2. TEORI PENUNJANG

2.1. Sistem Informasi

Pada masa sekarang ini, salah satu fungsi komputer adalah alat pengolahan data secara elektronik yang mempunyai peranan penting dalam membantu manajemen untuk memecahkan masalah yang dihadapi. Salah satu masalah manajemen tersebut adalah manajemen administrasi aktivitas bisnis utama seperti pembelian, penjualan, dan akutansi.

Pimpinan merupakan orang yang mempunyai tanggung jawab paling menentukan sukses tidaknya usaha yang dipimpinnya. Hal ini dimanifestasikan dalam kemampuan pimpinan untuk mengambil keputusan. Pengambilan keputusan bukanlah merupakan pekerjaan yang mudah, karena dibalik keputusan itu akan terdapat resiko-resiko tertentu apabila keputusan yang diambil adalah merupakan keputusan yang salah atau kurang tepat.

Bagi seorang pimpinan dari suatu bidang usaha, yang senantiasa menghadapi berbagai problema yang memerlukan pemecahan yang tepat dan ekonomis melalui pengambilan keputusan yang tepat pula. Sebagai seorang pimpinan maka ia bertugas membuat laporan-laporan yang dibutuhkan oleh bagian-bagian yang berkepentingan serta berhubungan dengan masalah-masalah yang ada, selain itu berfungsi pula sebagai pembuat keputusan, sedangkan unsur lainnya (pelaksana) yang akan menjalankan keputusan-keputusan tersebut.

Untuk menjamin tepatnya keputusan-keputusan yang diambil oleh seorang pimpinan, maka keputusan-keputusannya harus berdasarkan data-data yang benar-benar dapat dipercaya (data yang *reliable*), bukan data yang hanya berdasarkan perkiraan. Dalam hubungan inilah diperlukan suatu sistem pengolahan data yang tepat dan cepat serta dapat memberikan informasi kepada pimpinan di dalam mengambil atau menentukan suatu keputusan yang akan dibuat, sehingga dapat menjamin benarnya keputusan yang diambil tersebut. Dalam hubungannya dengan masalah-masalah pengambilan keputusan dan tersedianya data atau informasi untuk itu, maka suatu bidang usaha yang besar sifatnya memerlukan suatu bentuk sistem informasi yang dapat mengatasi

masalah pendapat dalam pengambilan keputusan oleh manajemen dengan memasukan unsur komputer sebagai alat dalam sistem itu.

Pentingnya peranan komputer di dalam sistem informasi, disebabkan besarnya volume data yang akan diolah maupun keanekaragaman bentuk data yang harus diolah pada suatu bidang usaha, sehingga untuk pekerjaan itu memerlukan waktu dan membutuhkan biaya yang cukup besar bila diolah secara sistem manual.

Disamping itu, pengolahan data yang kompleks secara manual tidak dapat menjamin ketelitian atau kebenarannya. Masalah pendataan dan pengolahan data tersebut dapat dipecahkan atau diselesaikan dengan mudah dalam waktu yang singkat oleh komputer.

2.2. Delphi Database Programming

2.2.1. *Delphi database*

Borland Delphi 5.0 yang untuk selanjutnya disingkat dengan *Delphi* merupakan program aplikasi database yang berbasis *Object Pascal* dari Borland. Selain itu, Delphi juga memberikan fasilitas pembuatan aplikasi *visual* seperti *Visual Basic*.

Beberapa komponen yang paling sering digunakan dalam pembuatan program aplikasi *database*, antara lain:

- **Data Module (TDataModule)** adalah sebuah kelas pada *Delphi* yang dikhususkan untuk menampung komponen-komponen *non visual*, umumnya adalah komponen-komponen milik palet *Data Access*.

  Untuk membuat *Data Module* pilih main menu *File|New Data Module*. Jika ingin mengakses satu *Data Module*, maka pada unit yang bersangkutan tambahkan *Data Module (File|Use Unit)*.

  Untuk operasi *database*, *Delphi* mempunyai kemampuan untuk mengakses berbagai jenis data dengan menggunakan *Borland Database Engine (BDE)* atau yang juga disebut dengan *IDAPI*. Tanpa menggunakan *BDE* tersebut, untuk mengakses *database* lokal atau *server* harus menggunakan fungsi dari *vendor* lain.

BDE adalah alat perantara yang memberikan akses yang terpisah atas data yang sedang diakses. *Programmer* tidak perlu menghiraukan lagi bagaimana pengaksesan data sesungguhnya yang dilakukan *BDE*. Dengan *BDE* kita dapat menghasilkan suatu aplikasi yang memiliki kemampuan akses beberapa jenis data sekaligus, misalkan: aplikasi untuk mengakses data dari *server sybase* (untuk sebuah fungsi) dan *server oracle* (untuk fungsi lain).

- **Alias** adalah nama lain yang diberikan untuk sebuah *database*, baik yang berada pada *disk* lokal ataupun jaringan. Keuntungan menggunakan *alias*, yaitu:
  1. Kemampuan memisahkan data dan kode program.
  2. Kemampuan menggunakan sekumpulan data yang berbeda dalam satu *alias*.
  3. Kemampuan memindahkan data aplikasi dalam *database* lokal ke *database client/server*.

  Cara membuat alias, antara lain:

  1. Menggunakan *BDE Configuration Utilitiy* atau *BDE Administrator*.
  2. Menggunakan *Database Desktop*.
  3. Menuliskan *alias* dalam program, kemudian menyimpannya dalam konfigurasi *file*.
- **Database Desktop (DBD)** dapat dianggap sebagai versi mini dari *Paradox* atau *dBase for Windows*. *DBD* menyediakan metode untuk membuat, melihat, mengedit, mengubah struktur, mengindeks, mengurutkan, membuat *query*, dan memanipulasi tabel, serta membuat *alias*.
- **Komponen DBGrid** digunakan untuk menampilkan data dalam bentuk tabel.
- **Komponen DBNavigator** digunakan untuk mendukung kemampuan navigasi dalam operasi tabel. Komponen ini berupa sekelompok tombol.
- **Komponen DBText** digunakan untuk menampilkan data dari sebuah *field* tertentu.
- **Komponen DBEdit** digunakan supaya *user* dapat meng-edit sebuah *field* data.
- **Komponen DBMemo** digunakan untuk memberikan presentasi *visual* dari sebuah *field* memo milik sebuah *database*.

- **Komponen DBImage** digunakan untuk memberikan persentasi *visual* dari sebuah *field* yang menyimpan data grafis.
- **Komponen DataSource (TDataSource)** digunakan untuk menghubungkan komponen *Table* atau *Query* dengan komponen *database visual* (mis: *DBGrid*). Komponen ini mempunyai fungsi utama mengambil data dari tabel yang sudah didefinisikan pada komponen *Table*, *Query*, atau *StoredProc*. Properti *Dataset* akan menghubungkan *DataSource* dengan *Table* atau *Query* yang ada.
- **Komponen Tabel (TTable) dan Query (TQuery)** yang berada pada palet *Data Access* adalah komponen utama yang digunakan untuk menghubungkan aplikasi dengan data. Komponen ini meyediakan akses langsung ke setiap *record* dan *field* dari berbagai macam jenis tabel, misalnya: *Paradox, dBase, Access, FoxPro, ODBC-compliant,* ataupun juga *SQL database* pada sebuah *remote server*, seperti *InterBase, Oracle, Sybase, MS-SQL Server, Informix, DB2*.

  Ada beberapa macam cara untuk mengakses nilai *field* dari sebuah tabel, yaitu:

  1. *NamaTabel[‘NamaField’]*, mis:
     ```
     Table1[‘kodecust’]:= ‘123’;
     ```
  2. *NamaTabel.FieldValues[‘NamaField’]*, mis:
     ```
     Table1.FieldValues[‘kodecust’]:= ‘123’;
     ```
  3. *NamaTabel.FieldByName(‘NamaField’).As<tipedata>*, mis:
     ```
     Tabel1.FieldByName(‘kodecust’).AsString:= ‘123’;
     Tabel1.FieldByName(‘kodecust’).AsBoolean:= true;
     Tabel1.FieldByName(‘kodecust’).AsInteger:= 100;
     ```

  Berikut ini contoh untuk menambahkan data pada akhir *record*:

  ```
  // set pointer ke akhir table dan siap menambah data
    Table1.Append;
       Table1[‘Nama’] := Edit1.Text;
       Table1[‘NRP’] := Edit1.Text;
    Table1.Post;
  ```

  Ada beberapa macam cara untuk mengakses nilai *field* dari sebuah *query*, yaitu:

1. *NamaQuery[‘NamaField’]*, mis:

```
Query1[‘kodecust’]:= ‘123’;
```

2. *NamaQuery.FieldValues[‘NamaField’]*, mis:

```
Query1.FieldValues[‘kodecust’]:= ‘123’;
```

3. *NamaQuery.FieldByName(‘NamaField’).As<tipedata>*, mis:

```
Query1.FieldByName(‘kodecust’).AsString:= ‘123’;
Query1.FieldByName(‘kodecust’).AsBoolean:= true;
Query1.FieldByName(‘kodecust’).AsInteger:= 100;
```

Berikut ini contoh untuk menambahkan data pada akhir *record*:

```
// Menutup query lebih dulu
Query1.Close;
// Menghapus perintah SQL yang ada
   Query1.SQL.Clear;
// Menambahkan perintah SQL
Query1.SQL.Add(‘SELECT * FROM Orders’);
      Query1.SQL.Add(‘WHERE NoOrder < 1000’);
// Mempersiapkan query
   Query1.Prepare;
// Mengaktifkan query
Query1.Open;
```

2.3. *Database*

Database pada dasarnya memiliki pengertian kumpulan data-data dan informasi yang terstruktur dalam suatu tabel dan relasi sehingga memudahkan dalam pengolahan data. *Database* ini digunakan khususnya untuk arus informasi atau data dalam jumlah yang besar. *Database* dipakai untuk menyimpan data sehingga dapat dimanipulasi dengan mudah.

Database yang baik harus tersusun dalam suatu struktur tertentu dan data yang ada terhubung dengan alur yang jelas dan tidak terbelit-belit sehingga memudahkan dalam pengaksesan dan manajemen dari data yang ada pada *database* tersebut.

Apa masalah yang timbul jika *database* tidak tersusun dengan baik? Berikut ini beberapa hal yang akan muncul jika *database* tidak memenuhi syarat sebagai *database* yang baik, (Riyanto, 2003).

- **Pengulangan Data (*Redundancy Data*)**

  Pengolahan data secara manual dan tidak adanya penamaan standar pada penamaan elemen-elemen data mengakibatkan data yang sama disimpan pada *database*.

- **Ketergantungan Data (*Data Dependency*)**

  Hal ini terjadi ketika hubungan antar data tidak diatur secara benar dalam *database*. Akibat yang muncul adalah data yang seharusnya berubah ketika ada data baru masuk atau ketika ada perubahan data pada tabel yang lain, tidak berubah sesuai dengan apa yang diharapkan.

- **Kepemilikan Data yang Tersebar**

  Kesalahan dalam pengelompokan data mengakibatkan data yang seharusnya mudah didapatkan akan menjadi berbelit-belit dalam pengaksesannya. Dampak yang muncul dari hal ini adalah suatu sistem yang tidak mampu memberikan informasi yang konsisten, andal, dan akurat.

2.4. *Data Flow Diagram* (DFD)

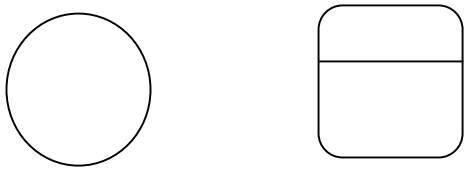
Data Flow Diagram (DFD) adalah representasi dari sebuah sistem secara grafis yang digambarkan dengan sejumlah simbol tertentu untuk menunjukkan perpindahan data dalam proses-proses suatu sistem. DFD merupakan suatu teknik grafik yang melukiskan arus informasi dan data serta perubahannya dari *input* ke *output*.

DFD menunjukkan perpindahan dan perubahan data dalam suatu sistem. Meskipun disebut arus data, namun penekanan pada DFD lebih pada prosesnya. Informasi dan perubahannya dalam DFD ditunjukkan dengan cara hirarki dalam bentuk diagram *level*. DFD *Level* 0 berisi entiti-entiti luar dari proses tunggal suatu sistem dengan *input* dan *output* data yang ditunjukkan dengan arah anak panah kedalam dan keluar. Diagram yang lebih detil lagi dari sistem tersebut dapat dibentuk dengan membagi proses pada *level* 0 DFD.

DFD menggunakan 4 macam simbol yaitu proses, arus data, simpanan data, dan kesatuan luar (*external entity*).

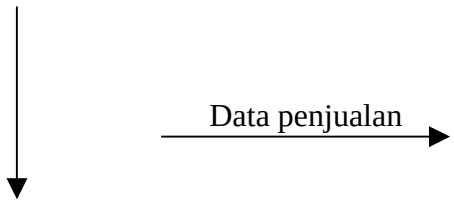
- Proses ialah simbol yang mengubah suatu data dari suatu bentuk menjadi bentuk yang lain. Atau dengan kata lain, proses menerima *input* data dan

mengeluarkan *output* data lain yang telah diproses. Simbol dari proses ada 2 macam, dapat dilihat pada gambar 2.1.



Gambar 2.1. Proses

- Arus data (*data flow*) ialah aliran yang menunjukkan perpindahan data dari satu bagian ke bagian yang lain dalam sebuah sistem. Data *flow* dalam DFD disimbolkan dengan tanda panah dan diberi nama atau keterangan di sampingnya yang menunjukkan data apa yang mengalir. Contoh dapat dilihat pada gambar 2.2.

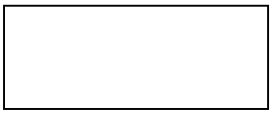


Gambar 2.2. Arus Data

- Simpanan data (*data storage*) ialah tempat penyimpanan data dalam suatu sistem, baik secara manual maupun secara elektronik. Simpanan data digunakan jika suatu proses perlu menggunakan data tersebut lagi kemudian. Simbol dari simpanan data dalam DFD ada 2 macam, seperti pada gambar 2.3.

Gambar 2.3. Simpanan Data

- Kesatuan luar (*external entity*) ialah seseorang, sekelompok orang, sebuah departemen didalam maupun diluar organisasi, atau sebuah sistem yang lain yang memberikan input untuk sistem yang ada atau menerima *output* dari sistem yang ada. *External entity* juga disebut *terminator*, karena merupakan batas dari sebuah sistem. Dalam DFD kesatuan luar disimbolkan dengan sebuah kotak persegi panjang seperti pada gambar 2.4.



Gambar 2.4. *External Entity*

2.5. *Entity Relationship Diagram* (ERD)

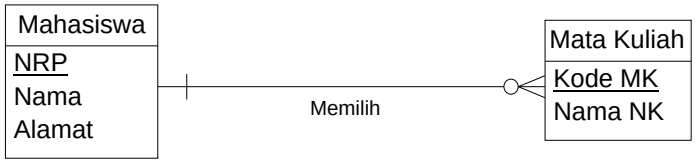
Sebuah ERD mendokumentasikan data sebuah perusahaan dengan cara menentukan data-data apa yang terdapat dalam tiap entiti dan bagaimana *relationship* (hubungan) antara sebuah entiti dengan yang lainnya. Di bawah ini akan dijelaskan beberapa hal yang digunakan dalam ERD:

- *Entity* (obyek data) dapat berupa *environmental element* (elemen di sekitar sistem yang berhubungan dengan sistem tersebut, *resource* (sumber daya yang berhubungan dengan sistem yang ada), dan transaksi yang sangat penting bagi usaha dagang sehingga dimodelkan dalam data. Contoh dari entiti ialah *customer*, pegawai, daftar presensi, dan sebagainya. Entiti digambarkan dengan sebuah kotak persegi empat dengan sebuah nama entiti (kata benda tunggal) di dalamnya, contoh pada gambar 2.5.

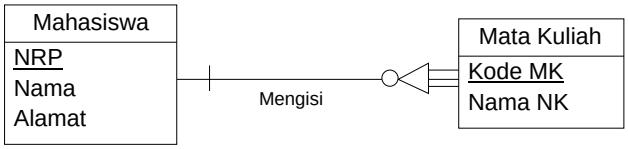
Gambar 2.5. Obyek Data

- *Attribute* dari sebuah entiti ialah karakteristik dari entiti tersebut. Sebagai contoh, atribut untuk entiti mahasiswa mungkin ialah NRP, Nama, Alamat, Sekolah asal, dan sebagainya. *Attribute value* ialah nilai dari setiap atribut untuk setiap entiti, misalnya NRP 26498005, 26498018, dan seterusnya seperti gambar 2.5.
- *Identifier* ialah atribut yang mengidentifikasikan sebuah entiti secara unik, contohnya ialah NRP untuk mahasiswa. Dalam ERD, *identifier* dituliskan dengan garis bawah, seperti pada gambar 2.5 dan gambar 2.6.
- *Relationship* ialah hubungan yang terjadi antara dua buah entiti, dan digambarkan dengan garis dan sebuah kata kerja disampingnya, contoh pada gambar 2.6. Dapat juga berupa garis dengan segitiga di tengah yang

melambangkan ketergantungan dengan entiti di atasnya (*dependent*), seperti pada gambar 2.7.

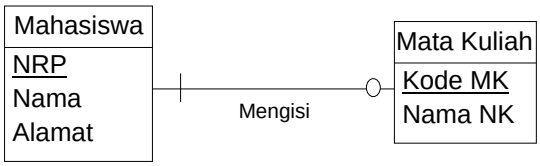


Gambar 2.6. *Relationship*



Gambar 2.7. *Dependent*

- *Connectivity* ialah jumlah yang menunjukkan berapa kali sebuah entiti muncul dalam relasi dengan entiti lainnya. Ada 3 jenis *connectivity*, yaitu:
    a. *One-to-one*, seperti pada gambar 2.8.



Gambar 2.8. *One-to-one*

    b. *One-to-many*, seperti pada gambar 2.6.
    c. *Many-to-many*, seperti pada gambar 2.9.

Gambar 2.9. *Many –to-many*

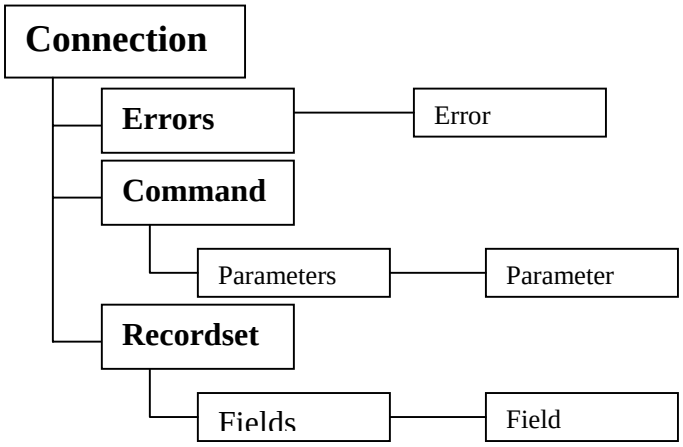
2.7. *ActiveX Data Objects* (ADO)

ADO adalah *interface* yang menjembatani layer OLE DB dengan syntax-syntax atau statement-statement yang akan digunakan dalam meminta dan meng-updata data dari sumber data. Tujuan dari ADO adalah konsistensi, objek model yang bebas bahasa programming yang mengijinkan aplikasi mengakses data tanpa melihat data source yang digunakan.

ADO mempunyai banyak *feature* yang membuatnya menjadi metode ideal dalam merancang aplikasi berbasis web dan juga aplikasi client/server dibandingkan metode yang lain. Beberapa feature tersebut adalah seperti berikut:

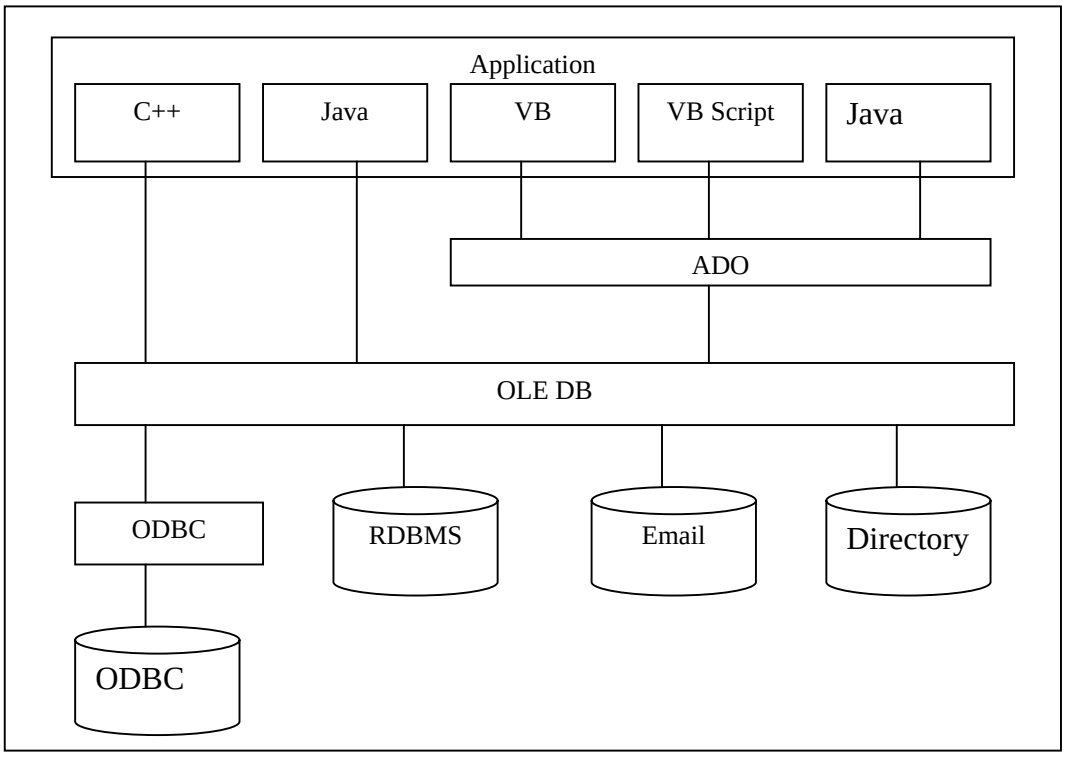
- Model Objeknya tidak terlalu besar (*small-object-model*) sehingga mudah untuk dipelajari.
- Menggunakan sedikit memory (*low-memory-footprint*).
- Akses data ke sumber data sangat cepat (*fast-access*)
- Koneksi ke sumber data dapat dibuat sambil jalan, tanpa harus mendeklarasikannya terlebih dahulu (*on-fly connections*)
- Mendukung type *cursor* yang berbeda-beda, termasuk *server-side cursor* (*cursor* pada sisi *server*)
- Mendukung penggunaan *Stored-Procedures* termasuk *input* dan *output* parameter.
- Mendukung penggunaan *Multiple recordset.*
- Mendukung penggunaan *Asynchronous queries* untuk mengambil data yang membutuhkan waktu lama dalam pengambilannya.

ADO mempunyai susunan/hirarki model objek sendiri. Tetapi, tidak seperti kebanyakan interface database yang lain (Jet dan RDO), hirarki yang ada tidak harus diikuti secara eksplisit. Untuk memulainya dapat dengan membuka objek Connection ataupun langsung mengakses object RecordSet untuk berhubungan dengan sumber data. Gambar 3.5 dibawah ini merupakan hirarki yang terdapat pada ADO.



Gambar 2.10. ADO Object Hierarchy

Service provider adalah "tenaga" sebenarnya yang terdapat didalam model objek ADO/OLE DB. Sementara ADO dan OLE DB menyediakan *syntax* yang sederhana dan mudah dipahami, *service provider*-lah yang sebenarnya menterjemahkan *syntax-syntax* tersebut kedalam bahasa yang dimengerti oleh sumber data. Contohnya service provider untuk SQL Server berbeda dengan *service* provider-nya Jet MDB, tetapi dengan ADO kita hanya menggunakan *syntax* yang sama untuk mengakses dua *database* tersebut.



Gambar 2.11.

Hubungan antara data source, teknologi data access dan bahasa pemrograman

2.8. Microsoft Access 2000.

2.8.1. Membuat File Database

Ada dua pilihan *database* yang bisa disusun dengan kotak dialog awal, yaitu sebuah *database* kosong dan juga database lengkap. *Database* kosong dihasilkan dari klik ganda pilihan pertama (*Blank Access Database*). Sedangkan klik ganda akan digunakan untuk membuat *database* menggunakan *wizard*.

2.8.2. Window Database

Database window Microsoft Access 2000 menyediakan berbagai cara dalam hal penampilan dan pengaturan objek *database*.

- ***Toolbar Database window*** berguna untuk operasi pengaturan objek *database* secara cepat, seperti pembuatan *database*, pembukaan ataupun pengaturan objek *database* lainnya.

- ***Bar Objects*** untuk menampilkan objek *database* secara lebih mudah, sebab orientasinya *vertical*.

- **Organisasi Objek *database* ke dalam *group*.** Klik *bar groups* untuk menampilkan kelompok, yang bias menampung objek *database* dari tipe yang berbeda.

- ***Shortcut* objek baru.** Pada *database window*, tersedia kumpulan *shortcut* untuk mengakses *database* dalam mode *design view* maupun *shortcut* untuk membuat *database* baru dengan cepat dan mudah, yaitu menggunakan *wizard*.

- **Pengaturan *database window*.** Ada cara baru untuk menangani objek didalam *database window*, yaitu untuk memilih objek anda tidak perlu melakukan klik, cukup meletakkan *pointer* saja diatasnya selama beberapa detik. Sedang untuk membukanya tidak perlu melakukan klik ganda, cukup satu kali saja.

- **Pemilihan dengan menuliskan nama objek.** Kini anda tidak perlu repot untuk mencari dan memilih objek, anda bisa menuliskan huruf awal dari nama objek, maka objek itu akan terpilih. Misalnya untuk memilih *table* ***Shippers*** cukup menuliskan Sh saja.

2.8.3. Mengatur Tabel

- Tentang tabel.

  Tabel adalah kumpulan data yang tersusun menurut aturan tertentu. Secara fisik, *table* berupa *grid* yang terdiri atas baris dan kolom. Baris menunjukkan record data dan kolom menunjukkan *field* data. Tabel bisa dipandang sebagai komponen utama pada *database*, karena table adalah dasar untuk menyusun komponen lainnya seperti *form*, *query* maupun *report*.

- Jenis Data Access 2000

Untuk menentukan jenis data *field*, kita harus mengetahui jenis-jenis data yang dikenal Access 2000. Berikut ini adalah tabel yang memuat jenis data tersebut:

Tabel 2.1. Tabel Jenis Data

Jenis Data	Keterangan dan Batasan
Text	Adalah data yang berupa teks dan tidak memerlukan angka maupun perhitungan, misalnya data tentang Nama, Alamat dan sebagainya. Karakter yang mampu ditampung sebanyak 255.
Memo	Adalah data berupa teks panjang, berguna untuk memberikan keterangan dan lain-lain. Panjangnya bisa mencapai 64.000 karakter.
Number	Adalah jenis data numeris atau angka biasa dan bukan mata uang ataupun angka dengan ketelitian tinggi. Bisa menampung sampai dengan 255.
Date/Time	Untuk menampung data waktu, yaitu hari, tanggal, jam, menit dan seterusnya.
Currency	Untuk menampung bilangan tanpa proses pembulatan pada saat perhitungan. Berguna untuk menampung mata uang dll. Bisa sampai 15 digit ketelitiannya.
Auto Number	Adalah bilangan yang secra otomatis dihasilkan oleh Access 2000 saat kita tambahkan *record* baru. Bilangan yang dihasilkan bisa urut, acak maupun *replication*.
Yes/No	Untuk menampung dua macam keadaan, ya atau tidak.

OLE Object	Menampung objek yang berasal dari aplikasi lain dari proses OLE (Obeject Linking and Embedding). Ukuran yang disediakan bisa sampai satu GygaByte.

- *Key* dan *Relasi*

  Key berguna sebagai wakil dari tabel untuk menunjukkan nilai unik suatu *field*. Dengan adanya *key*, proses pencarian menjadi lebih mudah karena tidak ada nilai yang kembar. Hal seperti ini akan sangat dirasakan manfaatnya pada suatu database dengan *record* yang besar

- Membuat *Query*

  Query adaalah sarana untuk mengatur data yang disimpan dalam tabel, sehingga hanya data tertentu yang dimunculkan dalam tabel. Karena menyertakan unsur urutan proses, maka *query* dikelompokkan dalam lingkup pemrograman.

  Berbeda dengan pengoperasian secara manual langsung, pengoperasian *query* bisa menggunakan seperangkat perintah yang membentuk susunan program dengan bahasa yang dikenal dengan SQL atau Structure Query Language.

  Dengan memanfaatkan *query*, maka penanganan *database* menjadi lebih mudah, karena kita lebih leluasa untuk menyusun tabel dengan kriteria baru sesuai dengan keperluan.