BAB II

LANDASAN TEORI

2.1 Pengenalan Database

Definisi :

Database adalah sekumpulan dari suatu informasi yang diorganisasikan dengan cara logik dan saling berhubungan (Sumber : *Kadir, Abdul. Dasar Pemrograman Delphi 5.0.* hal :465). Database juga mempunyai definisi yaitu suatu bentuk pengorganisasian data pada media eksternal (*disk*) dengan tujuan mempermudah pengaksesan (penyimpanan ataupun pengambilan data), ada pula yang mengartikan bahwa database sebagai tempat untuk sekumpulan berkas data terkomputerisasi. Melalui sistem basis data seorang user dapat melakukan berbagai fungsi seperti : menambahkan data, menghapus data, dan mengambil data.

Salah satu model database yang banyak digunakan adalah database relasional. Pada jenis database ini, sebuah database tersusun atas sejumlah tabel, dan biasanya terhubung dengan satu atau beberapa kata kunci.

2.2 Database Relasional

Definisi :

Database Relasional adalah database yang terdiri atau tersusun dari sejumlah tabel (Sumber : Kadir, Abdul. *Dasar Pemrograman Delphi 5.0.* hal :465). Setiap tabel mengandung baris dan kolom, yang disebut *record* dan field. Setiap *record* yang dimasukkan akan mengisi sebuah baris dalam tabel. Setiap

item data dalam sebuah *record* disimpan dalam sebuah field kunci dalam tabel.

Sebagai contoh, jika dibuat database yang mengandung dua buah tabel, maka kedua tabel tersebut dapat dihubungkan dengan menggunakan sebuah field yang terdapat pada kedua tabel itu. Sebuah hubungan terjadi diantara *record-record* dalam dua buah tabel jika sebuah nilai field kunci dalam sebuah *record* tabel sama pada tabel yang lain dan saling dihubungkan. Dengan menggunakan kapasitas dari tabel yang berbeda, dapat dibuat tabel yang berisi kumpulan data istimewa. Fasilitas ini membuat tabel pada database dapat diubah-ubah sesuai keperluan.

2.3 Kunci dan Indeks Tabel

Beberapa tabel dapat dihubungkan dengan mendefinisikan sebuah hubungan dari field tabel yang satu dengan field tabel yang lain. Field-field ini harus mempunyai indeks. Sebuah indeks adalah sebuah *file* yang digunakan database untuk menjaga lokasi *record* dalam sebuah tabel. Indeks ini mempermudah database untuk :

- Memelihara order sebuah tabel yang diurutkan.
- Mengumpulkan order.

Tabel Paradox mengandung indeks untuk menspesifikasikan order *record* diakses. Pada tabel Paradox, indeks primer (*primary indexes*) kadang disebut sebagai kunci atau *keys*.

2.3.1. Kunci

Definisi :

Kunci (*key*) adalah sebuah field dalam sebuah database yang digunakan sebagai sarana pengurutan (indeks) data di dalam database. Kunci ini sering disebut juga sebagai kunci tunggal (Sumber: Kristanto, Harianto, Ir. *Konsep Dan Perancangan Data Base.* hal:19)

Sebuah tabel paradox dapat mempunyai indeks, tapi hanya satu yang dipilih sebagai indeks primer. Dalam tabel Paradox, indeks primer disebut kunci. Sebuah tabel yang mempunyai kunci disebut tabel berkunci (*keyed table*).

Ketika membuat sebuah kunci, Database Desktop memberikan aturan tentang data yang dapat dikandung dalam field berkunci. Setiap nilai field harus unik sehingga tidak ada duplikasi dan kerancuan dalam pengaksesan maupun dalam merelasikan data-data yang ada dalam tabel. Dengan demikian Database Desktop dapat mengurutkan *record* tabel berdasarkan nilai dalam field yang didefinisikan sebagai kunci tabel.

Jika didefinisikan sebuah kunci pada tabel yang sudah mengandung data, Database Desktop memindahkan *record* tabel ke dalam order pengurutan yang benar. Lokasi *record* secara fisik ditunjukkan oleh pengurutan nilai field kunci dalam order menaik (A ke Z dan 0 ke 9). Sehingga *record* baru yang ditambahkan akan diletakkan pada posisi yang sesuai atau benar pada tabel yang

diurutkan, jadi belum tentu jika *record* baru akan berada pada urutan paling bawah pada urutan tabelnya. Jika digunakan lebih dari satu field sebagai kunci, maka indeksnya disebut kunci gabungan (*Composite key*).

2.3.2. Kunci Gabungan (Composite Key)

Definisi :

Kunci Gabungan (*Composite Key*) adalah kunci dalam sebuah database yang digunakan sebagai sarana pengurutan (indeks) data di dalam database dan terdiri atas lebih dari satu field (kumpulan field). (Sumber: Kristanto, Harianto, Ir. *Konsep Dan Perancangan Data Base.* hal:21)

Database Desktop membiarkan pembuatan nilai yang sama (duplikasi) dalam field individual kunci gabungan, sejauh nilai tersebut tidak diduplikasikan pada semua field kunci. Karena field kunci harus mempunyai keunikan pada setiap *record*nya.

Ketika dibuat sebuah kunci gabungan, Database Desktop membuat sebuah indeks gabungan primer (*primary composite index*), yang mengelola *record* dengan field pertama dari kunci (tergantung dari struktur tabelnya), kemudian field berikutnya.

2.3.3. Indeks

Definisi :

Indeks adalah sarana pengurutan data dalam sebuah database atau dalam tabel (Sumber: Alam, Agus.*Belajar Sendiri Borland Delphi 6.0.* hal:149)

Ketika dibuat sebuah indeks, Database Desktop membuat satu atau lebih *file* yang memuat nilai field yang diindeks dan lokasinya. Database Desktop mengacu pada *file* indeks ketika mengalokasikan dan menampilkan *record* dalam sebuah tabel.

2.4 Link

Definisi :

Link adalah hubungan (*hubungan*) antara tabel dengan kata lain adalah hubungan fisis atau konseptual antar objek (Sumber: Kristanto, Harianto, Ir. *Konsep Dan Perancangan Data Base.* hal:56)

.Hubungan antar tabel dilakukan jika nilai dari field yang satu dengan field yang diindeks pada tabel lain sama, jika terdapat nilai yang sama maka Database Desktop akan secara otomatis menghubungkan data dari satu tabel ke tabel yang lain.

Satu atau lebih field yang digunakan untuk mendefinisikan link harus diindeks karena Database Desktop harus menyamakan (*mathcing*) nilai dari field tersebut. Apabila sebuah indeks digunakan, maka Database Desktop mempunyai nilai terpelihara yang mempunyai *file* terpelihara yang mempunyai daftar lokasi dari semua *record* dalam tabel. Sehingga Database Desktop dapat menemukan dan menghubungkan (*link*) *record* dengan cara cepat dan efisien.

2.5 Objek dalam Database Desktop

Definisi :

Pada Database Desktop, komponen database yang mengirimkan, menampilkan, me-re*trieve*, dan mempresentasikan data disebut **objek** (Sumber: Kristanto, Harianto, Ir. *Seri Pemrograman Database menggunakan Delphi.* hal:4)

Objek utama yang sering dikerjakan pada Database Desktop adalah tabel, queri, dan *file* SQL.

Database Desktop menggunakan objek untuk mengirimkan, menampilkan, dan mepresentasikan informasi.

Suatu objek mengandung dua komponen yaitu :

- File dalam sebuah *disk*
- Tabel, query dan *file* SQL.

Database Desktop menggunakan ikon objek untuk mempresentasikan objek ketika diminimize. Setiap objek mempunyai ekstensi *file* yang berbeda.

2.5.1. Tabel

Definisi :

Tabel adalah suatu sarana dalam database yang digunakan untuk menyimpan data (Sumber : Kadir, Abdul. *Dasar Pemrograman Delphi 5.0.* hal :471).

Tabel mempunyai baris dan kolom. Setiap baris memuat informasi tentang sebuah item penting seperti nama barang, asal barang, harga serta hal yang lainnya. Sehingga baris bisa disebut sebagai ***record***. Setiap kolom memuat satu kategori data yang

mendukung *record* dan sering disebut **field**. Untuk menyebutkan isi dari field yang ada maka diberikan atribut atau judul yang menjelaskan isi dari field yang ada.

2.5.2. Query

Definisi :

Sebuah query Database Desktop adalah sebuah pertanyaan tentang data (Sumber : Alam, Agus, M. *Belajar Sendiri Borland Delphi 6.0.* hal :225) pada tabel yang dapat digunakan untuk:

- Menemukan atau memilih data dari sebuah tabel
- Menggabungkan data lebih dari satu tabel
- Membuat perhitungan data pada tabel

Database Desktop memberikan cara sederhana untuk menanyakan tentang data tabel. Dalam jendela Database Desktop Query, tinggal dipilih tabel yang ingin ditanyakan. Kemudian diberikan contoh data yang diinginkan, dan Database Desktop menjawab berdasarkan contoh. Hal ini disebut dengan Query By Example (**QBE**).

Database Desktop menyediakan tampilan query yang digunakan untuk mendefinisikan dan menjalankan sebuah query. Di dalam query, dapat dideskripsikan sebuah fasilitas pengeditan data sehingga akan dihasilkan sebuah tampilan tabel hasil dari pendeskripsian data yang dituliskan dalam query tersebut. Tampilan query menyederhanakan cara untuk membatasi tampilan data.

2.5.3. Desain Database

Dalam perancangan suatu basis data ada beberapa hal yang perlu diperhatikan, yaitu hierarki data dan manajemen data. Pita magnetik merupakan media penyimpanan berurutan yang paling populer, selain itu piringan magnetik merupakan cara utama untuk direct access. Namun karena perkembangan teknologi yang ada, ditemukan teknologi direct access yang baru yaitu *compact disk*.

Sebelum adanya masa basis data suatu perusahaan mengalami kesulitan dalam manajemen data mereka karena cara pengaturan data di penyimpanan sekunder. Usaha mula-mula dalam mengatasi masalah tersebut meliputi penyortiran dan penggabungan *file*, pemrograman yang ekstensif untuk mencari dan mencocokkan catatan *file*, serta indeks *file* dan kaitan yang dibangun ke dalam catatan data. Konsep database dibangun di atas indeks dan kaitan untuk mencapai suatu hubungan logis antara beberapa *file*.

Perangkat lunak yang mengelola database disebut sistem manajemen database (*Database Management System*-DBMS). Semua DBMS memiliki pengolah bahasa deskripsi data (*Data Description Language Processor*) yang digunakan untuk menciptakan database, serta pengelola database yang menyediakan isi database bagi pemakai. Orang yang bertanggung jawab atas database dan DBMS adalah pengelola database (*Database Administrator*) atau disingkat DBA.

2.5.3.1. Hierarki Data

Penyusunan data secara tradisional biasanya diorganisasikan dalam suatu tingkatan yang terdiri dari elemen, *record*, dan *file*. Dibawah ini adalah pembagian tingkatan data mulai dari yang terkecil

- ✓ Elemen data

  Merupakan unit yang terkecil, tidak dapat dibagi lagi menjadi unit yang berarti. Misalnya dalam catatan gaji, elemen dapat berupa nama, nip, upah perjam, dan lainnya.

- ✓ *Record*

  Record terdiri dari semua elemen data yang berhubungan dengan suatu objek atau kegiatan tertentu. Misalnya *record* tentang gaji, pendidikan karyawan, dan lainnya.

- ✓ File

  Semua *record* disusun menjadi suatu *file*. *File* merupakan kumpulan *record* yang berhubungan dengan subjek tertentu. Misalnya *file* karyawan yang berisi tentang data-data karyawan yang bersangkutan.

Jadi tingkatan tertinggi dari pengorganisasian data adalah *file* dan yang terendah adalah elemen data.

2.5.3.2. Manajemen Data

Manajemen data merupakan bagian dari manajemen sumber daya informasi yang mencakup semua kegiatan yang memastikan bahwa sumber daya data menjadi sesuai kebutuhan dan aman dari

gangguan dan tersedia bagi pemakai. Hal ini diperlukan karena data adalah sumber daya yang perlu dikelola dan dipelihara.

Kegiatan manajemen data mencakup beberapa hal, yaitu:

- ✓ Pengumpulan data

  Data yang diperlukan dikumpulkan dan dicatat dalam dokumen sumber (source document) yang fungsinya sebagai input bagi sistem.

- ✓ Integritas dan pengujian

  Data yang ada diperiksa untuk meyakinkan konsistensi dan akurasinya berdasarkan peraturan dan kendala yang telah ditentukan sebelumnya.

- ✓ Penyimpanan

  Data tersebut disimpan pada suatu medium yang aman seperti pita magnetik atau piringan magnetik.

- ✓ Pemeliharaan

  Adanya penambahan data, pengubahan data dan penghapusan data yang ada agar sumber daya data tetap mengikuti perkembangan jaman

- ✓ Keamanan

  Data dijaga dari kehancuran, kerusakan, atau penyalahgunaan.

- ✓ Organisasi

  Penyusunan data sedemikian rupa untuk memenuhi kebutuhan informasi pemakai.

- ✓ Pengambilan

  Tersedianya data yang mengikuti perkembangan jaman bagi user.

Dibandingkan sistem yang berbasis kertas DBMS (Database Management System) memiliki 4 keunggulan, yaitu:

- ➢ Kepraktisan

  Untuk sistem yang berbasis kertas akan menggunakan kertas yang sangat banyak dan akan memakan tempat untuk menyimpan informasi, sedangkan DBMS menggunakan media penyimpanan sekunder yang berukuran kecil tetapi padat informasi.

- ➢ Kecepatan

  Mesin dapat mengambil atau mengubah data lebih cepat daripada manusia.

- ➢ Mengurangi kejemuan

  Orang akan menjadi bosan jika melakukan hal yang sama dan berulang-ulang secar manual.

- ➢ Kekinian

  Informasi yang terdapat pada DBMS akan sesuai dengan keadaan perusahaan.

Keuntungan lain dengan menggunakan DBMS adalah adanya mekanisme sekuritas terhadap basis data berdasarkan wewenang user.

Pada umumnya untuk proses desain database, terlepas dari masalah yang ditangani dibagi menjadi:

- Perancangan basis data secara konseptual, merupakan upaya membuat model tetapi masih berupa konsep.
- Perancangan basis data secara logis, merupakan tahapan untuk memetakan model konseptual ke model basis data yang akan dipakai.
- Perancangan basis data secara fisis, merupakan tahapan untuk menuangkan perancangan basis data secara logis menjadi basis data fisis yang tersimpan pada media eksternal.

Sebuah database relasional tunggal dapat berisi sejumlah tabel. Desain yang baik membuat database mudah dipakai dan menyediakan keluwesan untuk mendukung kebutuhan yang akan datang. Ada beberapa langkah dalam membuat desain database tersebut, yaitu:

- Menentukan materi utama dalam aplikasi, karena setiap aplikasi melibatkan sejumlah besar kandungan yang properti dan relasinya mendasari aplikasi.
- Membuat tabel untuk masing-masing materi.
- Memilih sebuah kunci untuk masing-masing materi, setelah tabel telah terdefinisi maka ditentukan satu atau lebih field yang akan secara unik mengenali setiap *record* dalam tabel.

- Menambahkan atribut materi ke masing-masing tabel materi utama, maka harus dipikirkan informasi yang dibutuhkan oleh aplikasi untuk mengetahui tentang masing-masing materi utama yang telah ditentukan.
- Membuat tabel tambahan untuk atribut yang berulang, karena mungkin dalam penambahan atribut materi akan ditemukan sejumlah atribut yang terjadi lebih dari sekali untuk masing-masing kunci utama.
- Memastikan setiap field adalah atribut yang benar untuk kunci utama, maka perlu dikaji ulang setiap field dalam setiap tabel dan memeriksa bahwa kunci primer tabel memberikan pencarian yang logis bagi field tersebut.
- Mengkaji hubungan diantara tabel yang ada, perlu dikaji ulang semua tabel dalam database yang telah direncanakan, menentukan semua yang telah memiliki relasi nyata, dan memastikan bahwa ada cara untuk menggabungkannya

2.5.4. Teknik Normalisasi

Proses normalisasi merupakan proses pengelompokan data elemen menjadi tabel yang menunjukkan *entity* dan relasinya. Pada proses normalisasi selalu diuji pada beberapa kondisi. Apakah ada kesulitan pada saat menambah atau *insert*, menghapus atau *delete,* mengubah atau *update*, membaca atau *retrieve* pada satu database. Bila ada kesulitan pada pengujian tersebut maka relasi tersebut dipecahkan

pada beberapa tabel lagi atau dengan kata lain perancangan belumlah mendapat *database* yang optimal.

Ada beberapa bentuk dari normalisasi, yaitu:

- **Bentuk tidak normal (*Unnormalized Form*)**

  Bentuk ini merupakan kumpulan data yang direkam, tidak ada keharusan mengikuti suatu format tertentu, data dapat terduplikasi atau tidak lengkap. Data dikumpulkan apa adanya.

- **Bentuk normal kesatu (1NF/*First Normal Form*)**

  Bentuk normal kesatu mempunyai ciri, yaitu setiap data dibentuk dalam *Flat file*, data dibentuk satu demi satu *record* dan nilai dari field berupa "*Atomic Value*". Tidak ada satu set atribut yang berulang atau bernilai ganda. Tiap field hanya satu pengertian, bukan merupakan kumpulan kata yang mempunyai arti mendua, hanya satu arti saja dan juga bukanlah pecahan kata sehingga artinya lain. Misalnya atom adalah zat terkecil yang masih memiliki sifat dari induknya jika dipecah lagi maka ia tidak memiliki sifat induknya lagi. Misalnya: KELAS(Kode_kelas, nama_kelas, instruktur), merupakan bentuk 1NF karena tidak ada yang berganda dan setiap atribut merupakan satu pengertian tunggal.

- **Bentuk normal kedua (2NF/*Second Normal Form*)**

  Bentuk normal kedua mempunyai syarat yaitu bentuk normal kesatu. Atribut bukan kunci haruslah bergantung secara fungsi pada kunci utama atau primary *key*. Sehingga untuk membentuk normal kedua haruslah sudah ditentukan kunci-kunci *field*. Kunci *field* harus unik dan dapat mewakili atribut lain yang menjadi anggotanya. Misalnya: contoh relasi SISWA. Pada bentuk 1NF adalah

  SISWA(No_siswa, nama, wali_studi, kode_kelas)

  Disini terlihat bahwa kunci utama adalah nomor_siswa. Nama_siswa dan_wali studi bergantung secara fungsi pada No_siswa, tetapi kode_kelas bukanlah fungsi dari SISWA maka File SISWA dipecah menjadi dua relasi yaitu

  Relasi SISWA(No_siswa, nama, wali_studi) dan

  Relasi AMBIL KELAS(No_siswa, Kode_kelas)

- **Bentuk normal ketiga (3NF/*Third Normal Form*)**

  Untuk menjadi bentuk normal ketiga maka relasi haruslah dalam bentuk normal kedua dan semua atribute bukanlah primer tidak punya hubungan yang transitif. Dengan kata lain, setiap atribute bukan kunci haruslah bergantung hanya pada primary key secara menyeluruh.

- **Boyce-Codd normal form (BCNF)**

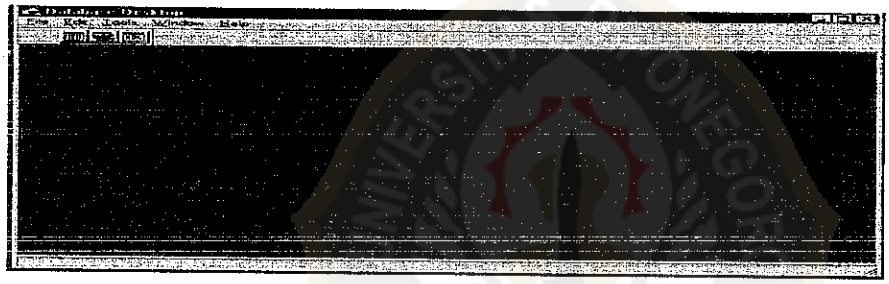
  BCNF mempunyai paksaan yang lebih kuat dari bentuk normal ketiga. Untuk menjadi BCNF, relasi harus dalam

bentuk normal kesatu dan setiap atribute harus bergantung fungsi pada atribute superkey.

2.6 Penggunaan Database Desktop

Untuk memanipulasi database secara interaktif (tanpa melalui form), Delphi menyediakan tool yang disebut Database Desktop. Melalui tool inilah database bisa dimanipulasi, sebelum memasuki ke tahap pemrograman.

Setelah mengaktifkan Database Desktop maka akan terlihat tampilan seperti berikut :



Gambar 1. Tampilan Database Desktop

2.6.1. Pengaturan Direktori Kerja Dan Alias

Definisi Direktori Kerja :

Sebuah direktori kerja Database Desktop adalah direktori yang digunakan untuk membuka dan menyimpan *file* (Sumber:*Pemrograman Database Menggunakan Delphi Jilid1*,hal:30)

Direktori kerja mengontrol setiap *file* yang ditampilkan dalam kotak dialog selama operasi Open dan Save. Direktori kerja

defaultnya adalah WORKING dibawah direktori sistem, sehingga dapat diubah direktori kerja sesuai dengan keinginan pemakai.

Definisi Direktori Privat :

Sebuah direktori yang digunakan menyimpan tabel temporer pada direktori yang tidak dapat digunakan bersama-sama dengan user lain (Sumber : Martina, Inge. *Seri Aplikasi Pemrograman Menggunakan Delphi.* hal :7)

Dalam lingkungan multi user, dibutuhkan tempat untuk meletakkan objek temporer yang berada dalam suatu tabel temporer pada direktori yang tidak dapat digunakan bersama-sama dengan user lain. Semua user Database Desktop membutuhkan direktori privat ketika menjalankan Database Desktop.

Direktori Privat defaultnya adalah PRIVATE, yang dibuat dibawah direktori Database Desktop utama pada hard *disk.* Direktori privat dapat pula diubah seperti yang diinginkan.

Definisi Alias :

Sebuah alias adalah nama yang dibuat sebagai shortcut ke sebuah direktori (Sumber: *Pemrograman Database Menggunakan Delphi Jilid1,*hal:30)

Untuk defaultnya, direktori kerja mempunyai alias: WORK: dan direktori privat : PRIV.

Keuntungan menggunakan Alias :

- Tidak memerlukan nama path yang panjang

- Dengan menggunakan alias, dapat dihubungkan ataupun tidak dihubungkan dari server database remote.
- Dapat merubah definisi sebuah alias kapan saja. Semua objek Database Desktop yang memakai alias secara otomatis akan memakai alias yang baru.

Ada dua macam alias yang perlu diperhatikan :

- Public aliases
- Project aliases

Public dan Project Alias

Public alias disimpan pada *file* konfigurasi BDE (Borland Database Engine). Public alias tersedia dari direktori kerja dan terlihat pada beberapa aplikasi yang menggunakan BDE. Sedangkan Project alias tersedia apabila digunakan Database Desktop, dan pada direktori kerja yang dibuat.

Jika direktori kerja dirubah, maka Database Desktop tidak akan me-load semua project alias yang terhubung pada direktori kerja lama. Hal ini terjadi karena Databae Desktop hanya me-load project alias yang spesifik dengan direktori kerja baru.

Jika sebuah project alias mempunyai nama sama dengan public alias, maka Database Desktop tidak akan me-load project alias.

Kotak dialog Alias Manager dapat digunakan untuk :

- Membuat alias baru
- Memodifikasi alias yang sedang digunakan

- Menghapus alias

2.6.2. Pembuatan Database Dengan Tabel Paradox

Sebagai tambahan, ketika dibuat tabel paradox, dapat dilakukan hal-hal seperti :

- Membuat kunci, atau indeks primer pada tabel
- Membuat indeks sekunder pada tabel
- Mendefinisikan validity checks untuk field individual
- Membangun tabel lookup untuk tabel lain
- Menentukan password untuk tabel atau field individual

Setelah dibuat sebuah tabel, dapat diurutkan *record*nya atau mencari data dengan perintah dan query.

Apabila ingin menambah atau menghapus field atau membuat perubahan struktur tabel, dapat dilakukan dengan menstruktur ulang tabel tersebut.

Perencanaan Pembuatan Tabel

Perencanaan adalah langkah pertama dalam membuat sebuah tabel. Disini perlu direncanakan apa yang perlu dibuat dan dimasukkan dalam sebuah tabel yang akan dibuat atau membuat layoutnya. Oleh karena itu perlu digunakan langkah-langkah berikut:

- Digunakan sedikit mungkin informasi pada setiap field untuk pemeliharaan data yang fleksibel dan mempercepat query. Misal pada field alamat dapat dibagi menjadi field jalan, kota, telepon, kode pos dan lain-lain.

- Membuat tabel yang lengkap. Tabel yang lengkap adalah tabel yang berisi informasi yang dibutuhkan secara jelas dan lengkap. Jika kemudian ditemukan field (informasi) yang lain, dapat ditambahkan pada tabel.
- Membuat tabel sesederhana mungkin untuk menghindari kompleksitas data tabel.
- Menghindari penduplikasian (redundancy) data. Contoh apabila akan dihubungkan dua buah tabel, perlu dihindari penggabungan field yang sama dari kedua tabel tersebut.
- Memahami jenis tabel yang dibutuhkan (Paradox atau dBASE) berdasarkan keuntungan masing-masing. Jika tabel Paradox mendukung password, validity checks, referential integrity, dan tipe field yang banyak jenisnya. Sedangkan tabel dBASE mendukung penghapusan dan kompatibel dengan dBASE yang ada sekarang.
- Untuk memilih tipe tabel yang akan digunakan, perlu dipertimbangkan :
  - Jenis tabel apa yang akan dipinjam strukturnya nanti.
  - Beberapa aturan untuk struktur tabel, seperti nama field, tipe, ukuran, dan aturan untuk menemukan field kunci yang diperbolehkan. Misalnya, Paradox memperbolehkan spasi dan penggunaaan tanda baca pada penamaan tabel, sedangkan dBASE tidak.

Dengan demikian, ada beberapa fasilitas dari tipe tabel yang dibutuhkan. Fasilitas-fasilitas tersebut akan memberikan pertimbangan sebelum membuat tabel. Untuk itu dapat dibuat sebuah tabel pertimbangan untuk memilih Paradox atau dBASE seperti di bawah ini:

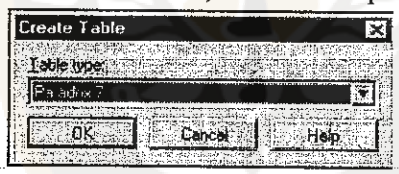
Tabel 1. Tabel Perbandingan Fasilitas Paradox dan dBASE

Fasilitas	Paradox	dBASE
Mempunyai validity checks, referential integrity, dan tabel look up	Ya	Tidak
Me-refresh data yang diubah	Ya	Tidak
Mempunyai urutan penomeran dalam tabel	Ya	Tidak
Memformat field memo sebaik field long integer, time, timestamp, autoincrement, dan bytes	Ya	Tidak

Membuat Tabel Paradox

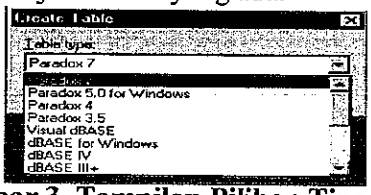
Untuk membuat sebuah tabel Paradox dari jendela Database Desktop dapat dilakukan langkah-langkah sebagai berikut :

1) Memilih New dari menu File, kemudian dipilih **Table.**



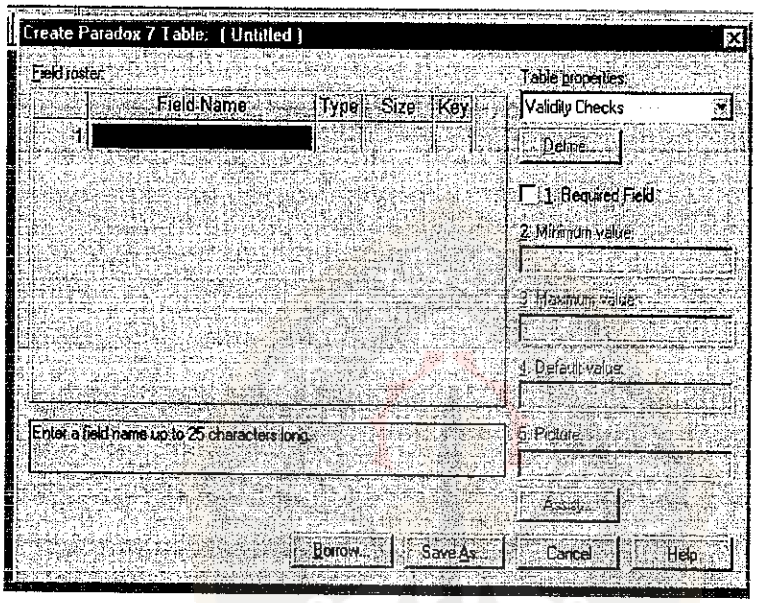
Gambar 2. Tampilan Pembuatan Tabel Paradox

2) Klik anak panah pada kotak pilihan Tabel Type untuk melihat daftar jenis-jenis tabel yang ada.



Gambar 3. Tampilan Pilihan Tipe Tabel

3) Pilih tabel Paradox versi 7.0 dan klik tombol **OK**. Selanjutnya Database Desktop akan menampilkan kotak dialog Create Paradox 7.0 Table. Kotak dialog ini adalah kotak dialog modal sehingga tidak dapat digunakan fasilitas DATABASE DESKTOP lain sampai mengklik tombol OK atau Cancel.



Gambar 4. Tampilan Tabel Paradox

Mendefinisikan Field

Untuk mendefinisikan sebuah field terdapat beberapa aturan :

- Panjang maksimum sebuah nama field adalah 25 karakter.
- Sebuah nama field tidak dapat dimulai dengan spasi kosong, tetapi dapat mengandung spasi kosong di dalamnya.
- Setiap nama field dalam sebuah tabel harus unik.
- Sebuah nama field seharusnya tidak mengandung karakter penting seperti koma (,), pipa (|), dan tanda seru (!).

Field dan Ukurannya

Tipe dan ukuran field pada tabel Paradox diperlihatkan pada tabel di bawah ini :

Tabel 2. Jenis dan Ukuran Field pada Tabel Paradox

Simbol	Ukuran	Tipe
A	1 – 225	Alpha
N		Number
$		Money
S		Short
I		Long Integer
#	0 – 32*	BCD
D		Date
T		Time
@		Timestamp
M	1 – 240**	Memo
F	0 – 240**	Formatted Memo
G	0 – 240***	Graphic
O	0 – 240***	OLE
L		Logical
±		Autoincrement
B	0 – 240***	Binary
Y	1 – 255	Bytes

Keterangan :

* Jumlah digit di belakang desimal

** Field Memo dan Formatted Memo dapat dipilih panjang karakternya (1 sampai 240 karakter untuk memo dan 0 sampai 240 karakter untuk formatted memo)

*** Optional

Field BLOP

Tipe field pada Paradox seperti memo, formatted memo, graphic, OLE, dan binary mengandung *file* binary large object atau disingkat sebagai BLOP.

Mengindeks

Database Desktop mengelola *record* tabel Paradox. Tabel tersebut mempunyai kunci yang menunjuk pada nilai dalam field kunci atau bisa disebut sebagai indeks primer. Sebagai defaultnya, semua indeks dikelola dan diakses dalam perintah menaik (A sampai Z, 0 sampai 9).

Ketika didefinisikan kunci gabungan untuk sebuah tabel, Database Desktop membuat sebuah indeks gabungan primer, yang mengelola *record* dengan field pertama sebagai kunci (tergantung struktur tabelnya), field kedua dan seterusnya.

Pada tabel Paradox, indeks sekunder mendefinisikan sebuah perintah tampilan alternatif untuk mengubah penampilan perintah *record*, meskipun lokasi *record* secara fisik dalam tabel tidak berubah, indeks sekunder juga digunakan untuk query, untuk mempercepat kemampuan, dan membangun hubungan antar tabel.

Aturan Membuat Field Kunci

Untuk mendefinisikan sebuah kunci harus diikuti aturan-aturan sebagai berikut :

- Sebuah tabel hanya mempunyai sebuah kunci. Kunci ini dapat dibuat dari satu atau lebih field.
- Sebuah kunci tidak dapat mengandung memo, formatted memo, graphic, OLE, binary, logical, atau field bytes.
- Jika sebuah kunci didefinisikan sebagai field tunggal, maka field tersebut harus field pertama pada Field Roster.

- Jika didefinisikan lebih dari satu field sebagai kunci, maka berarti dibuat sebuah composite key. Field-field ini dianggap sebagai grup, dan harus unik untuk setiap *record* dalam tabel. Composite key harus dimulai dari field pertama dalam Field Roster.
- Jika ditambah kunci pada sebuah tabel yang sebelumnya tidak mempunyai kunci atau mempunyai kunci yang lain, maka pengacauan kunci (*key violation*) akan terjadi.
- Data yang sudah dimasukkan ke dalam tabel mengacaukan aturan yang dibuat oleh kunci baru. Database Desktop menulis *record* yang kuncinya kacau (*key violating*) pada tabel temporer khusus yang disebut Keyviol.
- *Record* yang kuncinya dihapus dari tabel. Dapat diubah *record* dalam tabel Key sehingga mereka membutuhkan kunci, kemudian ditambahkan pada tabel asli dengan menggunakan Tools, Utilities, Add.
- Tabel tipe Paradox mempunyai lebih dari satu field yang didefinisikan sebagai field kunci. Field-field tersebut diatur seperti grup atau gabungan. Field kunci gabungan harus field pertama dari tabel. Dapat digunakan field kunci gabungan apabila semua nilai adalah unik.
- Ketika sebuah tabel mempunyai sebuah field kunci gabungan, nilai duplikasi dibiarkan dalam sebuah field kunci

individu, sepanjang tidak terduplikasi pada semua kunci atau harus unik.

- Database Desktop mengurutkan tabel yang mempunyai field kunci gabungan dengan memulai dari field pertama, kemudian baru field selanjutnya.

Validity Check

Pada tabel Paradox, validity checks adalah aturan yang dibebankan pada sebuah field untuk meyakinkan bahwa data yang dientri ke dalam field adalah suatu kebutuhan penting. Cara mendefinisikan sebuah validity check menjelaskan data apa yang dapat dimasukkan dalam sebuah field. Database Desktop menyediakan lima macam validity checks sebagai berikut:

Tabel 3. Tabel Validity Check

Validity Check	Artinya
Required Field	Setiap *record* tabel harus mempunyai sebuah nilai dalam field
Minimum Value	Nilai yang dimasukkan ke dalam field harus sama atau lebih besar daripada nilai minimum yang dispesifikasikan.
Maximum Value	Nilai yang dimasukkan ke dalam field harus sama atau lebih kecil daripada nilai maksimum yang dispesifikasikan.
Default Value	Nilai yang dispesifikasikan di sini akan dimasukkan secara otomatis di dalam field ini, jika tidak ada nilai yang dimasukkan.
Picture	Menentukan sebuah karakter string yang bertindak selaku template untuk nilai yang dapat dimasukkan pada field ini.

Ketika menyimpan tabel yang dibuat, Database Desktop akan menyimpan validity checks dalam sebuah *file* dengan nama tabel dan berekstensi. VAL.

Ketika memilih sebuah field dalam Field Roaster, Database Destop menampilkan validity Checks pada sisi kanan bawah jendela. Setelah memilih field, validity checks mengubah untuk merefleksikan constraint pada field yang tepilih.

Referential Integrity

Referential Integrity adalah sebuah field atau kumpulan field pada satu tabel (tabel anak) yang harus menunjuk pada kunci tabel yang lain (tabel induk). Hanya nilai yang aktif pada kunci tabel induk yang valid dengan field tabel anak. Database Desktop akan memeriksa keabsahan nilai dari sebuah tabel sebelum menerimanya pada tabel referential integrity.