BAB II

LANDASAN TEORI

2.1 Sejarah Apotek Ciwaruga

Apotek Ciwaruga berdiri tgl 28 Agustus 2002, dengan surat izin apotek, No : 442/017-SIA/3909-DinKes/2002 dan Apotek Ciwaruga merupakan salah satu badan usaha dari PT. Viktori Agung dengan akta notaris dan pejabat pembuat akta tanah, Ruli Iskandar, SH dengan No. 6 tanggal 15 Mei 2002 di Jakarta.

Apotek Ciwaruga pernah membuka cabang apotek di dalam lokasi PDAM kota Bandung di Jl. Badaksinga dengan kerjasama koperasi PDAM pusat Bandung pada tahun 2004-2006. Karena omset penjualannya yang kecil maka apotek tersebut diambil alih oleh koperasi PDAM pusat Bandung. Apotek Ciwaruga masih berencana membuka cabang apotek di tempat lain untuk mengembangkan bisnis dan pelayanan kepada masyarakat di sekitarnya.

Arti kata apotek adalah tempat menjual dan membuat atau meramu, meracik obat. Apotik juga merupakan tempat apoteker melakukan praktek profesi farmasi sekaligus menjadi paritel. Kata ini berasal dari kata bahasa Yunani yaitu *Apotheca* yang secara harfiah berarti ” Penyimpanan ”.

Menurut KepMenKes RI No. 1332/Menkes/SK/X/2002, disebutkan bahwa persyaratan-persyaratan apotek adalah sebagai berikut :

a. Untuk mendapatkan izin apotek, apoteker yang bekerjasama dengan pemilik sarana yang telah memenuhi persyaratan harus siap dengan tempat, perlengkapan termasuk sediaan farmasi dan perbekalan farmasi yang lain yang merupakan milik sendiri atau milik pihak lain.
b. Sarana apotek dapat didirikan pada lokasi yang sama dengan pelayanan komoditi yang lain di luar sediaan farmasi.
c. Apotek dapat melakukan kegiatan pelayanan komoditi yang lain di luar sediaan farmasi.

2.1.1 Visi dan Misi Apotek Ciwaruga

Visi menggambarkan rencana apotek kedepan, dan misinya menggambarkan sesuatu pelayanan yang lebih baik disbanding sebelumnya.

2.1.1.1 Visi

Visinya menjadi usaha apotek, praktik dokter, dan rumah sakit besar di Bandung Utara yang mengabdikan pada kepentingan dan peningkatan kesehatan masyarakat.

2.1.1.2 Misi

Adapun misi dari Apotek Ciwaruga adalah :

1. Memberikan pelayanan kesehatan kepada masyarakat
2. Memberikan tambahan pemahaman atau pengertian tentang obat-obatan

3. Membantu masyarakat yang tidak mampu untuk berobat ke dokter dan persalinan
4. Turut membantu dan mengembangkan serta meningkatkan perekenomian masyarakat di sekitarnya.

2.1.1.3 Tempat dan Kedudukan Apotek

Nama Apotek : Apotek Ciwaruga

No SIK : 442/017-SIA/3909-Dinkes/2002

Propinsi : Jawa Barat

Otonomi Daerah : Kabupaten Bandung

Kecamatan : Gegerkalong Hilir

Desa / Kelurahan : Gegerkalong

Alamat : Jl. Terusan Gegerkalong Hilir No.39

Kode Pos : 40111

Telepon : (022) 2017667

Status Apotek : Milik sendiri

Lokasi : Cukup strategis

Luas Apotek :

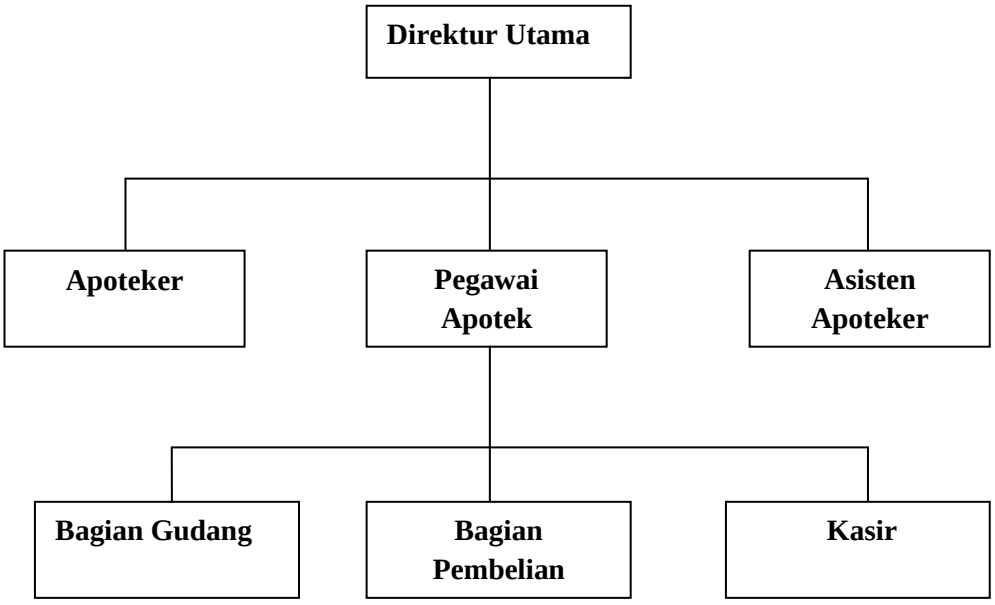
Luas Bangunan : 160 m^2

Luas Tanah : 84 m^2

2.1.1.4 Struktur Organisasi Apotek Ciwaruga

Struktur organisasi Apotek Ciwaruga secara umum terdiri :

1. Unsur pimpinan : Direktur utama.
2. Unsur penanggung jawab opersional : Apoteker dan asisten apoteker.
3. Unsur pelaksanaan pengadaan obat dan layanan farmasi: Asisten apoteker, pegawai apotek dan bagian gudang.
4. Unsur penunjang : Bagian pembelian dan kasir.



Gambar 2.1 Struktur Organisasi Apotek Ciwaruga

A. Uraian Jabatan

1. Direktur Utama

   Direktur utama memiliki tugas dan tanggung jawab sebagai berikut:

   A. Menyediakan sarana apotek termasuk bangunannya.

B. Direktur utama meliputi:

a. Menyusun perencanaan.

b. Mengorganisasi kegiatan.

c. Mengarahkan kegiatan.

d. Mengkoordinasikan kegiatan.

e. Melakukan evaluasi terhadap kegiatan.

f. Melaksanakan pengawasan.

g. Menentukan kebijakan.

h. Mengadakan rapat.

i. Mengambil keputusan.

j. Mengatur hubungan antara apotek dengan masyarakat dan instansi terkait.

C. Direktur utama selaku administrator bertugas menyelenggarakan administrasi meliputi:

a. Perencanaan.

b. Pengarahan.

c. Pengorganisasian.

d. Pengawasan.

e. Keuangan.

Dalam keadaan melakukan tugasnya, Pemilik Sarana Apotek (PSA) dibantu oleh apoteker, asisten apoteker, dan pegawai apotek.

2. Apoteker

Apoteker merangkap sebagai kepala apotek membantu Direktur utama dalam kegiatan-kegiatan sebagai berikut:

a. Memberikan konsultasi kepada konsumen.
b. Bertanggung jawab penuh atas kegiatan yang ada di apotek.
c. Bertanggung jawab atas prosedur aturan pengeluaran dan penyimpanan obat di apotek.
d. Membuat beberapa laporan kepada Dinas instansi terkait diantaranya DKBPOM (Dinas Kesehatan Badan Pemeriksaan Obat dan Makanan).
e. Memberikan pelatihan kepada asisten apoteker dan karyawan di apotek.

3. Asisten Apoteker

Asisten apoteker mempunyai tugas melaksanakan pengawasan dan pengerjaan harian proses transaksi apotek dan bertanggung jawab terhadap seluruh kegiatan di apotek sebagai berikut:

a. Memberikan konsultasi kepada konsumen.
b. Memberi harga pesanan obat yang diorder.
c. Bertanggung jawab atas prosedur keluar masuknya obat-obatan.
d. Membuat racikan atas obat resep dari dokter yang berupa racikan.
e. Melakukan pemasukan data pembelian obat menggunakan komputer.
f. Memantau persiapan obat di gudang.

g. Bertanggung jawab atas semua kegiatan apotek.
h. Melakukan pengawasan atas semua proses transaksi.
i. Membantu menyiapkan laporan yang diminta oleh apoteker dan direktur utama.
j. Bertanggung jawab atas semua laporan terhadap instansi terkait badan usaha komersial dan secara Dinas Kesehatan.
k. Bertanggung jawab atas semua laporan yang diperlukan.

4. Pegawai Apotek

Pegawai apotek membantu asisten apotek dalam kegiatan sebagai berikut:

1. Menyiapkan semua prosedur pembukuan secara adminstrasi yang benar.
2. Melakukan pencatatan atas semua biaya yang terjadi secara pembukuan.
3. Membuat pesanan obat ke supplier.
4. Melakukan pengarsipan atas semua *Purchase Order* (PO) pembelian obat dan pembayaran pajak.
5. Menyiapkan pesanan atas obat-obatan yang kosong.
6. Menyiapkan laporan pajak.
7. Menyiapkan jurnal untuk dibuat *cash flow* dan *neraca rugi laba.*

5. Bagian Gudang

Bagian gudang bertanggung jawab atas seluruh keberadaan barang di gudang.

1. Memberi laporan stok obat ke pegawai apotek.
2. Bertanggung jawab atas keluar masuk obat di gudang.
3. Melakukan *stock opname* obat per bulan dan per tahun.

6. Bagian Pembelian

Bagian pembelian mempunyai tugas dan tangung jawab atas segala *Purchase Order* (PO) obat ke supplier.

1. Mencari harga yang lebih murah ke supplier namun berkualitas.
2. Menjalin hubungan baik ke supplier.
3. Mengatur *Purchase Order* (PO) pesanan obat mana yang *urgent* itu yang diutamakan.

7 Kasir

Kasir membantu secara *team* pekerjaan asisten apoteker, pegawai apotek, bagian gudang, bagian pembelian dan bertanggung jawab atas semua transaksi pembayaran.

a. Melakukan proses transaksi pembelian
b. Melakukan proses tansaksi pembayaran obat dari supplier
c Betanggung jawab atas seluruh pembayaran pajak ke pemerintah
d. Bertanggung jawab atas semua setoran harian

2.2 Konsep Dasar Sistem

Adapun konsep dasar sistem antara lain pengertian sistem, karakteristik sistem dan klasifikasi sistem.

2.2.1 Pengertian Sistem

Terdapat dua kelompok pendekatan di dalam mendefinisikan siestem yaitu yang menekankan pada prosedurnya dan yang menekankan pada komponen atau elemennya.

Sistem menurut pendekatan prosedur adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan berkumpul bersama-sama untuk melakukan kegiatan atau untuk menyelesaikan suatu tujuan tertentu. Sistem menurut pendekatan elemen atau komponennya adalah kumpulan dari elemen-elemen yang saling berinteraksi untuk mencapai suatu tujuan tertentu.

2.2.2 Karakteristik Sistem

Suatu sistem mempunyai karakteristik atau sifat-sifat tertentu yaitu :

1. Komponen-komponen (*Components*), suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerja sama membentuk satu kesatuan. Komponen-komponen sistem dapat berupa suatu subsistem atau bagian dari sistem.
2. Batas (*Boundary*), batas sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batas suatu sistem menunjukkan rung lingkup (*scope*) dari sistem tersebut.
3. Lingkungan luar (*environments*), adalah apapun di luar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan juga dapat bersifat merugikan. Sistem tersebut yang menguntungkan merupakan energi dari sistem yang harus tetap dijaga dan

dipelihara, sedangkan yang merugikan sistem harus ditahan dan dikendalikan, kalau tidak maka akan mengganggu kelangsungan hidup dari sistem tersebut.

4. Penghubung (*interface*), penghubung suatu sistem merupakan media penghubung antar satu sub sistem dengan sub sistem yang lainnya. *Output* dari suatu sub sistem akan menjadi *input* bagi sub sistem yang lainnya dengan melalui penghubung sistem ini.
5. Masukan (*input*), adalah energi yang dimasukkan ke dalam sistem. *Input* bisa berupa input perawatan (*maintenance input*) dan *input* sinyal (*signal input*). *Maintenance input* adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi, sedangkan *signal input* adalah energi yang diproses untuk mendapatkan keluaran. Sebagai contoh didalam sistem kompiuter, program adalah *maintenance input* yang digunakan untuk mengoperasikan komputernya dan data adalah *signal input* untuk diolah menjadi informasi.
6. Keluaran (*output*), adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan. *Output* dapat merupakan masukan untuk sub sistem yang lain atau kepada supra sistem.
7. Pengolahan (*process*), suatu sistem mempunyai suatu bagian pengolahan yang akan merubah *input* menjadi *output.*
8. Sasaran (*objective*) dan tujuan (*goal*), suatu sistem pasti mempunyai tujuan atau sasaran. Sasaran dari sistem sangat menentukan sekali *input* yang dibutuhkan sistem dan *output* yang akan dihasilkan sistem.

2.2.3 Klasifikasi Sistem

Sistem dapat diklasifikasikan dari beberapa sudut pandang, diantaranya adalah sebagai berikut :

1. Sistem diklasifikasikan sebagai sistem abstrak (*abstract system*) dan sistem fisik (*physical system*). Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik.
2. Sistem diklasifikasikan sebagai sistem alamiah (*natural system*) dan sistem buatan manusia (*human made system*). Sistem alamiah adalah sistem yang terjadi melalui proses alam, tidak dibuat manusia. Sistem buatan manusia adalah sistem yang dirancang oleh manusia.
3. Sistem diklasifikasikan sebagai sistem tertentu (*deterministic system*) dan sistem tak tentu (*probabilistic system*). Sistem tertentu beroperasi dengan tingkah laku yang sudah dapat diprediksi, interaksi diantara bagian-bagiannya dapat dideteksi dengan pasti sehingga keluaran dari sistem dapat diramalkan. Sistem tak tentu adalah sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilitas.
4. Sistem diklasifikasikan sebagai sistem tertutup (*closed system*) dan sistem terbuka (*open system*). Sistem tertutup merupakan sistem yang tidak berhubungan dengan lingkungan luarnya. Sistem terbuka adalah sistem yang berhubungan dan terpengaruh dengan lingkungan luarnya.[1].

2.3 Komponen Dasar Informasi

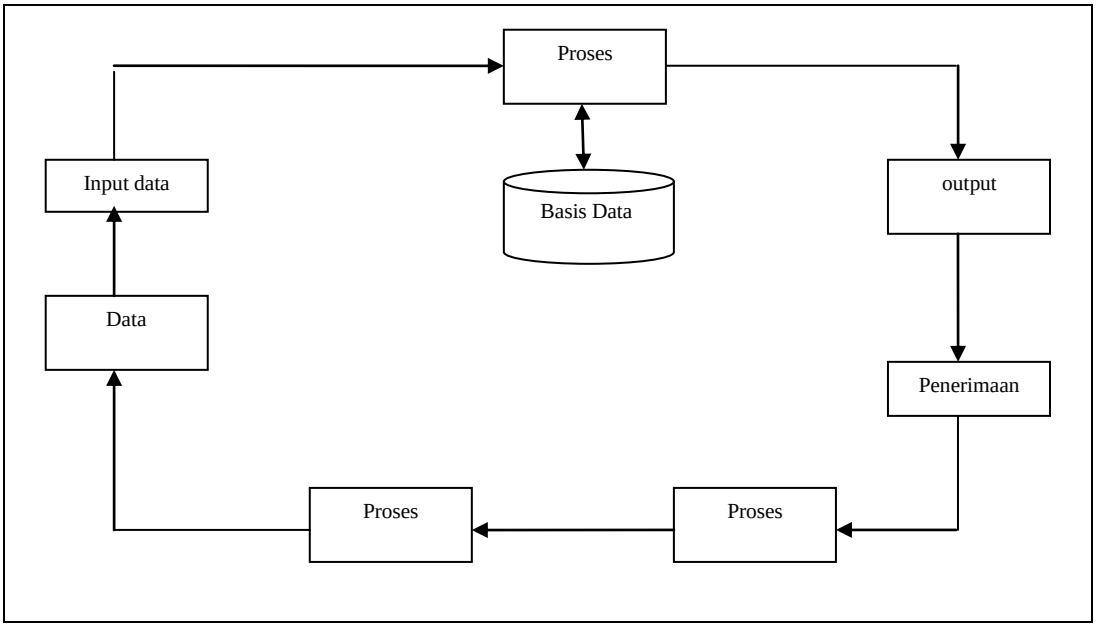
Adapun komponen dasar informasi diantaranya pengertian informasi, siklus informasi, komponen sistem informasi dan sistem informasi manajemen.

2.3.1 Pengertian Informasi

Informasi dapat didefinisikan sebagai hasil dari pengolahan data dalam suatu bentuk yang lebih berguna dan lebih berarti bagi penerimanya yang menggambarkan suatu kejadian-kejadian (*event*) yang nyata (*fact*) yang digunakan untuk pengambilan keputusan. Sumber dari informasi adalah data. Data adalah kenyataan yang menggambarkan kejadian-kejadian dan kesatuan nyata.

2.3.2 Siklus Informasi

Data merupakan bentuk yang masih mentah, belum dapat bercerita banyak, sehingga perlu diolah lebih lanjut. Data yang diolah melalui suatu model menjadi informasi, penerima kemudian menerima informasi tersebut, membuat suatu keputusan dan melakukan tindakan yang berarti menghasilkan suatu tindakan yang lain yang akan membuat sejumlah data kembali. Data tersebut akan ditangkap sebagai *input*, diproses kembali lewat suatu model dan seterusnya membentuk suatu siklus. Siklus ini oleh John Burch disebut dengan siklus informasi (*information cycle*) atau ada yang menyebutnya dengan istilah siklus pengolahan data (*data processing cycles*) seperti pada gambar 2.2.



Gambar 2.2 Siklus Informasi

2.3.3 Komponen Sistem Informasi

Sistem informasi terdiri dari komponen-komponen yang disebut dengan istilah blok bangunan (*building block*) yaitu:

1 Blok masukan (*input blok*), *input* mewakili data yang masuk kedalam sistem iformasi. *Input* disini termasuk metode-metode dan media untuk menangkap data yang akan dimasukkan, yang dapat berupa dokumen-dokumen dasar.

2 Blok model (*model block*), blok ini terdiri dari kombinasi prosedur logika dan model matematika yang akan memanipulasi data *input* dan data yang tersimpan di dasar data depan dengan cara tertentu untuk menghasilkan keluaran yang diinginkan.

3 Blok keluaran (*output block*), produk dari sistem informasi adalah keluaran yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta semua pemakai sistem.

4 Blok teknologi (*technology block*), teknologi merupakan kotak atau alat (*tool box*) dari pekerjaan sistem informasi. Teknologi terdiri dari (dua) bagian utama yaitu perangkat lunak (*software*) dan perangkat keras (*hardware*).

5 Blok basis data (*database block*), basis data merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya tersimpan di perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya.

6 Blok kendali (*control block*), blok kendali ini diterapkan agar sistem informasi dapat berjalan sesuai dengan yang diinginkan.[1]

2.3.4 Sistem Informasi Manajemen

Sistem informasi manajemen (*management information systems*) atau sering dikenal dengan singkatan (MIS) merupakan penerapan sistem informasi di dalam organisasi untuk mendukung informasi yang dibutuhkan untuk semua tingkatan manajemen. Sistem Informasi Manajemen (SIM) dapat didefinisikan sebagai kumpulan dari interaksi sistem-sistem informasi yang bertanggung jawab mengumpulkan dan mengolah data untuk menyediakan informasi yang berguna untuk semua tingkatan manajemen di dalam kegiatan perencanaan dan pengendalian.[2].

2.4 Konsep Dasar Basis Data

Konsep dasar basis data kumpulan dari *item* data yang saling berhubungan satu dengan yang lainnya yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, tersimpan di *hardware* komputer dan dengan *software* untuk melakukan manipulasi, untuk kegunaan tertentu ada juga yang mendefinisikan basis data adalah kumpulan informasi yang disimpan di dalam komputer untuk memperoleh informasi dari basis data tersebut.

1. Alasan Perlunya *Database*
   a. Basis data merupakan salah satu komponen penting dalam sistem informasi, karena merupakan dasar dalam menyediakan informasi.
   b. Basis data menentukan kualitas informasi : akurat, tepat pada waktunya dan relevan. Informasi dapat dikatakan bernilai bila manfaatnya lebih efektif dibandingkan dengan biaya mendapatkannya.
   c. Basis data mengurangi duplikasi data(data *redundancy*).
   d. Dengan mengaplikasikan basis data hubungan data dapat ditingkatkan.
   e. Basis data dapat mengurangi pemborosan tempat simpanan luar.
2. Bahasa Basis Data

Bahasa basis data terdiri atas : *Data Defination Languange* (DDL), merujuk pada kumpulan perintah yang dapat digunakan untuk mendefinisikan objek-objek basis data, seperti membuat sebuah tabel basis data atau indeks primer atau sekunder. *Data Manipulation Language* (DML), mengacu pada kumpulan perintah yang dapat digunakan untuk melakukan manipulasi data, seperti

penyimpanan data ke suatu tabel, kemudian mengubahnya dan menghapusnya atau hanya sekedar menampilkannya kembali.

3. Objektif Basis Data

Secara lebih lengkap, pemanfaatan basis data dilakukan untuk memenuhi sejumlah tujuan (objektif) seperti berikut :

a. Kecepatan dan kemudahan (*speed*)
b. Efisiensi ruang penyimpanan (*space)*
c. Keakuratan (*accuracy*)
d. Ketersediaan (*avaibility*)
e. Kelengkapan (*completeness*)
f. Keamanan (*security*)
g. Kebersamaan pemakaian (*sharability*)

Adapun konsep dasar basis data adalah pendahuluan, definisi basis data, tujuan basis data, dan manfaat/kelebihan basis data.[3]

2.4.1 Pendahuluan

Pemrosesan basis data sebagai perangkat andalan sangat diperlukan oleh berbagai instansi dan perusahaan. Dalam pengembangan sistem informasi diperlukan basis data sebagai media penyimpanan data. Kehadiran basis data dapat meningkatkan kinerja perusahaan dan dapat meningkatkan daya saing perusahaan tersebut. Basis data dapat mempercepat upaya pelayanan kepada pelanggan, menghasilkan informasi dengan cepat dan tepat sehingga membantu pengambilan keputusan untuk segra memutuskan suatu masalah berdasarkan

informasi yang ada. Banyak aplikasi yang dapat dibuat dengan berlandaskan pada basis data antara lain : semua transaksi perbankan, aplikasi pemesanan dan penjadwalan penerbangan, proses registrasi dan pencatatan data mahasiswa pada perguruan tinggi, aplikasi pemrosesan penjualan, pembelian dan pencatatan data pegawai beserta aktifisnya termasuk operasi penggajian pada suatu perusahaan, dan sebagainya. Beberapa informasi pada perusahaan *relative* seperti pencarian jumlah penjualan, mencari jumlah stok yang tersedia, barang apa yang paling laku dijual pada bulan ini dan berapa laba bersih perusahaan dapat diketahui dengan mudah dengan basis data.

Begitu banyak keuntungan yang dapat diperoleh dengan pemanfaatan basis data. Basis data dapat meningkatkan daya guna perangkat komputer yang mungkin tadinya hanya untuk keperluan *game* atau pengetikan dengan aplikasi *office.*

2.4.2 Definisi Basis Data

Basis data adalah kumpulan data yang saling berelasi. Data sendiri merupakan fakta mengenai obyek, orang dan lain-lain. Data dinyatakan dengan nilai (angka, deretan karakter atau simbol). Basis data dapat didefinisikan dalam berbagai sudut pandang seperti berikut :

1. Himpunan kelompok data yang saling berhubungan yang diorganisasi sedemikian rupa sehingga kelak dapat dimanfaatkan dengan cepat dan mudah.

2. Kumpulan data yang saling berhubungan disimpan secara bersama sedemikian rupa tanpa pengulangan (*redundancy*) yang tidak perlu untuk memenuhi kebutuhan.
3. Kumpulan *file*/tabel/arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronik.

2.4.3 Tujuan Basis Data

Basis data bertujuan untuk mengatur data sehingga diperoleh kemudahan, ketepatan dan kecepatan dalam pengambilan kembali. Untuk mencapai tujuannya, syarat sebuah basis data yang baik adalah sebagai berikut :

1. Tidak adanya redundansi dan inkonsistensi data, redundansi terjadi jika suatu informasi disimpan di beberapa tempat.
2. Kesulitan pengaksesan data, basis data memiliki fasilitas untuk melakukan pencarian informasi dengan menggunakan *query* ataupun dari *tool* untuk melihat tabelnya. Dengan fasilitas ini kita bisa secara langsung melihat data dari *software* DBMS-nya. Selain itu basis data bisa dihubungkan dengan program aplikasi sehingga memudahkan pengguna dalam mengakses informasi. Misalnya program aplikasi untuk kasir yang terhubung dengan basis data. Pengguna cukup menggunakan fasilitas pencarian ataupun laporan yang tersedia pada program aplikasi untuk mendapatkan informasi stok, laporan penjualan dan lain-lain. Dalam basis data, informasi yang diperoleh dari kumpulan data bisa berupa keseluruhan data, sebagian data, data dengan filter tertentu, data yang terurut ataupun data *summary*. Sebagai contoh sederhana

ketika kita ingin mencatat data alamat dan telepon dari kolega kita. Sebagian orang akan menggunakan buku alamat. Metode pencatatan dilakukan dengan menuliskan data setelah catatan terakhir. Ketika kita menginginkan informasi alamat seseorang, kita akan kesulitan mencari karena informasi yang tersaji tidak terurut. Ada juga orang mencatat dengan mengelompokan nama berdasarkan abjad. Hal ini akan lebih mempermudah pencarian karena kita tidak perlu membaca keseluruhan data, tetapi cukup dalam satu kelompok saja. Tapi masalah baru muncul ketika jumlah data untuk sekelompok data abjad tertentu terlalu banyak sedangkan kelompok abjad yang lain masih terlalu sedikit. Dalam metode ini ada banyak ruang yang tidak terpakai jika kita memberikan ruang yang sama untuk setiap kelompok. Dalam hal pencarian, kesulitan akan kita temui ketika informasi yang ingin kita cari dengan kata kunci sebagian namanya. Misalnya kita akan mencari alamat Anto, sementara yang tercatat dalam buku catatan adalah Mardianto. Tentu saja kita tidak akan dapat menemukannya dalam kelompok data dengan huruf depan A. selain itu tidak selamanya kata kunci yang diketahui adalah dari nama, tetapi bisa saja yang diketahui adalah nomor teleponnya, sedangkan yang ingin kita cari adalah alamat dan namanya. Hal ini merupakan masalah baru dari pencatatan data dengan buku. Basis data bisa memberikan solusi terhadap permasalahan-permasalahan tersebut di atas.

2.4.4 Manfaat/Kelebihan Basis Data

Banyak manfaat yang dapat kita peroleh dengan menggunakan basis data. Manfaat/kelebihan basis data diantaranya adalah :

1. Kecepatan dan kemudahan (*speed*)

   Dengan menggunakan basis data pengambilan informasi dapat dilakukan dengan dan mudah. Basis data memiliki kemampuan dalam mengelompokkan, mengurutkan bahkan perhitungan dengan matematika. Dengan perancangan yang benar, maka penyajian informasi akan dapat dilakukan dengan cepat dan mudah.

2. Kebersamaan pemakai (*sharability*)

   Sebuah basis data dapat digunakan oleh banyak *user* dan banyak aplikasi. Untuk data yang diperlukan oleh banyak bagian/orang, tidak perlu melakukan pencatatan di masing-masing bagian, tetapi cukup dengan satu basis data untuk dipakai bersama. Misalkan data mahasiswa dalam suatu perguruan tinggi, dibutuhkan oleh banyak bagian diantaranya : bagian akademik, bagian keuangan, bagian kemahasiswaan dan perpustakaan. Tidak harus semua bagian ini memiliki catatan data mahasiswa. Data cukup disediakan oleh sebuah basis data dan semua bagian bisa mengakses data tersebut sesuai dengan keperluannya.

3. Pemusatan kontrol data

   Karena cukup dengan satu basis data untuk banyak keperluan, pengontrolan terhadap data juga cukup dilakukan disatu tempat saja. Jika ada perubahan data

alamat mahasiswa misalnya, maka tidak perlu kita meng-*update* semua data di masing-masing bagian tetapi cukup hanya di satu basis data.

2.5 Konsep Dasar Sistem Basis Data

Adapun konsep dasar sistem basis data terdiri dari sistem dan sistem basis data.

2.5.1 Sistem

Sistem adalah sebuah tatanan yang terdiri atas sejumlah komponen fungsional (dengan tugas/fungsi khusus) yang saling berhubungan dan secara bersama-sama bertujuan untuk memenuhi suatu proses/pekerjaan tertentu. Sebagai contoh, sistem kendaraan terdiri dari : komponen starter, komponen pengapian, komponen penggerak, komponen pengerem, komponen kelistrikan, lampu dan lain-lain. Komponen-komponen tersebut diatas memiliki tujuan yang sama, yaitu untuk membuat kendaraan tersebut bisa dikendarai dengan nyaman dan aman. Contoh lain yaitu sistem perguruan tinggi yang terdiri dari dosen, mahasiswa, kurikulum dan lain-lain. Sistem ini bertujuan untuk menghasilkan mahasiswa-mahasiswa yang memiliki kemampuan di bidang ilmunya.

2.5.2 Sistem Basis Data

Sistem basis data merupakan perpaduan antara basis data dan Sistem Manajemen Basis Data (SMBD). Komponen-komponen sistem basis data meliputi :

1. Perangkat keras (*hardware*) sebagai pendukung operasi pengolahan data. Perangkat keras komputer adalah semua bagian fisik komputer. Contoh dari perangkat keras komputer yaitu : mouse, keyboard, monitor, CPU, memori dan lain-lain.
2. Sistem operasi (*operating system*) atau perangkat lunak untuk mengelola basis data. Sistem operasi merupakan suatu *software* sistem yang bertugas untuk melakukan kontrol dan manajemen *hardware* serta operasi-operasi dasar sistem, termasuk menjalankan *software* aplikasi seperti program-program pengolah kata dan *browser web*. Secara umum sistem operasi adalah *software* pada lapisan pertama yang ditaruh pada memori komputer pada saat komputer dinyalakan. Sementara *software-software* lainnya dijalankan setelah sistem operasi berjalan, dan sistem operasi akan melakukan layanan inti umum untuk *software-software* itu. Layanan inti umum tersebut seperti akses ke *disk*, manajemen memori, *schedulling task*, dan antar muka *user*. Dengan demikian masing-masing *software* tidak perlu lagi melakukan tugas-tugas inti umum tersebut, karena dapat dilayani dan dilakukan oleh sistem operasi. Contoh dari sistem operasi yang ada sekarang ini, yaitu DOS, Windows 98, Windows XP, Windows 2000, Windows NT, Linux, Macintosh dan lain-lain.
3. *Database Management Sistem* (DBMS). DBMS adalah *software* yang menangani semua akses ke basis data. Secara konsep apa yang terjadi adalah sebagai berikut :

a. *User* melakukan pengaksesan basis data untuk informasi yang diperlukannya menggunakan suatu bahasa manipulasi data, biasanya disebut SQL.
b. DBMS menerima *request* dari *user* dan menganalisa *request* tersebut.
c. DBMS menerima skema eksternal *user*, pemetaan eksternal/konseptual, skema konseptual, pemetaan konseptual/internal dan struktur penyimpanan.
d. DBMS mengeksekusi operasi-operasi yang diperlukan untuk memenuhi permintaan *user*. Contoh dari DBMS ini yaitu antara lain *Microsoft SQL, Server 2000, Oracle, MySQL, Interbase, Paradox, Microsftt Access* dan lain-lain.

4. Pemakai (*user).* Pemakai merupakan orang atau sistem yang akan mengakses dan merubah isi basis data. Beberapa jenis pengguna basis data yaitu :
   a. *Programmer* aplikasi : orang yang mengodekan aplikasi dengan bahasa pemrograman.
   b. *User* mahir : orang yang mampu menggunakan basis data secara langsung dengan menggunakan DBMS.
   c. *User* umum/*end user* : orang yang memakai basis data dengan menggunakkan perantara program aplikasi. Misalnya seorang kasir memasukkan data penjualan kedalam basis data dengan menggunakan aplikasi kasir.
   d. *User* khusus : bisa berupa sistem lain.

5. Aplikasi lain.

   Aplikasi lain merupakan *software* yang dibuat untuk memberikan *interface* kepada *user* sehingga lebih mudah dan terkontrol dalam mengakses basis data. Aplikasi lain ini merupakan komponen tambahan dalam sistem basis data yang sifatnya opsional.

2.6 Flow Map

Flow map merupakan diagram alir yang menunjukkan arus bagi dokumen, aliran data fisik entitas sistem informasi dan kegiatan operasi yang berhubungan dengan sistem informasi. Penggambarannya diawali dengan mengamati dokumen tersebut, proses apa yang terjadi terhadap dokumen tersebut dan seterusnya.

2.7 Model-model Data

Yang dimaksud dengan model data adalah sekumpulan *tool* konseptual untuk mendeskripsikan data, relasi-relasi antar data, *semantic* data dan *konsistensi konstrain*. Menyatakan hubungan antar rekaman yang tersimpan dalam basis data. Bermacam-macam model data terbagi dalam dua kelompok besar yaitu *object-based logical models* dan *record-based logical models*.

1. *Object-based logical models*

   Yang termasuk dalam kelompok ini diantaranya adalah *Entity Relationship* (ER) model dan *object-oriented model.*

   a. *Entity Relationship Model*

E-R model didasarkan atas persepsi terhadap dunia nyata yang terdiri dari sekumpulan *objek*, disebut *entitas* dan hubungan antar objek tersebut, disebut *relasi. Entitas* adalah *objek* didunia yang bersifat unik. Setiap *entitas* mempunyai *atribut* yang membedakannya dengan *entitas* lainnya. Contoh : *entitas* mahasiswa mempunyai *atribut* nim, nama, alamat dan tanggal lahir. Pemodelan data dengan model E-R menggunakan diagram E-R. Diagram E-R terdiri dari :

1. Kotak persegi panjang, menggambarkan himpunan *entitas.*
2. *Ellips*, menggambarkan *atribut-atribut entitas*.
3. *Diamond,* menggambarkan hubungan antara himpunan *entitas.*
4. Garis, yang menghubungkan antar objek dalam diagram E-R.

b. Object Oriented Model

Model berorientasi *objek* berbasiskan kumpulan *objek*. Setiap *objek* berisi :

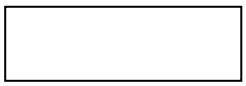
1. Nilai yang disimpan dalam *variable instant*, dimana *variable* "melekat" dengan *objek* itu sendiri.
2. Metode, operasi yang berlaku pada *objek* yang bersangkutan.
3. *Objek-objek* yang memiliki tipe nilai dan metode yang dikelompokkan dalam satu kelas. Kelas disini mirip dengan *abstrak* pada bahasa pemrograman.
4. *Sending message*, sebuah *objek* dapat mengakses data yang lain hanya dengan memanggil metode dari *objek* tersebut. Selain kedua model di atas, yang termasuk dalam kelompok *object-based logical model* adalah *semantic data model* dan *functional data model*.

2.8 Konsep *Entity Relationship Diagram* (ERD)

Perancangan basis data dengan menggunakan model *Entity Relationship* adalah dengan menggunakan *Entity Relationship Diagram* (ERD). Terdapat tiga notasi dasar yang bekerja pada model E-R yaitu : *entity sets, relationship sets* dan *attributes*. Sebuah *entity* adalah sebuah "benda" (*thing)* atau "*objek*" (*object*) di dunia nyata yang dapat dibedakan dari semua *objek* lainnya. *Entity sets* adalah sekumpulan *entity* yang mempunyai tipe yang sama. Kesamaan tipe ini dapat dilihat dari *atribut/property* yang dimiliki oleh setiap *entity* misalnya :

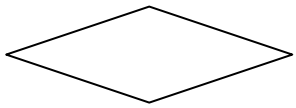
a. Kumpulan orang yang menyimpan uang pada suatu bank dapat didefinisikan sebagai *entity set* nasabah.
b. Kumpulan orang yang belajar di perguruan tinggi didefinisikan sebagai mahasiswa.

Entity set dilambangkan dengan bentuk persegi panjang, seperti pada gambar 2.3.



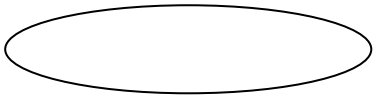
Gambar 2.3 Lambang *Entity Set*

Relationship adalah hubungan diantara beberapa *entity*. *Relationship set* adalah sekumpulan relasi yang mempunyai tipe yang sama. *Relationship set* digambarkan dengan *diamond* seperti tampak pada gambar 2.4.



Gambar 2.4 Lambang *Relationship Set*

Atribut merupakan sebutan untuk mewakili suatu *entity*. *Atribut* dalam ERD dilambangkan dengan bentuk *elips* seperti tampak pada gambar 2.5.



Gambar 2.5 Lambang *Atribut*

Ada beberapa istilah yang harus dipahami sebelum merancang basis data dengan menggunakan model *Entity Relationship* diagram, diantaranya yaitu :

1. *Superkey*

   Satu atau lebih *atribut* (*kumpulan atribut*) yang dapat membedakan setiap baris data dalam tabel secara unik. Sebagai ilustrasi diberikan contoh pada tabel 2.1.

Tabel 2.1 Tabel Mahasiswa

Nim	**Nama**	**Alamat**	**Jurusan**
1001	Rita	Yogyakarta	MI
1002	Aini		MI
1003	Rita	Magelang	SI
1004	Iko		TI

Dari tabel tersebut yang mungkin menjadi *superkey* adalah *atribut* Nim, *atribut* Nim dan Nama, *atribut* nim dan alamat, *atribut* nim dan jurusan, *atribut* nim, nama, alamat dan jurusan. *Atribut* nama tidak bisa menjadi *superkey* karena memungkinkan lebih dari satu baris yang memiliki nilai sama. Demikian pula

atribut alamat, *atribut* jurusan *atribut* nama dan alamat. *Atribut* nama dan jurusan, *atribut* alamat dan jurusan serta *atribut* nama, alamat dan jurusan.

2. *Kandidat Key*

   Kandidat key merupakan kumpulan *atribut* minimal yang membedakan setiap baris data dalam tabel secara unik. Untuk bisa menjadi *kandidat key*, suatu *atribut* harus memenuhi persyaratan sebagai berikut :

   a. Untuk satu nilai hanya mengidentifikasikan satu baris dalam satu relasi (unik)
   b. Tidak memiliki *subset* yang juga merupakan kunci relasi
   c. Tidak dapat bernilai *null.*

   Dengan kata lain, sebuah *kandidat key* adalah *superkey* yang paling sedikit jumlah *atributnya*. Dari contoh pada *superkey* dari tabel mahasiswa pada tabel 2.1 yang memenuhi persyaratan sebagai *kandidat key* adalah *atribut* nim.

3. Kunci Relasi (*Relation Keys*)/Kunci Utama (*Primary Key*)

   Nilai dari kunci relasi harus mengidentifikasikan sebuah baris yang unik di dalam sebuah relasi. Kunci relasi terdiri dari satu atau lebih *atribut-atribut* relasi. Agar bisa menjadi sebuah *primery key* sebuah *atribut* haruslah memenuhi persyaratan sebagai *kandidat key*. Dalam contoh tabel mahasiswa pada tabel 2.1 nim adalah kunci relasinya.

4. Kunci Alternatif (*Alternate Key*)

   Kunci yang tidak ada di dunia nyata, tetapi diadakan dan dijadikan *primery key*. Kunci *alternative* dibuat ketika tidak ada satupun *atribut* dalam sebuah relasi yang bisa mewakili relasi tersebut, atau ada yang bisa menjadi *kandidat key* tetapi tidak

cukup efektif untuk digunakan sebagai *primery key*. Sebagai contoh berikut ini adalah sebuah relasi *hobby* seperti pada tabel 2.2.

Tabel 2.2 Tabel *Hobby*

Nama	**Keterangan**
Hiking	
Climbing	Suka naik gunung itu lho
Renang	

Dari tabel 2.2 *kandidat key* pada tabel tersebut adalah nama. Tetapi untuk lebih efektif mungkin saja perancangan basis data akan memberikan kunci *alternative* berupa *kode_hobby*. Kode tersebut tidak pernah ada dalam dunia nyata karena kita tidak pernah tahu kalau *hiking* itu kodenya 1, atau renang itu kodenya R. tetapi kode ini diadakan dan dipilih sebagai kunci utama, maka kode_nilai disebut sebagai kunci *alternative.*

5. *Komposit Key*

   Primery key yang terdiri dari lebih dari 1 *atribut.*

6. *Foreign Key* (FK)

   Istilah FK juga banyak digunakan dalam perancangan. Sebuah FK adalah sekumpulan *atribut* dalam suatu relasi (misal A) sedemikian sehingga kumpulan *atribut* ini bukan kumpulan relasi A tetapi merupakan kunci dari relasi lain.

7. Kardinalitas Pemetaan

Kardinalitas pemetaan atau rasio kardinalitas menunjukkan jumlah *entity* yang dihubungkan ke satu *entity* lain dengan suatu *relationship sets*. Kardinalitas pemetaan meliputi :

a. Hubungan satu ke satu (*one to one*), yaitu suatu *entity* dalam A dihubungkan dengan maksimum satu *entity*.
b. Hubungan satu ke banyak (*one to many*) yaitu satu *entity* dalam A dihubungkan sejumlah *entity* dalam *entity* dalam B dihubungkan dengan maksimum satu *entity* dalam A.
c. Hubungan banyak ke satu (*many to one*) yaitu satu *entity* dalam A dihubungkan dengan maksimum satu *entity* B. Satu *entity* dalam B dapat dihubungkan dengan sejumlah *entity* dalam A.
d. Hubungan banyak ke banyak (*many to many*). Suatu *entity* dalam A dihubungkan dengan sejumlah *entity* dalam *entity* dalam B dihubungkan dengan sejumlah *entity* dalam A.

2.9 Tahap Pembuatan *Entity Relationship Diagram* (ERD)

Secara umum ada dua langkah dalam membuat ERD. Langkah pertama adalah membuat ERD awal untuk mendapatkan sebuah rancangan *database* yang minimal dapat mengakomodasi kebutuhan penyimpanan data terhadap sistem yang sedang kita tinjau. Setelah itu dapat dilakukan optimasi diagram E-R (*final design*) dengan mempertimbangkan *anomaly-anomaly* dan aspek-aspek efisiensi, kinerja dan *fleksibilitas*. Berikut akan dijelaskan tahap-tahap pembuatan ERD

awal. Untuk lebih memudahkan ilustrasi, akan diberikan kasus perancangan basis data dalam sistem informasi akademik. Adapun langkah-langkahnya adalah sebagai berikut :

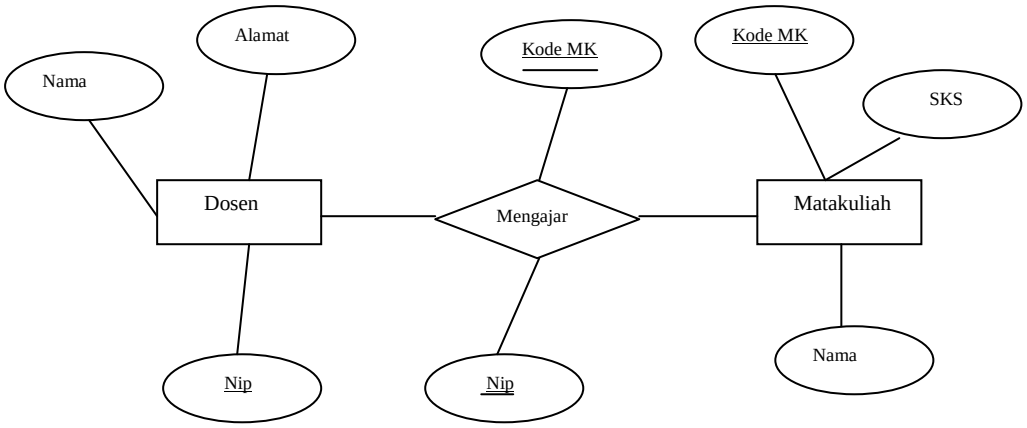
1. Mengidentifikasi dan menetapkan seluruh himpunan *entitas* (*entity)* yang akan terlihat. Misalnya *entitas* yang terlihat adalah mahasiswa, dosen, matakuliah.
2. Menentukan *atribut-atribut key* dari masing-masing himpunan *entitas.* Untuk *entitas* mahasiswa *atribut* yang terlibat yaitu : Nim, nama, alamat dan jurusan. Untuk *entitas* dosen *atribut* yang trlibat adalah : Nip, nama, alamat. Untuk *entitas* matakuliah, *atribut* yang terlibat adalah : Kode_matakuliah, nama, SKS.
3. Menentukan PK dari setiap *entitas* yang ada

   PK dari *entitas* mahasiswa : Nim.

   PK dari *entitas* dosen : Nip.

   PK dari *entitas* matakuliah : Kode_matakuliah.
4. Mengidentifikasi dan menetapkan seluruh himpunan relasi diantara himpunan *entitas* himpunan *entitas* yang ada beserta *foreign keynya.*

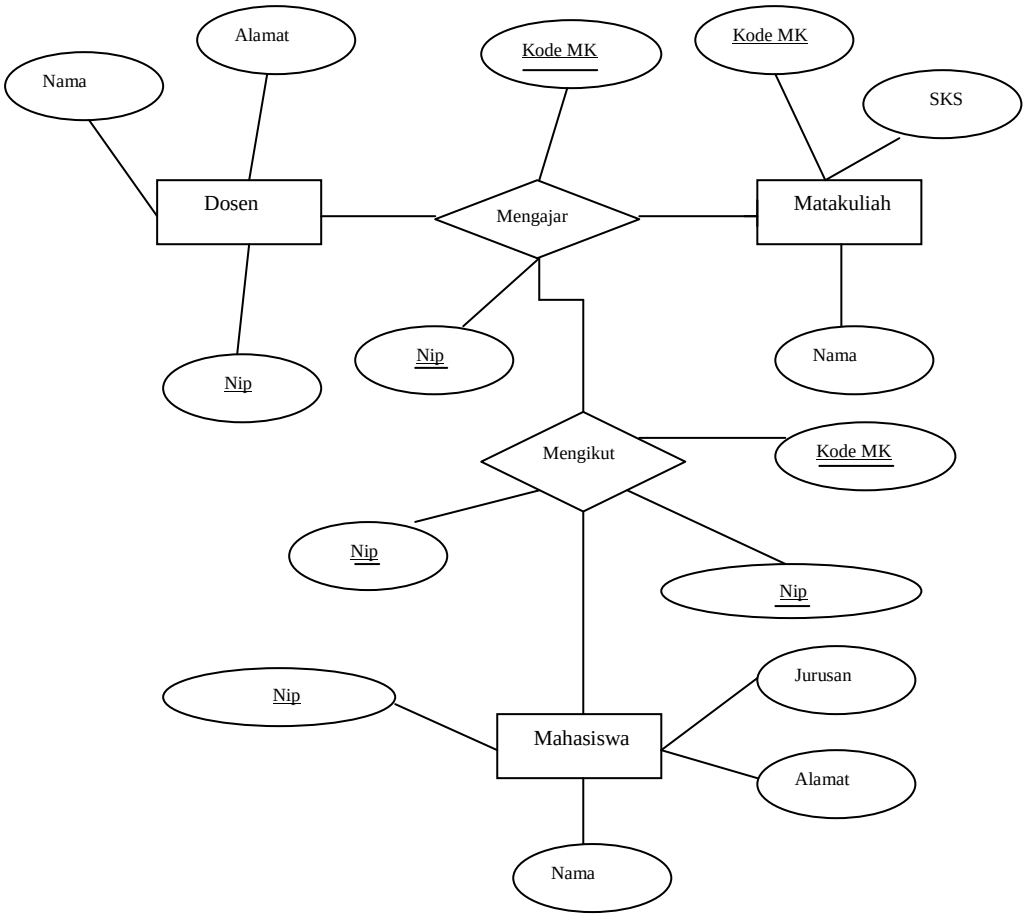
   Dosen dengan mata kuliah berelasi secara langsung dengan relasi mengajar yaitu dosen mengajar matakuliah bisa dilihat pada gambar 2.6.



Gambar 2.6 Contoh Relasi Mengajar

[Sumber : STMIK Amikom Yogyakarta, penerbit Andi]

Atribut pada gambar 2.6 ada beberapa *atribut* yang digarisbawahi yaitu *atribut* nip, pada *entity* dosen dan kode_matakuliah pada entity matakuliah. *Atribut-atribut* tersebut merupakan *primery key* dari *entity*. Sedangkan *atribut* yang diberi garis bawah ganda seperti *atribut* Nip dan kode_matakuliah pada relasi mengajar merupakan *foreign key*. Selain dosen dan matakuliah yang dihubungkan dengan relasi mengajar masih ada satu *entity* yang belum berelasi dalam sistem ini yaitu *entity* mahasiswa. *Entity* mahasiswa berhubungan dengan relasi matakuliah sekaligus berelasi dengan dosen, tetape relasi antara mahasiswa dengan kedua *entity* yang lain saling berkaitan (bersamaan). Mahasiswa mengikuti suatu matakuliah yang diajarkan oleh seorang dosen. Relasi seperti ini disebut dengan relasi agregasi. Dimana sebuah *entity* berelasi dengan yang lain. Jika digambarkan akan tampak seperti pada gambar 2.7.

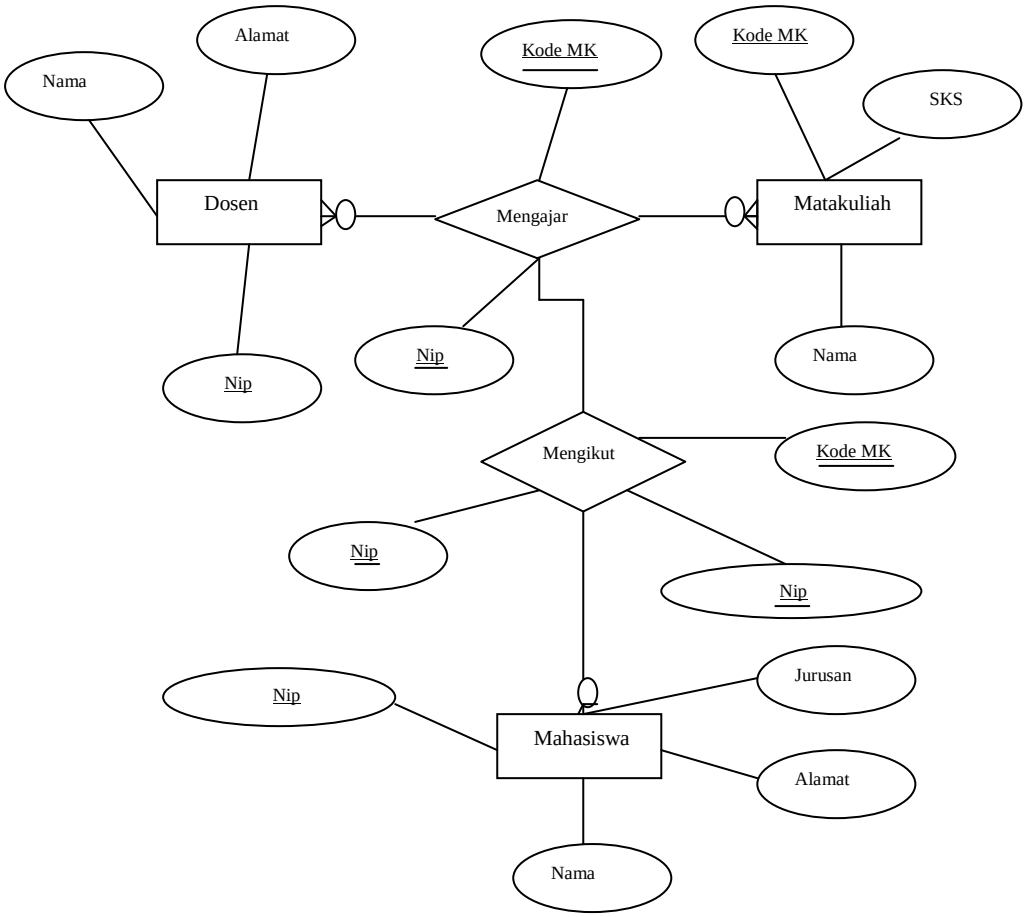


Gambar 2.7 Contoh Relasi Agregasi

[Sumber : STMIK Amikom Yogyakarta, penerbit Andi]

5. Menentukan derajat/kardinalitas relasi untuk setiap himpunan relasi

   Relasi antara dosen dengan matakuliah memiliki kardinalitas banyak ke banyak atau *many to many*, dimana satu dosen dapat mengajar banyak matakuliah dan satu matakuliah bisa diajarkan oleh banyak dosen. Demikian pula relasi antara mahasiswa dengan relasi mengajar yang menghubungkan antara *entity* dosen dengan matakuliah, memiliki kardinalitas banyak ke banyak. Berikut ini adalah gambar ERD setelah ditentukan derajat kardinalitasnya seperti pada gambar 2.8.



Gambar 2.8 ERD Dengan Derajat Kardinalitas

[Sumber : STMIK Amikom Yogyakarta, penerbit Andi]

Penggambaran ERD dapat pula dilengkapi dengan kamus data sehingga tidak perlu semua digambarkan atributnya dengan gambar ERDnya. Cara penulisan kamus data yaitu dengan menyebutkan *entity* diikuti dengan atribut-atributnya. Misalnya *entity* dosen akan ditulis sebagai berikut :

Dosen : {NIP, Nama,Alamat}

Dalam kamus data tersebut Nip diberi garis bawah untuk menandakan bahwa Nip merupakan *primery key* dari *entity* dosen. Setelah ERD awal lengkap dengan

kardinalitasnya selesai dibuat, langkah selanjutnya yang perlu dipikirkan adalah optimasi ERD. Optimasi dilakukan untuk tiga tujuan yaitu :

1. Efisiensi
2. Kinerja
3. Fleksibilitas.

Untuk meningkatkan efisiensi dan kinerja dalam optimasi ERD terkadang diperlukan pemberian kunci alternatif (*alternate key*) yaitu *atribut* yang sebenarnya tidak ada didunia nyata, tetapi ditambahkan dan dijadikan sebagai *primery key*. Sebagai contoh dalam relasi *hobby*, dimana dalam relasi tersebut hanya terdiri dari nama_*hobby*. Mengingat panjang data nama *hobby* cukup banyak, terkadang perlu dipertimbangkan untuk menambahkan kode_*hobby* yang sebenarnya hal ini tidak ada dalam dunia nyata. Dalam hal ini ada tiga bentuk pengodean yang bisa diberikan yaitu :

1. Sekuensial, jenis kode ini diberikan secara urut misalnya untuk kode pembelian terurut dari 1,2,3 dan seterusnya.
2. *Mnemonic*, jenis kode ini berupa karakter atau angka sebagai simbol, misalnya L dan P pada jenis kelamin yang artinya laki-laki dan perempuan.
3. Blok jenis pengodean ini terdiri dari beberapa bagian, misalnya Nim 02.11.2341. kode tersebut memiliki tiga bagian yang masing-masing memiliki makna tersendiri. Blok pertama bermakna tahun angkatan 2002. Blok kedua berisi 11 bermakna kode jurusan yang mewakili Teknik Informatika, sementara blok terakhir 2341 menggambarkan nomor urutan mahasiswa pada tahun angkatan dan jurusan tersebut.

Hal lain yang bisa dilakukan dalam usaha *optimasi* adalah melakukan dekomposisi atau pemecahan tabel. Dalam hal peningkatan *fleksibilitas, optimasi* dilakukan dengan cara berikut :

a. Penambahan *atribut*. Dengan ditambahkan *atribut* pada suatu *entity*/relasi diharapkan *entity*/relasi ini akan lebih *fleksibel* penggunaannya. Sebagai contoh relasi mengajar jika hanya memiliki *atribut* Nip dan kode_matakuliah berarti relasi ini hanya berlaku untuk satu periode waktu saja, namun jika ditambahkan *atribut* tahun dan semester maka relasi ini dapat menampung data untuk beberapa periode waktu.
b. Pemilihan domain *atribut* yang lebih luas. Sebagai contoh adalah pemilihan lebar data untuk *atribut* tahun. Sebelum tahun 2000, banyak sistem yang dibangun dengan rancangan lebar data untuk *atribut* tahun adalah dua digit. Dengan demikian untuk tahun 1982 hanya ditulis 82, untuk tahun 1998 hanya ditulis 98. Pada awalnya hal ini tidak bermasalah, tetapi ketika menginjak tahun 2000 terjadi kekacauan pada banyak sistem karena pencatatan tahun menjadi 00. Hal ini mempengaruhi perhitungan selisih alokasi waktu. Hal ini tidak akan terjadi jika tahun diberikan alokasi empat digit.
c. Generasi. Sebuah contoh kasus generasi adalah sebagai berikut. Diberikan dua buah *entitas* dosen dan mahasiswa sebagai anggota dalam sistem perpustakaan. Masing-masing dari kedua *entitas* tersebut memiliki *atribut IdAnggota*, nama dan alamat. Jika masing-masing jenis anggota direpresentasikan dalam sebuah relasi, maka jika ada penambahan jenis anggota misalnya karyawan, maka harus ditambahkan relasi baru. Tetapi jika dilakukan generasi, seluruh relasi

dari anggota perpustakaan dijadikan sebuah relasi anggota dengan *atribut IdAnggota,* nama, alamat dan ditambahkan dengan jenis.

d. Perubahan struktur *entitas* dari yang berorientasi kolom baris ke orientasi baris. Sebagai contoh sebuah relasi sirkulasi pada sistem perpustakaan yang berbentuk, seperti tampak pada tabel 2.3.

Tabel 2.3 Relasi Sirkulasi Berorientasi Kolom

Id_Ang	**TglPinjam**	**IdBuku1**	**IdBuku2**
1	1/1 06	A01	B54
3	3/2 06	D34	A43
…			

Relasi pada tabel 2.3 di atas berlaku untuk perpustakaan yang memiliki kebijakan meminjamkan buku dua eksemplar untuk tiap peminjaman yang terjadi. Untuk perpustakaan yang memiliki kebijakan lain, relasi ini tidak dapat digunakan. Atau untuk perpustakaan yang sama, pihak manajemen tidak diijinkan untuk mengubah kebijakan. Lain halnya jika relasi tersebut diubah dalam bentuk seperti tampak pada tabel 2.4.

Tabel 2.4 Relasi Sirkulasi Berorientasi Baris

Id_Ang	**TglPinjam**	**IdBuku**
1	1/1 06	AO1
1	1/1 06	B54
3	3/3 06	D34
3	3/2 06	A43
…		

2.10 Tahap Implementasi

Tahap implementasi merupakan upaya untuk membangun basis data fisik yang ditempatkan dalam *memori* sekunder. Adapun langkah-langkah untuk mengimplementasikan *Entity Relationship Diagram* (ERD) adalah sebagai berikut:

a. Transformasi dari model data yang telah dibuat ke skema basis data sesuai dengan DBMS yang dipilih.
b. Diagram ER → basis data
c. *Entity* → tabel-tabel/*file-file* data
d. *Atribut* → *field*.

Transformasi dasar :

1. Setiap himpunan *entitas* akan diimplementasikan sebagai sebuah tabel.
2. Relasi dengan derajat 1-1 yang menghubungkan dua buah himpunan *entitas* akan direpresentasikan dalam bentuk penambahan *atribut-atribut* relasi kesalah satu tabel yang mewakili kedua himpunan *entitas.*
3. Relasi dengan derajat 1-N yang menghubungkan dua buah himpunan *entitas* juga akan direpresentasikan dalam bentuk perancangan atau pemberian *atribut key* dalam himpunan *entitas* pertama (yang berderajat 1) ke himpunan kedua (yang berderajat N).
4. Relasi dengan derajat N-N yang menghubungkan dua himpunan *entitas* akan di fujudkan dalam bentuk tabel khusus, yang memiliki *field* tepatnya (*foreign key)* yang berasal dari *key-key* kedua himpunan *entitas* yang dihubungkan.

5. Relasi tunggal

   a. Derajat 1-1

      Penggunaan *field key* dua kali pada satu tabel 1 dengan nama dan fungsi berbeda.

   b. Derajat 1-N

      Dibentuk tabel baru yang mendapatkan *field* dari *atribut* relasi yang ditambah dengan *atribut key* dari himpunan *entitasnya*. Sebagai contoh akan diberikan ER diagram seperti pada gambar 2.9.



Gambar 2.9 (ERD) *Entity Relationship Diagram*

Adapun langkah implementasinya adalah sebagai berikut :

1. Mengubah *entitas* menjadi tabel dengan *fieldnya* adalah *atribut* dari *entitas* tersebut. Dengan demikian tabel yang terbentuk pada tahap ini adalah :
   a. Tabel dosen, *field-fieldnya* : Nip, Nama, Alamat.

b. Tabel matakuliah, *field-fieldnya* : Kode_MK, Nama, SKS.

c. Tabel mahasiswa, *field-fieldnya* : Nim, Nama, Alamat.

d. Tabel jurusan, *field-fieldnya* : Kd_Jurusan, Nama.

2. Mengubah relasi antar *entitas* dengan derajat kardinalitas banyak ke banyak menjadi tabel baru. Dari gambar 2.6 relasi yang derajat kardinalitasnya banyak ke banyak adalah :

   a. Relasi mengajar, yang menghubungkan dosen dengan matakuliah. Relasi ini dirubah menjadi tabel mengajar dengan *field* : Kode_MK, kode_MK_prasyarat.

   b. Relasi prasyarat, yang menghubungkan antara matakuliah dengan matakuliah. Relasi ini diubah menjadi tabel persyarat dengan field : Nim, Kode_MK-Prasyarat.

   c. Relasi mengikuti, yang menghubungkan antara mahasiswa dengan relasi mengajar. Relasi ini diubah menjadi tabel mengikuti dengan *field* : Nim, Kode_MK, Nip.

Menambahkan *field* pada *entity* yang dihubungkan oleh relasi dengan derajat satu kebanyak. Dari gambar 2.6 relasi yang berderajat satu ke berderajat banyak adalah relasi memiliki yang menghubungkan antara jurusan dengan mahasiswa. Relasi ini mengakibatkan penambahan *field* Kd_ jurusan sebagai *primery key* dari *entity* jurusan yang berderajat satu ke *entity* mahasiswa yang berderajat banyak. Dengan demikian tabel mahasiswa memiliki *field* : Nim. Nama, Alamat dan Kd_jurusan.

2.11 Metode Perancangan Sistem

Metode perancangan sistem terdiri dari diagram konteks, DFD dan ERD.

Pemodelan analisis representasi teknis pertama dari simbol, yang terdiri atas :

1. Analisis terstruktur atau metode pemodelan klasik adalah aktivitas pembangunan model dengan menggunakan notasi yang sesuai dengan prinsip analisis operasional yang dapat menciptakan model yang menggambarkan muatan dan aliran informasi (*data control*), membagi sistem secara fungsional dan menggambarkan esensi dari apa yang harus dibangun.
2. *Entity Relationship Diagram* (ERD) adalah notasi yang digunakan untuk melakukan aktivitas pemodelan data.
3. *Data Flow Diagram* (DFD) memberikan informasi tambahan yang digunakan selama analisis *domain* informasi dan fungsi sebagai dasar dari pemodelan fungsi.

2.11.1 Diagram Konteks

Diagram konteks (*context diagram*) menggambarkan hubungan antara sistem dengan entitas luarnya. Diagram konteks berfungsi sebagai transformasi dari suatu proses yang melakukan transformasi dari *input* menjadi data *output*. Entitas yang dimaksud adalah entitas yang mempunyai hubungan langsung dengan sistem. Suatu diagram konteks selalu mengandung satu dan hanya satu proses saja. Proses ini mewakili proses dari seluruh sistem. Diagram konteks ini menggambarkan hubungan *inpu*t atau *output* antara sistem dengan dunia luarnya (kesatuan luar).

2.11.2 *Data Flow Diagram* (DFD)

Data Flow Diagram (DFD) atau Diagram Aliran Data (DAD) memperlihatkan hubungan fungsional dari nilai yang dihitung oleh sistem termasuk nilai masukan, keluaran serta tempat penyimpanan internal Diagram Aliran Data (DAD) merupakan gambaran garis yang memperlihatkan aliran data dari sumbernya dengan objek kemudian melewati proses yang mentransformasinya ke tujuan yang lain. *Data Flow Diagram* (DFD) digunakan pada metode pengembangan sistem yang terstruktur (*structured analysis and design*). Simbol DFD adalah sebagai berikut :

1. *External entity* (kesatuan luar)/*bondary* (batas sistem) setiap sistem mempunyai *bondary* yang memisahkan suatu sistem dengan lingkaran luar. Sistem menerima *input* dan mengeluarkan *output* pada lingkungan luarnya. *External entity* merupakan kesatuan di lingkungan luar sistem yang dapat berupa orang, organisasi atau sistem lain di lingkungan luar yang akan memberi *input* atau menerima *output* dari sistem.
2. *Data Flow Diagram* (DFD) diberi simbol panah yang mengalir diantara proses simpanan data (*data store*) dan kesatuan luar (*external entity*).
3. Proses adalah kegiatan yang dilakukan oleh orang, mesin atau komputer dari hasil suatu arus data yang masuk kedalam proses untuk dihasilkan arus data yang akan keluar dari proses. Setiap proses harus diberikan penjelasan lengkap meliputi identifikasi proses dan pemroses.

4. *Data store* (simpanan) adalah simpanan dari data yang dapat berupa *file database* dari sistem komputer, suatu arsip atau catatan manual, suatu nilai acuan manual dan suatu agenda atau buku.

2.12 Kamus Data/*Data Dictionary* (DD)

Kamus data adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan kamus data analisis sistem dapat mendefinisikan data yang mengalir dari sistem dengan lengkap. Kamus data dibuat pada tahap analisis sistem digunakan sebagai alat komunikasi antara analisis sistem dengan *user* sedangkan pada tahap perancangan sistem, kamus data digunakan untuk merancang *input* laporan-laporan basis data. Kamus data dibuat berdasarkan arus data yang ada di *Data Flow Diagram* (DFD) yang sifatnya global. Kamus data harus dapat mencerminkan keterangan tentang data yang dicatat, maka kamus data harus memuat nama arus data, alias, bentuk data, data penjelasan, periode, volume dan struktur data seperti pada tabel 2.5.

Tabel 2.5 Notasi Kamus Data

No	Notasi	Arti Notasi Kamus Data
1	=	Artinya terbentuk dari (*is composed*) atau terdiri dari (*consist of)*
2	+	And (dan)
3	[ ]	Salah satu dari/memilih salah satu dari elemen-elemen data dari dalam kurung berikut
4	[	Sama dengan simbol “[ ]” pemisah pilihan dari tanda [ ]
5	N {} M	Iterasi mulai dari nilai minimum N kali dan nilai maksimum N kali
6	*	Keterangan setelah tanda ini komentar
7	@	Petunjuk (*key field*)

2.13 Pengenalan Borland Delphi 7.0

Delphi adalah sebuah aplikasi yang digunakan untuk pengembangan dengan memanfaatkan keistimewaan konsep-konsep antarmuka grafis dalam Microsoft Windows. Aplikasi yang dihasilkan Delphi berkait erat dengan Windows itu sendiri sehingga dibutuhkan pengetahuan bagaimana cara kerja dari Windows. Dalam pemrograman visual banyak istilah dan konsep untuk menyebut "sesuatu" yang membentuk sebuah aplikasi. Istilah-istilah tersebut memiliki arti yang sama dalam lingkungan pemrograman visual yang lain, seperti misalnya *objek*, *property*, dan *event*. Disini akan membahas mengenai konsep-konsep pemrograman tersebut beserta pemrograman *event-driven.*

2.13.1 Pemrograman *Event-Driven*

Pemrograman *event drivent* telah ada sebelum GUI (*Graphical User Intereface*) dibuat dan diimplementasikan dalam beberapa cara. Dengan diperkenalkannya *mouse*, pemrograman *event driven* banyak disukai oleh para pemakai dan perancang program. Sebelum ada pemrograman *event driven*, pemrograman *top-down* digunakan untuk membuat kode yang menangani banyak pemrosesan. Selain itu aplikasi yang dirancang dengan teknik ini mudah untuk diperbaiki, disamping kodenya menjadi luwes. Namun aplikasi yang dibuat dengan metode ini biasanya memiliki menu yang erat pada proses yang ada dalam program.

Pemrograman *event driven* tidak mencoba menggantikan pemrograman prosedural, namun melengkapi dengan kerangka yang dapat membedakan antara

user interface dan proses tertentu dalam aplikasi. Delphi dan aplikasi *event driven* lainnya menyediakan kerangka tersebut sehingga *programmer* akan lebih terkonsentrasi pada masalah aplikasinya. Kunci pokok dalam pemrograman *even-driven* adalah bagaimana menentukan *event* yang muncul, namun hanya *event* tertentu yang harus diproses, yaitu yang berhubungan dengan aplikasi.

2.13.2 Memahami *Objek*

Istilah *objek* ini sedikit berubah sejak diperkenalkan pertama kali dalam pemrograman *objek.* Kata *objek* sekarang digunakan lebih bebas karena banyak aplikasi yang menggunakan istilah ini untuk menyatakan diri sebagai aplikasi berorientasi *objek.*

2.13.3 Arti dari *Objek*

Definisi dari sebuah *objek* adalah sesuatu yang dapat melakukan hal-hal tertentu. Di dalam Delphi contoh *objek* adalah *button, label, list box* dan lain-lain. *Objek* adalah komponen-komponen untuk membuat sebuah aplikasi.

2.13.4 Pembuatan *Objek*

Pemrograman visual merupakan dimensi baru dalam pembuatan aplikasi karena dapat langsung menggambarkan *objek-objek* ke layar sebelum dieksekusi. Dalam lingkungan pengembangan *visual,* sekarang *objek* yang dapat anda buat hasilnya langsung tampil di layar. *Objek* yang dibuat itu akan sama hasilnya pada saat program dijalankan. Dengan demikian anda tidak perlu lagi melakukan

pengubahan kode program secara manual. Setelah *objek* diletakkan dalam suatu *form*, maka semua *attribute objek* tersebut akan disimpan sebagai suatu kode program yang dapat langsung dijalankan.

2.13.5 *Property*

Selain memiliki kemampuan untuk menanggapi *event, objek* juga memiliki *property.* Dalam *property* antara lain didapatkan informasi mengenai warna, tinggi, lebar, posisi sebuah *objek* dan lain sebagainya. Nilai pada tiap *property* dapat mempengaruhi cara *objek* ditampilkan atau cara *objek* bekerja. Bagi sebuah *objek* mirip dengan *variable* lokal dalam prosedur. *Property* langsung berkaitan dengan *objek* dalam digunakan oleh proses-proses yang ada dalam *objek.*

2.13.6 *Event*

Semua aplikasi Windows memakai metode *event-driven* untuk mengelola interaksi antar program dan pemakainya. Semua *event* yang muncul ditimbulkan oleh pemakai atau oleh suatu operasi dalam sistem Windows sendiri. Kebanyakan kode program yang akan anda tulis dalam Delphi digunakan untuk menangani *event* dari pemakai atau sistem.

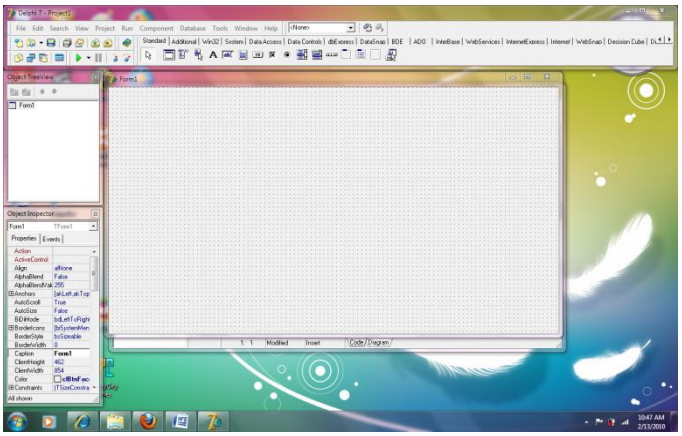
2.13.7 Memulai Delphi

Pada contoh berikut penulisan menggunakan Windows 7.0, jadi untuk memulai menjalankan program Delphi 7.0, dapat dilakukan dengan cara berikut, pilih tombol start, all program, Borland Delphi 7.0, lihat gambar 2.10.



Gambar 2.10 Memulai Delphi

Setelah anda klick Delphi 7.0 dari layar di atas, maka akan ditampilkan layar utama Borland delphi seperti gambar 2.11.



Gambar 2.11 Jendela Utama Delphi 7.0

Setelah Delphi dipanggil, maka pada layar monitor akan ditampilkan empat Windows yaitu:

1. Windows Utama

   Windows Utama merupakan pusat pengaturan di dalam Delphi. Di dalam Windows ini terdapat dua buah elemen yang masing-masing memiliki fungsi khusus, yaitu menu *bar* dan *speed bar*.

2. *Object Inspector*

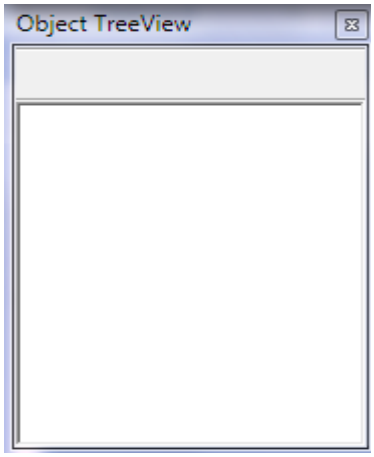
   Jendela *object inspector* ini memiliki dua buah tab yaitu *tab properties* dan tab event. Tab *properties* menampilkan semua properti untuk objek yang terpilih dalam Windows *form*, sedang tab *event* mendata semua *events* yang akan ditanggapi oleh *objek* bersangkutan seperti pada gambar 2.12.



Gambar 2.12. Jendela *Object Inspector*

3. *Object Tree View*

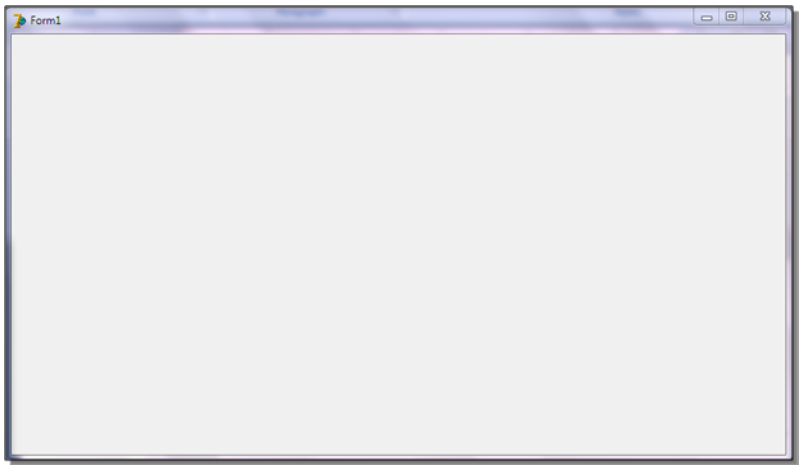
   Object tree view digunakan untuk menampilkan semua komponen dan *object* yang ada pada *form* dalam struktur pohon dan menggambarkan hubungan diantaranya seperti pada gambar 2.13.



Gambar 2.13 Jendela *Object TreeView*

2. Windows *Form*

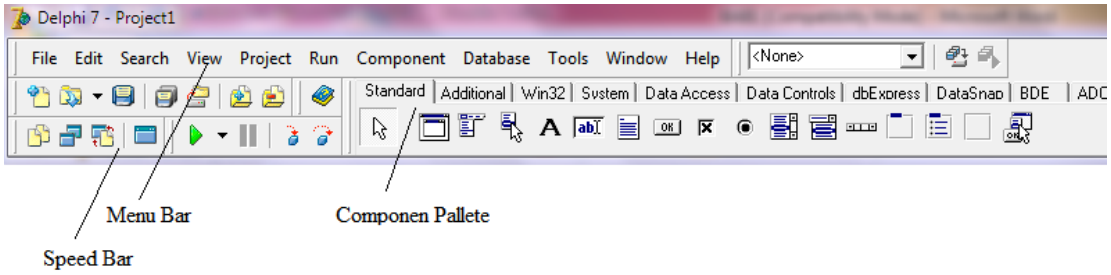
   Ketika pertama memulai Delphi, windows ini biasanya ditampilkan dengan judul "*Form* 1" dan terletak di bagian kanan windows editor program. *Form* adalah area dimana nanti anda merancang windows bagi aplikasi baru yang sedang dibuat. Sebuah aplikasi dapat berisi beberapa *form* dan minimal harus memiliki sebuah *form* seperti pada gambar 2.14.



Gambar 2.14 Jendela *Form*

3. Menggunakan Windows Utama

Fungsi-fungsi untuk mengatur proses pengembangan dalam Delphi diakses melalui windows utama, windows utama ini terdiri dari tiga buah elemen seperti pada gambar 2.15.



Gambar 2.15 Menggunakan Windows Utama Menu *Bar*.

c. Menu *Bar*

Menu ini dipakai untuk mengatur semua windows yang ada dalam lingkungan kerja Delphi. Menu ini juga dipakai untuk mengelola proses desain aplikasi, mengatur lingkungan kerja Delphi, serta memberikan fasilitas petunjuk segera (*online-help*).

d. Speed Bar

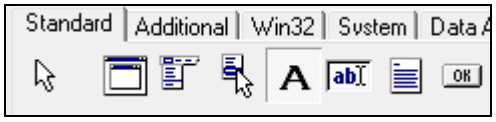
Speed bar muncul di bagian kiri pemisah windows utama, *speed bar* menyediakan akses yang cepat bagi operasi-operasi yang sering digunakan, seperti membuka *file*, menyimpan, *cut*, *paste* dan lain-lain.

e. Component Pallette

Pallete komponen terletak pada bagian kanan *speed bar pallete* komponen ini dipakai untuk memilih *objek* yang akan digunakan dalam pembuatan aplikasi dan kemudian meletakkan *objek* tersebut pada rancangan *form.*

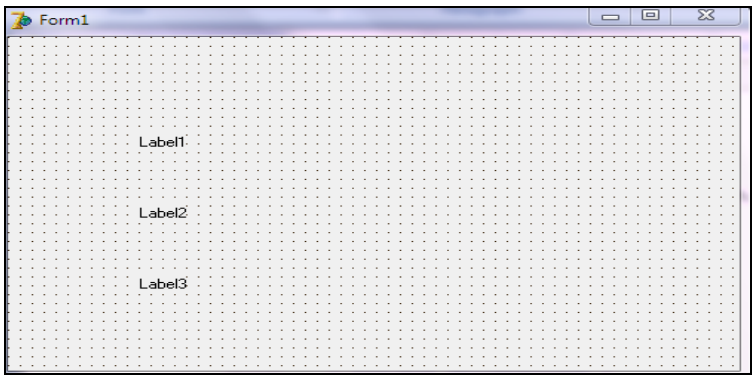
2.12.8. Membuat Program Pertama dengan Delphi

Untuk mencoba pertama kali membuat program dengan Delphi, pada latihan ini akan dicoba bagamana membuat program untuk menampilkan judul-judul sinetron yang sedang beredar saat ini. Komponen yang digunakan adalah komponen standar seperti pada gambar 2.16.



Gambar 2.16 Komponen Standard

Pertama pada jendela *form*1, tempelkan tiga buah label seperti pada gambar 2.17.



Gambar 2.17 Menempatkan Label dalam *Form*

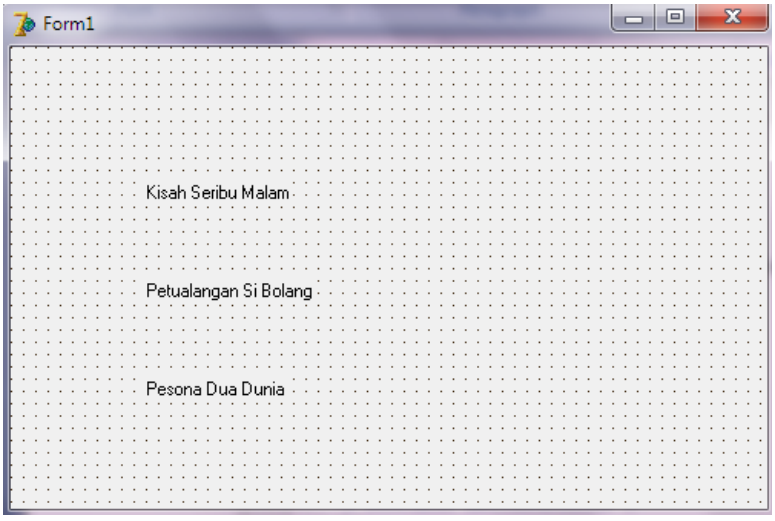
Langkah berikutnya adalah:

1. Click label 1, kemudian pada jendela *object inspector*, ubah *property caption* menjadi “Kisah seribu malam”
2. Ulangi ke langkah 1, untuk merubah *property* lainnya.

Table 2.6 Tabel Label

NO	***Object***	***Caption***
1	Label1	Kisah Seribu Malam
2	Label2	Petualangan Si Bolang
3	Label3	Pesona Dua Dunia

3. Di jendela *form* tulisan tadi akan tampil seperti gambar 2.18.

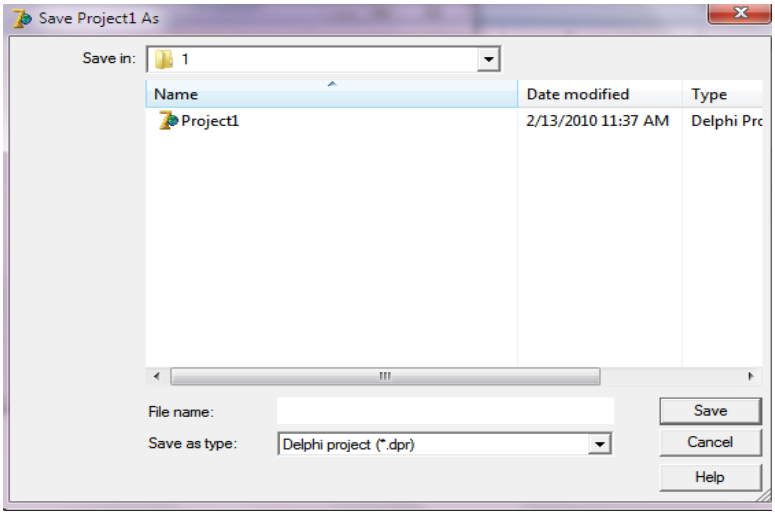


Gambar. 2.18 Label yang Telah Berubah

2.12.9 Menyimpan Program

Jika telah selesai membuat program yang pertama, selanjutnya adalah menyimpannya supaya nanti dapat dibuka kembali. Langkah untuk menyimpan program adalah sebagai berikut:

1. Pilih menu *file*, kemudian pilih *save project as* seperti pada gambar 2.19.
2. Masukkan nama *file* untuk unit dan nama *project* anda kemudian klik *save.*



Gambar 2.19 Menyimpan *Project*

2.12.10 Menjalankan Program

Untuk menjalankan program, dapat dilakukan dengan cara memilih menu *run*, atau dengan menekan tombol F9 seperti pada gambar 2.20.[5].



Gambar 2.20 Hasil *Run*