit memberikan pengendalian yang lebih ketat karena anda memaksa operator memasukkan data yang sah sebelum masuk ke data control berikutnya, tetapi pengujian dilakukan di berbagai tempat sehingga menyulitkan pemeliharaan program. Menggunakan event BeforePost memberi keuntungan pengendalian yang terpusat tetapi membuat anda lepas dari data control.

Pada kesempatan ini saya akan menunjukkan kedua cara tersebut karena kedua-duanya perlu anda ketahui, mana yang nantinya anda pilih untuk digunakan dalam proyek anda saya persilahkan anda menilai sendiri. Tetapi sebelum kita membahas teknik pengujian tersebut ada satu hal yang perlu anda ketahui yaitu exceptions.

## Exceptions

*Exceptions* adalah situasi yang muncul karena terjadi kesalahan ketika sebuah atau sekelompok perintah dijalankan. Kondisi *exception* tersebut ditangani secara khusus melalui *Exception block*. Ada 2 macam *exception block*, yaitu :

1. *try statements except exceptionBlock end.*

Apabila perintah di bagian *statements* menyebabkan munculnya kesalahan atau diistilahkan *raising an error* maka program akan melompat ke bagian *except* sesuai dengan jenis kesalahan yang ingin ditangkap. Sebagai contoh :

```
09:
10: try
11:   X := Y/Z;
12: except
13:   on EZeroDivide do ShowMessage('Divide by zero');
14: end;
15:
```

apabila Z berisi angka selain 0 saat baris 11 dieksekusi maka tidak ada kesalahan terjadi dan program akan melompat ke baris 15, tetapi apabila Z berisi angka 0 saat baris 11 dieksekusi maka akan muncul kesalahan (*error*) pembagian dengan 0 dan menyebabkan sistem mengirim pesan *exectopn EZeroDivide*, pesan kesalahan ini menyebabkan program melompat ke baris 13 dan kemudian menjalankan perintah *ShowMessage*.

2. *try statement1 finally statement2 end.*

Blok *try..finally ..end* digunakan untuk memastikan bahwa perintah-perintah di *statements2* dijalankan tidak peduli apakah perintah di *statement1* memunculkan *exceptions* atau tidak. Sebagai contoh :

```
09:
10: try
11:   Table1.Open;
12:   while not EOF do
13:     ....
14:     .....
15:   : .....
16:
17: 20: finally
18: 21:   Table1.Close;
19: 22: end;
20:
21: 23:
```

akan membuat program menjalankan perintah di baris 11 – 19, dan meskipun pada saat baris 11-19 dieksekusi muncul kesalahan, program akan melanjutkan ke baris 21. Dengan demikian apapun yang terjadi kita memastikan bahwa *Table1* pasti ditutup setelah perintah di baris 11-19 dieksekusi.

## Menguji input menggunakan event OnExit

Kita akan menggunakan komponen DBGolongan sebagai contoh pengujian keabsahan input menggunakan event OnExit.

1. Aktifkan form GolonganFrm.
2. Pilih komponen DBGolongan di form.
3. Aktifkan Object Inspector (F11) dan kemudian pilih tab Events.
4. Double-click isian di samping event OnExit untuk membuat event handler OnExit.
5. Tulis Listing 6-8 sebagai event handler OnExit dari DBGolongan. Kita akan menguji apakah isi dari DBGolongan merupakan angka dengan cara mengubah text menjadi angka menggunakan perintah `IntToStr`, apabila perintah ini memunculkan kesalahan maka kita menganggap bahwa isian keliru.

**Listing 7-17. Event handler OnExit dari DBNIP**

```
procedure TGolonganFrm.DBGolonganExit(Sender: TObject);
var
  value:integer;
begin
  //jika DBGolongan.Text tidak kosong maka
  if DBGolongan.Text <> '' then
  begin
    //cek apakah isi DBGolongan berupa angka dan tidak ada karakter
    try
      value:=StrToInt(DBGolongan.Text);
    except
      ShowMessage('Golongan harus berupa angka bulat');
      //kembali ke DBGolongan
      ActiveControl:=DBGolongan;
    end; //try
  end; //if DBGolongan.Text <> ''
end;
```

6. Kompilasi program dan jalankan, tambahkan satu record dan coba isi DBGolongan dengan angka maupun karakter bukan angka.

Untuk tiap data control yang membutuhkan pengujian keabsahan data input, anda dapat menggunakan metoda seperti Listing 6-8.

## Menguji input menggunakan BeforePost

Pengujian keabsahan input menggunakan event *BeforePost* hanya dapat dilakukan apabila perintah `Post` diletakkan di dalam *exception block*.

Latihan berikut ini akan menunjukkan bagaimana menguji input menggunakan event *BeforePost*. Kita akan memastikan bahwa sebelum record baru disimpan, semua field tidak kosong. Apabila ada field yang kosong maka dimunculkan *exception* `Abort` yang mengakibatkan munculnya event `OnPostError`. Event handler `OnPostError` mempunyai pola sebagai berikut :

```

procedure TDataPersonalia.GolonganTblPostError(DataSet: TDataSet;
  E: EDatabaseError; var Action: TDataAction);
begin
end;

```

dimana DataSet berisi komponen dataset yang terkait dengan event ini, E merupakan exception yang berisi pesan kesalahan, dan Action merupakan parameter yang menentukan apa yang akan dilakukan dengan kesalahan tersebut. Action dapat diisi dengan salah satu dari nilai di Tabel 6.x

**Tabel 7-17. Nilai Action**

Action	Keterangan
daFail	Batalkan operasi dan tampilkan pesan kesalahan
daAbort	Batalkan operasi tanpa menampilkan pesan kesalahan
daRetry	Ulangi operasi yang menyebabkan kesalahan. Kesalahan harus dibetulkan terlebih dahulu di event handler sebelum mengembalikan nilai ini

Event OnPostError digunakan untuk membatalkan perintah Post dengan mengisi parameter Action dengan daAbort.

1. Aktifkan form PegawaiFrm.
2. Ubah event handler OnClick dari SimpanBtn menjadi seperti Listing 6-9.

**Listing 7-18. Perubahan event handler OnClick dari SimpanBtn**

```

procedure TGolonganFrm.SimpanBtnClick(Sender: TObject);
begin
  try
    DataPersonalia.GolonganTbl.Post;
  finally
    TambahBtn.Enabled:=True;
    SimpanBtn.Enabled:=False;
    UbahBtn.Enabled:=True;
    HapusBtn.Enabled:=True;
    BatalBtn.Enabled:=False;
  end;
end;

```

3. Aktifkan form DataPersonalia.
4. Gunakan menu File | Use Unit | GolonganFrm untuk menghubungkan DataPersonalia dengan form GolonganFrm, karena kita akan menggunakan komponen data control di GolonganFrm.
5. Pilih komponen tabel GolonganTbl dan kemudian pindah ke Object Inspector.
6. Pilih tab Events dan buat event handler BeforePost dengan melakukan double-click pada isian BeforePost.

7. Tulis Listing 6-9 sebagai event handler BeforePost dan Listing 6-10 sebagai event handler OnPostError.

**Listing 7-19. Event handler BeforePost dari GolonganTbl**

```
procedure TDataPersonalia.GolonganTblBeforePost(DataSet: TDataSet);
var
  Valid:Boolean;
begin
  //cek isi masing-masing data control tidak kosong
  with GolonganFrm do
  begin
    Valid:=(DBGolongan.Text <> '') and (DBGaPok.Text <> '');
    if not Valid then
    begin
      ShowMessage('Semua field harus diisi, '
        +' tidak boleh ada yang kosong');
      //batalan perintah, raise exception Abort
      Abort;
    end; //if not Valid
  end; //with PegawaiFrm
end;
```

**Listing 7-20. Event handler OnPostError dari GolonganTbl**

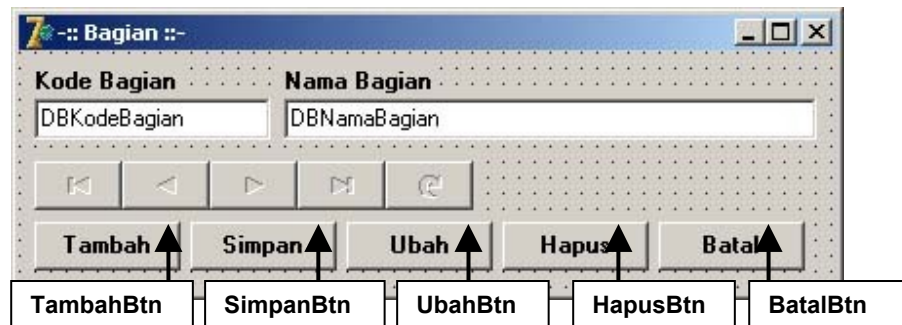
```
procedure TDataPersonalia.GolonganTblPostError(DataSet: TDataSet;
  E: EDatabaseError; var Action: TDataAction);
begin
  Action:=daAbort;
end;
```

8. Kompilasi program dan jalankan. Cobalah menambah record dan kemudian menyimpan record baru dengan salah satu field dikosongkan / tidak diisi, anda akan mendapat pesan 'Semua field harus diisi, tidak boleh ada yang kosong' dan kemudian proses menambah record dibatalkan.

Kita sudah mempelajari bagaimana menggunakan event untuk menangani pengujian keabsahan input sebelum input tersebut disimpan, berikut ini kita akan menguji keabsahan input menggunakan field dari tabel lain..

**Mengolah data Bagian**

1. Buat sebuah form baru dan simpan dengan nama BagianForm.pas
2. Tambahkan dua buah Label, dua buah DBEdit dan empat buah Button seperti Gambar 6.17 dan gunakan Tabel 6.7 untuk mengatur properti dari DBEdit dan Button.



**Gambar 7.72. Tata letak BagianFrm**

**Tabel 7-18. Properti komponen di BagianFrm**

Komponen : DBKodeBagian		Komponen : DBNamaBagian	
Properti	Isi	Properti	Isi
DataSource	DataPersonalial.DSBagian	DataSource	DataPersonalial.DSBagian
DataField	KodeBagian	DataField	Nama
Komponen : DBNavigator			
Properti	Isi		
DataSource	DataPersonalial.DSBagian		

3. Buat event handler OnClick dari TambahBtn, SimpanBtn, HapusBtn, UbahBtn dan BatalBtn seperti Listing 6-12.

**Listing 7-21. Event handler OnClick tombol-tombol dari form BagianFrm**

```

procedure TBagianFrm.TambahBtnClick(Sender: TObject);
begin
  DataPersonalial.BagianTbl.Append;
  TambahBtn.Enabled:=False;
  SimpanBtn.Enabled:=True;
  UbahBtn.Enabled:=False;
  HapusBtn.Enabled:=False;
  BatalBtn.Enabled:=True;
end;

procedure TBagianFrm.SimpanBtnClick(Sender: TObject);
begin
  try
    DataPersonalial.BagianTbl.Post;
  finally
    TambahBtn.Enabled:=True;
    SimpanBtn.Enabled:=False;
    UbahBtn.Enabled:=True;
    HapusBtn.Enabled:=True;
    BatalBtn.Enabled:=False;
  end;
end;

```

```

procedure TBagianFrm.UbahBtnClick(Sender: TObject);
begin
  DataPersonalia.BagianTbl.Edit;
  TambahBtn.Enabled:=False;
  SimpanBtn.Enabled:=True;
  UbahBtn.Enabled:=False;
  HapusBtn.Enabled:=False;
  BatalBtn.Enabled:=True;
end;

procedure TBagianFrm.HapusBtnClick(Sender: TObject);
begin
  DataPersonalia.BagianTbl.Delete;
  TambahBtn.Enabled:=True;
  SimpanBtn.Enabled:=False;
  UbahBtn.Enabled:=True;
  HapusBtn.Enabled:=True;
  BatalBtn.Enabled:=False;
end;

procedure TBagianFrm.BatalBtnClick(Sender: TObject);
begin
  DataPersonalia.BagianTbl.Cancel;
  TambahBtn.Enabled:=True;
  SimpanBtn.Enabled:=False;
  UbahBtn.Enabled:=True;
  HapusBtn.Enabled:=True;
  BatalBtn.Enabled:=False;
end;

```

4. Buat event handler OnActivate dari BagianFrm seperti Listing 6-13.

#### **Listing 7-22. Event handler OnActivate dari BagianFrm**

```

procedure TBagianFrm.FormActivate(Sender: TObject);
begin
  TambahBtn.Enabled:=True;
  SimpanBtn.Enabled:=False;
  UbahBtn.Enabled:=True;
  HapusBtn.Enabled:=True;
  BatalBtn.Enabled:=False;
end;

```

5. Aktifkan MainFrm dan buat event handler OnClick dari sub menu Bagian seperti Listing 6-14.

#### **Listing 7-23. Event handler OnClick dari sub menu Bagian di MainFrm**

```

procedure TMainFrm.Bagian1Click(Sender: TObject);
begin
  //cek apakah BagianTbl sudah dibuka
  if not DataPersonalia.BagianTbl.Active then
    DataPersonalia.BagianTbl.Open;

  //tampilkan form BagianFrm
  BagianFrm.ShowModal;
end;

```

## Mengolah data Pegawai

1. Tambahkan satu dataset Table dan satu dataSource di DataPersonalia. Atur agar properti DataSource1 dan dataset Tabel1 berisi seperti Tabel 6.7.

**Tabel 7-19. Properti Table1**

**Komponen : Table1**

Properti	Isi
DatabaseName	DBPersonalia
TableName	Pegawai.db
Name	PegawaiTbl

**Komponen : DataSource1**

Properti	Isi
DataSet	PegawaiTbl
Name	DSPegawai

2. Dengan menggunakan menu File | New | Form buat sebuah form dan kemudian atur agar properti Name berisi PegawaiFrm. Simpan form sebagai PegawaiForm.pas.
3. Aktifkan MainForm dan kemudian buat event handler OnClick dari sub menu Pegawai seperti Listing 6-15.

**Listing 7-24. Event handler OnClick dari sub menu Pegawai**

```
procedure TMainFrm.PegawaiClick(Sender: TObject);
begin
    //cek apakah PegawaiTbl sudah dibuka, jika belum buka tabel
    if DataPersonalia.PegawaiTbl.Active = False then
        DataPersonalia.PegawaiTbl.Open;

    //cek apakah GolonganTbl sudah dibuka, jika belum tabel
    if DataPersonalia.GolonganTbl.Active = False then
        DataPersonalia.GolonganTbl.Open;

    //tampilkan form pegawai
    PegawaiFrm.ShowModal;
end;
```

4. Hubungkan DataModulePersonalia dengan PegawaiFrm melalui menu File | Use Unit | DataModulePersonalia.
5. Tambahkan beberapa data control, dbnavigator dan button seperti Gambar 6.18. Gunakan cara yang sama saat membuat form GolonganFrm untuk mengatur tombol dari DBNavigator.

**Gambar 7.73. Form PegawaiFrm.**

6. Gunakan Tabel 6.9 untuk mengatur properti dari data control di PegawaiFrm.

**Tabel 7-20. Properti data control dari PegawaiFrm.**

Komponen : DBNIP	
Properti	Isi
DataSource	DataPersonalia.DSPegawai
DataField	NIP

Komponen : DBNama	
Properti	Isi
DataSource	DataPersonalia.DSPegawai
DataField	Nama

Komponen : DBTglMasuk	
Properti	Isi
DataSource	DataPersonalia.DSPegawai
DataField	TglMasuk

Komponen : DBPosisi	
Properti	Isi
DataSource	DataPersonalia.DSPegawai
DataField	Posisi

Komponen : DBNavigator1	
Properti	Isi
DataSource	DataPersonalia.DSPegawai

7. Buat event handler OnClick dari masing-masing tombol seperti Listing 6-16.

**Listing 7-25. Event handler OnClick dari tombol-tombol di PegawaiFrm**

```
procedure TPegawaiFrm.TambahBtnClick(Sender: TObject);
begin
    DataPersonalia.PegawaiTbl.Append;
    TambahBtn.Enabled:=False;
    SimpanBtn.Enabled:=True;
```

```

    UbahBtn.Enabled:=False;
    HapusBtn.Enabled:=False;
    BatalBtn.Enabled:=True;
end;

procedure TPegawaiFrm.SimpanBtnClick(Sender: TObject);
begin
    try
        DataPersonalia.PegawaiTbl.Post;
    finally
        DataPersonalia.PegawaiTbl.Cancel;
        SimpanBtn.Enabled:=False;
        UbahBtn.Enabled:=True;
        HapusBtn.Enabled:=True;
        TambahBtn.Enabled:=True;
        BatalBtn.Enabled:=False;
    end;
end;

procedure TPegawaiFrm.UbahBtnClick(Sender: TObject);
begin
    DataPersonalia.PegawaiTbl.Edit;
    TambahBtn.Enabled:=False;
    SimpanBtn.Enabled:=True;
    UbahBtn.Enabled:=False;
    HapusBtn.Enabled:=False;
    BatalBtn.Enabled:=True;
end;

procedure TPegawaiFrm.HapusBtnClick(Sender: TObject);
begin
    DataPersonalia.PegawaiTbl.Delete;
end;

procedure TPegawaiFrm.BatalBtnClick(Sender: TObject);
begin
    DataPersonalia.PegawaiTbl.Cancel;
    TambahBtn.Enabled:=True;
    SimpanBtn.Enabled:=False;
    UbahBtn.Enabled:=True;
    HapusBtn.Enabled:=True;
    BatalBtn.Enabled:=False;
end;

```

8. Buat event handler untuk menangani event OnActivate dari PegawaiFrm seperti di Listing 6-17.

**Listing 7-26. Event handler OnActivate dari PegawaiFrm**

```

procedure TPegawaiFrm.FormActivate(Sender: TObject);
begin
    TambahBtn.Enabled:=True;
    SimpanBtn.Enabled:=False;
    UbahBtn.Enabled:=True;
    HapusBtn.Enabled:=True;
    BatalBtn.Enabled:=False;
end;

```

9. Buat event handler OnExit untuk data control DBNIP. Disini dilakukan pengujian untuk memastikan bahwa NIP menggunakan format P-nnn, dengan nnn berisi angka dari 0 sampai 9.

**Listing 7-27. Event handler OnExit dari DBNip untuk pengujian keabsahan input.**


```
procedure TPegawaiFrm.DBNIPExit(Sender: TObject);
var
  NIP:string;
  Valid:Boolean;
begin
  NIP := DBNip.Text;
  if NIP <> '' then
  begin
    //uji validitas input
    Valid:= (NIP[1] = 'P') and (NIP[2] = '-')
    and (NIP[3] in ['0'..'9']) and (NIP[4] in ['0'..'9'])
    and (NIP[5] in ['0'..'9']);

    //jika input tidak valid
    if not Valid then
    begin
      //tampilkan pesan kesalahan
      ShowMessage('NIP harus dalam format P-nnn, '
        +' dengan n antara 0 s/d 9');

      //kembali ke DBNip
      ActiveControl:=DBNip;
    end;
  end;
end;
```

### Menguji input menggunakan field dari tabel lain.

Langkah-langkah berikut ini akan menunjukkan kepada anda bagaimana memastikan bahwa isi data control DBGolongan akan selalu ada di tabel Golongan.db.

10. Tambahkan data control DBLookupComboBox  ke form PegawaiFrm sehingga anda memperoleh Gambar 6.19.



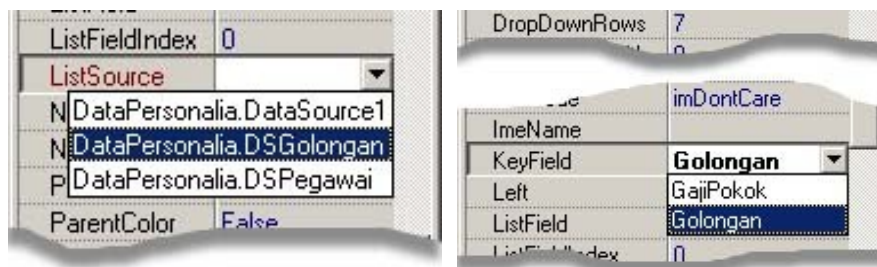
The screenshot shows a Windows form titled "Pegawai". It contains several data controls: "Nomor Induk Pegawai" (DBNIP), "Nama" (DBNama), "Tanggal Masuk" (DBTglMasuk), "Kode Bagian" (DBKodeBagian), "Posisi" (DBPosisi), and "Golongan" (DBGok). Below these controls are five buttons: "Tambah", "Simpan", "Ubah", "Hapus", and "Batal".

**Gambar 7.74. Data control DBLookupComboBox**

11. Dengan tetap memilih DBLookupComboBox1 gunakan Tabel 6.10 dan Gambar 6.20 untuk mengatur properti dari DBLookupComboBox1.

**Tabel 7-21. Properti DBLookupComboBox**

Properti	Isi
DataSource	DataPersonalia.DSPegawai
DataField	Golongan
ListSource	DataPersonalia.DSGolongan
KeyField	Golongan
Name	DBGolonganCmb



**Gambar 7.75. Properti ListSource dan KeyField**

12. Kompilasi program dan cobalah mengisi Golongan, anda hanya akan mendapatkan isian golongan seperti yang anda isi melalui GolonganFrm.

Fungsi properti ListSource adalah menghubungkan data control tersebut ke tabel yang akan digunakan untuk menentukan isi data control, sedangkan properti KeyField berisi field dari tabel penguji yang digunakan sebagai kunci untuk mencari isian data control. Data control nantinya hanya dapat diisi dengan isian yang berasal dari tabel penguji pada field yang ditentukan. Selain menggunakan DBLookupComboBox anda juga dapat menggunakan DBLookup ListBox untuk keperluan yang sama, perbedaannya hanya terletak di tampilan data control saja.

Bagaimana dengan data control KodeBagian ? Bukankah kita juga harus membatasi pengisian kode bagian hanya untuk data yang ada di tabel Bagian ?

13. Ganti DBKodeBagian dengan data control DBLookupComboBox seperti Gambar 6.21.



**Gambar 7.76. Penggantian komponen KodeBagian**

14. Atur agar properti dari DBLookupComboBox tersebut seperti Tabel 6.

**Tabel 7-22. Properti dari DBLookupKodeBagianCmb**

Properti	Isi
DataSource	DataPersonalia.DSPegawai
DataField	KodeBagian
ListSource	DataPersonalia.DSBagian
KeyField	KodeBagian
Name	DBLookupKodeBagianCmb

15. Kompilasi dan jalankan program.

## **BAB 7 MENCARI RECORD DAN MENYARING RECORD**

### **APA YANG AKAN KITA PELAJARI ?**

- Mencari record berdasarkan kriteria tertentu
- Menyaring record menggunakan relasi master-detail
- Menyaring record menggunakan filter

**WAKTU LATIHAN : 3 JAM**

## Mencari record

Ada 3 cara mencari record, yaitu :

1. FindKey
2. FindNearest
3. Locate
4. Lookup

### FindKey

Perintah FindKey digunakan untuk mencari record berdasarkan field index. FindKey akan mencari record yang isi field indexnya memenuhi kriteria yang diinginkan dan apabila record tersebut ditemukan maka posisi record aktif akan diletakkan pada record tersebut dan mengembalikan TRUE sedangkan apabila tidak menemukan record yang memenuhi kriteria maka FindKey akan mengembalikan FALSE dan posisi record berada di kondisi EOF. FindKey dituliskan sebagai :

```
function FindKey(const KeyValues: array of const): Boolean;
```

Dengan KeyValues merupakan array yang berisi kriteria yang diinginkan. Dalam contoh berikut ini Tabel Pegawai menggunakan satu buah index yaitu NIP. Perintah FindKey(['P-006']) akan menyebabkan posisi record berada di record nomor 5 dan FindKey mengembalikan TRUE sedangkan perintah FindKey(['P-005']) akan menyebabkan posisi record tidak berubah dan FindKey mengembalikan FALSE. Gambar 7.1 mengilustrasikan hal tersebut.



NIP	Nama	TglMasuk	Golongan	KodeBagian	Posisi
P-001	Edhi	28/09/2004	1	1	
P-002	Rudi	28/01/2002	1	1	1
P-003	Umi	28/09/2004	2	1	1
P-004	Rudi Susanto	31/01/2004	1	1	2
P-006	Ina	02/01/2003	1	1	2
P-009	Yuni	03/03/2004	1	1	1

Gambar 8.77. Isi Tabel Pegawai

### FindNearest

Perintah FindNearest akan meletakkan posisi record pada record pertama yang memenuhi kriteria yang diinginkan atau record pertama yang lebih besar dari kriteria yang ditentukan. Apabila dua kondisi tersebut tidak dapat ditemukan maka FindNearest akan meletakkan posisi record ke record terakhir. Perintah FindNearest mempunyai sintak :

```
procedure FindNearest(const KeyValues: array of const);
```

Dengan menggunakan Tabel Pegawai seperti Gambar 7.1, maka perintah FindNearest(['P-005']) akan menyebabkan posisi record berada di record nomor 5 atau menunjuk ke record

dengan NIP sama dengan 'P-006', sedangkan perintah `FindNearest(['P-003'])` akan menyebabkan posisi record berada record nomor 3.

## Locate

Berbeda dengan perintah `FindKey` dan `FindNearest` yang menggunakan field index sebagai tempat pencarian, perintah `Locate` tidak memerlukan field index dan dapat menggunakan sembarang field sebagai lokasi pencarian. Perintah `Locate` menggunakan sintak penulisan sebagai berikut :

```
function Locate(const KeyFields: String; const KeyValues: Variant; Options: TLocateOptions): Boolean;
```

Parameter *KeyFields* berisi nama field yang akan digunakan sebagai lokasi pencarian, apabila ada lebih dari satu field maka masing-masing field dipisah dengan tanda koma (;), parameter *KeyValues* merupakan array yang berisi kriteria pencarian, elemen-elemen di dalam array *KeyValues* harus sama dengan jumlah field dalam *KeyFields*. Parameter *Options* berisi metoda pencarian seperti Tabel 7.1.

**Tabel 8-23. Parameter Options pada Locate**

Options	Keterangan
loCaseInsensitive	Pencarian mengabaikan huruf besar dan kecil, sehingga 'MARET' , 'maret', 'Maret' akan dianggap sama.
loPartialKey	Kriteria pencarian dapat diisi dengan sebagian dari kata yang dicari, misalnya 'HAM' akan cocok dengan 'HAM', 'HAMMER', 'HAMBURGER'

Dengan menggunakan Tabel Pegawai seperti Gambar 7.1, perintah `Locate ('Nama','RUDI', [loCaseInsensitive])` akan menyebabkan posisi record berada di record nomor 2, perintah `Locate ('Nama','RUDI',[])` akan menyebabkan posisi record tidak berubah dan `Locate` mengembalikan nilai FALSE, perintah `Locate('Nama','In', [loPartialKey])` akan menyebabkan posisi record menuju ke record 5 dan `Locate` mengembalikan nilai TRUE, perintah `Locate('Nama','UM', [loCaseInsensitive, loPartialKey])` mengembalikan nilai TRUE dengan posisi record berada di nomor 3.

Apabila anda ingin menggunakan dua buah field sebagai lokasi pencarian maka parameter *KeyValues* harus berupa variant array dan dapat dibuat menggunakan perintah `VarArrayOf` seperti contoh berikut ini :

```
Locate ('NIP;Nama',VarArrayOf(['P-002','Rudi'],[]))
```

yang berarti mencari record dimana field NIP berisi 'P-002' dan field Nama berisi 'Rudi'.

## Lookup

Perintah `Lookup` sama seperti perintah `Locate` tetapi perintah ini tidak mengubah lokasi record aktif. Perintah ini dapat digunakan untuk memeriksa apakah ada record yang memenuhi kriteria yang diinginkan tanpa mengubah posisi record serta mengambil field

tertentu yang memenuhi kriteria tersebut. Perintah Lookup menggunakan sintak penulisan sebagai berikut :

```
function Lookup(const KeyFields: String; const KeyValues: Variant; const ResultFields: String): Variant;
```

Parameter *KeyFields* merupakan string yang berisi daftar nama field yang menjadi lokasi pencarian, apabila ada lebih dari satu field maka digunakan pemisah ; (titik koma), parameter *KeyValues* berisi kriteria pencarian dan apabila ada lebih dari satu field maka digunakan perintah *VarArrayOf* untuk membuat array variant, parameter *ResultFields* merupakan string yang berisi nama field yang akan dikembalikan sebagai hasil pencarian, dan apabila ada lebih dari satu field yang dikembalikan maka field-field tersebut dipisahkan dengan tanda ; (titik koma).

Dengan perintah :

```
1. var
2. V:Variant;
3. C:String;
4. begin
5. V:= Table1.Lookup('Nama','Rudi','NIP');
6. if not (VarType(V) in [varNull]) then
7. begin
8. C := V;
9. end
10. else
11. begin
12. C:="";
13. end;
14. end;
```

pada baris 5, perintah *Table1.Lookup ('Nama','Rudi','NIP')* akan mencari record yang field Nama berisi 'Rudi' dan apabila ditemukan akan mengembalikan isi field NIP dari record tersebut ke variabel V, tetapi apabila tidak ditemukan maka dikembalikan *varNull*. Baris 6 menguji apakah V bertipe *varNull* dan apabila tidak bertipe *varNull* maka pencarian sukses dan hasilnya diambil untuk disimpan ke variabel C (baris 8).

### ***Membuat aplikasi Presensi***

Aplikasi Presensi digunakan oleh karyawan untuk memasukkan data kehadirannya. Karyawan memasukkan Nip-nya dan kemudian program akan mencari apakah NIP tersebut tercatat di database, apabila ditemukan maka program akan menyimpan NIP, tanggal dan jam kehadiran ke tabel Presensi.

1. Buat aplikasi baru melalui menu File | New | Application.
2. Ubah nama Form1 menjadi PresensiPgwFrm dan simpan dengan nama PresensiPegawaiForm.pas serta Presensi.dpr sebagai nama proyek.

3. Buat form data module melalui menu File | New | Data Module
4. Ubah nama data module menjadi DataPresensiPgw dan simpan dengan nama DataPresensi.pas.
5. Tambahkan 1 datasource dan 1 dataset, kemudian atur properti dari dataset serta datasource tersebut seperti pada Tabel 7.x

**Tabel 8-24. Properti dataset dan datasource**

**Komponen : DataSource**

Properti	Isi
Name	DSPegawai

**Komponen : DataSet**

Properti	Isi
Name	PegawaiTbl
Databasename	DBPersonalia
TableName	Pegawai.Tbl

6. Tambahkan 1 datasource dan 1 dataset, kemudian atur properti dari dataset dan datasource tersebut seperti pada Tabel 7.3

**Tabel 8-25. Properti dataset dan datasource**

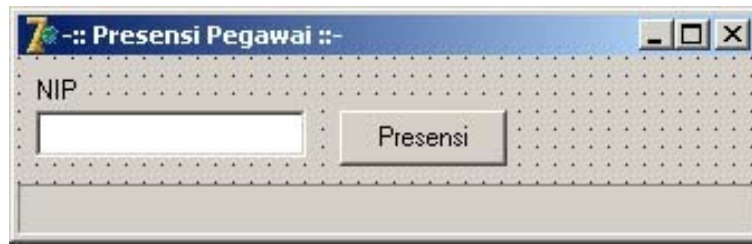
**Komponen : DataSource**

Properti	Isi
Name	DSPresensi

**Komponen : DataSet**

Properti	Isi
Name	PresensiTbl
Databasename	DBPersonalia
TableName	Presensi.Tbl

7. Pindah ke form PresensiPgwFrm.
8. Tambahkan 1 buah TEdit dan atur agar properti Name menjadi ENIP.
9. Tambahkan 1 buah TPanel dan atur agar properti Align dari TPanel tersebut menjadi alBottom.
10. Tambahkan 1 buah TButton dan atur agar properti Name menjadi PresensiBtn.
11. Atur agar lokasi komponen seperti Gambar 7.2



**Gambar 8.78. Tata letak PresensiPgwForm**

12. Hubungkan DataPresensi ke PresensiPgwFrm melalui menu File | Use Unit | DataPresensi.
13. Buat event handler OnClick dari tombol PresensiBtn seperti di Listing 7.x

**Listing 8-28. Event handler OnClick dari PresensiBtn**

```
procedure TPresensiPgwFrm.PresensiBtnClick(Sender: TObject);
var
  tgl:string;
begin
  //buka Tabel Presensi jika tidak aktif
  if not DataPresensiPgw.PegawaiTbl.Active then
    DataPresensiPgw.PegawaiTbl.Open;
  //buka Tabel Presensi jika tidak aktif
  if not DataPresensiPgw.PresensiTbl.Active then
    DataPresensiPgw.PresensiTbl.Open;

  //cek apakah nip ada di PegawaiTbl
  if DataPresensiPgw.PegawaiTbl.FindKey([ENIP.Text]) then
    begin
      //ubah tgl sekarang menjadi string
      DateTimeToString(tgl,'dd-mm-yyyy hh:nn:ss',Now());

      //NIP ditemukan, simpan NIP dan Tanggal Presensi ke tabel Presensi
      DataPresensiPgw.PresensiTbl.AppendRecord([ENIP.Text,Now(),Now()]);

      //tampilkan informasi kehadiran dari pegawai ini
      Panell1.Caption:=Format('%s - %s hadir pada %s ',[ENIP.Text,
        DataPresensiPgw.PegawaiTbl.FieldName('NAMA').AsString, tgl]);
    end
  else //NIP tidak ditemukan
    begin
      Panell1.Caption:=Format('%s tidak ditemukan di'
        +' database',[ENIP.Text]);
    end; //else if PegawaiTbl.Find
end;
```

14. Kompilasi program dan masukkan data presensi, cobalah memasukkan NIP yang tidak ada di tabel Pegawai.db untuk melihat apa yang terjadi.

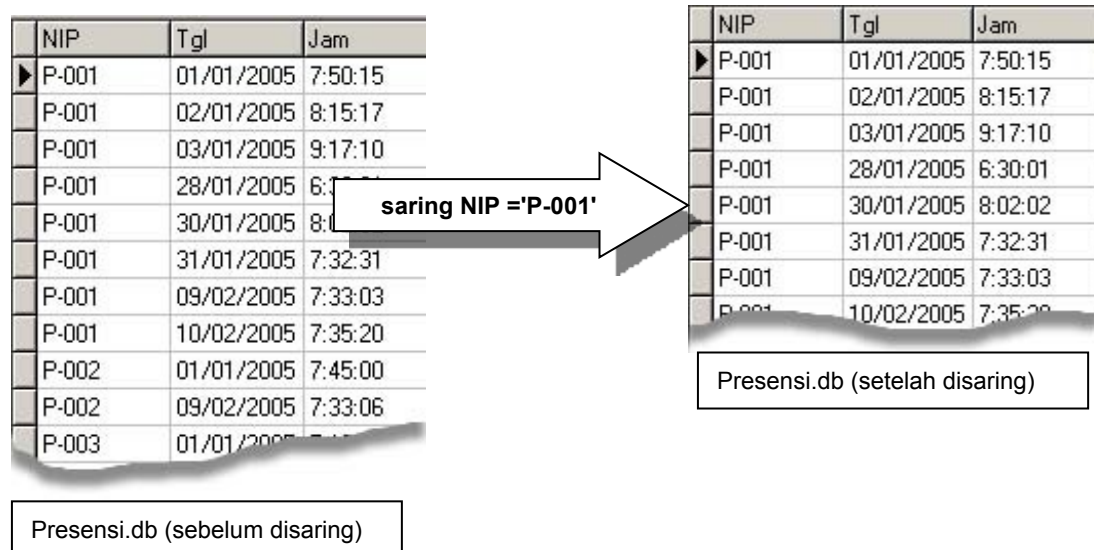
### **Menyaring record**

Menyaring record merupakan tindakan dimana kita hanya menampilkan record-record yang memenuhi kriteria tertentu. Misalnya, apabila kita hanya menampilkan record dari

Presensi.db yang field NIP-nya berisi 'P-001' maka kita akan memperoleh tampilan seperti Gambar 7.3.

Ada dua cara untuk menyaring record, yaitu :

1. Menggunakan field dari tabel lain (relasi Master-Detail)
2. Menggunakan properti Filter.



NIP	Tgl	Jam
P-001	01/01/2005	7:50:15
P-001	02/01/2005	8:15:17
P-001	03/01/2005	9:17:10
P-001	28/01/2005	6:30:01
P-001	30/01/2005	8:02:02
P-001	31/01/2005	7:32:31
P-001	09/02/2005	7:33:03
P-001	10/02/2005	7:35:20
P-002	01/01/2005	7:45:00
P-002	09/02/2005	7:33:06
P-003	01/01/2005	7:45:00

saring NIP = 'P-001'

NIP	Tgl	Jam
P-001	01/01/2005	7:50:15
P-001	02/01/2005	8:15:17
P-001	03/01/2005	9:17:10
P-001	28/01/2005	6:30:01
P-001	30/01/2005	8:02:02
P-001	31/01/2005	7:32:31
P-001	09/02/2005	7:33:03
P-001	10/02/2005	7:35:20

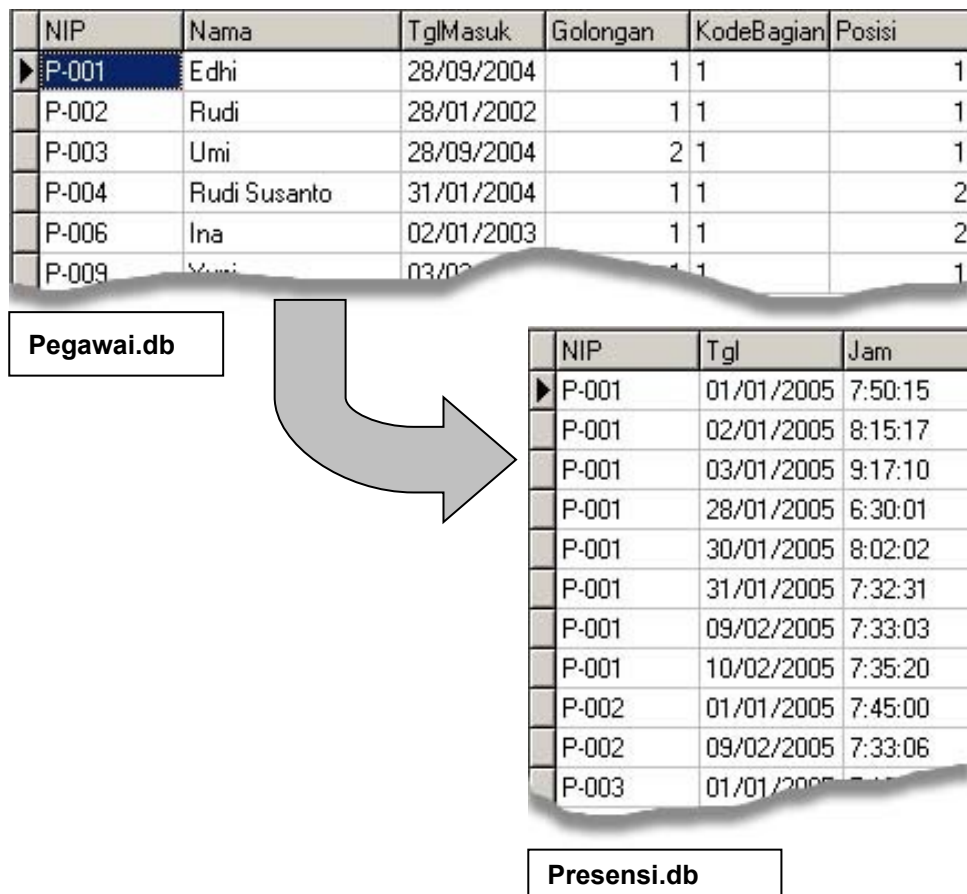
Presensi.db (sebelum disaring)

Presensi.db (setelah disaring)

**Gambar 8.79. Menyaring record**

### Relasi Master-Detail

Relasi master-detail atau relasi one-to-many adalah relasi antara dua tabel dimana satu record di salah satu tabel mempunyai relasi dengan beberapa record di tabel lain. Apabila record aktif di tabel master berubah maka record aktif yang dapat diakses di tabel detail juga mengikuti perubahan di tabel master. Sebagai contoh, apabila tabel Pegawai.db menjadi master sedangkan tabel Presensi.db menjadi detail maka meletakkan record aktif di posisi record dengan NIP sama dengan 'P-001' akan menyebabkan dataset yang mewakili Presensi.db hanya menampilkan / memberikan record-record dengan field NIP sama dengan 'P-001'. Tetapi apabila record aktif di Pegawai.db menunjuk ke record dengan NIP = 'P-006' maka dataset untuk Presensi.db akan berada di posisi EOF karena tidak ada record di Presensi.db yang mempunyai field NIP = 'P-006'.



**Gambar 8.80. Relasi Master-Detail**

Latihan berikut ini akan memanfaatkan relasi master-detail untuk menampilkan data presensi sesuai dengan NIP Pegawai.

1. Ambil aplikasi Sistem Personalia dengan File | Open Project | SistemPersonalia.dpr
2. Tambahkan satu buah dataset dan satu datasource di DataPersonalia, dataset dan datasource tersebut akan kita gunakan untuk mewakili Presensi.db. Atur agar properti dari dataset dan datasource tersebut seperti Tabel 7.4

**Tabel 8-26. Isi properti dataset dan datasource untuk Presensi.db**

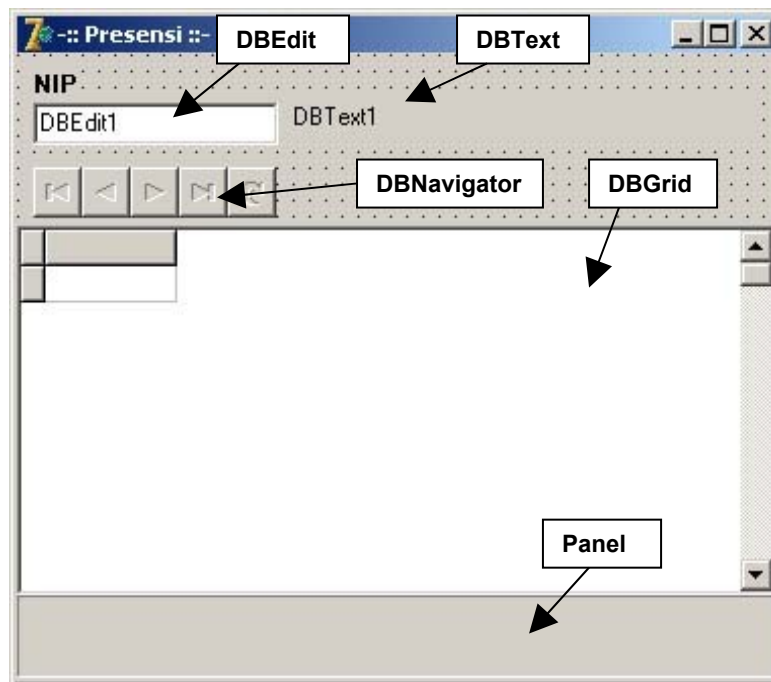
**Komponen : DataSource**

Properti	Isi
Name	DSPresensi

**Komponen : DataSet**

Properti	Isi
Name	PresensiTbl
Databasename	DBPersonalia
TableName	Presensi.Tbl

3. Buat form baru dan atur agar nama form berisi PresensiFrm serta disimpan sebagai PresensiForm.pas.
4. Hubungkan PresensiFrm dengan DataModulePersonalia melalui menu File | Use Unit | DataModulePersonalia.
5. Tambahkan DBEdit, DBText, DBNavigator, DBGrid serta TPanel sehingga anda memperoleh tata letak seperti Gambar 7.5. Atur agar properti dari masing-masing komponen seperti Tabel 7.x.



**Gambar 8.81. Tata letak PresensiFrm**

**Tabel 8-27. Properti dari komponen-komponen di PresensiFrm.**

**Komponen : DBEdit1**

Properti	Isi
Name	DBNip
DataSource	DataPersonalia.DSPegawai
DataField	NIP

**Komponen : DBText1**

Properti	Isi
Name	DBText1
DataSource	DataPersonalia.DSPegawai
DataField	Nama

**Komponen : DBNavigator1**

Properti	Isi
Name	DBNavigator1
DataSource	DataPersonalia.DSPegawai

---


**Komponen : DBGrid1**

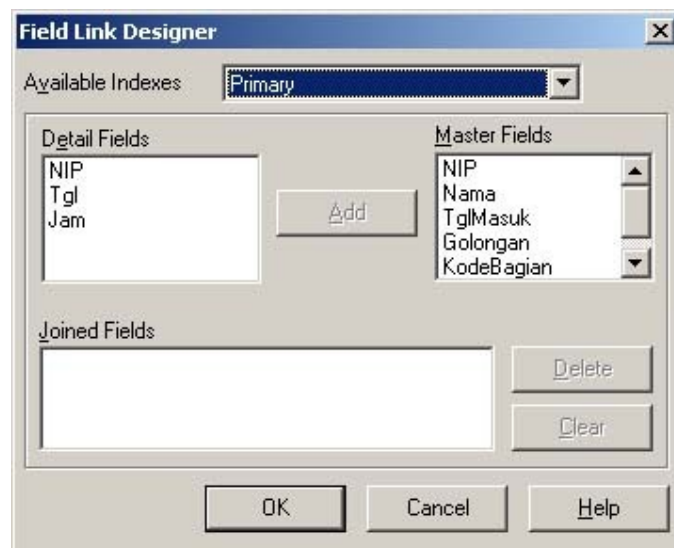
Properti	Isi
Name	DBGrid1
DataSource	DataPersonalia.DSPegawai

- Atur agar properti VisibleButtons dari DBNavigator1 hanya menampilkan tombol nbFirst, nbPrior, nbNext, nbLast.
- Pindah ke form DataModulePersonalia dan pilih komponen PresensiTbl.
- Aktifkan Object Inspector (F11) dan atur agar properti MasterSource dari PresensiTbl berisi DSPegawai.



**Gambar 8.82. MasterSource dari PresensiTbl**

- Klik kotak kosong disamping properti MasterFields dan kemudian double-click tombol  sehingga anda memperoleh tampilan jendela Field Link Designer (Gambar 7.7).



**Gambar 8.83. Field Link Designer**

10. Pilih NIP pada Detail Fields dan NIP pada Master Fields (Gambar 7.8).



**Gambar 8.84 Membuat Relasi Master-Detail**

11. Klik tombol Add sehingga Joined Fields berisi seperti Gambar 7.9.



**Gambar 8.85. Menghubungkan field NIP dari PresensiTbl ke NIP dari PegawaiTbl**

12. Klik tombol OK, terlihat sekarang MasterFields berisi NIP dan IndexField berisi NIP (Gambar 7.10)



**Gambar 8.86. Relasi Master-Detail PegawaiTbl -> PresensiTbl.**

13. Pindah ke MainForm dan hubungkan MainForm ke PresensiFrm melalui menu File | Use Unit | PresensiFrm.
14. Buat event handler OnClick untuk sub menu Presensi seperti pada Listing 7-2.

### Listing 8-29. Event handler OnClick untuk sub menu Presensi1

```
procedure TMainFrm.Presensi1Click(Sender: TObject);
begin
  //cek apakah PresensiTbl sudah dibuka, jika belum buka tabel
  if not DataPersonalia.PresensiTbl.Active then
    DataPersonalia.PresensiTbl.Open;

  //cek apakah PegawaiTbl sudah dibuka, jika belum buka tabel
  if not DataPersonalia.PegawaiTbl.Active then
    DataPersonalia.PegawaiTbl.Open;

  //tampilkan form PresensiFrm
  PresensiFrm.ShowModal;
end;
```

15. Kompilasi program, pilih menu Presensi dan kemudian lakukan navigasi, perhatikan setiap kali NIP dari pegawai berubah maka DBGrid hanya menampilkan record presensi yang NIP-nya sesuai dengan Pegawai.

### Menyaring record menggunakan properti Filter

Menyaring menggunakan properti Filter dilakukan apabila kita ingin menampilkan record-record berdasarkan kriteria tertentu. Misal, kita ingin menampilkan record presensi yang mempunyai tanggal presensi di antara tanggal 12/02/2005 sampai dengan 25/02/2005, atau menampilkan record pegawai yang golongannya sama dengan 1, dan sebagainya.

Secara umum, untuk menyaring record menggunakan properti filter dilakukan dengan langkah-langkah berikut ini :

1. Isi properti filter dengan pernyataan-kondisional.
2. Atur opsi-opsi dari penyaringan (Tabel 7.7).
3. Atur agar properti Filtered bernilai TRUE, nilai FALSE menyatakan bahwa record tidak disaring.

Pernyataan kondisional adalah pernyataan dengan bentuk "Field operator syarat", misalnya : NIP = 'P-001', dimana NIP merupakan Field, tanda = merupakan operator dan 'P-001' merupakan syarat. Anda dapat menghubungkan lebih dari satu pernyataan-kondisional melalui operator AND, OR, NOT. Operator yang dapat digunakan beserta arti dari masing-masing operator diperlihatkan pada Tabel 7.6.

**Tabel 8-28. Operator Penyaringan**

Operator	Contoh	Keterangan
<	Golongan < 1	Menampilkan record yang field Golongan berisi angka lebih kecil dari 1.
<=	Golongan <= 1	Menampilkan record yang field Golongan berisi angka lebih kecil dari 1 atau sama dengan 1.
>	Golongan > 3	Menampilkan record yang field Golongan berisi angka lebih besar dari 3.

>=	Golongan >= 4	Menampilkan record yang field Golongan berisi angka lebih besar dari 4 atau sama dengan 4.
=	Tipe = 'A'	Menampilkan record yang field Tipe berisi karakter 'A'.
<>	sks <> 3	Menampilkan record yang field sks tidak berisi angka 3.
AND	(sks <= 3) and (tipe = 'P')	Menampilkan record yang field sks berisi angka lebih kecil atau sama dengan 3 dan field tipe berisi karakter 'P'.
OR	(Golongan <> 3) or (Posisi <> 'P')	Menampilkan record yang field Golongan tidak berisi angka 3 atau record yang field Posisi tidak berisi karakter 'P'.
NOT	NOT (Golongan = 3)	Menampilkan record yang field Golongan tidak berisi angka 3
*	Nama = 'M*'	Menampilkan record yang field Nama diawali dengan huruf 'M', misalnya 'MARDI', 'MONA', 'MIRNA'.
	Nama = 'PA*'	Menampilkan record yang field Nama diawali dengan huruf 'PA', misalnya 'PARDI', 'PARNO', 'PARTO', tetapi 'PUDJI', 'PERNE', 'PORNA' tidak akan ditampilkan

Hasil penyaringan juga dipengaruhi oleh properti FilterOptions seperti Tabel 7.7.

**Tabel 8-29. FilterOptions**

Options	Keterangan
foCaseInsensitive	Filter tidak dipengaruhi oleh huruf besar kecil
foNoPartialCompare	Apabila FilterOptions tidak berisi foNoPartialCompare maka asterik (*) diperlakukan sebagai karakter <i>wildcard</i> , sedangkan apabila berisi foNoPartialCompare maka karakter asterik (*) diperlakukan sebagai karakter biasa

Kita akan menggunakan metoda menyaring record menggunakan filter untuk menampilkan record presensi pada bulan tertentu. Untuk itu kita harus menyaring record menggunakan field Tgl dari tabel Presensi.db, dimana kita akan menyaring record yang memenuhi kondisi tgl >= tanggal 1 bulan tertentu dan tgl <= tgl terakhir dari bulan tertentu. Misalnya, apabila kita ingin menampilkan record presensi pada bulan Januari tahun 2005 maka pernyataan kondisional akan menjadi '(tgl >= '01/01/2005') and (tgl <= '31/01/2005)', tetapi untuk bulan Pebruari 2005 maka pernyataan kondisional akan menjadi '(tgl >= '01/02/2005') and (tgl <= '28/02/2005)'.

Bagaimana kita mengetahui tanggal terakhir dari bulan tertentu ? Algoritma untuk menentukan tanggal terakhir dari bulan tertentu diperlihatkan pada Listing 7-3.

### Listing 8-30. Algoritma menentukan tanggal terakhir bulan tertentu

```
//buat tanggal 1/Bulan/Tahun dalam string
TglAwalStr:=Format('%d/%d/%d',[1,Bulan,Tahun]);

//ubah tanggal dalam string menjadi bertipe TDate
TglAwal:= StrToDateTime(TglAwal);

//hitung jumlah hari dalam bulan di TglAwal
JumlahHariBulan:=DaysInMonth(TglAwal);

//buat tanggal terakhir dalam bulan tersebut dalam string
TglAkhir:=Format('%d/%d/%d',[JumlahHariBulan,Bulan,Tahun]);
```

Setelah tanggal awal dan tanggal terakhir diketahui maka saringan dapat disusun menggunakan perintah seperti pada Listing 7-4.

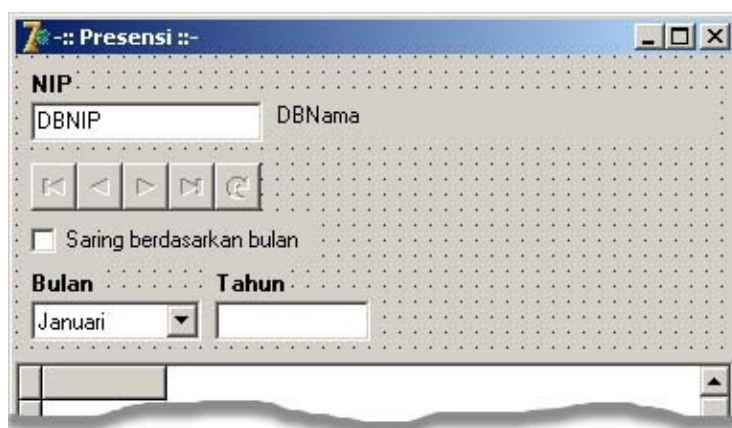
### Listing 8-31. Algoritma membuat saringan berdasarkan tanggal awal dan tanggal akhir

```
FilterStr:=Format('(Tgl >= %s) and (Tgl <= %s)',[QuotedStr(TglAwal),
QuotedStr(TglAkhir)]);
```

Perhatikan kita menggunakan perintah QuotedStr untuk membuat string yang berada di dalam tanda petik. Misalnya, perintah 'Nama : '+QuotedStr('Edhi') akan menghasilkan string 'Nama : 'Edhi' '.

Latihan berikut ini akan menunjukkan bagaimana menyaring record presensi berdasarkan tanggal presensi.

1. Pindah ke form PresensiFrm.
2. Tambahkan satu buah ComboBox, satu buah CheckBox dan satu buah Edit. Atur agar tata letak dari komponen-komponen tersebut seperti Gambar 7.11.



Gambar 8.87. Tambahan komponen PresensiFrm

3. Gunakan Tabel 7.8 untuk mengatur properti dari komponen CheckBox, ComboBox dan Edit.

**Tabel 8-30. Properti komponen tambahan di PresensiFrm**

**Komponen : CheckBox**

Properti	Isi
Name	SaringTabelCheck
Caption	Saring berdasarkan bulan

**Komponen : ComboBox**

Properti	Isi
Name	BulanCmb
Items	Januari Pebruari Maret April Mei Juni Juli Agustus September Oktober Nopember Desember
ItemIndex	0
Style	csDropDownList

**Komponen : Edit**

Properti	Isi
Name	ETahun
Text	kosong

4. Tambahkan deklarasi metoda BuatFilterBulan pada bagian public dari PresensiFrm seperti Listing 7.

**Listing 8-32. Deklarasi metoda BuatFilterBulan**

```
type
  TPresensiFrm = class(TForm)
  :
  :
  private
    { Private declarations }
  public
    { Public declarations }
    function BuatFilterBulan(bulan,tahun:integer):string;
  end;
```

5. Tulis implementasi dari metoda `BuatFilterBulan` di bawah bagian implementasi.

**Listing 8-33. Implementasi metoda `BuatFilterBulan`**

```
function TPresensiFrm.BuatFilterBulan(bulan,tahun:integer):string;
var
  FilterStr,TglAwal,TglAkhir:String;
  JumlahHariBulan:integer;
  TahunIni,BulanIni,HariIni:word;
begin
  FilterStr:='';
  if SaringTabelCheck.Checked then
  begin
    //buat tanggal awal bulan tahun
    TglAwal:=Format('%d/%d/%d',[1,Bulan,Tahun]);

    //ambil jumlah hari pada bulan tahun
    JumlahHariBulan:=DaysInMonth(StrToDateTime(TglAwal));

    //buat tanggal akhir dari bulan tahun
    TglAkhir:=Format('%d/%d/%d',[JumlahHariBulan,Bulan,Tahun]);

    //gunakan TglAwal dan TglAkhir untuk memfilter tabel PresensiTbl
    FilterStr:=Format('(Tgl >= %s) and (Tgl <= %s)',
      [QuotedStr(TglAwal), QuotedStr(TglAkhir)]);
  end;
  //Kembalikan hasil filter;
  Result:=FilterStr;
end;
```

6. Buat event handler `OnClick` dari `SaringTabelCheck` seperti pada Listing 7-7. Event handler ini kita gunakan untuk membuat saringan yang sesuai apabila pemakai mengubah status properti `Check` dari `SaringTableCheck` menjadi `True` dan melepaskan saringan apabila properti `Check` berisi `False`.

**Listing 8-34. Event handler `OnClick` dari `SaringTabelCheck`**

```
procedure TPresensiFrm.SaringTabelCheckClick(Sender: TObject);
begin
  with DataPersonalia do
  begin
    PresensiTbl.Filtered:=SaringTabelCheck.Checked;
    if (SaringTabelCheck.Checked) and (ETahun.Text <> '') then
      PresensiTbl.Filter:=BuatFilterBulan(BulanCmb.ItemIndex+1,
        StrToInt(ETahun.Text));
    PresensiTbl.Refresh;
  end; //with DataPersonalia do
end;
```

7. Buat event handler `OnChange` dari `BulanCmb` seperti Listing 7-8. Event handler ini kita gunakan untuk menangani situasi dimana pemakai mengubah bulan saringan sehingga kita harus menyesuaikan pernyataan-kondisional.

**Listing 8-35. Event handler OnChange dari BulanCmb**

```
procedure TPresensiFrm.BulanCmbChange(Sender: TObject);  
begin  
    SaringTabelCheckClick(Self);  
end;
```

8. Kompilasi program dan jalankan, pilih menu presensi dan cobalah melakukan saringan menggunakan bulan.

## **BAB 8 CALCULATED FIELD dan LOOKUP FIELD**

### **APA YANG AKAN KITA PELAJARI ?**

- Apa itu Lookup Field?
- Apa itu Calculated Field ?
- Membuat event handler untuk field bertipe calculated

**WAKTU LATIHAN : 3 JAM**

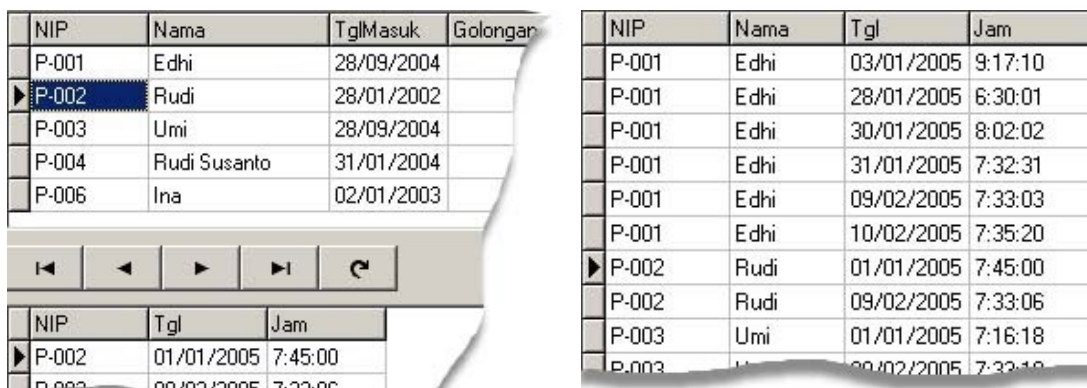
## Calculated Field

*Calculated Field* merupakan field yang tidak disimpan secara permanen di tabel. *Calculated Field* digunakan untuk menampilkan hasil perhitungan terhadap satu record tertentu tetapi hasil perhitungan tersebut tidak disimpan di tabel. Isi dari *Calculated Field* ditentukan saat dataset menerima event *OnCalculatedField*.

## Lookup Field

*Lookup Field* juga merupakan field yang diambil dari tabel lain tetapi seolah-olah merupakan bagian dari tabel yang mempunyai lookup field tersebut. Dengan menggunakan *Lookup field* kita dapat menggabungkan isi dari beberapa tabel menjadi satu tampilan data control.

Sebagai ilustrasi perhatikan Gambar 8.1.a dan Gambar 8.1.a Pada Gambar 8.1.a, untuk memperlihatkan NIP dan NAMA kita mengambil dari tabel Pegawai.db sedangkan data presensi diambil dari tabel Presensi.db. Kedua tabel kemudian direlasikan menggunakan relasi Master-Detail seperti yang kita pelajari pada Bab 7. Masing-masing field ditunjukkan isinya dengan menggunakan data control yang berbeda, tetapi pada Gambar 8.1.a kita memasukan field Nama yang diambil dari Pegawai.db seolah-olah sebagai bagian dari tabel Presensi.db.



NIP	Nama	TglMasuk	Golongan
P-001	Edhi	28/09/2004	
P-002	Rudi	28/01/2002	
P-003	Umi	28/09/2004	
P-004	Rudi Susanto	31/01/2004	
P-006	Ina	02/01/2003	

NIP	Tgl	Jam
P-002	01/01/2005	7:45:00
P-002	09/02/2005	7:33:06

NIP	Nama	Tgl	Jam
P-001	Edhi	03/01/2005	9:17:10
P-001	Edhi	28/01/2005	6:30:01
P-001	Edhi	30/01/2005	8:02:02
P-001	Edhi	31/01/2005	7:32:31
P-001	Edhi	09/02/2005	7:33:03
P-001	Edhi	10/02/2005	7:35:20
P-002	Rudi	01/01/2005	7:45:00
P-002	Rudi	09/02/2005	7:33:06
P-003	Umi	01/01/2005	7:16:18
P-003	Umi	09/02/2005	7:33:18

(a) (b)

**Gambar 9.88. (a) Relasi Master-Detail ; (b) LookupField**

Dalam latihan berikut ini kita akan menggunakan calculated field dan lookup field untuk mengolah gaji pegawai. Kita akan menggunakan calculated field untuk menghitung besar gaji kotor, gaji bersih dan pajak dari pegawai tersebut dan memanfaatkan lookup field untuk menampilkan nama pegawai seolah-olah sebagai field dari tabel Gaji.db

1. Tambahkan satu dataset dan datasource ke dalam DataPersonalia. Kita akan menggunakan dataset dan datasoruce tersebut sebagai representasi dari tabel Gaji.db.
2. Atur agar properti dari dataset dan datasource tersebut seperti pada Tabel 8.1

**Tabel 9-31. Properti dataset dan datasource representasi Gaji.db**

**Komponen : DataSet1**

Properti	Isi
Name	GajiTbl
DatabaseName	DBPersonalia
TableName	Gaji.db

**Komponen : Datasource1**

Properti	Isi
Name	DSGaji
Dataset	GajiTbl

3. Sekarang, kita akan secara eksplisit menyatakan field-field mana dari tabel Gaji.db yang akan digunakan. Anda dapat menggunakan teknik ini untuk menentukan apakah sebuah field akan ditampilkan di data control atau tidak.
4. Pilih dataset GajiTbl dan kemudian double-click GajiTbl, anda akan memperoleh jendela Field Editor seperti Gambar 8.2.



**Gambar 9.89. Field Editor dari GajiTbl**

5. Letakkan kursor di Field Editor dan kemudian klik kanan mouse sehingga anda memperoleh menu popup seperti Gambar 8.3.



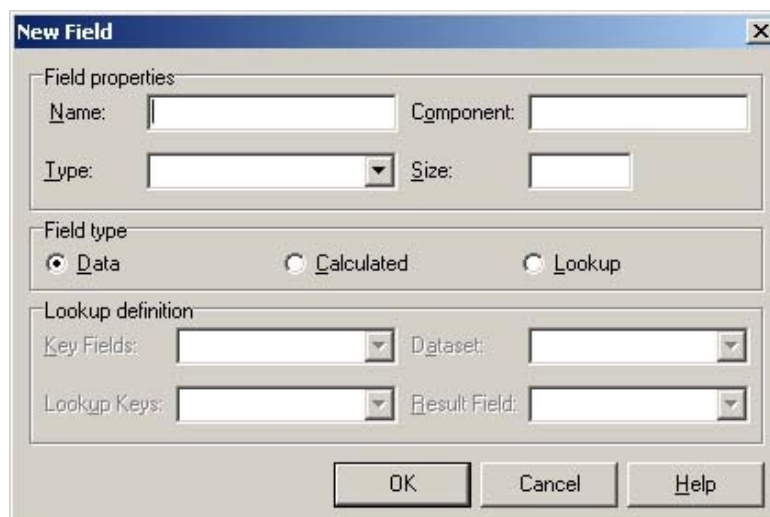
**Gambar 9.90. Menu popup Field Editor.**

6. Pilih menu **Add all fields**. Menu **Add all fields** akan menambahkan semua field dari GajiTbl ke dalam Field Editor, untuk memilih field tertentu anda dapat menggunakan menu **Add Fields**. Gambar 8.4. memperlihatkan Field Editor sudah berisi semua field dari GajiTbl.



**Gambar 9.91. Isi Field Editor dari GajiTbl**

7. Show Time!!! Kita akan menambahkan sebuah field bertipe Lookup Field untuk menampilkan nama pegawai. Nama pegawai diambil dari tabel Pegawai.db yang diwakili oleh dataset PegawaiTbl.
8. Klik kanan di Field Editor dari GajiTbl dan kemudian pilih menu New field sehingga anda memperoleh tampilan seperti Gambar 8.5.



**Gambar 9.92. Kotak dialog New Field**

9. Fungsi dari masing-masing input pada kotak dialog New Field diperlihatkan pada Tabel 8.2

**Tabel 9-32. Fungsi input pada kotak dialog New Field**

Groupbox : Field Properties	
Input	Keterangan
Name	Nama field, apabila diisi dengan nama field yang sudah ada di tabel maka anda dapat mengubah tipe data dari field tersebut.
Component	Nama komponen. Delphi secara otomatis membuat nama komponen sesuai dengan isi dari Name. Sebaiknya anda tidak mengubah isi input ini.

Type	Tipe data.
Size	Panjang field, hanya dapat diisi untuk tipe data tertentu, seperti string, varbytes dan sebagainya.
<b>Groupbox : Field Type</b>	
Data	Pilih tipe ini apabila anda ingin mengubah tipe data dari field di table
Calculated	Pilih tipe ini apabila anda ingin membuat calculated field
Lookup	Pilih tipe ini apabila anda ingin membuat field dari tabel lain menjadi bagian dari tabel ini. Pilihan ini akan mengaktifkan Groupbox Lookup Definition
<b>Groupbox : Lookup Definition</b>	
Key Fields	Menentukan field yang akan digunakan sebagai penghubung.
Dataset	Menentukan dataset yang mewakili tabel yang akan digabungkan.
Lookup Keys	Menentukan field di tabel kedua yang akan dihubungkan dengan field KeyFields. Pencarian didasarkan pada kesamaan isi antara Key Fields dan Lookup Keys. Nama field dari Key Fields dan Lookup Keys tidak perlu sama.
Result Field	Field yang akan digunakan sebagai hasil pencarian.

10. Isi kotak dialog New Field seperti Gambar 8.6.

**Gambar 9.93. Membuat lookup field Nama**

11. Klik OK dan field editor akan mempunyai satu field baru. Klik field Nama dan kemudian geser field Nama tersebut sehingga berada di bawah field NIP (Gambar 8.7).



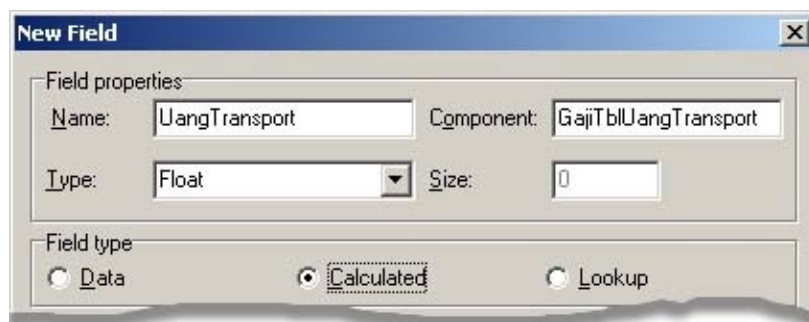
**Gambar 9.94. Tambahan field Nama**

12. Mumpung kita ada di Field Editor, saya akan menunjukkan kepada anda beberapa hal yang dapat anda lakukan dengan Field Editor. Kita dapat mengubah nama tampilan field (Display Label) melalui properti DisplayLabel. DisplayLabel merupakan properti yang digunakan oleh DBGrid untuk mengisi judul kolom, apabila anda tidak mengubah isi properti DisplayLabel maka Delphi akan menggunakan nama field sebagai DisplayLabel, tentunya tidak elok apabila kita menggunakan judul kolom seperti "NAMA\_IKAN\_TAWAR", akan lebih bagus apabila judul kolom tersebut menjadi "NAMA IKAN TAWAR".
13. Klik field GajiPokok dan kemudian aktifkan Object Inspector (F11), pilih properti DisplayLabel dan kemudian ubah isinya menjadi *Gaji Pokok* (Gambar 8.8)



**Gambar 9.95. Mengubah isi properti DisplayLabel**

14. Lakukan hal yang sama untuk mengubah DisplayLabel dari field HariMasuk menjadi "Jumlah Kehadiran".
15. Setelah menambah satu field dengan tipe Lookup Field, tahap berikutnya kita akan menambahkan field dengan tipe Calculated Field untuk menampilkan hasil perhitungan Uang Transport (UT), Gaji Kotor (GK), Pajak (PJK), dan Gaji Bersih (GB) dari masing-masing pegawai.
16. Klik kanan Field Editor dan kemudian pilih menu New Field.
17. Gunakan Gambar 8.9 sebagai panduan untuk membuat Calculated Field dengan nama UangTransport.
18. Lakukan hal yang sama dengan langkah 16 dan 17 untuk membuat GajiKotor, Pajak dan GajiBersih.



**Gambar 9.96. Calculated Field UangTransport**

19. Setelah kita membuat field-field bertipe Calculated maka tahap berikutnya adalah menentukan isi dari Calculated Field. Seperti telah saya jelaskan di awal bab ini, field bertipe Calculated Field diisi melalui event OnCalculatedField.
20. Untuk itu pilih GajiTbl dan kemudian aktifkan Object Inspector (F11), buat event handler OnCalculatedField seperti Listing 8-1.

**Listing 9-36. Event handler OnCalculatedField**

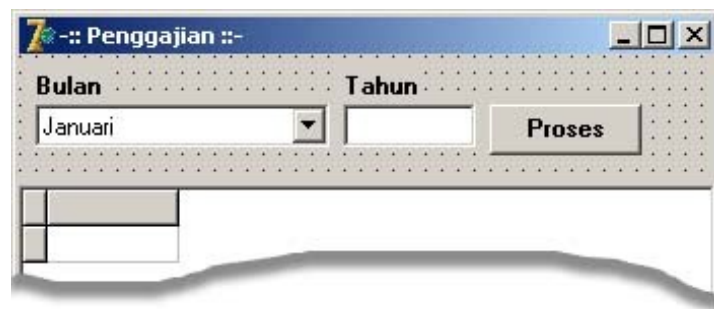
```
procedure TDataPersonalia.GajiTblCalcFields(DataSet: TDataSet);
var
  UT, GK, PJK, GB: Real;
begin
  with GajiTbl do
  begin
    //hitung uang transport berdasarkan jumlah kehadiran
    UT:=FieldByName('Transport').AsInteger *
        FieldByName('HariMasuk').AsInteger;

    //hitung gaji kotor = GajiPokok + Uang Transport
    GK:=FieldByName('GajiPokok').AsFloat + UT;

    //hitung pajak = Gaji Kotor * 0.10
    PJK:= GK * 0.10;
    //hitung gaji bersih = Gaji Kotor - Pajak
    GB := GK - PJK;

    //isikan hasil perhitungan ke masing-masing field
    FieldByName('UangTransport').AsFloat:= UT;
    FieldByName('GajiKotor').AsFloat:= GK;
    FieldByName('Pajak').AsFloat:= PJK;
    FieldByName('GajiBersih').AsFloat:=GB;
  end;
end;
```

21. Buat sebuah form baru dan ubah properti Name dari form tersebut menjadi GajiFrm, simpan form dengan nama GajiForm.pas.
22. Hubungkan DataModulePersonalia ke GajiFrm melalui menu File | Use Unit | DataModulePersonalia.
23. Tambahkan komponen ComboBox, Edit, DBGrid dan Button seperti Gambr 8.10. Atur properti dari masing-masing komponen seperti Tabel 8.3.



**Gambar 9.97. Tata letak GajiFrm**

**Komponen : ComboBox**

Properti	Isi
Name	BulanCmb
Items	Januari Pebruari ... Desember
Style	csDropDownList

**Komponen : Edit**

Properti	Isi
Name	ETahun
Text	kosongkan

**Komponen : DBGrid**

Properti	Isi
Name	DBGridGaji
Datasource	DataPersonalia.DSGaji

**Komponen : Button**

Properti	Isi
Name	ProsesBtn
Caption	Proses

24. Buat event handler OnClick dari ProsesBtn untuk menghitung gaji pegawai pada bulan tertentu. Kita akan menggunakan relasi Master-Detail antara PegawaiTbl dengan GolonganTbl dan PresensiTbl secara programming untuk memperoleh informasi mengenai UangTransport per hari berdasarkan golongan pegawai serta jumlah hari masuk dari pegawai tersebut.

**Listing 9-37. Event handler OnClick dari ProsesBtn**

```

procedure TGajiFrm.ProsesBtnClick(Sender: TObject);
var
  Transport, JumlahHariMasuk: integer;
  GaPok: Real;
  NIP: string;
begin
  with DataPersonalia do
  begin
    //kosongkan filter dari gaji
    GajiTbl.Filter:='';
    GajiTbl.Filtered:=False;

    //cek apakah tabel-tabel sudah dibuka
    if not PegawaiTbl.Active then
      PegawaiTbl.Open;

    if not PresensiTbl.Active then
      PresensiTbl.Open;
  end;
end;

```

```

if not GolonganTbl.Active then
    GolonganTbl.Open;

    //buat relasi MasterDetail antara Golongan dengan Pegawai
    //relasi ini akan kita gunakan untuk mengambil uang transport per
    //hari berdasarkan golongan pegawai
    GolonganTbl.MasterSource:=DSPegawai;
    GolonganTbl.MasterFields:='Golongan';

    //buat relasi MasterDetail antara Presensi dengan Pegawai
    //relasi ini akan kita gunakan untuk mengambil jumlah hari masuk
    PresensiTbl.MasterSource:=DSPegawai;
    PresensiTbl.MasterFields:='NIP';

    //buat filter untuk memperoleh kehadiran pegawai berdasarkan
    //bulan tertentu
    PresensiTbl.Filter:=PresensiFrm.BuatFilterBulan(
        BulanCmb.ItemIndex+1, StrToInt(ETahun.Text));
    //ambil data pegawai dari tabel pegawai
    while not PegawaiTbl.Eof do
        begin
            //ambil jumlah hari masuk dari pegawai ini
            JumlahHariMasuk:=PresensiTbl.RecordCount;

            //ambil gajipokok dan transport per hari dari pegawai ini
            GaPok:=GolonganTbl.FieldName('GAJIPOKOK').AsFloat;
            Transport:=GolonganTbl.FieldName('TRANSPORT').AsInteger;

            //ambil NIP pegawai
            NIP:=PegawaiTbl.FieldName('NIP').AsString;

            //cek apakah record ini sudah disimpan di tabel Gaji.db,
            //jika ya maka gunakan Edit
            // sedangkan jika belum pernah disimpan gunakan AppendRecord
            if not GajiTbl.Locate('NIP;Bulan',
                VarArrayOf([NIP,BulanCmb.ItemIndex+1]), [loCaseInsensitive])
                then
                    GajiTbl.AppendRecord([NIP,BulanCmb.ItemIndex+1,
                        GAPOK,JumlahHariMasuk,Transport])
                else
                    begin
                        With GajiTbl do
                            begin
                                Edit;
                                FieldByName('GAJIPOKOK').AsFloat:=GaPok;
                                FieldByName('HARIMASUK').AsInteger:=JumlahHariMasuk;
                                FieldByName('TRANSPORT').AsInteger:=Transport;
                                Post;
                            end;
                        end;
                        DataPersonalia.PegawaiTbl.Next;
                    end;

            //filter tampilan gaji pegawai berdasarkan bulan tertentu
            GajiTbl.Filter:=Format('Bulan = %d', [BulanCmb.ItemIndex+1]);
            GajiTbl.Filtered:=True;
        end; //with DataPersonalia
    end;

```

25. Pindah ke MainForm, hubungkan GajiFrm ke MainForm melalui menu File | Use Unit | GajiForm.
26. Buat event handler OnClick dari sub menu Gaji seperti pada Listing 8-2.

**Listing 9-38. Event handler OnClick dari sub menu Gaji**

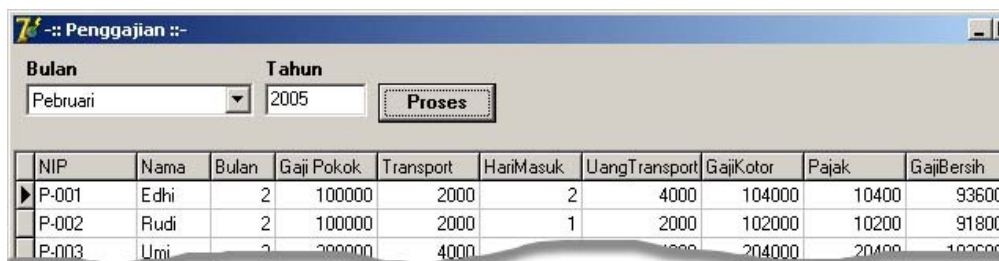
```
procedure TMainFrm.GajiClick(Sender: TObject);
var
  Year, Month, Day: Word;
begin
  if not DataPersonalia.GajiTbl.Active then
    DataPersonalia.GajiTbl.Open;

  //ambil tahun, bulan dan hari ini
  DecodeDate(Now(), Year, Month, Day);

  //isi komponen ETahun dengan tahun sekarang
  GajiFrm.ETahun.Text := IntToStr(Year);

  //tampilkan form GajiFrm
  GajiFrm.ShowModal;
end;
```

27. Kompilasi program dan jalankan. Pilih sub menu Gaji dan kemudian pilih bulan pada BulanCmb, setelah itu klik tombol Proses untuk memproses gaji pada bulan tersebut. Gambar 8.11 memberi contoh hasil proses penggajian untuk bulan Pebruari.



The screenshot shows a window titled "7 Pengajian :-". It contains a form with a "Bulan" dropdown menu set to "Pebruari" and a "Tahun" text box set to "2005". A "Proses" button is next to them. Below the form is a table with 10 columns: NIP, Nama, Bulan, Gaji Pokok, Transport, HariMasuk, UangTransport, GajiKotor, Pajak, and GajiBersih. The table contains three rows of data.


NIP	Nama	Bulan	Gaji Pokok	Transport	HariMasuk	UangTransport	GajiKotor	Pajak	GajiBersih
P-001	Edhi	2	100000	2000	2	4000	104000	10400	93600
P-002	Rudi	2	100000	2000	1	2000	102000	10200	91800
P-003	Umi	2	200000	4000			204000	20400	183600

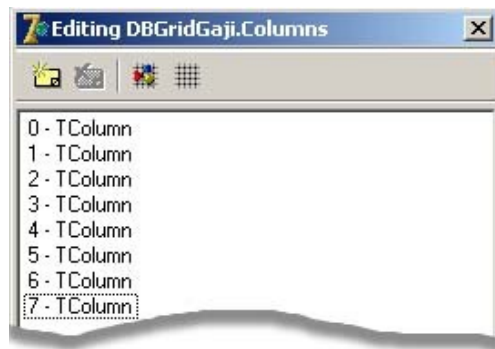
**Gambar 9.98. Hasil proses penggajian bulan Pebruari 2005**

28. Meskipun fungsi menghitung gaji sudah dapat berfungsi dengan baik, tetapi tampilan hasil perhitungan tidak terlalu bagus. Perhatikan DBGridGaji tetap menampilkan field Bulan, meskipun pada BulanCmb sudah tercantum bulan yang bersangkutan. Kita akan membuang kolom Bulan dari mengubah tampilan DBGridGaji.
29. Tutup program dan kembali ke Delphi.
30. Pindah ke GajiFrm dan pilih DBGridGaji. Double-click DBGridGaji untuk menampilkan Columns Editor. (Gambar 8.12)



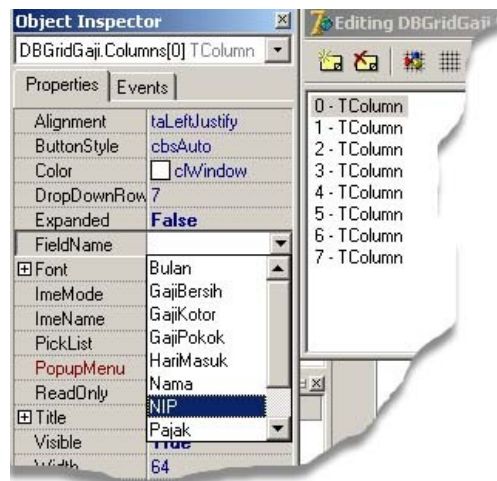
**Gambar 9.99. Columns Editor**

31. Klik tombol  untuk menambah kolom. Klik beberapa kali sehingga anda memperoleh 7 buah TColumns (Gambar 8.13).



**Gambar 9.100. Menambah kolom di DBGrid**

32. Kita akan menghubungkan setiap kolom dengan field pada GajiTbl. Klik 0 – TColumn dan kemudian aktifkan Object Inspector (F11). Ubah properti FieldName dengan NIP (Gambar 8.14).



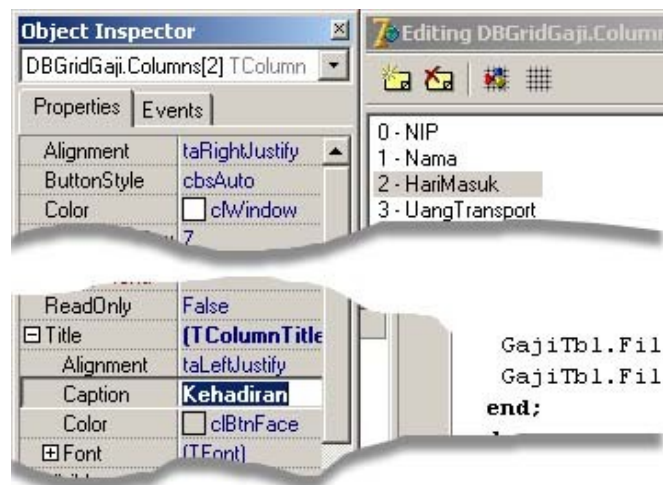
**Gambar 9.101. Menghubungkan 0-TColumn dengan field NIP**

33. Lakukan hal yang sama untuk menghubungkan kolom yang lain ke field yang sesuai. Gunakan Tabel 8.3 sebagai panduan.

**Tabel 9-33. Menghubungkan kolom DBGridGaji dengan Field GajiTbl**

Kolom	Field
1-TColumn	Nama
2-TColumn	HariMasuk
3-TColumn	UangTransport
4-TColumn	GajiPokok
5-TColumn	GajiKotor
6-TColumn	Pajak

34. Perhatikan perubahan pada DBGridGaji. Judul kolom masih menggunakan nama field, anda dapat mengubah judul kolom melalui properti Title dari masing-masing kolom.
35. Masih menggunakan Column Editor dari DBGridGaji, apabila Column Editor anda tidak tampak anda dapat memunculkan kembali dengan meng-double-click DBGridGaji. Pilih salah satu kolom, dalam contoh ini saya memilih kolom HariMasuk.
36. Aktifkan Object Inspector dan kemudian klik tombol + di samping Title. Ubah isi properti Caption menjadi Kehadiran. Anda juga dapat menentukan perataan judul kolom ditampilkan melalui properti Alignment (Gambar 8.15).



**Gambar 9.102. Mengubah judul kolom HariMasuk**

37. Perhatikan perubahan dari DBGridGaji. Lakukan hal yang sama untuk kolom-kolom lain.

## **BAB 9 POSTGRESQL**

### **APA YANG AKAN KITA PELAJARI ?**

- Sejarah PostgreSQL?
- Dimana memperoleh PostgreSQL ?
- Memasang PostgreSQL di Windows ?

### **WAKTU LATIHAN : 1 JAM**

## **Sejarah PostgreSQL**

PostgreSQL merupakan salah satu Object-Relational Database Management System (ORDBMS) yang didasarkan pada POSTGRES versi 4.2. POSTGRES versi 4.2 dikembangkan di Departemen Berkeley Computer Science pada Universitas California. POSTGRES pertama kali dikembangkan oleh Profesor Michael Stonebraker dan didanai oleh Defense Advanced Research Projects Agency (DARPA), The Army Research Office (ARO), The National Science Foundation (NSF) serta ESL, Inc.

Konsep-konsep yang digunakan di dalam Postgres dipresentasikan pertama kali melalui makalah *The Design of POSTGRES* sedangkan definisi mengenai data dijelaskan dalam makalah *The POSTGRES data model*.

Dalam perkembangannya POSTGRES telah melalui berbagai perubahan sejak pertama kali diperkenalkan. Tahun 1987 merupakan tahun pertamakali POSTGRES dioperasikan dan pada tahun 1988 didemonstrasikan pada konferensi ACM-SIGMOD. Pada tahun 1989, Versi 1, seperti dijelaskan dalam makalah *The Implementation of POSTGRES*, mulai digunakan oleh beberapa pemakai di luar Universitas California. Menanggapi beberapa kritik mengenai sistem aturan (*rule system*) maka sistem aturan POSTGRES diperbaiki dan pada tahun 1990 keluar versi 2 dari POSTGRES. Versi 3 diluncurkan sekitar tahun 1991 dan mempunyai tambahan kemampuan untuk mengelola sistem penyimpanan lebih dari satu (*multiple storage managers*). Sampai versi Postgres95, pengembangan Postgres masih ditujukan untuk menangani masalah *portability* dan *reliability*.

POSTGRES telah digunakan dalam berbagai macam riset dan aplikasi, mulai dari sistem analisa keuangan, aplikasi pemantauan kemampuan mesin jet, database untuk pergerakan asteroid, database informasi medis dan berbagai sistem GIS. Dalam perkembangannya, Ilustra Information Technology, belakangan digabungkan ke Informix yang dimiliki oleh IBM, mengambil kode-kode POSTGRES dan mengembangkannya untuk keperluan komersial.

POSTGRES sejak awal dirancang sebagai sistem terbuka (open-source) dan boleh digunakan, diubah, didistribusikan oleh siapa saja. Pada tahun 1994, Andrew Yu dan Jolly Chen menambahkan interpreter SQL ke POSTGRES dan sejak itu POSTGRES berganti nama menjadi Postgres95.

Pada tahun 1996, nama Postgres95 ternyata tidak dapat bertahan, sehingga kemudian diubah menjadi PostgreSQL. Perubahan nama tersebut diharapkan memberikan kaitan yang kuat antara versi asli POSTGRES dan penambahan kemampuan bahasa SQL. Saat itu, versi PostgreSQL ditetapkan mulai versi 6.0. PostgreSQL mendukung secara luas versi baku SQL:2003 dan memberikan berbagai kemampuan seperti :

- complex queries
- foreign keys
- triggers

- views
- transactional integrity
- multiversion concurrency control

Selain itu, PostgreSQL juga dapat diperluas oleh pemakai dengan menambahkan berbagai kemampuan seperti :

- Menentukan tipe data sesuai keperluan pemakai
- Membuat fungsi-fungsi khusus
- Membuat operator
- Membuat fungsi aggregate
- Membuat metoda index
- Membuat *procedural languages (P/L)*

### **Memperoleh PostgreSQL**

PostgreSQL hanya dapat dijalankan di sistem operasi Unix, Linux, Windows 2000 / 2003 / XP dan tidak dapat dijalankan di Windows95 / Windows ME serta Windows98. Saat ini PostgreSQL telah mencapai versi 8.0.1.

PostgreSQL dapat diunduh secara bebas di situs resmi PostgreSQL, yaitu : <http://www.postgresql.org>. Selain mendownload PostgreSQL, anda juga dapat mendownload berbagai aplikasi yang berhubungan dengan PostgreSQL, seperti :

- pgAdmin : Aplikasi untuk mengelola database PostgreSQL
- phpAdmin : Aplikasi berbasis web dengan kemampuan sama seperti pgAdmin
- psqLODBC : Driver untuk menghubungkan aplikasi anda dengan PostgreSQL melalui ODBC.

Untuk keperluan latihan ini, anda harus mengunduh (*download*) distribusi PostgreSQL untuk windows (versi terakhir postgresql-8.0.1.zip).

### **Instalasi PostgreSQL**

Agar anda dapat merasakan situasi yang lebih realistis dalam mengembangkan aplikasi yang menggunakan remote database maka saya menyarankan agar anda menyediakan minimal dua buah komputer, satu komputer difungsikan sebagai database server dan komputer lain difungsikan sebagai tempat mengembangkan aplikasi. Meskipun demikian, anda juga dapat menggunakan satu buah komputer yang difungsikan sebagai server PostgreSQL sekaligus juga klien dari PostgreSQL.

## Memasang PostgreSQL

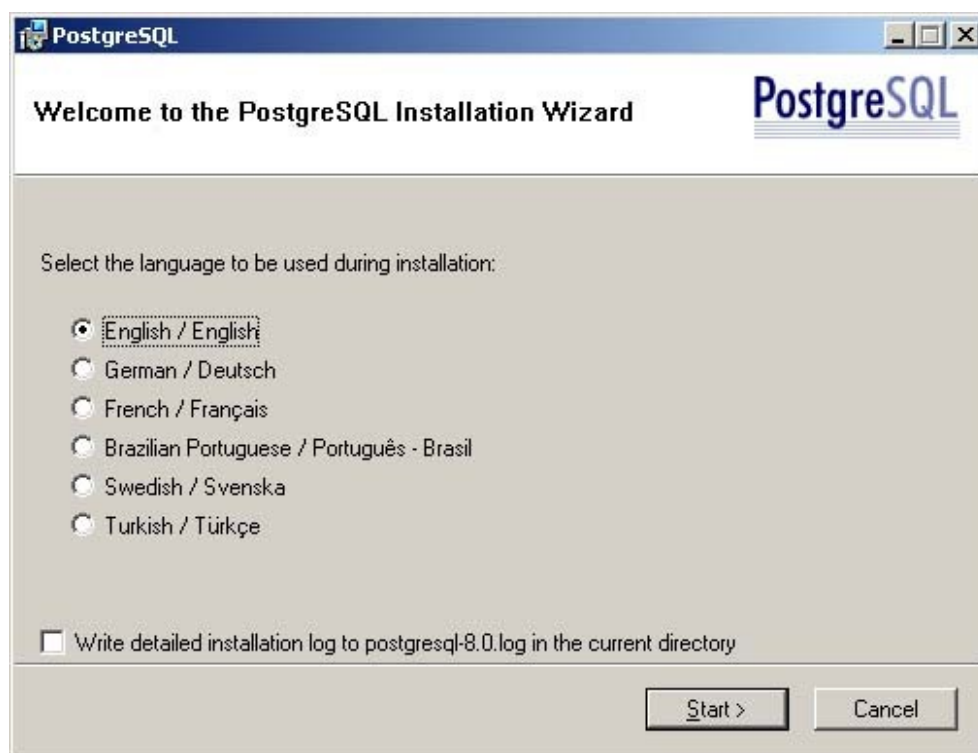
Setelah anda mengunduh postgresql-8.0.1.zip, ekstrak file tersebut menggunakan aplikasi seperti WinZip atau 7-Zip di direktori sementara, pada contoh ini saya menggunakan D:\TEMP\PostgreSQL. Anda akan memperoleh file-file seperti Gambar 9.1.



**Gambar 10.103. File instalasi PostgreSQL**

Setelah anda memperoleh file-file instalasi maka ikuti langkah-langkah berikut ini untuk memasang PostgreSQL di komputer anda.

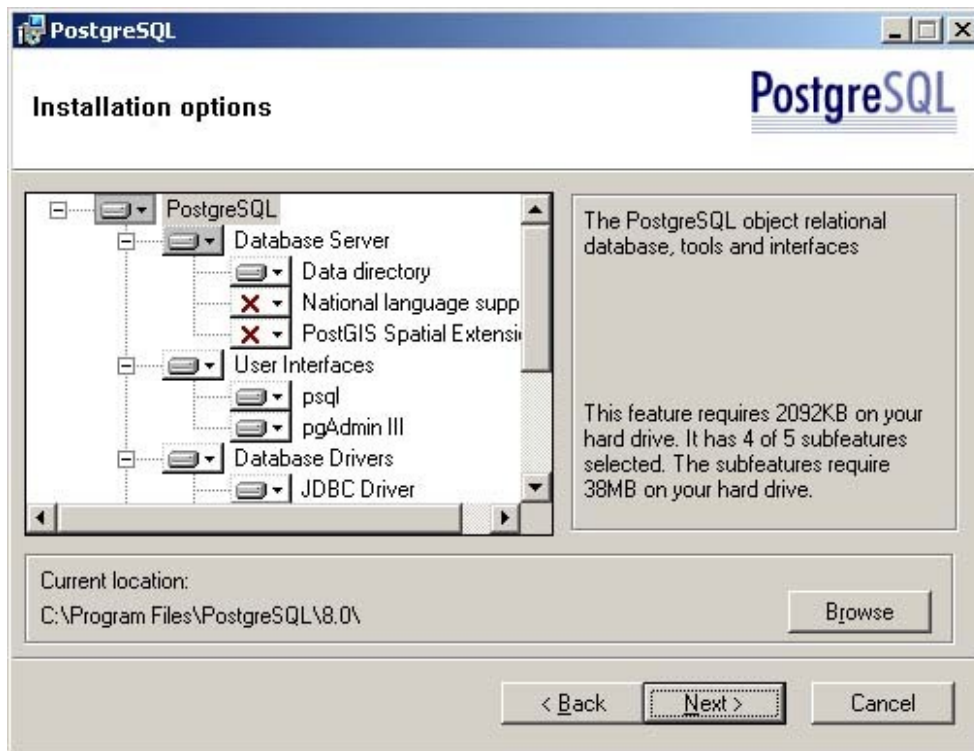
1. Double-click file postgresql-8.0.msi.
2. Pilih English sebagai bahasa Instalasi dan kemudian klik tombol Start (Gambar 9.2)




**Gambar 10.104. Kotak dialog Welcome**

3. Tunggu beberapa saat, klik Next saat muncul peringatan "It is strongly recommended...".
4. Klik Next pada kotak dialog Installation Notes. Anda sebaiknya membaca terlebih dahulu beberapa catatan penting yang ditampilkan.

- Setelah kotak dialog Installation Notes, program instalasi akan menampilkan kotak dialog Installation Options.



**Gambar 10.105. Pilihan instalasi**

- Kita tidak akan mengubah pilihan instalasi. Pilihan yang diberi tanda  berarti pilihan tersebut tidak dipasang. Untuk mengubah pilihan dipasang atau tidak, klik tombol disamping pilihan tersebut dan kemudian pilih sub menu *Entire feature will be unavailable* apabila anda tidak ingin memasang pilihan tersebut dan pilih sub menu *Will be installed on local hard drive* untuk memasang pilihan tersebut. Apabila anda ingin mengubah lokasi pemasangan, klik Browse dan kemudian pilih direktori dimana PostgreSQL akan dipasang.
- Setelah semua pilihan pemasangan ditentukan, klik tombol Next. Program Instalasi akan menampilkan kotak dialog *Service Configuration* (Gambar 9.4). Arti dari masing-masing pilihan pada kotak dialog *Service Configuration* ditunjukkan pada Tabel 9.1.

**Tabel 10-34. Arti pilihan di Service Configuration**

Pilihan	Keterangan
Install Service	Apabila pilihan ini dipilih maka PostgreSQL akan dijalankan sebagai service. Service merupakan program yang otomatis dijalankan oleh Windows saat Windows dijalankan pertama kali. Sebaiknya anda memilih pilihan Install Service ini untuk memudahkan administrasi PostgreSQL.
Service Name	Nama Service. Nama ini akan ditampilkan di Service Manager (Start   Administrative Tools   Services). Sebaiknya anda tidak mengubah nama service ini.
Account Name	Nama user untuk menjalankan service. Untuk alasan keamanan

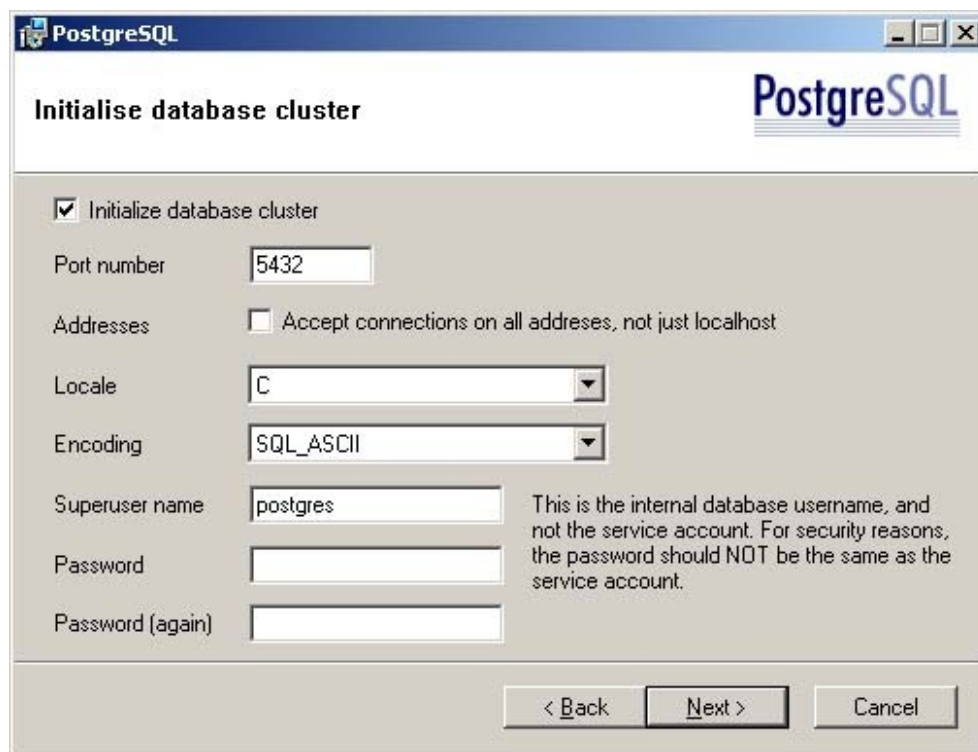
	sebaiknya anda tidak menggunakan account yang menjadi anggota grup administrator. Apabila account yang anda isi belum dibuat maka program instalasi akan secara otomatis membuat account tersebut.
Account Domain	Nama domain dimana account tercatat. Gunakan nama komputer anda sebagai nama domain apabila anda ingin agar account hanya tercatat di komputer dimana postgresQL dipasang dan tidak tercatat di PDC (Primary Domain Controller). Hubungi administrator jaringan apabila anda ingin menggunakan account di PDC.
Account Password	Kata sandi untuk account yang akan dibuat. Apabila anda mengosongkan pilihan ini maka program instalasi akan membuat password sendiri.
Verify Password	Isikan kata sandi yang sama dengan yang anda isi di pilihan <i>Account Password</i> .

**Gambar 10.106. Konfigurasi Service**

8. Program instalasi mungkin anda menanyakan beberapa pertanyaan yang terkait dengan pembuatan account. Misalnya, apabila kata sandi yang anda buat dinilai lemah dan mudah ditebak maka program instalasi akan menawarkan kata sandi yang lain.
9. Setelah pengaturan service selesai dilakukan, program instalasi akan menampilkan kotak dialog untuk mengatur inisialisasi (persiapan) database. Arti dari masing-masing pilihan diperlihatkan pada Tabel 9.2. Gambar 9.5 menunjukkan kotak dialog *Initialise database cluster*.

**Tabel 10-35. Arti pilihan di *Initialise database cluster***

Pilihan	Keterangan
Initialize database cluster	Klik pilihan ini apabila anda ingin menginisialisasi database yang digunakan oleh PostgreSQL. Apabila anda tidak memilih pilihan ini maka anda harus melakukan inisialisasi database secara manual melalui program initdb. Saya sarankan anda memilih pilihan ini.
Port number	Nomor port yang akan digunakan oleh aplikasi untuk berhubungan dengan PostgreSQL melalui jaringan TCP/IP.
Address	Klik pilihan ini apabila anda ingin PostgreSQL dapat dihubungi dari komputer lain.
Locale	Pilihan definisi lokal seperti koma, tanda uang (Rp, \$) dan sebagainya
Encoding	Pilih SQL_ASCII untuk menyatakan bagaimana sistem mengartikan karakter.
Superuser name	Nama user pemilik database. Nama user ini tidak sama dengan Account Name pemilik service.
Password	Kata sandi
Password (again)	Kata sandi (ulangi memasukkan kata sandi).



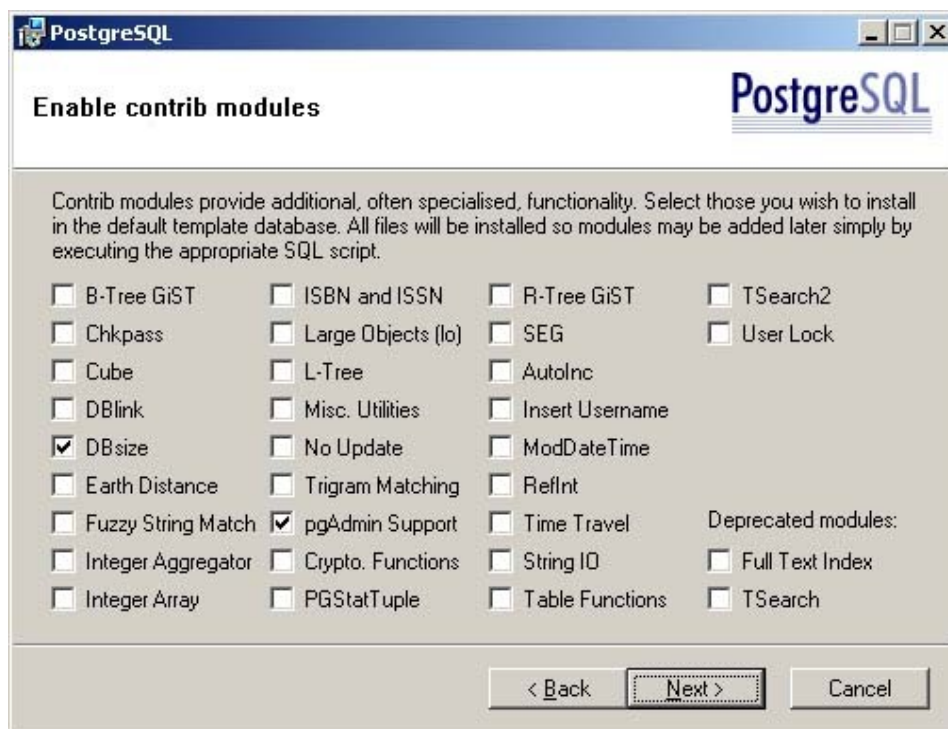
**Gambar 10.107. Kotak dialog *Initialise database cluster***

- Setelah pilihan service dan persiapan database disiapkan, program instalasi akan menawarkan pilihan bahasa yang digunakan sebagai procedural languages untuk membuat trigger. Pilihan baku adalah PL/pgsql. Klik tombol Next.



**Gambar 10.108. Pilihan *procedural languages***

11. Berikutnya, anda akan diberi pilihan modul-modul yang dipasang bersamaan PostgreSQL. Pilih pilihan baku seperti Gambar 9.7.



**Gambar 10.109. Pilihan modul-modul PostgreSQL**

12. Klik Next dan program instalasi akan mulai memasang dan mengatur konfigurasi PostgreSQL. Setelah pemasangan selesai anda akan memperoleh kotak dialog seperti Gambar 9.8.



**Gambar 10.110. Selesai memasang PostgreSQL**

## **BAB 10 Structured Query Language (SQL)**

### **APA YANG AKAN KITA PELAJARI ?**

- Structured Query Language
- Data Definition Language (DDL)
- Data Manipulation Language (DML)

**WAKTU LATIHAN : 2 JAM**

## SQL

SQL (Structured Query Language) merupakan bahasa yang digunakan untuk mengelola database. SQL tidak hanya digunakan di RDBMS (Remote Database Management System) tetapi juga dapat digunakan di Local Database System, tentu saja dengan batasan-batasan tertentu. Satu hal yang perlu ditekankan disini, **SQL bukan bahasa pemrograman! SQL digunakan untuk menunjukkan kepada sistem database apa yang kita inginkan dan bukan bagaimana melakukan sesuatu seperti keinginan kita.**

Kita memberitahu sistem database melalui pengiriman *query* (pertanyaan), sistem database akan memberi jawaban terhadap *query* tersebut dengan mengirimkan record-record yang memenuhi pertanyaan tersebut.

### Komponen SQL

Meskipun SQL bukan bahasa pemrograman, akan tetapi SQL menyediakan berbagai perintah yang sangat berguna dalam memelihara database relational. Perintah-perintah di dalam SQL dapat dikelompokkan ke dalam 3 kelompok, yaitu :

- Data Definition Language (DDL) : Bagian dari SQL yang digunakan untuk mendefinisikan database, mengubah struktur database dan membuang database.
- Data Manipulation Language (DML) : Bagian dari SQL yang digunakan untuk memanipulasi data di dalam database seperti : menyisipkan data, mengubah data, mengambil data atau menghapus data.
- Data Control Language (DCL) : Bagian dari SQL yang digunakan untuk meningkatkan keamanan data dan mencegah kerusakan data.

### Data Definition Language (DDL)

Seperti telah disebutkan di atas, DDL digunakan untuk mendefinisikan database. Perintah-perintah yang termasuk di dalam DDL antara lain :

- CREATE TABLE
- CREATE VIEW
- ALTER TABLE
- DROP TABLE
- DROP VIEW

### CREATE TABLE

Perintah *CREATE TABLE* digunakan untuk membuat tabel baru. Di MSSQL 2000 secara mendasar perintah ini mempunyai sintaks sebagai berikut :

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } ] TABLE table_name ({
column_name data_type [ DEFAULT default_expr ] [ column_constraint
[ ... | table_constraint | LIKE parent_table [ { INCLUDING | EXCLUDING
} DEFAULTS ] } [, ... ])[ INHERITS ( parent_table [, ... ] ) ]
```

```
[ WITH OIDS | WITHOUT OIDS ][ ON COMMIT { PRESERVE ROWS | DELETE ROWS
| DROP } ][ TABLESPACE tablespace ]
```

dengan `column_constraint` dapat berisi :

```
[ CONSTRAINT constraint_name ] { NOT NULL | NULL | UNIQUE [ USING
INDEX TABLESPACE tablespace ] |
PRIMARY KEY [ USING INDEX TABLESPACE tablespace ] |
CHECK (expression) | REFERENCES reftable [ ( refcolumn ) ]
[ MATCH FULL | MATCH PARTIAL | MATCH PLE ][ ON DELETE action ]
[ ON UPDATE action ] } [ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY
DEFERRED | INITIALLY IMMEDIATE ]
```

dan `table_constraint` sebagai :

```
[ CONSTRAINT constraint_name ] { UNIQUE ( column_name [, ... ] )
[USING INDEX TABLESPACE tablespace ] |
PRIMARY KEY ( column_name [, ... ] )
[ USING INDEX TABLESPACE tablespace ]
CHECK ( expression ) | FOREIGN KEY ( column_name [, ... ] )
REFERENCES reftable [ ( refcolumn [, .] ) ]
[ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ] [ ON DELETE action ]
[ ON UPDATE action ] }
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED |
INITIALLY IMMEDIATE ]
```

Anda tidak perlu bingung dengan banyaknya pilihan dalam memberikan perintah `CREATE TABLE`, umumnya kita hanya menggunakan sebagian kecil dari perintah-perintah tersebut. Sebagai contoh, perintah berikut ini :

```
CREATE TABLE Pegawai (NIP char (10) NOT NULL PRIMARY KEY, Nama varchar
(20) NOT NULL, Golongan int)

CREATE TABLE GaPok (Golongan smallint NOT NULL PRIMARY KEY, GajiPokok
float)
```

akan membuat tabel `Pegawai` dan tabel `GaPok`. Tabel `Pegawai` menggunakan field `NIP` sebagai `PRIMARY KEY` dan tabel `GaPok` menggunakan field `Golongan` sebagai `PRIMARY KEY`.

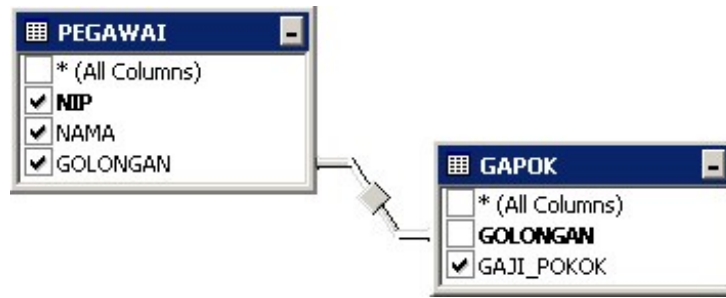
## CREATE VIEW

View merupakan struktur logis dari sebuah data, umumnya view merupakan gabungan dari beberapa tabel. Sintak perintah `Create View` sebagai berikut :

```
CREATE [ OR REPLACE ] VIEW name [ ( column_name [, ...] ) ] AS query
```

Di dalam perintah `CREATE VIEW` tersebut *query* menggunakan format seperti pada perintah `SELECT`.

Sebagai contoh, apabila kita ingin melihat informasi atau data mengenai pegawai lengkap dengan gaji pokok sesuai dengan golongannya maka informasi tersebut dapat diperoleh dari gabungan antara tabel `Pegawai` dan `GaPok` seperti pada Gambar 10.1.



**Gambar 11.111. Relasi Pegawai dan GaPok**

Misalkan informasi tersebut dihasilkan dari view bernama PEGAWAI\_VIEW, maka PEGAWAI\_VIEW dapat dibuat melalui perintah :

```
CREATE VIEW VIEW_PEGAWAI AS
SELECT GAPOK.GAJI_POKOK, PEGAWAI.NIP, PEGAWAI.NAMA, PEGAWAI.GOLONGAN
FROM GAPOK INNER JOIN PEGAWAI ON GAPOK.GOLONGAN = PEGAWAI.GOLONGAN
```

Perhatikan tabel Pegawai dihubungkan dengan tabel GaPok melalui perintah *INNER JOIN* menggunakan field GOLONGAN.

## ALTER TABLE

Perintah ALTER TABLE digunakan untuk mengubah konfigurasi tabel. Perintah ALTER TABLE menggunakan sintak sebagai berikut :

```
ALTER TABLE [ ONLY ] name [ * ] action [, ... ]
ALTER TABLE [ ONLY ] name [ * ] RENAME [ COLUMN ] column TO new_column
ALTER TABLE name RENAME TO new_name
```

dengan *action* salah satu dari :

```
ADD [ COLUMN ] column type [ column_constraint [ ... ] ]
DROP [ COLUMN ] column [ RESTRICT | CASCADE ]
ALTER [ COLUMN ] column TYPE type [ USING expression ]
ALTER [ COLUMN ] column SET DEFAULT expression
ALTER [ COLUMN ] column DROP DEFAULT
ALTER [ COLUMN ] column { SET | DROP } NOT NULL
ALTER [ COLUMN ] column SET STATISTICS integer
ALTER [ COLUMN ] column SET STORAGE { PLAIN | EXTERNAL | EXTENDED
| MAIN }
ADD table_constraint
DROP CONSTRAINT constraint_name [ RESTRICT | CASCADE ]
CLUSTER ON index_name
SET WITHOUT CLUSTER
SET WITHOUT OIDS
OWNER TO new_owner
SET TABLESPACE tablespace_name
```

Sebagai contoh, apabila kita ingin menambahkan field Alamat dan No Telepon di tabel Pegawai maka perintah yang diberikan adalah :

```
ALTER TABLE pegawai ADD COLUMN alamat varchar(50),
ADD COLUMN kota varchar(30), ADD COLUMN no_telp varchar(16)
```

## DROP TABLE dan DROP VIEW

Perintah DROP TABLE digunakan untuk membuang tabel dari database sedangkan perintah DROP VIEW digunakan untuk membuang view dari database. Anda harus berhati-hati ketika menggunakan perintah ini karena perintah ini sekali diberikan tidak dapat dibatalkan.

## Data Manipulation Language (DML)

Perintah-perintah yang berada di kelompok DML digunakan untuk melakukan manipulasi data di tabel. Beberapa perintah yang cukup penting untuk diketahui antara lain :

- INSERT
- SELECT
- UPDATE
- DELETE

### INSERT

Perintah INSERT digunakan untuk memasukkan data ke dalam sebuah tabel. Perintah ini mempunyai sintak sebagai berikut :

```
INSERT INTO table [ ( column [, ...] ) ] { DEFAULT VALUES | VALUES ( {  
expression | DEFAULT } [, ...] ) | query }
```

Sebagai contoh, perintah berikut ini :

```
INSERT INTO Pegawai (NIP, Nama, Golongan, Alamat, Kota, No_Telepon)  
values ('P-001', 'Edhi Nugroho', 1, 'Jl P No 3', 'Semarang',  
'0246789')
```

akan menyisipkan satu buah record ke dalam tabel Pegawai.

### SELECT

Perintah SELECT merupakan perintah yang sangat penting karena digunakan untuk mengambil sejumlah record dari satu atau lebih tabel. Perintah SELECT mempunyai sintak sebagai berikut :

```
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]  
* | expression [ AS output_name ] [, ...]  
[ FROM from_item [, ...] ]  
[ WHERE condition ]  
[ GROUP BY expression [, ...] ]  
[ HAVING condition [, ...] ]  
[ { UNION | INTERSECT | EXCEPT } [ ALL ] select ]  
[ ORDER BY expression [ ASC | DESC | USING operator ] [, ...] ]  
[ LIMIT { count | ALL } ]  
[ OFFSET start ]  
[ FOR UPDATE [ OF table_name [, ...] ] ]
```

dengan *from\_item* dapat berisi salah satu dari :

```
[ ONLY ] table_name [ * ] [ [ AS ] alias [ ( column_alias  
[, ...] ) ] ]  
( select ) [ AS ] alias [ ( column_alias [, ...] ) ]  
function_name ( [ argument [, ...] ] ) [ AS ] alias [ ( column_alias  
, ...)| column_definition [, ...] ) ]
```

```
function_name ( [ argument [, ...] ] ) AS ( column_definition
[, ...] )
from_item [ NATURAL ] join_type from_item [ ON join_condition |
USING ( join_column [, ...] ) ]
```

Contoh-contoh berikut ini menunjukkan bagaimana menggunakan perintah SELECT untuk mengambil record seperti yang kita inginkan.

```
SELECT * FROM Pegawai
```

Perintah di atas akan mengambil semua record yang ada di tabel Pegawai.

```
SELECT * FROM Pegawai WHERE NIP = '0001'
```

Perintah di atas akan mengambil record di tabel Pegawai yang field NIP berisi '0001'

```
SELECT nip, nama FROM Pegawai
```

akan mengambil semua record dari tabel Pegawai tetapi hanya untuk field NIP dan NAMA saja.

```
SELECT nip, nama FROM Pegawai WHERE Golongan > 1 and Kota = 'Semarang'
```

akan mengambil field NIP, NAMA dari record-record yang field Golongan berisi lebih besar dari 1 dan field Kota berisi 'Semarang'.

```
SELECT nip, nama FROM pegawai where nama like '%Rudi%'
```

akan menghasilkan semua record yang isi field nama berisi kata Rudi, seperti 'Rudi Sujarwo', 'Masrudi', 'Surudi Sudiro' dan sebagainya, tetapi

```
SELECT nip, nama FROM pegawai where nama like '%Rudi'
```

hanya akan menghasilkan record yang isi field nama diawali dengan Rudi, sehingga apabila ada record-record berisi 'Rudi Sujarwo', 'Masrudi', 'Surudi Sudiro' maka hanya record yang berisi 'Rudi Sujarwo' yang memenuhi kriteria tersebut.

```
SELECT DISTINCT (NAMA), Golongan FROM Pegawai
```

akan mengambil semua record dari Pegawai tetapi record-record yang isi field NAMA-nya sama hanya diambil satu kali.

```
SELECT nip, nama, golongan, gaji_pokok, gaji_pokok * 0.10 as pajak
FROM view_pegawai
```

akan mengambil semua record dari view\_pegawai dan menambahkan / menghitung gaji\_pokok \* 0.10 serta menampilkan hasil perhitungan sebagai field pajak.

```
SELECT * FROM pegawai ORDER BY nama
```

akan mengambil semua record dari tabel Pegawai dan mengurutkan record berdasarkan field nama.

```
SELECT MAX(golongan) As gol_terbesar FROM pegawai
```

akan mengambil field golongan yang berisi nilai golongan terbesar dari tabel Pegawai.

## UPDATE

Perintah UPDATE dapat digunakan untuk mengubah isi satu atau beberapa record yang memenuhi kriteria yang kita tentukan. Perintah UPDATE mempunyai sintak sebagai berikut :

```
UPDATE [ ONLY ] table SET column = { expression | DEFAULT } [, ...]  
[ WHERE condition ]
```

dengan *condition* berisi ekspresi yang menyatakan kriteria record-record yang akan diubah.

Perintah seperti di bawah ini :

```
UPDATE pegawai SET nama = 'Rudi S', golongan = 4 WHERE nip = 'P-001'
```

akan mengubah isi field nama menjadi 'Rudi S' dan field golongan menjadi berisi 4 bagi record yang isi field nip-nya sama dengan 'P-001'.

## DELETE

Menghapus record dapat dilakukan dengan mengirimkan perintah DELETE. Perintah DELETE mempunyai perintah sebagai berikut :

```
DELETE FROM [ ONLY ] table [ WHERE condition ]
```

sehingga perintah seperti

```
DELETE FROM pegawai WHERE nip = 'P-002'
```

akan menghapus semua record yang isi field nip sama dengan 'P-002', sedangkan

```
DELETE FROM pegawai WHERE nama like '%RUDI%'
```

akan menghapus semua record yang field nama berisi kata-kata 'RUDI'.

## **BAB 11 MEMBUAT KONEKSI KE REMOTE DATABASE**

### **APA YANG AKAN KITA PELAJARI ?**

- Metoda-metoda koneksi ke remote database server
- Membuat koneksi BDE ke remote database server
- Membuat koneksi ODBC ke remote database server
- Membuat koneksi dbExpress ke remote database server

**WAKTU LATIHAN : 2 JAM**

## **Membuat koneksi ke server**

Bagaimana menghubungkan Delphi ke server database ? Ada tiga cara yang dapat anda lakukan, yaitu :

- Menggunakan BDE
- Menggunakan ODBC
- Menggunakan dbExpress

### **Menggunakan BDE**

Delphi sudah menyediakan sejumlah driver BDE untuk melakukan koneksi ke server database. Server database yang sudah tersedia driver BDE-nya antara lain :

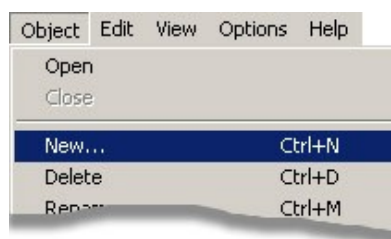
- Interbase / Firebird
- Oracle
- Sybase
- MS-SQL Server

Dalam contoh berikut ini, kita akan membuat koneksi ke database yang dikelola menggunakan MS-SQL. Untuk itu anda harus sudah berhasil memasang MS-SQL server sebelum mencoba langkah-langkah berikut ini. Sebelum membuat koneksi menggunakan BDE Administrator anda harus mencatat informasi-informasi berikut ini :

- Nama atau nomor ip server database, pada contoh ini digunakan localhost.
- Nama database yang akan digunakan, pada contoh ini digunakan database Northwind.
- Nama user yang akan digunakan untuk melakukan koneksi, pada contoh ini digunakan nama user = sa.

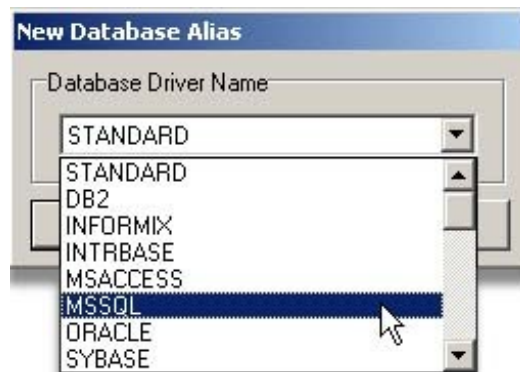
Lakukan langkah-langkah berikut ini untuk membuat sebuah alias yang menghubungkan aplikasi anda ke server database.

1. Jalankan BDE Administrator
2. Pilih menu Object | New



**Gambar 12.112. Membuat alias baru**

- Pilih driver MSSQL (Gambar 11.2)



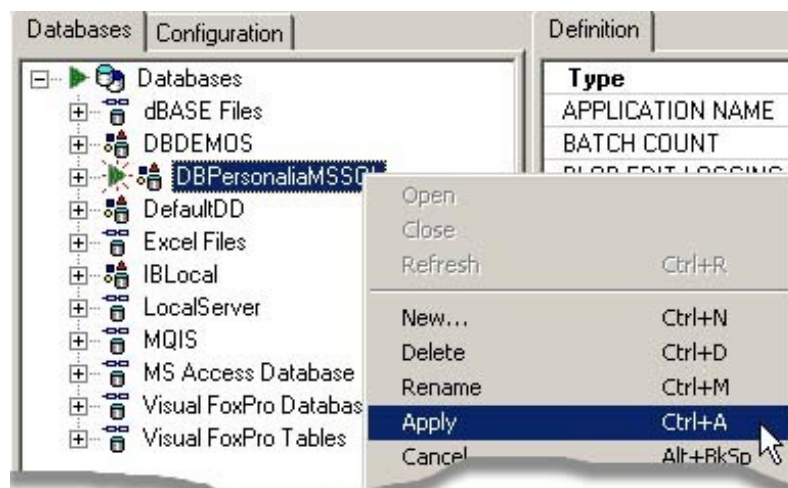
**Gambar 12.113 Driver MSSQL**

- Atur agar konfigurasi dari alias ini seperti pada Gambar 11.3

Definition	
<b>Type</b>	MSSQL
APPLICATION NAME	
ENABLE SCHEMA CACHE	FALSE
HOST NAME	localhost
LANGDRIVER	
QUERY TIME	
SCHEMA CACHE TIME	-1
SERVER NAME	localhost
SQLPASSTHRU MODE	SHARED AUTOCOMMIT
SQLQRYMODE	
TDS PACKET SIZE	4096
USER NAME	sa

**Gambar 12.114 Konfigurasi BDE MSSQL**

- Simpan konfigurasi sebagai DBPersonaliaMSSQL (Gambar 11.4)



**Gambar 12.115 Menyimpan konfigurasi sebagai DBPersonaliaMSSQL**

### Menggunakan ODBC

ODBC (Open Data Base Connectivity) merupakan bakuan yang dibuat oleh Microsoft bagi para pembuat database sistem. Bakuan tersebut memberikan syarat-syarat bagi pembuatan driver yang digunakan untuk melakukan akses ke database sistem. Apabila database sistem yang anda gunakan tidak didukung oleh BDE maka anda dapat menggunakan alternatif ODBC untuk menghubungkan aplikasi Delphi anda dengan database sistem.

Konfigurasi koneksi melalui ODBC disimpan sebagai DSN (Data Source Names). Windows membedakan DSN ke dalam tiga kelompok, yaitu :

- User DSN : DSN ini hanya dapat digunakan oleh user yang membuat DSN.
- System DSN : DSN ini dapat digunakan oleh semua pemakai yang login ke mesin dimana DSN dibuat.
- File DSN : DSN ini berupa file konfigurasi dan dapat digunakan oleh pemakai yang mempunyai hak pemakaian bersama (sharing).

Untuk membuat alias koneksi melalui ODBC, lakukan langkah-langkah berikut ini :

1. Jalankan ODBC manager melalui menu (windows) Start | Control Panel | Administrative Tools (Gambar 11.5)



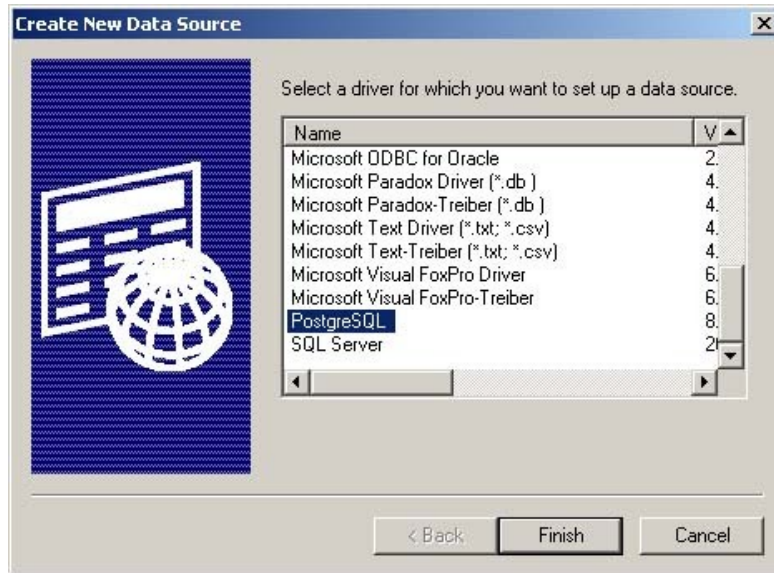
**Gambar 12.116 ODBC Manager**

2. Pilih tabstop System DSN.



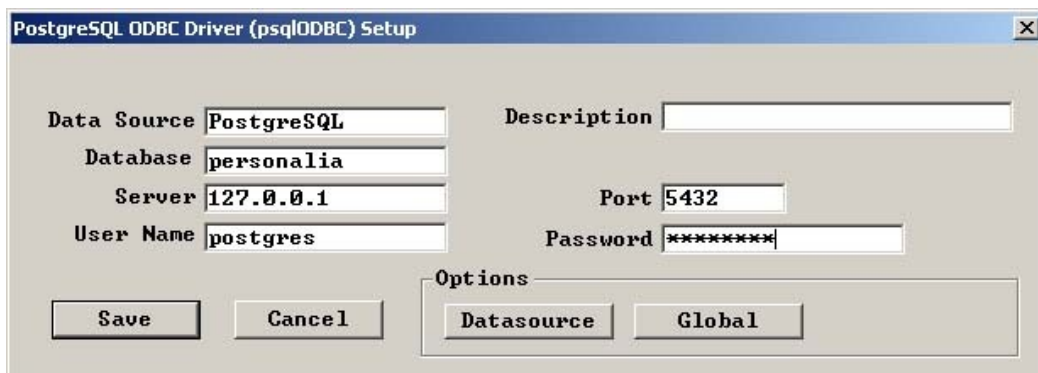
**Gambar 12.117. System DSN**

3. Klik tombol Add, pilih driver PostgreSQL Server. **Pemilihan driver tergantung kepada sistem database yang anda gunakan dan sebelumnya anda harus sudah memasang (instal) driver tersebut, pada contoh ini kita menggunakan driver PostgreSQL untuk melakukan koneksi ke server database PostgreSQL.**



**Gambar 12.118 Driver ODBC**

4. Klik tombol Finish. Setelah memilih driver yang akan digunakan, langkah berikutnya adalah mengatur konfigurasi server. Pengaturan konfigurasi sangat bergantung kepada sistem database yang anda gunakan, saya sarankan untuk melihat buku petunjuk dari sistem database yang anda gunakan mengenai bagaimana mengatur konfigurasi DSN bagi sistem database tersebut. Pada contoh ini, saya menggunakan PostgreSQL sebagai server database.



**Gambar 12.119 Konfigurasi koneksi PostgreSQL**

5. Klik tombol Save. Perhatikan SystemDSN mempunyai satu tambahan konfigurasi, yaitu PostgreSQL. Anda dapat mengubah konfigurasi dengan memilih dsn dan kemudian klik tombol Configure.



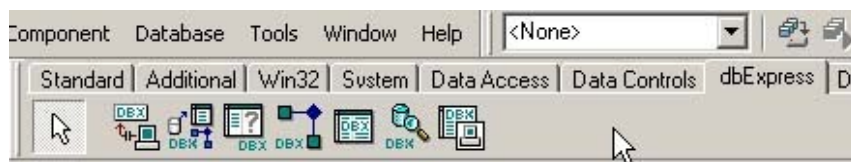
**Gambar 12.120 DSN PostgreSQL**

### Menggunakan dbExpress

Konfigurasi dbExpress hanya dapat dilakukan melalui Delphi. Ingat, tidak semua sistem database mempunyai atau menyediakan driver dbExpress. Apabila aplikasi anda dirancang untuk bekerja dengan berbagai macam sistem database maka dbExpress bukan pilihan yang tepat. Untuk sistem yang dirancang dapat bekerja dengan berbagai macam sistem database maka pilihan paling aman adalah melalui konfigurasi ODBC.

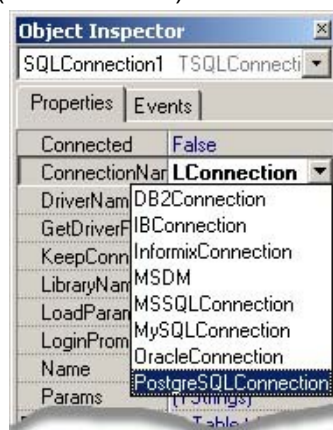
**Untuk menjalankan langkah-langkah berikut ini anda harus sudah memasang dbExpress driver untuk PostgreSQL. Versi demo driver tersebut dapat anda unduh (download) di <http://www.vitavoom.com>.**

1. Jalankan Delphi
2. Tambahkan satu komponen SQLConnection  dari palette dbExpress.



**Gambar 12.121 Palette dbExpress**

3. Di **Object Inspector**, atur agar properti ConnectionName menggunakan PostgreSQLConnection (Gambar 11.11)



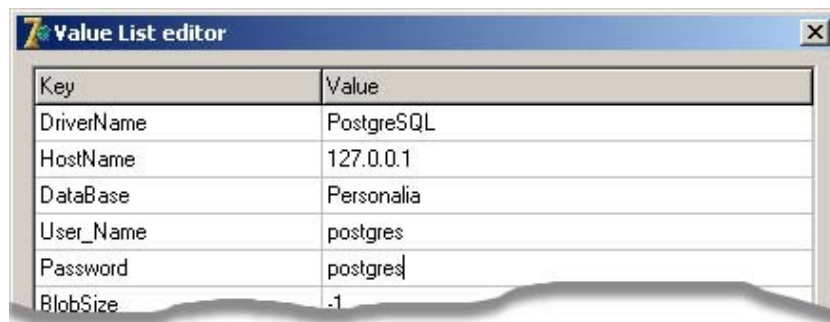
**Gambar 12.122 Menggunakan konfigurasi koneksi MSSQLConnection**

4. Delphi akan mengambil konfigurasi dari koneksi tersebut dan kemudian mengisi properti lain sesuai dengan konfigurasi tersebut.
5. Langkah berikutnya adalah mengatur konfigurasi server yang akan dikoneksikan dengan aplikasi anda.
6. Klik ganda properti Params. Delphi akan menampilkan kotak dialog untuk mengatur parameter server. Beberapa fungsi dari parameter yang penting diperlihatkan di Tabel 11.1

**Tabel 12-36. Fungsi parameter pada koneksi dbExpress**

Parameter	Keterangan
HostName	Alamat IP atau nama mesin dimana server database berada
DatabaseName	Nama database yang akan digunakan
User_Name	Nama user untuk login ke database server
Password	Kata sandi (password) untuk login ke database server

7. Atur agar isi dari parameter seperti pada Gambar 11.13. Khusus untuk parameter User\_Name dan Password sesuaikan dengan konfigurasi dari database server anda.
















**Gambar 12.123. Parameter untuk koneksi ke database Personalia**

### ***Komponen database sesuai dengan metoda koneksi***

Pemakaian komponen database bergantung kepada metoda koneksi yang anda gunakan. Tabel 11.2 memberikan ringkasan tentang komponen-komponen database sesuai dengan koneksi.

**Tabel 12-37 Komponen database sesuai dengan jenis koneksi**

Jenis Koneksi	Komponen	Icon
BDE / ODBC	TQuery	

	TStoredProc	
	TDatabase	
	TSession	
	TBatchMove	
	TUpdateSQL	
	TNestedTable	
dbExpress	TSQLConnection	
	TSQLDataSet	
	TSQLQuery	
	TSQLStoredProc	
	TSQLTable	
	TSQLMonitor	
	TSimpleDataSet	

## Mengirim query ke server

Delphi menyediakan dua macam metoda untuk mengirim *query* ke server, yaitu :

- ExecSQL : Metoda ini digunakan apabila *query* tidak mengembalikan data, seperti misalnya INSERT, UPDATE, DELETE.
- Open : Metoda ini digunakan apabila *query* mengembalikan data, seperti : SELECT.

## **BAB 12 MENGGUNAKAN TQUERY**

### **APA YANG AKAN KITA PELAJARI ?**

- TDatabase dan TQuery
- Menggunakan TQuery untuk menyisipkan data
- Menggunakan TQuery untuk mengambil data
- Menggunakan TQuery untuk menghapus data

**WAKTU LATIHAN : 2 JAM**

## ***TDatabase***

Komponen TDatabase merupakan penghubung dari komponen dataset ke alias / konfigurasi server. Dengan menggunakan TDatabase maka koneksi ke database server dari komponen-komponen dataset dapat diatur melalui satu tempat.

Apa bedanya aplikasi yang menggunakan TDatabase dan tidak ? Setiap kali komponen dataset melakukan hubungan ke server database maka hubungan tersebut disebut *session*. Setiap session harus diotorisasi atau disahkan oleh server database melalui mekanisme login. Apabila aplikasi anda mempunyai 5 buah komponen TDataset maka ada 5 session yang harus ditangani oleh aplikasi anda. Hal semacam ini tentunya cukup merepotkan karena pemakai harus login sebanyak 5 kali!.

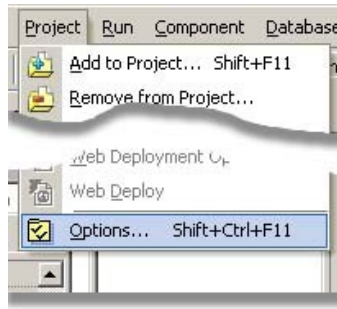
Tetapi, situasinya akan berbeda sangat jauh apabila session tersebut ditangani oleh komponen TDatabase. Dengan menghubungkan komponen dataset ke TDatabase maka session ke server database cukup dilakukan oleh komponen TDatabase. Properti yang penting dari TDatabase diperlihatkan pada Tabel 12-1.

**Tabel 13-38 Properti dari TDatabase**

<b>Properti</b>	<b>Keterangan</b>
Alias	Nama Alias atau DSN yang akan digunakan sebagai konfigurasi koneksi
Connected	True = melakukan koneksi ke server; False = menutup koneksi
DatabaseName	Nama database yang akan digunakan. Apabila DatabaseName sama dengan nama alias atau DSN maka properti Alias dapat dikosongkan.
Name	Nama komponen
Params	Daftar parameter yang digunakan untuk melakukan koneksi, apabila tidak diisi maka koneksi menggunakan konfigurasi seperti yang tercatat di Alias atau DSN.

Berikut ini, kita akan membuat aplikasi untuk mengimplementasikan sistem personalia dengan menggunakan BDE.

1. Buat aplikasi baru melalui menu File | New | Application.
2. Ubah properti Name dari Form1 menjadi GapokFrm.
3. Simpan aplikasi sebagai Personalia.dpr.
4. Tambahkan satu form bertipe Data Module melalui menu File | New | Data Module.
5. Pilih menu Project | Options.




**Gambar 13.124 Menu Project | Options**

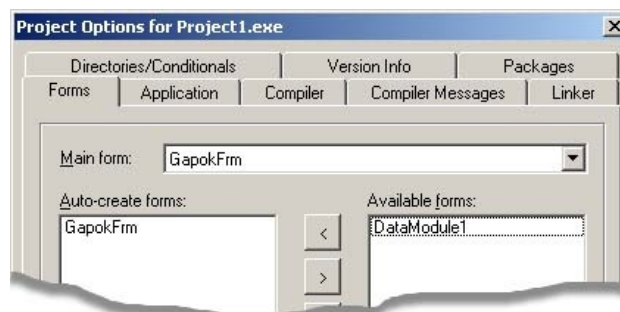
6. Delphi akan menampilkan kotak dialog seperti Gambar 12.2. Pilih tab stop Forms.



**Gambar 13.125 Jenis Forms**

**Apa bedanya Auto Create Forms dan Available Forms?** Auto Create Forms adalah forms yang akan dibuat secara otomatis oleh Delphi (tepatnya oleh aplikasi anda) sedangkan Available Forms adalah forms yang secara eksplisit harus anda buat sendiri melalui perintah Create.

7. Pilih form DataModule2 dan kemudian klik tombol  untuk memindahkan DataModule2 dari Auto Create Forms ke Available Forms (Gambar 12.3). Klik tombol OK.



**Gambar 13.126 Available Forms**

8. Tambahkan komponen TDatabase ke form DataModule2.
9. Atur agar properti dari komponen TDatabase seperti pada Tabel 12-2

**Tabel 13-39 Konfigurasi TDatabase**

Properti	Keterangan
Alias	PostgreSQL
Connected	False
DatabaseName	DBPersonalia

### **TQuery**

*Query* ke server database yang menggunakan koneksi BDE atau ODBC dapat dikirim melalui komponen TQuery. Properti dari TQuery diperlihatkan pada Tabel 12.3

**Tabel 13-40 Properti dari TQuery**

Properti	Keterangan
DatabaseName	Nama Alias atau DSN yang akan digunakan sebagai konfigurasi koneksi, anda juga dapat menggunakan isi DatabaseName dari TDatabase.
Active	True = melakukan koneksi ke server; False = menutup koneksi
SQL	Berisi <i>query</i> yang akan dikirim ke server.
Params	Berisi isi parameter dari query, apabila query membutuhkan parameter
Datasource	Berisi datasource yang digunakan sebagai Master dalam relasi Master-Detail

1. Tambahkan satu buah TQuery ke DataModule2 dan atur agar properti dari TQuery tersebut seperti Tabel 12-4.

**Tabel 13-41 Properti dari Query1**

Properti	Keterangan
DatabaseName	DBPersonalia
Name	QGaPok



**Gambar 13.127 Pengaturan properti Query1**

2. Tambahkan satu buah TDataSource, dari palette DataAccess. Atur agar properti dari Datasource tersebut seperti Tabel 12.5

**Tabel 13-42 Properti dari Query1**

Properti	Keterangan
Dataset	QGapok
Name	DSGapok

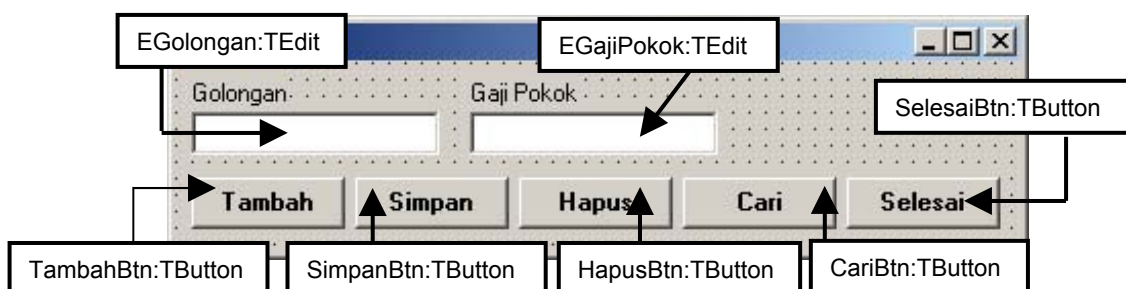


**Gambar 13.128. DataModule1**

### Memasukkan data menggunakan TQuery

Meskipun anda dapat langsung memasukkan query ke dalam properti SQL dari TQuery, tetapi untuk contoh berikut ini, kita akan melihat bagaimana menggunakan TQuery secara programming. Hey!! Bukankah anda sedang belajar membuat program ???

1. Pindah ke GaPokFrm (masih ingat caranya ? Tekan F12 sampai anda melihat Code Editor dan kemudian pilih tab GaPokForm, tekan F12 sehingga anda melihat form GapokFrm).
2. Masukkan beberapa komponen seperti pada Gambar 12.2.



**Gambar 13.129 Tata letak GapokFrm**

3. Buat event handler OnCreate untuk GapokFrm. Di dalam event OnCreate tersebut kita akan membuat form DataModule1 dan kemudian membuat koneksi ke server database. Apabila koneksi gagal dilakukan maka aplikasi akan dihentikan. Selain itu, kita juga mematikan (disabled) tombol Simpan, Batal, Hapus dan Cari, serta semua komponen TEdit.

#### Listing 13-39. Event OnCreate dari GapokFrm

```
procedure TGapokFrm.FormCreate(Sender: TObject);
begin
    //buat form DataModule1
    DataModule1:=TDataModule1.Create(Self);

    //lakukan koneksi ke server
    DataModule1.Datasource1.Connected:=True;

    //cek apakah koneksi berhasil dilakukan, apabila gagal
    //maka aplikasi dihentikan
    if not DataModule1.Datasource1.Connected then
        Application.Terminate;

    //matikan tombol yang tidak boleh digunakan
    SimpanBtn.Enabled:=False;
    HapusBtn.Enabled:=False;
    CariBtn.Enabled:=False;
    BatalBtn.Enabled:=False;

    //matikan komponen Edit
    EGolongan.Enabled:=False;
    EGajiPokok.Enabled:=False;
end;
```

4. Buat event handler OnClick untuk tombol TambahBtn (Listing 12-2). Di dalam event handler ini, kita akan mengaktifkan tombol Simpan, Batal dan komponen TEdit.

#### Listing 13-40 Event handler OnClick dari TambahBtn

```
procedure TGapokFrm.TambahBtnClick(Sender: TObject);
begin
    EGolongan.Enabled:=True;
    EGajiPokok.Enabled:=True;
    EGolongan.Text:='';
    EGajiPokok.Text:='';
    SimpanBtn.Enabled:=True;
    BatalBtn.Enabled:=True;
    ActiveControl:=EGolongan;
end;
```

5. Buat event handler OnClick untuk tombol Simpan. Di dalam event handler ini, kita akan mengaktifkan tombol Simpan (Listing 12-3).

#### Listing 13-41 Event handler OnClick dari SimpanBtn

```
procedure TGapokFrm.SimpanBtnClick(Sender: TObject);
begin
    //cek apakah golongan dan gajipokok diisi, apabila tidak batalkan
    if (EGolongan.Text <> '') and (EGajiPokok.Text <> '') then
        begin
            with DataModule1.QGapok do
                begin
                    Close; //matikan komponen
                    SQL.Clear; //bersihkan query sebelumnya
                end
            end
        end;
```

```

        //siapkan query
        SQL.Add('INSERT INTO gapok (Golongan,Gaji_Pokok) '
            + ' values (:Golongan, :Gaji_Pokok)');
        //siapkan parameter
        ParamByName('GOLONGAN').AsInteger:=StrToInt(EGolongan.Text);
        ParamByName('GAJI_POKOK').AsFloat:=StrToFloat(EGajiPokok.Text);
        Prepare; //siapkan
        ExecSQL; //eksekusi query
        Close;
    end;//with DataModule1.QGapok
end //if EGolongan.Text...
else
begin
    MessageDlg('Golongan dan atau Gaji Pokok tidak '
        + ' boleh kosong',mtWarning,[mbOk],0);
end;//else if EGolongan.Text...

//atur tombol dan form
SimpanBtn.Enabled:=False;
BatalBtn.Enabled:=False;
TambahBtn.Enabled:=True;
EGolongan.Text:='';
EGajiPokok.Text:='';
EGolongan.Enabled:=False;
EGajiPokok.Enabled:=False;
end;

```

6. Buat event handler OnClick untuk tombol BatalBtn seperti Listing 12-4.

#### Listing 13-42 Event handler OnClick dari BatalBtn

```

procedure TGapokFrm.BatalBtnClick(Sender: TObject);
begin
    //atur tombol dan form
    SimpanBtn.Enabled:=False;
    BatalBtn.Enabled:=False;
    TambahBtn.Enabled:=True;
    EGolongan.Text:='';
    EGajiPokok.Text:='';
    EGolongan.Enabled:=False;
    EGajiPokok.Enabled:=False;
end;

```

7. Kompilasi program dan jalankan. Cobalah meng-klik tombol Tambah, mengisi data dan kemudian klik tombol Simpan untuk menyimpan atau tombol Batal untuk membatalkan proses penambahan data. Untuk melihat isi dari tabel GaPok anda dapat menggunakan program-program sesuai dengan server database yang anda gunakan.

### Query dengan parameter

Apabila anda perhatikan Listing 12-3 maka anda akan melihat bahwa kita mengirim perintah INSERT dalam bentuk :

```
INSERT INTO gapok (Golongan,Gaji_Pokok) values (:Golongan,
:Gaji_Pokok)
```

Query seperti di atas disebut sebagai *query dengan parameter*. Kita menggunakan *query dengan parameter* apabila query yang kita kirim digunakan untuk berbagai macam data. Untuk memasukkan data golongan = 1 dan gaji\_pokok = 100000 ke dalam tabel gapok maka kita mengirim perintah sql :

```
INSERT INTO gapok (Golongan,Gaji_Pokok) values (1,100000)
```

dan untuk memasukkan data golongan = 2, gaji\_pokok = 200000 maka kita mengirim perintah sql :

```
INSERT INTO gapok (Golongan,Gaji_Pokok) values (2,200000)
```

Perhatikan, kita mengirim perintah sql yang sama tetapi dengan nilai yang berbeda. Delphi menyediakan solusi dari situasi ini dengan menyediakan mekanisme *query dengan parameter*. Query tersebut diubah menjadi :

```
INSERT INTO gapok (Golongan,Gaji_Pokok) values (:Golongan,
:Gaji_Pokok)
```

dimana :Golongan dan :Gaji\_Pokok disebut sebagai parameter. Perintah untuk mengakses parameter menggunakan bentuk :

```
Query1.ParamsByName('<nama_parameter>').As<tipe_data> := <isian>;
```

Pada saat memberikan perintah ParamsByName, <tipe\_data> diganti sesuai dengan tipe field yang akan menerima data tersebut. <tipe\_data> yang sering digunakan ditunjukkan pada Tabel 12-6.

**Tabel 13-43. <tipe\_data> untuk parameter query**

Tipe data	<tipe_data>
Integer	AsInteger
String	AsString
Float	AsFloat
DateTime	AsDateTime

Sehingga untuk mengirim perintah sql :

```
INSERT INTO gapok (Golongan,Gaji_Pokok) values (1,100000)
```

maka kita melakukannya dalam dua tahap, yaitu :

1. Memasukkan query ke dalam properti SQL melalui perintah :

```
SQL.Add('INSERT INTO gapok (Golongan,Gaji_Pokok) values '
+' (:Golongan,:Gaji_Pokok)');
```

2. Memasukkan parameter melalui perintah :

```
//siapkan parameter
ParamByName('GOLONGAN').AsInteger:=StrToInt(EGolongan.Text);
ParamByName('GAJI_POKOK').AsFloat:=StrToFloat(EGajiPokok.Text);
```

3. Menghubungkan parameter dengan query melalui perintah :

```
Prepare; //siapkan
```

4. Mengirim query ke server melalui perintah :

```
ExecSQL; //eksekusi query
```

### Mengambil record menggunakan TQuery

Setelah kita berhasil menggunakan TQuery untuk memasukkan record baru maka pada bagian ini kita akan mempelajari bagaimana mengambil record menggunakan TQuery, selain itu anda juga akan mempelajari bagaimana membuat tampilan semacam DBGrid.

Perintah sql untuk mengambil seluruh isi tabel gapok adalah :

```
SELECT * FROM gapok
```

Perintah sql SELECT tersebut dikirim menggunakan metoda Open karena perintah SELECT mengembalikan record-record yang memenuhi kriteria pengambilan tersebut. Untuk mengambil isi field dari masing-masing record dapat dilakukan dengan perintah FieldByName(<nama\_field>).As<tipe\_data>. Listing 12-5 memberikan algoritma untuk mengambil record dari tabel.

#### Listing 13-43 Algoritma mengambil record



```
:
Query1.SQL.Add(<query berisi select>);
Query1.Open; //kirim query
while not Query1.EOF do //selama masih ada record
begin
    var_field_1 := Query1.FieldByName(<nama_field_1>).As<tipe_data>;
    var_field_2 := Query1.FieldByName(<nama_field_2>).As<tipe_data>;
    :
    dan seterusnya
    Query1.Next; //ambil record berikutnya
end;
:
```

Kita akan menggunakan komponen TListView sebagai tempat untuk menampilkan isi dari tabel GaPok. Komponen TListView mempunyai properti Items yang bertipe TListItem dan digunakan untuk menyimpan isian TListView. Kita dapat menampilkan TListView dalam salah satu dari tiga tampilan, yaitu :

- vsIcon : Items ditampilkan sebagai kumpulan icon.
- vsList : Items ditampilkan sebagai daftar.
- vsReport : Items ditampilkan sebagai baris dan kolom, dengan tiap kolom mempunyai judul kolom

- vsSmallIcon : Items ditampilkan sebagai icon tetapi dengan ukuran kecil.

Dengan mengatur beberapa properti lain maka kita dapat meniru tampilan DBGrid dalam menampilkan record.

1. Tambahkan komponen TListView  dari *Component Palette* Win32.
2. Klik ganda komponen ListView1. Delphi akan menampilkan editor Column.
3. Klik tombol  dua kali untuk membuat dua buah judul kolom seperti Gambar 12.



**Gambar 13.130 Kolom dari ListView1.**

4. Masih menggunakan Editor Column, pindah ke Object Inspector (F11). Isi properti Caption dari 1-TListColumn dengan Gaji Pokok.



**Gambar 13.131 Mengisi properti Caption dari 1-ListColumn.**

5. Lakukan hal yang sama dengan 0-TListColumn tetapi menggunakan Caption Golongan.
6. Kembali ke GaPokFrm. Pilih ListView1 dan kemudian aktifkan Object Inspector (F11).
7. Atur agar properti dari ListView1 seperti Tabel 12-6.

**Tabel 13-44 Properti ListView1**

Properti	Isi properti
Name	GapokView
ReadOnly	True
RowSelect	True
ViewStyle	vsReport

8. Setelah mengatur properti ListView1 seperti Tabel 12-6 maka anda akan memperoleh tampilan GaPokFrm seperti Gambar 12.9.



**Gambar 13.132 Tata letak GapokFrm dengan TreeView**

9. Buat deklarasi metoda private dari GapokFrm dengan nama AmbilGapok seperti Listing 12-5.

**Listing 13-44 Deklarasi metoda AmbilGapok**

```
:<dipotong>
private
{ Private declarations }
  procedure AmbilGapok;
public
{ Public declarations }
end;
```

10. Tulis implementasi dari metoda AmbilGapok seperti Listing 12-6.

**Listing 13-45 Implementasi metoda AmbilGapok**

```
procedure TGapokFrm.AmbilGapok;
var
  Item:TListItem; //variabel untuk menampung item baru
begin
  //bersihkan GapokView
  GapokView.Items.Clear;
  with DataModule1.QGapok do
  begin
    //siapkan query
    Close;
    SQL.Clear;

    //query mengambil isi tabel gapok
    SQL.Add('select * from gapok');
    Prepare;
    //kirim isi query
    Open;
    //selama masih ada record
    while not EOF do
    begin
      //tambahkan satu buah item
      Item:=GapokView.Items.Add;
      //isi item dengan isi record
      Item.Caption := IntToStr(FieldByName('GOLONGAN').AsInteger);
      Item.SubItems.Add(
```

```

        IntToStr(FieldByName('GAJI_POKOK').AsInteger));
        //ambil record berikutnya
        Next;
    end; //while not EOF
    //tutup query
    Close;
end; //with DataModule1.Gapok..
end;

```

- Ubah event handler OnCreate dari GapokFrm dengan menambahkan baris untuk menjalankan AmbilGapok seperti pada Listing 12-7.

#### Listing 13-46 Perubahan event OnCreate dari GapokFrm

```

procedure TGapokFrm.FormCreate(Sender: TObject);
begin
    //buat form DataModule1
    :dipotong, tetap seperti Listing sebelumnya
    //ambil isi gapok
    AmbilGapok; //tambahkan baris ini untuk menjalankan AmbilGapok
end;

```

- Sesuaikan pula event handler OnClick dari SimpanBtn menjadi seperti Listing 12-9.

#### Listing 13-47 Penyesuaian event handler OnClick dari SimpanBtn

```

procedure TGapokFrm.SimpanBtnClick(Sender: TObject);
begin
    : dipotong, sama seperti sebelumnya
    ExecSQL; //eksekusi query
    Close;
    //ambil isi gapok
    AmbilGapok; //tambahkan perintah ini untuk mengambil isi gapok
    : dipotong, sama seperti sebelumnya
end;

```

- Hasil dari program diperlihatkan pada Gambar 12.10.

The screenshot shows a Windows-style application window titled "GaPok". It features two input fields at the top labeled "Golongan" and "Gaji Pokok". Below these fields is a row of six buttons: "Tambah", "Simpan", "Batal", "Hapus", "Cari", and "Selesai". At the bottom of the window is a data table with two columns: "Golongan" and "Gaji Pokok". The table contains three rows of data. The first row is highlighted with a blue selection bar. The data in the table is as follows:

Golongan	Gaji Pokok
1	100000
2	200000
3	400000

Gambar 13.133 Isi tabel gapok.

## Menggunakan konstanta sebagai *query*

Seringkali *query* yang kita gunakan tidak hanya digunakan oleh satu metoda atau procedure tetapi juga digunakan oleh berbagai procedure. Tentunya, tidak efisien apabila setiap kali akan menggunakan *query* tersebut kita harus menuliskan isi *query*, akan lebih mudah mengelola *query* dan lebih efisien apabila *query* tersebut anda tuliskan sebagai sebuah konstanta dan kemudian ketika akan menggunakan *query* tersebut, kita cukup menuliskan nama konstantanya.

1. Tulis dua buah konstanta untuk mewakili perintah sql INSERT dan SELECT seperti pada Listing 12-10.

### Listing 13-48 Konstanta untuk mewakili INSERT dan SELECT

```
implementation
const
  //query
  AMBIL_GAPOK = 'SELECT * FROM gapok order by golongan';
  INSERT_GAPOK = 'INSERT INTO gapok (Golongan,Gaji_Pokok) '
    + ' values (:Golongan, :Gaji_Pokok)';
```

2. Sesuaikan event handler OnClick dari SimpanBtn menjadi seperti Listing 12-11.

### Listing 13-49 Penyesuaian event handler OnClick dari SimpanBtn

```
procedure TGapokFrm.SimpanBtnClick(Sender: TObject);
begin
  //cek apakah golongan dan gajipokok diisi, apabila tidak batalkan
  if (EGolongan.Text <> '') and (EGajiPokok.Text <> '') then
  begin
    with DataModule1.QGapok do
    begin
      Close; //matikan komponen
      SQL.Clear; //bersihkan query sebelumnya
      //siapkan query
      SQL.Add(INSERT_GAPOK);
      //siapkan parameter
      ParamByName('GOLONGAN').AsInteger:=StrToInt(EGolongan.Text);
      ParamByName('GAJI_POKOK').AsFloat:=StrToFloat(EGajiPokok.Text);
      Prepare; //siapkan
      ExecSQL; //eksekusi query
      Close;
    end;
  end;
```

## Mengubah isi record

Perintah sql untuk mengubah isi record adalah :

```
UPDATE <table> set <field> = <isi_baru> [WHERE <ekspresi>]
```

Misal, kita ingin mengubah record dengan golongan = 1 dengan field gaji\_pokok semula berisi 100.000 menjadi berisi 200.000, maka perintah sql yang dikirim adalah :

```
UPDATE gapok SET gaji_pokok = 200000 WHERE golongan = 1
```

Dalam program ini kita akan memberikan fasilitas mengubah isi record dengan algorithma sebagai berikut :

1. Setiap kali pemakai memilih salah satu baris di ListView maka tombol UbahBtn diaktifkan.
2. Pada saat pemakai meng-klik tombol UbahBtn maka program akan mengambil isi baris yang dipilih dimana golongan diperoleh dari properti Selected.Caption dan gaji\_pokok diperoleh dari properti Selected.SubItems[0], isi properti tersebut dipindahkan ke komponen TEdit dan kemudian komponen TEdit diaktifkan. Pada saat yang bersamaan, tombol TambahBtn dan HapusBtn dimatikan sedangkan tombol SimpanBtn dan BatalBtn diaktifkan.
3. Untuk membedakan apakah data yang akan disimpan merupakan data baru atau data lama maka kita gunakan atribut fModeSimpan. fModeSimpan diisi 0 melalui event onClick dari TambahBtn apabila data merupakan data baru dan diisi 1 apabila data merupakan data lama melalui event onClick dari UbahBtn.
4. Pemakai menyimpan data dengan meng-klik tombol SimpanBtn. Dengan menguji apakah fModeSimpan berisi 0 atau 1 maka kita kirimkan perintah sql yang sesuai, yaitu INSERT apabila fModeSimpan berisi 0 dan UPDATE apabila fModeSimpan berisi 1.

Implementasi dari algorithma di atas dapat dilakukan sebagai berikut :

1. Tambahkan konstanta untuk mewakili perintah sql UPDATE serta mode simpan seperti ditunjukkan oleh Listing 12-12.

#### **Listing 13-50 Konstanta untuk mewakili perintah UPDATE**

```
implementation
const
  MODE_INSERT = 0;
  MODE_UPDATE = 1;
  //query
  AMBIL_GAPOK = 'SELECT * FROM gapok order by golongan';
  INSERT_GAPOK = 'INSERT INTO gapok (Golongan,Gaji_Pokok) '
    + 'values (:Golongan, :Gaji_Pokok)';
  UPDATE_GAPOK = 'UPDATE gapok SET Gaji_Pokok = :Gaji_Pokok '
    + 'where Golongan = :Golongan';
```

2. Buat event handler OnSelectItem dari GapokView. Event OnSelectItem akan dimunculkan apabila pemakai memilih salah satu bari dari GapokView. Kita akan memanfaatkan event ini untuk menyalakan dan mematikan tombol UbahBtn. Listing 12-13 menunjukkan isi dari event handler OnSelectItem.

#### **Listing 13-51 Event handler OnSelectItem dari GapokView**

```
procedure TGapokFrm.GapokViewSelectItem(Sender: TObject;
  Item: TListItem; Selected: Boolean);
begin
```

```

if Selected then UbahBtn.Enabled:=True else UbahBtn.Enabled:=False;
end;

```

3. Buat event handler OnClick dari UbahBtn seperti pada Listing 12-14.

#### Listing 13-52 Event handler OnClick dari UbahBtn

```

procedure TGapokFrm.UbahBtnClick(Sender: TObject);
begin
if GapokView.Selected <> nil then
begin
//isi komponen Edit dengan baris yang dipilih
EGolongan.Text:=GapokView.Selected.Caption;
EGajiPokok.Text:=GapokView.Selected.SubItems[0];

//hanya GajiPokok yang boleh diedit
EGolongan.Enabled:=False;
EGajiPokok.Enabled:=True;

//sesuaikan tombol
SimpanBtn.Enabled:=True;
BatalBtn.Enabled:=True;
TambahBtn.Enabled:=False;
HapusBtn.Enabled:=False;

//pindahkan kursor ke komponen EGajiPokok
ActiveControl:=EGajiPokok;

//catat mode simpan
fModeSimpan:=MODE_UPDATE;
end; //if GapokView.Selected..
end;

```

4. Sesuaikan event handler OnClick dari SimpanBtn sehingga nampak seperti Listing 12-15.

#### Listing 13-53 Penyesuaian event handler OnClick dari SimpanBtn

```

procedure TGapokFrm.SimpanBtnClick(Sender: TObject);
begin
//cek apakah golongan dan gajipokok diisi, apabila tidak batalkan
if (EGolongan.Text <> '') and (EGajiPokok.Text <> '') then
begin
with DataModule1.QGapok do
begin
Close; //matikan komponen
SQL.Clear; //bersihkan query sebelumnya
//siapkan query
case fModeSimpan of
MODE_INSERT : SQL.Add(INSERT_GAPOK);
MODE_UPDATE : SQL.Add(UPDATE_GAPOK);
end; //fModeSimpan
//siapkan parameter
ParamByName('GOLONGAN').AsInteger:=StrToInt(EGolongan.Text);
ParamByName('GAJI_POKOK').AsFloat:=StrToFloat(EGajiPokok.Text);
Prepare; //siapkan
ExecSQL; //eksekusi query
Close;

```

```

        //ambil dan tampilkan isi gapok
        AmbilGapok;
    end; //with DataModule1.QGapok
end //if EGolongan.Text...
else
begin
    MessageDlg('Golongan dan atau Gaji Pokok tidak boleh '
        + ' kosong',mtWarning,[mbOk],0);
end; //else if EGolongan.Text...

//atur tombol dan form
SimpanBtn.Enabled:=False;
BatalBtn.Enabled:=False;
TambahBtn.Enabled:=True;

EGolongan.Text:='';
EGajiPokok.Text:='';
EGolongan.Enabled:=False;
EGajiPokok.Enabled:=False;
end;

```

5. Sesuaikan event handler OnClick dari TambahBtn seperti Listing 12-16.

#### Listing 13-54 Penyesuaian event handler OnClick dari TambahBtn

```

procedure TGapokFrm.TambahBtnClick(Sender: TObject);
begin
    EGolongan.Enabled:=True;
    EGajiPokok.Enabled:=True;
    EGolongan.Text:='';
    EGajiPokok.Text:='';
    SimpanBtn.Enabled:=True;
    BatalBtn.Enabled:=True;
    ActiveControl:=EGolongan;
    fModeSimpan:=MODE_INSERT;
end;

```

## Menghapus record

Dengan menggunakan algoritma yang hampir sama dengan mengubah isi record, kita akan membuat fasilitas untuk menghapus isi record. Pemakai memilih isi record yang akan dihapus dari GapokView dan kemudian mengklik tombol HapusBtn untuk membuang record tersebut.

1. Sesuaikan event handler OnSelectItem dari GapokView menjadi seperti Listing 12-17.

#### Listing 13-55 Penyesuaian event OnSelectItem.

```

procedure TGapokFrm.GapokViewSelectItem(Sender: TObject;
    Item: TListItem; Selected: Boolean);
begin
    if Selected then
    begin
        HapusBtn.Enabled:=True;
        UbahBtn.Enabled:=True;
    end
    else
    begin

```

```

    HapusBtn.Enabled:=False;
    UbahBtn.Enabled:=False;
end;
end;

```

2. Buat event handler OnClick dari HapusBtn seperti pada Listing 12-18.

#### Listing 13-56 Event handle OnClick dari HapusBtn

```

procedure TGapokFrm.HapusBtnClick(Sender: TObject);
var
    Gol:integer;
begin
    //cek apakah ada baris yang dipilih
    if GapokView.Selected <> nil then
        begin
            //ambil golongan yang dipilih
            Gol:= IntToStr(GapokView.Selected.Caption);

            with DataModule1.QGapok do
                begin
                    Close;
                    SQL.Clear;

                    //siapkan query
                    SQL.Add(DELETE_GAPOK);
                    ParamByName('Golongan').AsInteger:=Gol;
                    Prepare;

                    //kirim query
                    ExecSQL;
                    Close;
                    AmbilGapok;
                end; //with DataModule1...
            end; //if GapokView..
        end;
    end;

```