

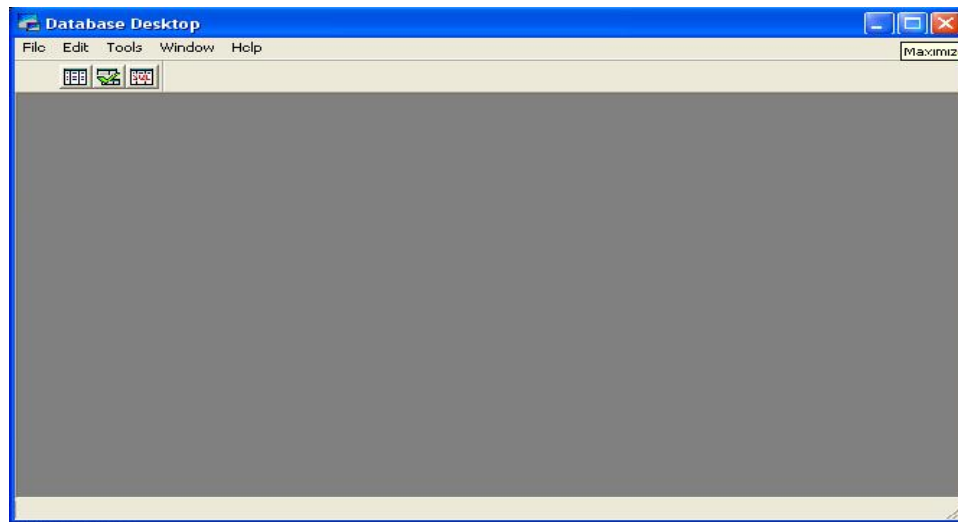
DATA BASE DESKTOP :

I. Database Desktop

Database desktop adalah fasilitas yang dimiliki Delphi untuk membantu membuat, menampilkan, mengurutkan, dan mencari data yang ada didalam tabel. Database Desktop dapat menangani tabel dari Paradox, dBase, dan tabel SQL

Cara memanggil Database Desktop :

1. Klik tombol Start yang ada ditaskbar, lalu pilih Programs, Borland Delphi 7, Database Desktop
2. Jika sudah berada pada didalam Delphi, pilih menu Tools | Database Desktop



Database Desktop

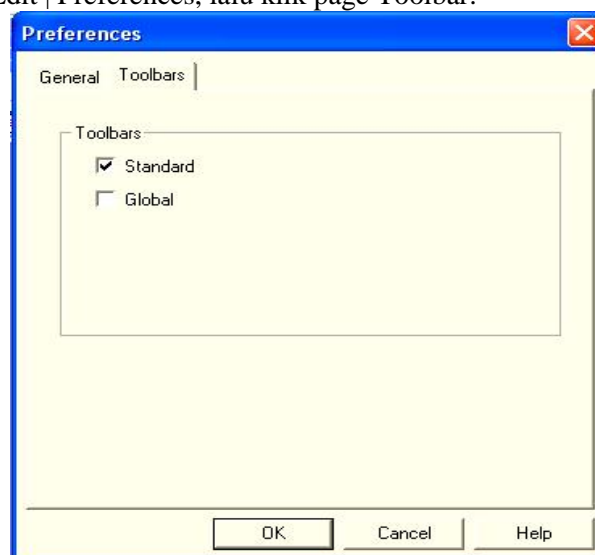
II. Fasilitas – fasilitas

Database desktop memberikan kemudahan dalam pemakaian, di antaranya :

1. Dapat menangani nama file panjang
2. Toolbar yang dapat dipindahkan serta toolbar tambahan, pemakai dapat menentukan apakah akan menampilkan beberapa toolbar atau menampilkan toolbar lain selain toolbar default.

Untuk mengubah tampilan toolbar :

- a. Pilih menu Edit | Preferences, lalu klik page Toolbar.



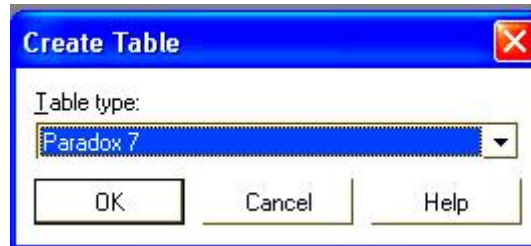
Kotak dialog Preference

3. Toolbar standard adalah toolbar default untuk setiap jendela anak. Toolbar Global adalah toolbar dengan tombol – tombol untuk menampilkan setiap object Database Desktop, yaitu Tabel, Query dan File SQL
4. Sistem menu yang lebih memudahkan pencarian. Menu Alias Manager dan Password berada di level teratas dari menu Tools.

III. Membuat File Database :

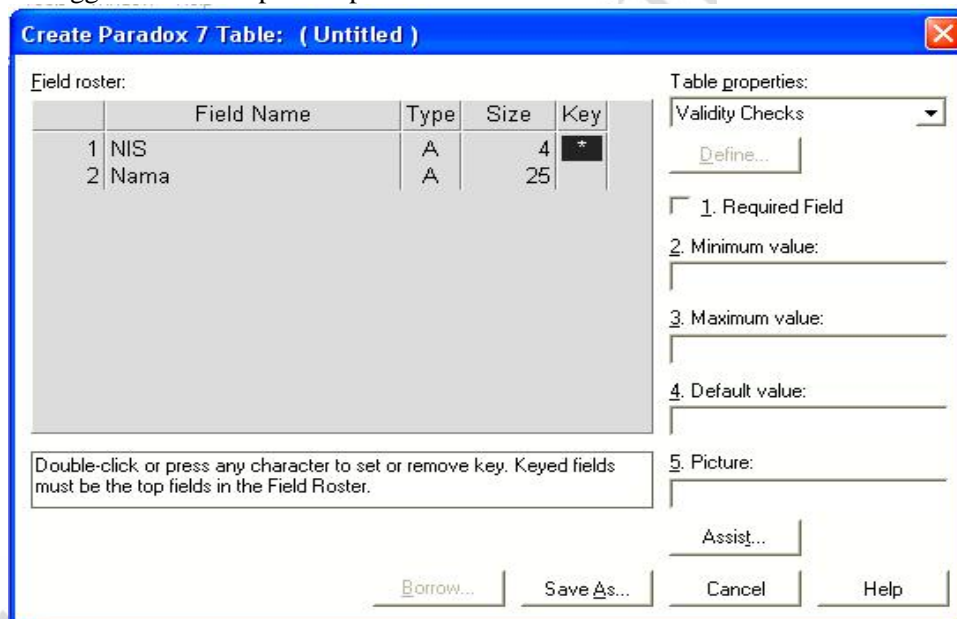
Dalam format Paradox, satu file database hanya berisi satu tabel database. Jadi agak berbeda dengan format MS – Access, yang memungkinkan membuat beberapa tabel dalam satu file database Paradox, lakukanlah langkah – langkah berikut :

1. Pada Database Desktop pilih File | New | Table sehingga muncul tampilan sebagai berikut :



Format Database

2. Pilih tipe atau format tabel yang diinginkan, misalnya paradox 7 lalu klik Ok, sehingga muncul tampilan seperti ini




Membuat Struktur Tabel Database

3. Pada Kolom field Name, tentukan nama field yang diinginkan. Nama field adalah nama pengenalan suatu kolom dalam tabel database. Aturan untuk pemberian nama field adalah seperti berikut ini :
 - a. Maksimum panjangnya 25 karakter
 - b. Tidak boleh diawali spasi, tetapi boleh mengandung spasi disarankan agar tidak ada spasi
 - c. Harus unik, yaitu tidak boleh ada yang sama dalam satu tabel
 - d. Jangan menggunakan tanda koma (,), tanda pipe (|) dan tanda seru (!).
 - e. Hindarkan penggunaan kata perintah SQL, misalnya SELECT, WHERE, COUNT, dll
4. Tentukan tipe field pada kolom type. Tipe field ini menentukan tipe data yang dapat ditampung dalam field. Untuk menampilkan daftar field yang tersedia, tekan tombol spasi atau klik kanan. Tipe field yang sering dipakai adalah sebagai berikut :

- a. A (Alpha), untuk menampung kumpulan karakter huruf, angka dan karakter ASCII yang dapat tercetak. Lebar field tipe ini antara 1 – 255 byte
 - b. N (Number), untuk menampung data angka yang dapat dihitung. Jangkauan yang dapat disimpan adalah dari –10307 samapai 10308 dengan 15 angka digit signifikan.
 - c. \$ (Money), sama dengan number tetapi default – nya data ditampilkan dengan desimal dan pemisah ribuan (sama dengan currency). Karakter pemisah desima dan pemisah ribuan tergantung dari Regional Setting dari sistem operasi Windows yang terdapat pada control panel. Tipe field ini sangat cocok untuk angka yang menunjukkan nilai uang.
 - d. S (Short), untuk menampung bilangan bulat antara –32,767 samapai 32,767.
 - e. I (Long Integer), untuk menampung bilangan bulat dengan nilai antara – 2147483648 sampai 2147483647
 - f. D (date), untuk menampung data tanggal sampai dengan 31 Desember 9999
 - g. T (Time), untuk menampung data waktu dalam 24 jam sampai hitungan mili detik
 - h. M (Memo), untuk menampung data memo. Data memo biasanya dipakai untuk menyimpan data seperti tipe Alpha, tetapi isinya dapat sangat besar dan dapat terdiri atas beberapa baris. Disisi lain, biasanya dipakai jika tidak semua record membutuhkannya, misalnya catatan prestasi karyawan, catatan kriminal, dll. Data memo disimpan dalam file terpisah (berakhiran *.MB).
 - i. F (Formatted Memo), untuk menampung data memo yuang dapat mempunyai format, misalnya font tertentu, warna text dan lain – lain.
 - j. G (Graphic), untuk menampung data gambar. Perlu diperhatikan bahwa data gambar sebtulnya disimpan dalam file yang terpisah. Jadi yang tersimpan dalam tabel hanyalah informasi yang menghubungkan dengan data gambar tersebut.
 - k. L (Logical), untuk menampung data tipe boolean True dan False
5. Tentukan lebar field pada kolom size (jika diperlukan)
 6. Pada kolom Key, programmer dapat memberi tanda * dengan menekan sembarang tombol. Field yang diberi tanda key harus berupa urutan field dari yang paling atas. Field yang diberi tanda key akan dipakai sebagai kunci pengurutan (Index) primer.
 7. Lengkapi struktur tabel seperti yang diinginkan sesuai dengan kebutuhan program.
 8. Klik tombol Save As sampai muncul kotak dialog Save Table As. Pada isian Save in pilihlah folder yang akan dipakai untuk menyimpan data, misalnya C:\LatDelphi. Pada isian File Name isilah nama file databsnya, misalnya siswa
 9. Klik tombol Save
 10. Untuk keluar dari Database Desktop, pilih menu file lalu exit.

IV. Memodifikasi File Database

Cara memodifikasi database yang sudah ada dalam disk hampir sama dengan cara membuatnya, yaitu :

1. Panggil program Database Desktop dengan memilih menu Tools lalu Databse Desktop (melalui Delphi)
2. Pilih menu **File, Open, Table** lalu tekan nama file database yang akan dibuka dilanjutkan dengan mengklik tombol open
3. Pilih menu **Table** lalu Restructure atau cukup klik icon restructure  sampai muncul tampilan seperti waktu akan membuat struktur tabel
4. Jika ingin menambah field, programmer tinggal menambahkannya
5. Jika ingin mengubah urutan field, klik nomor field yang akan diubah urutannya lalu seret (drag) ke tempat yang diinginkan.,
6. Jika ingin menghapus field, klik pada nomor field yang akan dihapus lalu tekan tombol Ctrl + Delete

7. Klik tombol **Save** untuk menyimpan hasil modifikasi dengan nama yang sama atau klik tombol **Save as** jika ingin hasil modifikasi disimpan kedalam file database baru.

V. Menambahkan Index

Index data adalah file yang berisi urutan data pada suatu tabel database. Index data diperlukan terutama untuk pencarian data dengan cepat dan juga untuk pengelompokkan data, index data disimpan dalam file berakhiran ***.PX**

Dalam file database Paradox, ada dua macam index data yaitu Index **Primer** dan index **secunder**. Index primer sering disebut sebagai **key**. Key (kunci) adalah sebuah field atau beberapa field yang dipakai untuk membuat data dalam tabel teratur. Key dipakai sebagai referensi suatu record. Oleh karena itu, key mempunyai sifat – sifat sebagai berikut :

1. Mencegah duplikasi record pada key yang sama, karena key dipakai sebagai referensi suatu record
2. Jika ada perubahan data, index data key selalu ter-update.

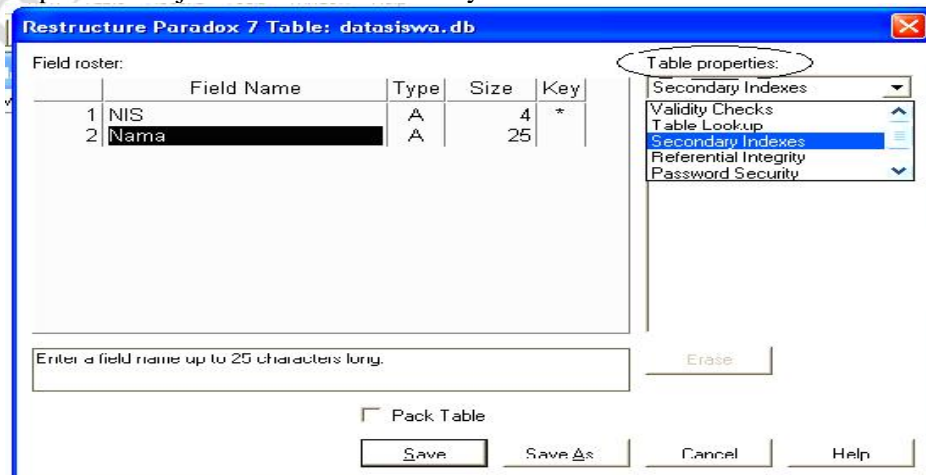
Perlu diingat bahwa field atau beberapa field yang dipakai sebagai key harus berada pada urutan paling atas (paling kiri jika dilihat dalam tabel) dari field lainnya.

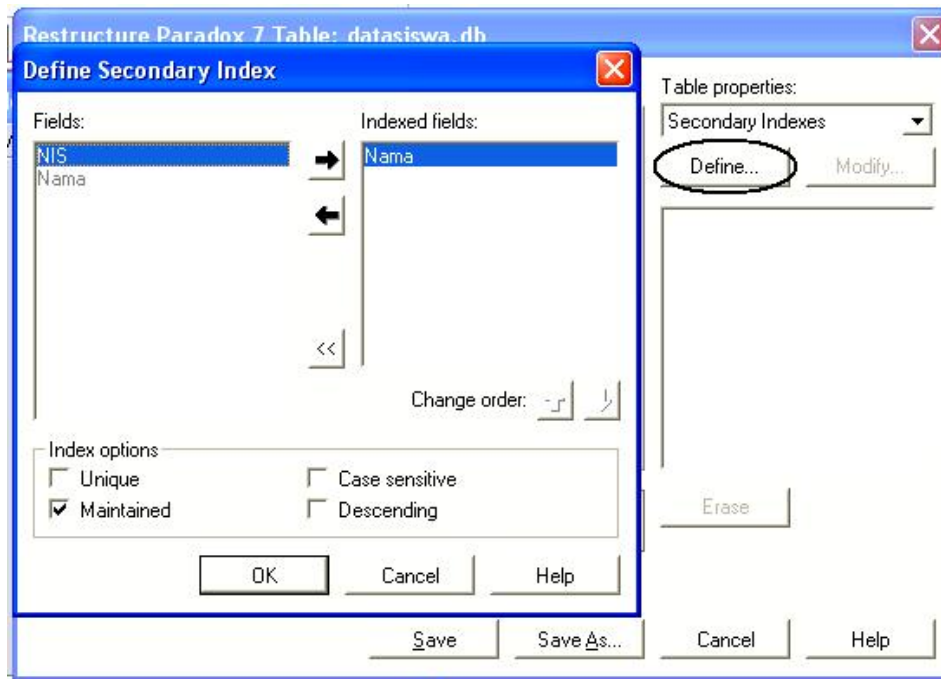
Berbeda dengan index primer, index sekunder dapat mempunyai kunci index dari field atau kumpulan field mana saja. Dalam satu tabel database, dapat dibuat beberapa index sekunder sekaligus. Index sekunder biasa dipakai untuk keperluan sebagai berikut :

1. Sebagai alternatif untuk mendapatkan urutan data dalam tabel
2. Sebagai kunci penghubung dalam relasi data atau tabel lain dengan tabel tersebut.
3. Mempercepat proses pencarian suatu data, yaitu sebagai kunci pencarian.

Untuk membuat index data, caranya hampir sama dengan memodifikasi data, yaitu sebagai berikut :

1. Panggil program Database Desktop dengan memilih menu **Tools** lalu **Database Desktop**.
2. Pilih menu **File | Open | Table** lalu tekan nama file database yang akan dibuka dilanjutkan dengan mengklik tombol **Open**
3. Pilih menu **Table** lalu **Restructure** atau cukup klik icon Restructure sampai muncul tampilan seperti waktu akan membuat struktur tabel.
4. Untuk membuat index primer (key), cukup klik ganda pada kolom key atau tekan sembarang tombol pada field yang diinginkan.
5. Untuk membuat index sekunder, pilih pilihan Secondary indexes pada ComboBox **Table Properties** yang terletak di sebelah kanan atas. Setelah itu , klik tombol **Define** sampai muncul jendela Define Secondary Index.





Mendefinisikan kunci Index Sekunder

6. Klik ganda pada nama field di kotak disebelah kiri yang akan dipakai sebagai kunci index sekunder, misalnya nama. Nama field yang telah dipilih sebagai kunci index akan dicopy ke kotak sebelah kanannya. Programmer dapat membuat satu kunci index yang terdiri atas beberapa field.
7. Tentukan pilihan pengurutan data berdasarkan kunci index yang bersangkutan. Pilihan pengurutan data yang tersedia seperti berikut ini :
 - a. **Unique**, dipakai jika diinginkan kunci bersifat unik. Pilihan ini menyebabkan jika pada tabel terdapat dua record yang sama, maka yang dicatat dalam index – nya hanya satu record, yaitu record yang pertama dicatat.
 - b. **Case Sensitive**, dipakai jika diinginkan index membedakan urutan huruf besar dan kecil, sehingga urutan menggunakan urutan nomor ASCII – nya (huruf besar lebih kecil atau lebih dahulu dibanding huruf kecil).
 - c. **Maintained**, dipakai jika diinginkan index otomatis selalu diupdate jika ada perubahan dalam tabel, misalnya ditambah, dihapus dan diedit. Pilihan ini juga mempercepat proses saat query. Pilihan maintained hanya bisa dipakai jika tabel juga mempunyai index primer (key). Untuk contoh sekarang, pilihan maintained sebaiknya dipakai. Database dengan index non – maintained hanya cocok dipakai pada database yang hanya untuk dibaca (read only).
 - d. **Descending**, jika diinginkan arah pengurutannya menurun.
8. Klik OK
9. Ketikkan nama index yang diinginkan. Perlu diperhatikan, nama index tidak boleh sama persis dengan nama field.



Menentukan nama index

10. Klik OK
11. Jika ingin membuat index sekunder lain pada tabel yang sama, ulang ke prosedur nomor 5
12. Klik save untuk menyimpan lagi tabel.

VI. Object Database Desktop

Komponen – komponen dalam database desktop yang menyimpan, menampilkan, dan mencari data dinamakan object – object database desktop. Object – object utama adalah tabel, query, dan file SQL

Database desktop memakai icon object untuk menyatakan object yang diminimisasi. Setiap object mempunyai ekstensi yang berbeda. Berikut ekstensi (akhiran) dari object :

Ektensi	Tipe Object
.CFG	File konfigurasi
.DB	Tabel Paradox
.DBF	Tabel dBase
.DBT	Memo dari tabel dBase
.FAM	Daftar file – file yang berhubungan
.INI	File konfigurasi
.LCK	File lock
.MB	Memo untuk tabel Paradox
.PX	Index utama dari tabel Paradox
.QBE	Query yang disimpan
.SQL	File SQL
.TV	Setting view untuk tabel paradox
.VAL	Pemeriksaan validitas dan referensial integrity untuk tabel paradox
.Xnn .Ynn	Index sekunder pada field tunggal untuk tabel paradox

VII. Working Directory dan Private Directory

Working Directory adalah direktori default yang dipakai database desktop untuk membuka dan menyimpan file. Working directory mengontrol file – file yang akan ditampilkan pada kotak dialog pada saat operasi membuka dan menyimpan file.

Pada saat menginstal Database Desktop pada drive local (bukan pada drive jaringan), database desktop membuat sebuah direktory yang bernama **Working** dibawah direktory sistem. Anda dapat mengubahnya, caranya yaitu :

1. Pilih menu **file | Working Direktory**. Ditampilkan kotak dialog Set Working Direktory
2. Pilih Browse, cari folder dimana database disimpan yang ingin dijadikan sebagai folder default tempat penyimpanan database



Set Working Direktory

Database Desktop memberi nama **Work** untuk working directory.

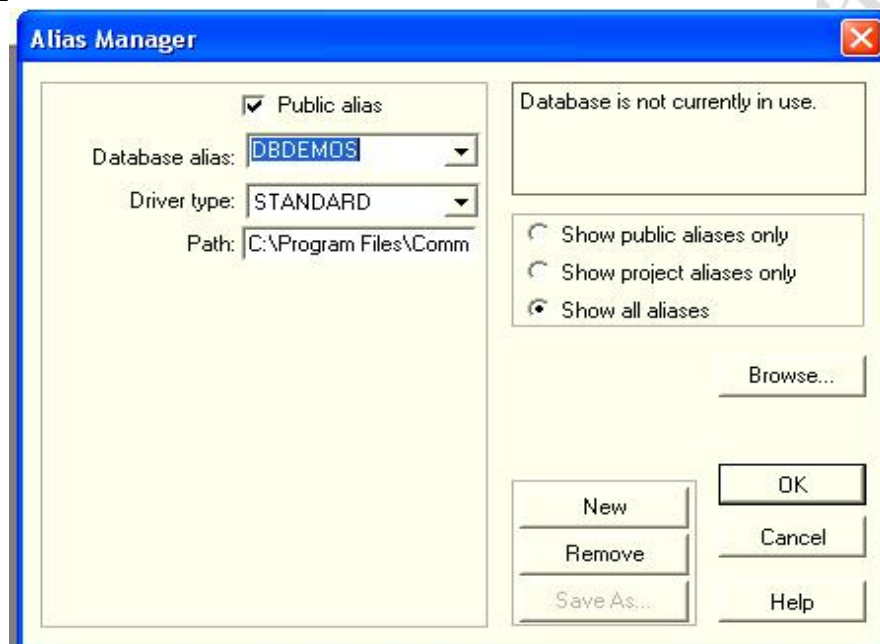
Jika programmer mengubah working directory, akan membawa efek – efek sebagai berikut :

1. Alias untuk proyek pada working directory menjadi aktif, dan alias pada direktori sebelumnya menjadi tidak aktif
2. jika programmer mempunyai tabel atau file SQL yang sedang dibuka, akan ditutup secara otomatis. Jika sudah terjadi perubahan, programmer akan diminta untuk menyimpannya.

VIII. Memakai Alias

Alias dipakai untuk memberi nama logis pada sebuah direktori. Alias sangat berguna karena anda tidak perlu menuliskan nama path yang panjang (c:\ alamat folder yang dituju) dan membuat file lebih mudah dipakai. Defaultnya direktori kerja (working directory) diberi alias WORK dan private directory sebagai alias PRIV

Untuk membuka kotak dialog alias manager, dari Database Desktop, pilih menu **Tools | Alias Manager**



Alias Manager

Option – option

Public Alias.

Pilih kotak cek ini untuk membuat sebuah alias sebagai public alias, artinya dapat dipakai oleh semua aplikasi yang memakai BDE (Borland Database Engine). Jika kotak cek ini tidak dipilih, berarti alias yang dibuat adalah project alias yang hanya dapat dipakai oleh aplikasi Database Desktop pada direktori kerja.

Database Alias.

Pilih sebuah alias dari daftar. Untuk membuat alias baru, pertama pilih **New** lalu ketik nama (alias) baru.

Driver Type.

Pilih driver yang anda inginkan. Daftar drop – down menampilkan semua driver yang terhubung. Jika Anda ingin membuat database dari tabel – tabel Paradox / atau dBase, pilih STANDARD.

Path.

Ketikkan direktori lengkap termasuk huruf drive-nya

Show Options

Show Public Alias Only.

Hanya menampilkan public alias

Show Project Aliases Only

Hanya menampilkan project alias

Show All Aliases

Menampilkan semua alias (public alias dan project alias)

Browse

Pilih option ini untuk mencari sebuah directory

New

Untuk membuka kotak kosong di mana anda dapat mengetikkan sebuah alias baru. Setelah memilih New, button menjadi Keep New.

Keep New

Membuat alias temporer yang ada hanya sampai anda keluar

Remove

Untuk membuang alias yang dipilih

Save As

Untuk menyimpan alias dan dapat dipakai lagi. Anda akan melihat kotak dialog Save File As. Defaultnya, Database Desktop menyimpan public alias di file IDAPI32.CFG dan project alias di PDOXWORK.CFG. anda akan ditanya apakah akan menulis ulang (overwrite) file .CFG yang ada. Dengan overwrite, database desktop menambah alias baru tanpa mengubah setting konfigurasi yang ada.

Untuk membuat alias baru, dari kotak dialog alias manager

1. Pilih New
2. Ketikkan nama alias pada kotak Database Alias. Misalnya dbSiswa
3. Pilih driver dari daftar Driver Type, dan pilih Standard
4. Ketikkan direktori lengkap termasuk huruf driver-nya di kotak path. Anda dapat memilih Browse untuk memilih direktory. Ketikkan direktori tempat anda menyimpan file DataSiswa.dbf (misalnya), c:\data siswa
5. Pilih public alias
6. Pilih Save As untuk menyimpan alias, defaultnya public alias disimpan di file IDAPI32.CFG. anda akan ditanya apakah akan menulis ulang file yang ada. Pilih Yes
7. Tutup Alias Manager dengan tombol close.

Pengecekan Validitas

Perhatikan gambar database desktop pada halaman 2, gambar tersebut merupakan jendela default jika anda masuk ke dalam menu **Table | Restructure**. Perhatikan field **Table Properties**. Pilihan default adalah **Validity Checks**, dipakai untuk menyatakan aturan yang diberlakukan pada sebuah field sehingga data yang dimasukkan sesuai dengan criteria tertentu. Paradox memberikan 5 jenis pengecekan validitas :

1. **required field**
menyatakan field yang dipilih pada **Field Roster** adalah field yang harus diisi (anda harus memasukkan sebuah nilai untuk field tersebut pada setiap record didalam table)
2. **Minimum**
Menyatakan nilai minimum untuk field yang dipilih pada **Field Roster**. Nilai yang anda isikan ke dalam field harus lebih besar atau sama dengan nilai minimum yang dinyatakan disini
3. **Maximum**
Menyatakan nilai maksimum untuk field yang dipilih pada **Field Roster**. Nilai yang anda isikan ke dalam field harus lebih kecil atau sama dengan nilai maksimum yang dinyatakan disini.

Catatan :

Pengecekan nilai maksimum dan nilai minimum hanya dapat dipakai pada field dengan tipe alpha, number, short, long integer, money, timestamp, time and date.

4. **Default**
Menyatakan nilai default untuk sebuah field. Jika sebuah field mempunyai pengecekan validasi ini, Database Desktop akan memasukkan nilai yang anda nyatakan disini, jika anda tidak memasukkan nilai lain pada saat mengubah field ini. Pada saat menambahkan record baru. Anda dapat mengganti nilai default dengan berpindah ke field tersebut dan mengetik nilai baru. Anda juga dapat menghapus nilai default dan field tetap kosong jika field tersebut tidak mempunyai pengecekan **Required Field**. Anda dapat memakai nilai default pada field dengan tipe alpha, number, short, long integer, money, logical dan tipe tanggal (date, time dan timestamp)
5. **Picture**
Dipakai untuk membatasi tipe – tipe informasi yang dapat dimasukkan ke dalam sebuah field. Jika sebuah field mempunyai pengecekan ini, anda menyatakan sebuah string sebagai template untuk nilai – nilai yang dapat dimasukkan ke dalam sebuah field. Contoh, jika anda memakai template (###) ### - ### (format untuk nomor telepon) dan anda memasukkan nilai 0225222131, database desktop akan memformat nilai tersebut menjadi (022)522-2131. contoh lain :

Picture	Deskripsi
#&#&#&	Kode pos Canada, contoh 1A2B3C
#####[-#####]	Kode pos amereika contoh 12345 atau 12345-6789
*!	Setiap karakter adalah huruf besar
{ Ya, Tidak }	Berisi “Ya” atau “Tidak”

Catatan :

Jika anda menyatakan pengecekan validitas picture pada sebuah table yang sudah berisi data, Database desktop tidak akan memformat data yang sudah ada atau memvaliditasi data yang sidah ada untuk mengecek apakah sudah sesuai

Tombol **Assist** dibawah **Picture** dipakai untuk membuka kotak dialog **Picture Assistance** dan anda dapat memilih atau mengubah sebuah picture yang sudah didefinisikan.



Jendela Picture Assistance

Option – option yang ada :

1. **Picture**
Dipakai untuk memasukkan picture. Anda dapat mengisinya dengan isi dari listbox **Sample Pictures**.
2. **Verify Syntax**
Dipakai untuk mengecek apakah database desktop dapat menerima picture yang anda berikan. Jika sintaks benar, ditampilkan sebuah pesan yang menyatakan picture tersebut benar.
3. **Restore Original**
Jika anda membuat kesalahan, pilih restore original untuk kembali ke semula
4. **Sample Value**
Ketikkan sebuah nilai lalu pilih Test Value untuk mengetahui apakah picture bekerja
5. **Test value**
Pilih tombol ini untuk mengetahui apakah database desktop dapat menerima nilai yang anda ketikkan ke dalam kotak Sample Value
6. **Sample Picture**
Database desktop sudah menyediakan beberapa picture standar. Klik tombol drop – down untuk menampilkan daftar picture yang sudah ada. Jika anda memilih salah satu picture, anda akan melihat penjelasan pada area pesan.
7. **Add to List**
Dipakai untuk membuka kotak dialog **Save Picture** dan anda dapat mendeskripsikan picture anda dan menambahkannya ke dalam daftar **sample picture**. Deskripsi yang anda masukkan akan ditampilkan di area pesan pada saat anda memilih picture tersebut dari daftar Sample Pictures
8. **Delete From List,**
Dipakai untuk menghapus sebuah picture dari daftar Sample Pictures
9. **Use**
Untuk mengkopi sebuah picture dari daftar sample picture ke dalam kotak teks picture.

Membuang Pengecekan Validasi

Setelah menyatakan pengecekan validitas, suatu saat mungkin anda ingin membuangnya. Caranya :

1. Pilih salah satu field dari daftar field Roster
2. Pilih Validity Checks dari daftar Table Properties
3. Pada Panel dibawahnya, buang pengecekan validitas yang anda inginkan.

Referential Integrity

Referential integrity berarti sebuah field atau kumpulan field pada sebuah tabel (tabel anak) harus mengacu kunci dari tabel lain (table bapak). Hanya nilai – nilai key dari tabel bapak yang dapat diisikan pada field tertentu dari tabel anak.

Dengan referential integrity, database desktop mengecek validitas sebuah nilai sebelum memasukkannya ke dalam tabel.

Database desktop memudahkan anda menyatakan referential integrity. Anda tidak dapat menyatakan referential integrity di antara file **.DBF**, tabel – tabel paradox 3.5 atau tabel yang tidak mempunyai kunci. Anda dapat memakai file – file **.DB** dan juga beberapa tabel – tabel SQL Server untuk menyatakan referential integrity.

Pada saat membuat atau mengubah hubungan referential integrity, Database Desktop membuat sebuah index pada field – field referential integrity jika index belum ada. Index tersebut diberi nama dengan nama field (jika field tunggal) atau nama dari referential integrity (jika field majemuk). Index akan muncul pada daftar **Secondary Indexes** dari daftar **Table Properties** . Jika anda menghapus referential integrity, Database Desktop tidak secara otomatis menghapus index tersebut. Anda harus menghapusnya secara manual.

Beberapa aturan referential integrity :

1. Anda hanya dapat membuat referential integrity di antara field – field yang sama yang berisi nilai – nilai yang sesuai. Nama field tidak harus asalkan tipe dan ukurannya sama.
2. Anda dapat membangun referential integrity antar tabel – tabel pada directori yang sama.
3. Tabel bapak dari referential integrity harus mempunyai kunci
4. Jika anda mendefinisikan referential integrity pada sebuah tabel yang sudah berisi data, beberapa nilai yang sudah ada mungkin tidak sesuai dengan field kunci dari tabel bapak. Jika hal ini Database desktop akan menempatkan record – record tersebut pada tabel temporary KeyViol pada directory private anda.

Contoh :

1. Buatlah sebuah tabel dengan struktur sebagai berikut dan beri nama **kyw.DB**:

No	Field Name	Type	Size	Key
1	NIP	A	4	*
2	Nama-Kyw	A	25	
3	Kode_Bag	A	1	
4	Kode_Jbt	A	1	
5	Tgl_masuk	D		
6	Gaji_Pokok	N		
7	Menikah	A	5	

2. Isikan data dari tabel yang anda buat minimal 5 record dengan isian pada kode_bag : A, B, C, D, secara acak, dan data yang lainnya terserah anda, sesuai dengan type field yang sudah ada pada tabel diatas

Table : kyw.db							
kyw	Nip	Nama	Kode_bag	Kode_jbt	TglMasuk	GajiPokok	Menikah
1	0000	Setiawan	A	1	09/02/2001	1.000.000,00	TIDAK
2	0001	RAHMAN	B	1	02/10/1999	1.400.000,00	YA
3	0002	MARWOTO	C	3	28/12/2001	1.000.000,00	TIDAK
4	0003	PARTO	D	2	13/02/2000	1.000.000,00	TIDAK



- Buatlah sebuah tabel baru dengan nama **Bagian.DB** yang disimpan di directory yang sama dengan **kyw.db**, struktur tabel dan isi tabel baru adalah sebagai berikut :

No	Field Name	Type	Size	Key
1	Kode_bag	A	1	*
2	Nama_bag	A	20	

- Isikan datanya sebanyak 3 buah yaitu Kode_bag A, B, C, dengan nama bagian terserah anda

Selanjutnya kita akan mendefinisikan referential integrity L

- Anda harus menjadikan directory tempat **Kyw.db** dan **Bagian.db** sebagai working directory. Pilih menu **File | Working Directory**. Pilih directory tersebut.
- Buka tabel **Karyawan.Db**
- Pilih menu **Table | Restructure** lalu dari daftar **Table Properties** pilih **Referential Integrity** dan klik tombol **Define**. Ditampilkan dialog **referential integrity**

- Dari daftar fields pilih Kode_bag lalu klik tombol  atau tekan Alt + A. **Kode_bag** akan muncul di **child fields**
- dari daftar tables klik **Bagian.Db** lalu klik tombol  . Field **Kode_bag** akan muncul di **Parent Fields**.
- Klik Ok. Simpan referential integrity tersebut dengan nama **Bag-Kyw**.
- Klik OK dan anda akan kembali kestruktur tabel, sekali lagi tekan Save. Ditampilkan **Keyviol.DB** karena tabel berisi sebuah data dengan kode_bag = D

Database desktop memberikan dua aturan pengubahan (Update Rule) untuk tabel – tabel yang memakai referential integrity. Update Rule menyatakan apa yang akan terjadi jika seorang user mencoba mengubah atau menghapus data pada tabel bapak yang mempunyai record – record pada tabel akan berhubungan dengannya.

Prohibit

Menyatakan anda tidak dapat mengubah atau menghapus sebuah data kunci pada tabel master (bapak) jika ada record – record pada tabel anak (transaksi) yang nilainya berhubungan. Misalnya, jia ada Kode_bag = A pada tabel karyawan.db, anda tidak dapat mengubah atau menghapus record pada **Bagian.db** dengan kode_bag = A.

Cascade

Cascade menyatakan bahwa perubahan pada tabel master secara otomatis akan dibuat juga pada tabel transaksi (anak). Jika anda menghapus sebuah nilai kunci pada tabel master, record – record pada tabel anak yang bergantung padanya juga akan dihapus. Cascade adalah pilihan default.

Kotak dialog referential integrity juga memberikan pilihan **Strict Referential Integrity**. Pilihan ini dipakai untuk memproteksi data sehingga tidak rusak oleh Database Desktop versi sebelumnya. **Strict Referential Integrity** menyatakan Paradox untuk Dos tidak dapat mengakses tabel yang mempunyai referential integrity.

APLIKASI DATABASE

Pendahuluan

Dengan aplikasi database, anda dapat berinteraksi dengan informasi yang tersimpan didalam database. Database menyediakan struktur informasi yang dapat dibagi ke beberapa aplikasi. Delphi memakai database relasional, artinya informasi disimpan di dalam tabel dalam bentuk baris dan kolom. Pada saat merancang aplikasi database, anda harus mengerti tentang struktur data tersebut, lalu anda dapat merancang antarmuka (user interface) untuk menampilkan atau mengubah data.

Memakai Database

Delphi mempunyai banyak komponen untuk mengakses databse. Komponen – komponen tersebut dikelompokkan ke dalam page – page pada Component Palette, menurut mekanisme pengaksesannya. Delphi 7 mempunyai beberapa page baru yaitu BDE, dbExpress, Data Snap.

Page BDE, berisi komponen – komponen yang memakai Borland Database Engine (BDE). BDE berisi sejumlah fungsi – fungsi API untuk berinteraksi dengan database. Page ini berisi komponen – komponen yang pada versi sebelumnya berada pada page **Data Access**, kecuali komponen **Data Source**.

Page ADO, berisi komponen – komponen yang memakai ActiveX Data Objects (ADO) untuk mengakses database melalui OLEDB. ADO adalah standard Microsoft. Ada banyak driver untuk berhubungan dengan server – server database yang berbeda. Dengan komponen – komponen ADO, anda dapat mengintegrasikan aplikasi anda dengan lingkungan ADO

Page dbExpress, berisi komponen – komponen untuk mengembangkan aplikasi pada platform lain, misalnya berhubungan dengan Linux. Anda dapat mengakses database dengan menggunakan dataset *unidirectional* artinya pengaksesan yang cepat dengan overhead minimal. Dataset unidirectional tidak memerlukan buffer data didalam memori sehingga lebih cepat dan memerlukan resource lebih sedikit

Page interbase, berisi komponen – komponen untuk mengakses database secara langsung.

Page DataAccess berisi komponen – komponen untuk berhubungan dengan berbagai mekanisme akses.

PENCARIAN DATA

Pencarian data merupakan operasi yang sangat penting dalam manajemen database. Untuk melakukan suatu operasi terhadap database, misalnya mengedit data, mencetak data dan lain – lain, biasanya harus diawali dengan pencarian data, karena penunjuk record harus berada pada record yang akan diproses.

Pada dasarnya pencarian data ada dua macam, yaitu pencarian data tanpa index (urutan data) dan pencarian data menggunakan index. Pencarian data pada data tanpa index harus dilakukan secara berurutan (sekuensial), sedangkan pencarian data pada data yang mempunyai index dapat dilakukan dengan tidak berurutan.

Pencarian data berurutan dilakukan dengan cara membandingkan satu record pada posisi tertentu dengan suatu kriteria. Jika record tidak sesuai dengan kriteria, pencarian, dilanjutkan ke record dibawahnya secara berurutan satu per satu. Jika ditemukan record yang memenuhi syarat, pencarian dapat dihentikan. Keuntungan menggunakan pencarian berurutan (sekuensial) adalah tidak diperlukan index data sehingga ukuran database tidak membesar, selain itu dapat menggunakan kriteria yang sangat bervariasi dan tidak terbatas, sedangkan kerugiannya adalah waktu proses pencarian bisa lama jika datanya besar.

Sebagai ilustrasi, waktu yang diperlukan untuk mencari data dengan pencarian berurutan adalah misalnya kita mempunyai satu juta record dan data yang dicari terletak di record yang ke 50, maka program akan memeriksa data hanya sebanyak 50 record sehingga proses pencarian berlangsung cepat. Namun, jika record yang dicari terletak pada record ke 750.000, program akan memeriksa record satu per satu sebanyak 750 000 kali, yang akan memakan waktu cukup lama.

Pencarian pada database dengan index data, dilakukan dengan membandingkan suatu kriteria dengan data tengah suatu set data. Jika tidak sesuai, akan dilanjutkan dengan membandingkan lagi kriteria dengan data tengah dari setengah set data yang mungkin, demikian seterusnya sampai record ditemukan atau set data tidak dapat dibagi lai. Waktu yang diperlukan untuk mencari data akan sangat pendek atau cepat, walaupun datanya sangat besar.

Sebagai ilustrasi, jika data terurut dengan index berjumlah satu juta record dan posisi record yang dicari pada posisi ke 750 000 maka pertama kali program akan membandingkan kriteria yang dicari dengan record yang ditengah, yaitu posisi ke 500 000. jika tidak sama dan kriteria lebih kecil dari data dalam posisi tersebut, program akan membandingkan kriteria dengan data tengah dari data ke 500 001 sampai ke satu juta, yaitu pada posisi ke 750 000 dan ternyata sama.

Pencarian Data Berurutan

Untuk melakukan pencarian data secara berurutan (sekuensial) dapat digunakan fungsi locate, object Tquery atau pernyataan SELECT dalam SQL. Akan tetapi, pada bagian ini kita tidak akan mempelajari fasilitas tersebut, karena beberapa keterbatasannya. Kita hanya akan mencoba membuat cara pencarian dengan program buatan sendiri, sehingga lebih luwes (fleksibel).

Langkah – langkah secara umum untuk melakukan pencarian data secara berurutan adalah sebagai berikut :

1. Bawa penunjuk record (pointer) ke awal file jika perlu
2. Jika data belum akhir file, bandingkan record sekarang dengan kriteria pencarian
3. Jika record sesuai dengan kriteria, pencarian dapat dihentikan.
4. Jika record tidak sesuai dengan kriteria, majukan pointer satu record ke record berikutnya, setelah itu ulang ke prosedur nomor 2.

Contoh, kita akan membuat program untuk mencari nama barang, dengan sifat pencarian akan menganggap huruf besar sama dengan huruf kecil dan teks yang dicari tidak harus nama depan. Untuk itu, lakukanlah langkah – langkah berikut ini :

1. Buka database desktop, buat sebuah tabel baru yang terdiri atas

Field Name	Type	Size	Key
Kode	A	6	*
Nama	A	30	
Satuan	A	6	
Hbeli	\$		
Hjual	\$		

Simpan dengan nama **tBarang** kedalam directory anda, dan buat alias baru dengan nama Barang.

2. Buka sebuah project baru. Setelah itu simpan unitnya dengan nama uCariNamaBarang, dan Projectnya dengan nama Pbarang pada folder anda.
3. Tambahkan object berikut pada form

Object	Properties	Setting
Tabel1	Active	True
	Name	TBarang
DataSource1	DataSet	Tbarang
	Name	DsBarang
DBGrid	DataSource	DsBarang
	Name	DbBarang
DbNavigator	DataSource	DsBarang
	Name	DbBarang
Label1	Caption	Nama
Edit1	Name	EdNama
	Text	
Button1	Caption	Cari
	Name	BtCari
Button2	Caption	Selesai
	Name	btSelesai

4. tekan tombol F12 atau klik tombol **Toggle Form / unit** sehingga muncul jendela editor kode program. Kemudian tambahkan pernyataan berikut ini pada deklarasi variabel global (diatas pernyataan implementation)

```
var
    form1 : TForm1;
    ada   : integer ;
```

5. tekan tombol F12 sehingga muncul kembali tampilan form.
6. Klik ganda pada object edCari, kemudian ketikkan kode programnya seperti berikut ini :

```
procedure TForm1.edCariChange(Sender: TObject);
begin
    // var ada bersifat global
    ada:=0;
end;
```

Baris diatas menunjukkan bahwa jika ada perubahan pada object edCari, program mengisi variable ada dengan angka nol, yang nantinya dianggap program sebagai tanda pencarian dimulai dari awal.

7. Tekan tombol F12 sehingga muncul kembali tampilan form
8. Klik ganda pada object btCari lalu ketikkan kode programnya sebagai berikut :

```
procedure TForm1.btCariClick(Sender: TObject);
begin
    //variable ada bersifat global
    if ada = 0 then
        // mulai pencarian baru
        // bawa ke record paling atas
        Table1.First
    else
        //pernah mencari dan ketemu
        //ke satu record berikutnya
        Table1.Next;

    //ulang sampai akhir file
    while not Table1.Eof do
        begin
            //cari posisi string
            ada:=pos(UpperCase(edCari.Text),
                UpperCase(Table1.fieldbyname('nama').AsString));
            if ada > 0 then
                //jika ketemu
                break;
                Table1.Next;
            end;
            //jika tidak ketemu
            if ada=0 then
                begin
                    beep;
                    Table1.First;
                end
            end;
        end;
```

9. tekan tombol F12 sehingga muncul kembali tampilan form
10. Klik ganda pada btSelesai lalu ketikkan kode programnya sebagai berikut :

```
procedure TForm1.btSelesaiClick(Sender: TObject);
begin
    Application.Terminate;

end;
```

11. Simpan unit dan projectnya dengan memilih file | save all

Pada program diatas, digunakan property **Eof** (end of file). Property ini dipakai untuk mengetahui apakah penunjuk record sudah menunjuk akhir file. Jika penunjuk record berada di record terakhir, property **Eof** masih bernilai False, tetapi kalau kemudian dicoba menggerakkan penunjuk record ke bawah, **Eof** akan bernilai True. Jika dataset kosong (tidak ada record), **Eof** juga bernilai True.

Kebalikan **Eof** adalah **Bof** (beginning of file). Property ini dipakai untuk mengetahui apakah penunjuk record berada di record pertama, property **Bof** masih bernilai False, tetapi kalau kemudian dicoba menggerakkan penunjuk record ke atas, **Bof** akan bernilai True.

Pada program diatas, kita juga sudah menggunakan beberapa fungsi dan procedure untuk menggerakkan penunjuk record. Sekarang kita akan melihat fungsi dan procedure yang dapat dipakai untuk menggerakkan penunjuk record tersebut secara detail

First

Dipakai untuk menggerakkan penunjuk record (record pointer) ke posisi paling atas dalam suatu dataset. Selain itu, pemanggilan procedure ini akan mengakibatkan beberapa hal, antara lain :

- Mengaktifkan record paling atas, sehingga semua perubahan terhadap record aktif akan tercatat pada record tersebut
- Property Bof menjadi true, property Eof menjadi False
- Menyiarkan adanya perubahan record, sehingga kontrol data, fungsi dan procedure lain yang berhubungan dengan dataset dapat meng-update.

Last

Dipakai untuk menggerakkan penunjuk record ke posisi paling bawah dalam suatu dataset. Selain itu, pemanggilan procedure ini akan mengakibatkan beberapa hal, antara lain :

- Mengaktifkan record paling bawah, sehingga semua perubahan terhadap record aktif akan tercatat pada record tersebut.
- Property Eof menjadi true, property Bof menjadi false
- Menyiarkan adanya perubahan record, sehingga kontrol data, fungsi dan procedure lain yang berhubungan dengan dataset dapat meng-update.

Next

Dipakai untuk menggerakkan penunjuk record satu record di bawah (setelah) posisi record sekarang. Pemanggilan procedure ini akan mengakibatkan beberapa hal, antara lain :

- Property Bof dan Eof menjadi False
- Mengaktifkan record di bawah record sekarang, sehingga semua perubahan terhadap record aktif akan tercatat pada record tersebut.
- Property Eof menjadi True jika pada saat memanggil procedure ini penunjuk record sudah berada pada record terakhir.
- Menyiarkan adanya perubahan record, sehingga kontrol data, fungsi dan procedure lain yang berhubungan dengan dataset dapat meng – update

Prior

Dipakai untuk menggerakkan penunjuk record satu record diatas (sebelum) posisi record sekarang. Pemanggilan procedure ini akan mengakibatkan beberapa hal, antara lain :

- Property Bof dan Eof menjadi False
- Mengaktifkan record diatas record sekarang, sehingga semua perubahan terhadap record aktif akan tercatat pada record tersebut
- Property Bof menjadi True jika pada saat memanggil procedure ini, penunjuk record sudah berada pada record pertama.
- Menyiarkan adanya perubahan record, sehingga kontrol data, fungsi dan procedure lain yang berhubungan dengan dataset dapat meng – update

MoveBy(n)

MoveBy merupakan fungsi. Fungsi ini dipakai untuk menggerakkan penunjuk record ke record sebelumnya atau ke record sesudahnya sebanyak n record. Jika n positif, penunjuk record digreakkan ke arah record sesudahnya (ke bawah). Sedangkan jika n negatif, penunjuk record digerakkan ke arah record sebelumnya (ke atas).

Nilai kembalian (return value) dari fungsi ini adalah beberapa record sesungguhnya penunjuk record digerakkan. Biasanya nilai kembalian ini sama dengan n. Misalnya jika diberikan perintah **MoveBy(10)**, biasanya nilai kembalian fungsi juga 10, kecuali jika sebelum mencapai pergerakkan penunjuk record sebanyak 10 record, sudah mencapai Eof, nilai kembaliannya kurang dari 10.

Pemanggilan fungsi ini akan mengakibatkan beberapa hal :

- property Bof dan Eof menjadi False
- Mengakibatkan record tujuan menjadi aktif, sehingga semua perubahan terhadap record aktif akan tercatat pada record tersebut.
- Property Bof menjadi True jika pada saat memanggil procedure ini nilai n negatif dan melebihi jumlah record di atasnya (seperti menabrak batas dataset).
- Property Eof menjadi True jika pada saat memanggil procedure ini, nilai n positif dan melebihi jumlah record dibawahnya (seperti menabrak batas dataset)
- Menyiarkan adanya perubahan record, sehingga adanya perubahan record, sehingga kontrol data, fungsi dan procedure lain yang berhubungan dengan dataset dapat meng – update.

Pencarian Dengan Index

Delphi menyediakan fungsi dan procedure yang cukup lengkap untuk mencari data dengan index. Fungsi dan procedure tersebut adalah **SetKey**, **GotoKey**, **GotoNearest**, **EditKey**, **FindKey**, dan **FindNearest**. Procedure dan fungsi SetKey, GotoKey, GotoNearest dan EditKey adalah bawaan dari Borland Delphi versi sebelumnya, yang tidak akan kita bahas. Pada bagian ini kita hanya akan membahas fungsi FindKey dan FindNearest.

Fungsi FindKey dipakai untuk mencari data pada dataset yang ter-index, sesuai kriteria pencarian. Jika data ditemukan, penunjuk record akan berada pada record pertama yang memenuhi kriteria dan nilai kembalian FindKey berisi True. Jika data tidak ditemukan, penunjuk record tidak akan bergerak dan nilai kembalian FindKey berisi False.

Fungsi FindNearest hampir sama dengan FindKey. Bedanya hanyalah bahwa pada FindNearest, jika data tidak ditemukan penunjuk record akan dibawa ke record pertama yang nilainya lebih besar dari kriteria pencarian. Selain itu, FindNearest hanya bisa dipakai pada pencarian data tipe string.

Secara umum langkah – langkah untuk melakukan pencarian data dengan index adalah sebagai berikut :

1. Tentukan index yang aktif dengan menggunakan property IndexName. Jika yang digunakan adalah index primer (key), property IndexName tidak perlu diset (diisi).
2. Buka Tabel
3. Gunakan fungsi FindKey atau FindNearest untuk mencari data.

Pencarian dengan FindKey

Supaya dapat lebih memahami, sekarang kita akan mencoba dengan sebuah contoh program, Untuk itu lakukan langkah – langkah berikut ini :

1. Buat sebuah Form Baru, simpan unitnya dengan Nama uCariBarang1
2. Tambahkan object berikut pada form

Object	Properties	Setting
Tabel1	Active	True
	Name	Tbarang

DataSource1	DataSet	Tbarang
	Name	DsBarang
DBGrid	DataSource	DsBarang
	Name	DbBarang
DbNavigator	DataSource	DsBarang
	Name	DbBarang
Label1	Caption	Nama
Edit1	Name	EdNama
	Text	
Button1	Caption	Cari
	Name	BtCari
Button2	Caption	Selesai
	Name	btSelesai

3. Klik ganda pada daerah kosong dalam object Form, kemudian ketikkan kode programnya sebagai berikut :

```
procedure TForm2.FormCreate(Sender: TObject);
begin
    btcari.Default:=True;

end;
```

Baris program diatas menunjukkan bahwa pada saat program dijalankan, Object btCari menjadi Default, sehingga jika ditekan tombol Enter akan mengakibatkan hal yang sama jika Object ditekan enter

4. Tekan Tombol F12 sehingga muncul kembali Form
5. Klik ganda pada object btCari, kemudian ketikkan kode programnya seperti berikut ini :

```
procedure TForm2.btcariClick(Sender: TObject);
begin
    // aktifkan index nama
    tbBarang.IndexName:='NamaBrg';
    //cari isi object edCari
    tbBarang.FindKey([edCari.Text]);
end;
```

Baris diatas menunjukkan bahwa jika object edCari diklik, program akan mengaktifkan index berdasar nama barang. Setelah itu program akan mencari isi object edCari pada index yang aktif. Perlu diingat bahwa kunci index adalah index primer (key), index tidak perlu diaktifkan, kecuali sudah ditentukan index aktif lainnya. Perhatikan pula bahwa pada saat pencarian, program tidak membedakan huruf besar dan huruf kecil. Selain itu, data yang dicari (isi object edCari) harus diketik lengkap.

6. Tekan tombol F12 sehingga muncul kembali tampilan form
7. Klik ganda pada object btSelesai lalu ketikkan kode programnya seperti berikut ini :

```
procedure TForm2.btSelesaiClick(Sender: TObject);
begin
    Application.Terminate;

end;
```

8. Simpan datanya lalu jalankan dengan menggunakan tombol F9 jangan lupa mensetting Project | Options form yang akan dijalankan

Pencarian dengan FindNearest

Oleh karena pada FindKey criteria pencarian harus ditulis lengkap, maka FindKey hanya cocok untuk melakukan pencarian pada data tipe string yang pendek atau data tipe selain string. Khusus untuk pencarian data tipe string, kita dapat menggunakan fungsi FindNearest. Keistimewaan FindNearest adalah criteria yang dicari tidak harus diketik lengkap, cukup depannya saja.

Sekarang, kita akan mencoba membuat sebuah program contoh penggunaan fungsi FindNearest. Adapun langkah – langkahnya adalah sebagai berikut :

1. Buat form baru pada project Pbarang yang telah dibuat sebelumnya
2. tambahkan object berikut :

Object	Properties	Setting
Tabel1	Active	True
	Name	Tbarang
DataSource1	DataSet	Tbarang
	Name	DsBarang
DBGrid	DataSource	DsBarang
	Name	DbBarang
DbNavigator	DataSource	DsBarang
	Name	DbBarang
Label1	Caption	Nama
Edit1	Name	EdNama
	Text	
Button1	Caption	Selesai
	Name	BtSelesai

3. Klik ganda pada object edCari, kemudian ketikkan kode programnya sebagai berikut :

```
procedure TForm3.edCariChange(Sender: TObject);
begin
  with tbBarang Do
  begin
    //aktifkan index nama
    IndexName:='NamaBrg';
    //cari isi object edCari
    FindNearest([edCari.text]);
  end;
end;
```

4. Perhatikan bahwa pada penulisan di atas kita dapat menggunakan pernyataan **With tbBarang Do** diikuti blok kode program, sehingga semua penulisan **tbBarang** dalam blok kode program setelah pernyataan tersebut tidak diperlukan lagi.
5. Simpan dulu unitnya dalam project yang sama dengan nama uCariBarang2
6. Jalankan program setelah mensetting Project | Options penjalan mainprogram yang dijalankan pertama kali.

Memakai Table, DataSource, DBGrid, DBEdit, DBNavigator

Aplikasi database memungkinkan user berinteraksi dengan informasi yang disimpan dalam database. Untuk membuat aplikasi database sederhana, anda memerlukan beberapa komponen, diantaranya :

1. **DataSet**
Untuk menyatakan kumpulan record dari database. Dataset dapat berasal dari sebuah tabel, beberapa kolom dan beberapa record dari sebuah tabel, atau informasi dari beberapa tabel.
2. **DataSource**
Bertindak sebagai perantara antar antarmuka user dengan dataset
3. **DataAware**
Adalah komponen – komponen antarmuka user, terdapat pada page **Data Controls**

Cara Mengakses Tabel

1. Buat sebuah aplikasi. Tambahkan komponen Tabel (dari tab BDE) ke dalam form. Komponen table adalah komponen nonvisual, artinya pada saat aplikasi dijalankan komponen ini tidak tampak. Beberapa karakteristik dari komponen nonvisual adalah anda boleh meletakkannya dimana saja dan komponen ini tidak dapat diubah ukurannya.
2. Isi pada **Database** dari komponen **Table** dengan yang diinginkan, atau sesuai dengan alias yang diinginkan, anda tinggal klik tombol panah pada property tersebut, atau menentukan path yang anda inginkan.
3. Ubah property **TableName** sesuai dengan nama table yang diinginkan
4. Ubah property **Active = true**. Hal ini menyatakan tabel akan langsung dibuka pada saat aplikasi dijalankan. Jika False, tabel tidak akan diakses.

Menambahkan DataSource

Delphi mengakses informasi dari sekumpulan informasi yang dinamakan **DataSet**. DataSource dipakai untuk mendeteksi perubahan – perubahan yang terjadi pada data dan mendeteksi keadaan dari DataSet.

Merancang Laporan Menggunakan Rave Reports

Ringkasan merancang laporan dengan Rave Reports :

1. Buat object Form yang akan dipakai untuk mencetak
2. Kaitkan form dengan sumber database, menggunakan object **Table** atau **Query** dari page **BDE** atau object lain yang setara. Tentukan isi property untuk mengambil datanya, misalnya property **TableName** atau property **SQL**. Ubah pula property **Active**-nya menjadi **True**.
3. Tambahkan object koneksi data dari page **Rave**, misalnya menggunakan **RvCustomConnection**, **RvDataConnection**, **RvTableConnection** atau **RvQueryConnection**. Object koneksi ini setara dengan object **DataSource** dalam page **DataAccess** yang biasa dipakai sebagai sumber data oleh object lain.
4. Ubah property object koneksi data yang dipakai, sehingga object koneksi data terkait dengan sumber datanya. Property yang diubah bergantung koneksi yang anda pakai, bisa property **Table**, property **FieldAliasList** atau property **DataSet**.
5. Buat file project laporan dengan memanggil program RaveReports lewat menu **Tools** lalu **Rave Designer** dalam menu Delphi.
 - 5.1. Buat file project baru (File lalu New) atau edit yang ada di layar atau buka file project lama
 - 5.2. Kaitkan file project dengan data yang aktif dalam Delphi dengan memilih menu **File** lalu **New Data Object**.
 - 5.2.1. Pilih **Direct Data View** lalu klik **Next**
 - 5.2.2. Pilih koneksi data yang aktif lalu klik **Finish**
 - 5.3. Gunakan menu **Tools** lalu **Report Wizard** untuk membuat layout laporan atau gunakan cara manual dengan memanfaatkan object – object dalam page – page yang ada.
 - 5.4. Simpan file project laporan
 - 5.5. Keluar dari program Rave Reports dan kembali ke program delphi
6. Dalam program delphi, klik ganda pada icon RvProject dalam page Rave sehingga ditambahkan object RvProject ke dalam Form.
7. Isi property **ProjectFile**-nya dengan nama file project laporan yang sudah Anda buat, lengkap dengan nama directorynya. Ubah pula property **Name**-nya jika perlu.
8. Tambahkan object **Button** ke dalam form. Setelah itu lakukan klik ganda pada object Button tersebut dan ketikkan kode programnya untuk menjalankan ProjectFile. Contoh isi program untuk memanggil adalah sebagai berikut :
Procedure TfrmCetakBarang.BtCetakClick (sender : Tobject) ;
Begin
 RvLat27.Execute ;
End;
9. Sampai disini, project dan unit Delphi dapat disimpan dan dijalankan.

Object yang sering dipakai untuk merancang Laporan

Object atau icon yang dipakai dalam merancang laporan berjumlah cukup banyak, sehingga kita tidak akan membahas semuanya. Berikut penjelasan ringkas dari beberapa icon atau object yang sering dipakai dalam merancang laporan dengan Rave Reports.

1. Always Show Band Headers
Dipakai untuk menampilkan header dari band atau pita yang dipakai untuk panduan peletakan object – object dalam band tersebut.
2. Text
Terdapat dalam page Standard, dipakai untuk membuat teks yang isinya bersifat tetap, seperti judul tabel, header dan lain – lain.

3. Region
Terdapat dalam page Report, dipakai untuk menentukan area laporan. Region ini perlu dibuat sebagai tempat anda meletakkan object lain.
4. Band
Terdapat dalam page Report, dipakai untuk membuat header atau footer atau teks lain yang dicetak sekali pada tiap halaman. Untuk mengatur cara pencetakan Band, dapat dilakukan lewat property BandStyle.
5. DataBand
Terdapat dalam Page Report, dipakai untuk menampung data dalam tiap record database. Akan dicetak berulang sesuai dengan record yang dicetak.
6. DataText
Terdapat dalam page Report, dipakai untuk menampilkan suatu field dalam database dan dapat dikombinasikan dengan teks lain. Cara menampilkannya dapat diatur lewat property DataField dan DataView.
7. DataMemo
Terdapat dalam page Report, dipakai untuk mencetak teks yang panjang dengan kemampuan melipat kata (word wrap).
8. CalcText
Terdapat dalam page report, dipakai untuk menampilkan operasi perhitungan sederhana seperti Sum, Average, Count, Min, dan Max dari suatu field.
9. Bitmap
Terdapat dalam page standard, dipakai untuk mencetak gambar bitmap (*.bmp) yang terdapat dalam disk. Anda dapat menghubungkan object ini dengan gambarnya lewat property Image.
10. MetaFile
Terdapat dalam tab standard, dipakai untuk mencetak gambar metafile (*.wmf) yang terdapat dalam disk. Anda dapat menghubungkan object ini dengan gambarnya lewat property Image.