

BAB III

LANDASAN TEORI

3.1 Pengertian Aplikasi

Aplikasi adalah program siap pakai yang dapat digunakan untuk menjalankan perintah – perintah dari pengguna aplikasi tersebut dengan tujuan mendapatkan hasil yang lebih akurat dan sesuai dengan tujuan pembuatan aplikasi tersebut. Aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu teknik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputansi yang diinginkan atau diharapkan maupun pemrosesan data yang diharapkan. Aplikasi secara umum adalah alat terapan yang difungsikan secara khusus dan terpadu sesuai kemampuan yang dimilikinya. Aplikasi merupakan suatu perangkat komputer yang siap pakai bagi user.

Menurut Jogiyanto (1999:12) adalah penggunaan dalam suatu komputer, instruksi (instruction) atau pernyataan yang disusun sedemikian rupa sehingga dapat memproses input menjadi output.

Menurut Rachmad Hakim S, Aplikasi adalah perangkat lunak yang digunakan untuk tujuan tertentu, seperti mengolah dokumen, mengatur Windows dan permainan (game), dan sebagainya.

Menurut KBBI (1998:52) aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan

bahasa pemrograman tertentu. Aplikasi adalah suatu program komputer yang dibuat untuk mengerjakan dan melaksanakan tugas khusus dari pengguna.

3.2 Pengertian Data

Data adalah suatu yang belum mempunyai arti bagi penerimanya dan masih memerlukan adanya suatu pengolahan. Data bisa berupa suatu keadaan, gambar, suara, huruf, angka, matematika, bahasa maupun simbol – simbol lainnya yang bisa kita gunakan sebagai bahan untuk melihat lingkungan objek, kejadian atau suatu konsep.

Menurut Turban et al. (2005, p38), data adalah deskripsi dasar tentang sesuatu, kejadian, dan transaksi yang ditangkap, direkam, diklasifikasikan, namun tidak terorganisir, untuk menyampaikan suatu arti khusus. Sedangkan menurut McLeod and Schell (2007, p12), data terdiri dari fakta dan gambaran secara umum tidak dapat digunakan oleh user.

3.3 Pengertian Pengolahan Data

Pengolahan data adalah segala macam pengolahan terhadap data atau kombinasi – kombinasi dari berbagai macam pengolahan terhadap data untuk membuat data itu berguna sesuai dengan hasil yang diinginkan dan dapat segera dipakai. Para ahli juga mempunyai pendapatnya masing – masing tentang pengertian data.

Menurut Jogiyanto H.M “Pengolahan Data adalah manipulasi dari data kedalam bentuk yang lebih berguna dan berarti.

Menurut Kristanto (2003:8), pengolahan data adalah waktu yang digunakan untuk menggambarkan perubahan bentuk data menjadi informasi yang memiliki kegunaan.

Sedangkan Menurut George R. Terry, Phd, pengolahan data adalah serangkaian operasi informasi yang direncanakan guna mencapai tujuan atau hasil yang diinginkan. (Martin M. Lipschutz, 1990).

Dengan demikian dapat disimpulkan bahwa “Pengolahan Data merupakan kegiatan yang dilakukan dengan menggunakan masukan berupa data dan menghasilkan informasi yang bermanfaat untuk tujuan sesuai dengan yang direncanakan.

3.4 Microsoft Visual Basic 6.0

Microsoft Visual Basic pada dasarnya adalah sebuah bahasa pemrograman komputer. Bahasa pemrograman adalah perintah-perintah atau instruksi yang dimengerti oleh komputer untuk melakukan tugas-tugas tertentu. *Visual basic* (yang sering juga disebut dengan *VB*) selain disebut sebagai sebuah bahasa pemrograman, juga sering disebut sebagai sarana (*tool*) untuk menghasilkan program-program aplikasi berbasis *windows*.

Beberapa kemampuan atau manfaat dari *Visual Basic* di antaranya seperti :

1. Untuk membuat program aplikasi berbasis *windows*.
2. Untuk membuat objek-objek pembantu program seperti kontrol

ActiveX, file Help, aplikasi *internet*, dan sebagainya.

3. Menguji program (*debugging*) dan menghasilkan program akhir berakhiran *EXE* yang bersifat *executable* atau dapat langsung dijalankan.

Visual basic merupakan sebuah pengembangan terakhir dari bahasa *BASIC*. *BASIC* (*Beginner's All-purpose Symbolic Instruction Code*) adalah sebuah bahasa pemrograman kuno yang merupakan awal dari bahasa-bahasa pemrograman tingkat tinggi lainnya. *BASIC* dirancang tahun 1950-an dan ditujukan untuk dapat digunakan oleh para *programmer* pemula. Biasanya *BASIC* diajarkan untuk para pelajar sekolah menengah yang baru mengenal komputer, serta digunakan untuk mengembangkan program-program cepat saji yang ringan dan menyenangkan. Walaupun begitu, peran *BASIC* lebih dari sekedar itu saja. Banyak para *programmer* andal saat ini memulai karirnya dengan mempelajari *BASIC*. *Visual basic* masih tetap mempertahankan beberapa sintaks atau format penulisan program yang pernah dipakai oleh *BASIC* karena di dalamnya juga sudah mengandung kaidah-kaidah pemrograman yang cukup andal.

Sejak dikembangkan pada tahun 80-an, *Visual Basic* kini telah mencapai versinya yang ke-6. Beberapa keistimewaan utama dari *Visual Basic 6.0* ini diantaranya seperti :

1. Menggunakan *platform* pembuatan program yang diberi nama *Developer Studio*, yang memiliki tampilan dan sarana yang sama dengan *Visual C++* dan *Visual J++*. Dengan begitu dapat bermigrasi

atau belajar bahasa pemrograman lainnya dengan mudah dan cepat tanpa harus belajar dari nol lagi.

2. Memiliki *compiler* andal yang dapat menghasilkan *file executable* yang lebih cepat dan lebih efisien dari sebelumnya.
3. Memiliki beberapa tambahan sarana *Wizard* yang baru. *Wizard* adalah sarana yang mempermudah di dalam pembuatan aplikasi dengan mengotomatisasi tugas-tugas tertentu.
4. Tambahan kontrol-kontrol baru yang lebih canggih serta peningkatan kaidah struktur bahasa *Visual Basic*.
5. Kemampuan membuat *ActiveX* dan fasilitas *internet* yang lebih banyak.
6. Sarana akses data yang lebih cepat dan andal untuk membuat aplikasi *database* yang berkemampuan tinggi.
7. *Visual Basic 6.0* memiliki beberapa versi atau edisi yang disesuaikan dengan kebutuhan pemakainya.

Seperti aplikasi-aplikasi komersil lainnya, *Visual Basic 6.0* juga dipasarkan dalam berbagai jenis atau versi. Beberapa versi dari *Visual Basic 6.0* yang ada di pasaran di antaranya adalah :

1. *Standard Edition/Learning Edition* : ini adalah versi standar yang sudah mencakup berbagai sarana dasar dari *Visual Basic 6.0* untuk mengembangkan aplikasi.
2. *Professional Edition* : versi ini memberikan berbagai sarana ekstra yang dibutuhkan oleh para *programmer profesional*. Misalnya kontrol-kontrol tambahan, dukungan untuk pemrograman *internet*, *compiler* untuk

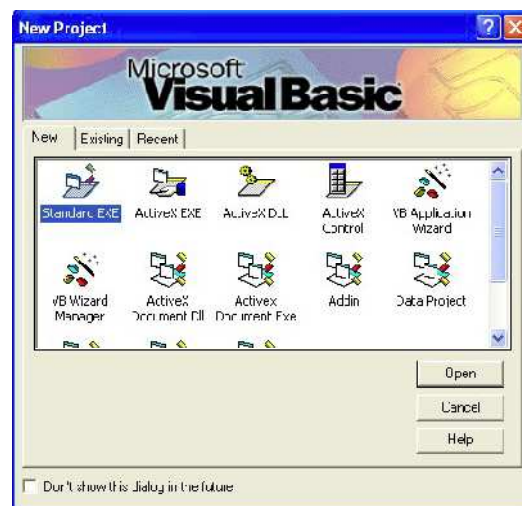
membuat *file Help*, serta sarana pengembangan *database* yang lebih baik.

3. *Enterprise Edition* : versi ini dikhususkan untuk para *programmer* yang ingin mengembangkan aplikasi *remote computing* atau *client/server*. Biasanya versi ini digunakan untuk membuat aplikasi pada jaringan.

3.4.1 Tampilan Layar Visual Basic

Main Windows (Jendela Utama) terdiri dari *title bar* (baris judul), *menu bar*, dan *toolbar*. Jendela utama menampilkan lokasi dari *form* yang aktif. Untuk memulai program baru dapat dilakukan dengan cara :

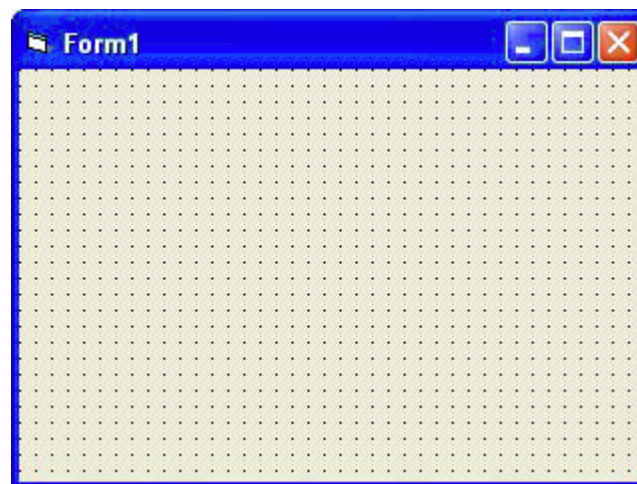
1. Klik *Start, All Program, Microsoft Visual Basic 6.0 lalu Visual Basic 6.0*
2. Dari *New Project*, pilih standard *EXE (Visual Basic IDE)* untuk memulai program baru.



Gambar 3.1. Jendela utama *Visual Basic*

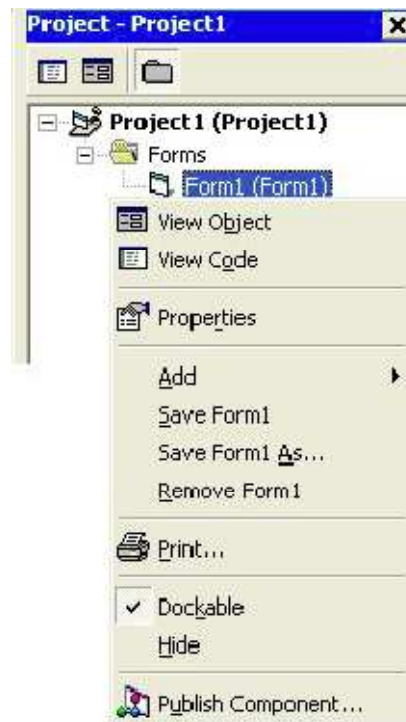
3. Klik tombol *Open*, setelah itu *file* bernama *Project1* dengan sebuah *file* bernama *Form1* akan tampil. Sebuah *file project* dapat menampung beberapa *file form*.

Form Windows (Jendela *Form*) adalah pusat dari pengembangan aplikasi *Visual Basic*. *Form* merupakan bagian dari setiap *file project*. Setiap satu *file* dapat menampung beberapa *file form* (tergantung dari kebutuhan).



Gambar 3.2. Jendela *Form*

Project Windows (Jendela *Proyek*) menampilkan daftar dari *form* dan modul proyek. *Proyek* merupakan kumpulan dari modul *form*, modul *class*, modul standar, dan *file* sumber yang membentuk suatu aplikasi. *Project Windows* disebut juga dengan *Project Explorer* yang berguna untuk melakukan penambahan, perubahan properti, penghapusan, dan penyimpanan sebuah *file project* atau *file form*.



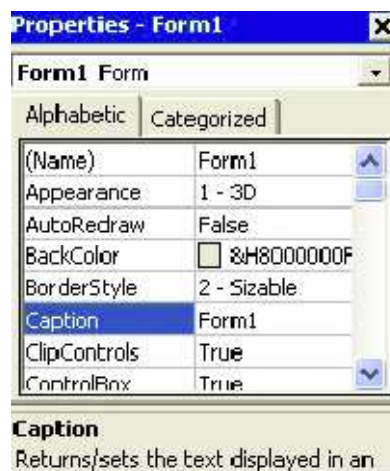
Gambar 3.3. Jendela Proyek



Gambar 3.4. Toolbox

Properties Windows (Jendela Properti) berisi daftar struktur *setting* properti yang digunakan pada sebuah objek terpilih. *Properties* berguna untuk mengatur nama, *caption*, warna, *icon*, *cursor*, dan segala sesuatu yang berhubungan dengan properti dari

form, *button*, *label*, dan lain-lain. Jenis data dari properti ini demikian banyaknya dan berbeda untuk setiap jenis *input* dari masing-masing kontrol.



Gambar 3.5 Jendela *property*

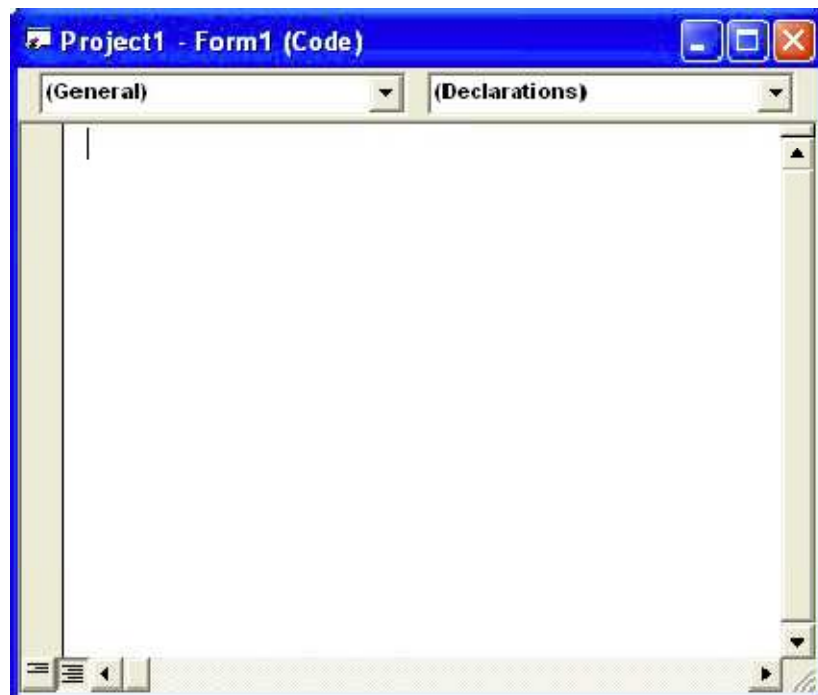
Form Layout Windows (Jendela Layout Form) menampilkan posisi *form* relatif terhadap layar monitor.



Gambar 3.6. Jendela *Layout Form*

Windows Code (Jendela Kode) adalah tempat untuk menuliskan kode program. *Code* adalah sekumpulan baris perintah yang akan dijalankan oleh suatu *event*, yang sering disebut dengan prosedur. Pada *window* ini terdapat fasilitas *editing* yang cukup

lengkap. Jika ingin melakukan klik ganda pada sebuah objek yang berupa kontrol atau *form* maka *windows code* ini akan langsung aktif dan membuka *cursor* ke tempat penulisan program yang terkait dengan objek tersebut. *Event* adalah peristiwa atau kejadian yang diterima oleh suatu objek. *Event* ada banyak sekali dan itu tergantung dari objeknya. Sebuah objek bisa berbeda eventnya dengan objek lain. *Event* tersebut ada bermacam-macam antara lain *click*, *load*, *active*, *keydown*, *keypress*, *change*, *mousemove*, dan lain- lain.



Gambar 3.7. *Windows code*

3.4.2 Variabel dan Konstanta pada *Visual Basic 6.0*

Variabel adalah lokasi di memori komputer tempat *Visual Basic* menyimpan sementara informasi. Aturan penamaan variabel, yaitu :

1. Maksimum 40 karakter. Jika lebih, yang dikenali hanya 40 karakter pertama saja.
2. Tidak boleh ada spasi.
3. Yang diperbolehkan adalah huruf, angka atau garis bawah.
4. Karakter pertama harus huruf.
5. Tidak boleh menggunakan kata kunci, objek, properti, dan metode *Visual Basic*.

Konstanta adalah variabel yang nilainya tidak berubah. Konstanta bekerja seperti variabel tetapi tidak dapat mengubah nilainya pada saat program berjalan. Agar sebuah konstanta dapat dipakai semua prosedur dalam program, sebaiknya dibuat konstanta dalam modul standar dengan kata kunci `public`.

3.4.3 *Data Report*

Data report adalah pelengkap VB untuk membuat laporan. Secara umum langkah pembuatan laporan dengan data report adalah membuat `DataEnvironment`, membuat `DataReport`, menghubungkan `DataReport` dengan `DataEnvironment` melalui properti `DataSource` dan `DataMember`, lalu membuat perintah menampilkan `DataReport`

3.4.4 *Data Environment*

Data Environment merupakan fasilitas yang disediakan Microsoft Visual Basic 6.0 yang digunakan media penghubung program yang di buat dengan database, biasanya dalam program digunakan untuk membuat laporan.

3.5 *Database*

Basis data atau *database* merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di perangkat komputer dan digunakan perangkat lunak untuk memanipulasinya. *Database* merupakan salah satu komponen yang penting dalam sistem informasi karena merupakan basis dalam menyediakan informasi bagi para pemakai.

Beberapa definisi basis data dari beberapa orang ahli basis data adalah sebagai berikut :

1. *Database* adalah sekumpulan *data store* (bisa dalam jumlah yang sangat besar) yang tersimpan dalam *magnetic disc, optical disc, magnetic drum* atau media penyimpanan sekunder lainnya.
2. *Database* adalah sekumpulan program – program aplikasi umum yang mengeksekusi dan memproses data secara umum (seperti pencarian, peremajaan, penambahan, dan penghapusan terhadap data).
3. *Database* terdiri dari data yang akan digunakan atau diperuntukan terhadap *user*, dimana masing – masing *user* akan menggunakan data

tersebut sesuai dengan tugas dan fungsinya, dan *user* lain dapat juga menggunakan data tersebut dalam waktu yang bersamaan.

4. *Database* adalah koleksi terpadu dari data – data yang saling berkaitan dari suatu *enterprise* (perusahaan, instansi pemerintah atau swasta).

Database yang sudah tersedia dalam suatu media penyimpanan tidak akan pernah bisa diakses tanpa adanya suatu perangkat lunak aplikasi yang familiar dengannya, misalkan saja perangkat lunak yang berbasis *database*. Kumpulan / gabungan *database* dengan perangkat lunak aplikasi berbasis *database* tersebut dinamakan *Database Management System (DBMS)*.

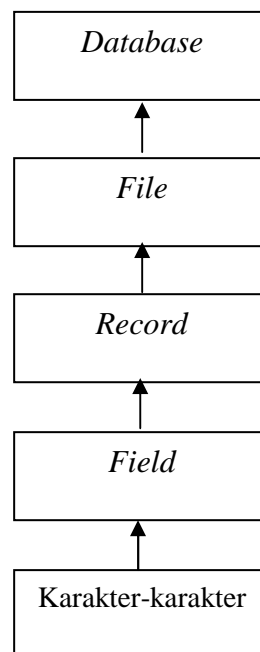
Microsoft Access (atau *Microsoft Office Acces*) adalah sebuah program aplikasi basis data komputer relasional yang ditujukan untuk kalangan rumahan dan perusahaan kecil hingga menengah. Aplikasi ini merupakan anggota dari beberapa aplikasi *Microsoft Office*, selain tentunya *Microsoft Word*, *Microsoft Excel*, dan *Microsoft PowerPoint*. Aplikasi ini menggunakan mesin basis data *Microsoft Jet Database Engine* dan juga menggunakan tampilan grafis yang intuitif sehingga memudahkan pengguna.

Microsoft Access dapat menggunakan data yang disimpan di dalam format *Microsoft Access*, *Microsoft Jet Database Engine*, *Microsoft SQL Server*, *Oracle Database* atau semua *container basis data* yang mendukung standar *ODBC*. Para pengguna/programmer yang mahir dapat menggunakannya untuk mengembangkan perangkat lunak aplikasi yang kompleks, sementara para programmer yang kurang mahir dapat

menggunakannya untuk mengembangkan perangkat lunak aplikasi yang sederhana. *Access* juga mendukung teknik – teknik pemrograman berorientasi objek, tetapi tidak dapat digolongkan ke dalam perangkat bantu pemrograman berorientasi objek.

3.5.1 Jenjang Data

Sampai dengan membentuk suatu *database*, data mempunyai jenjang mulai dari karakter – karakter, *record,file* dan kemudian *database*. Jenjang ini dapat digambarkan sebagai berikut :



Gambar 3.8. Jenjang data

1. Karakter – karakter

Karakter merupakan bagian data yang terkecil, dapat berupa karakter *numeric*, huruf ataupun karakter khusus yang membentuk suatu *field*.

2. *Field*

Suatu *field* menggambarkan suatu atribut dari *record* yang menunjukkan suatu item dari data, seperti nama, alamat dan lain sebagainya. Kumpulan dari *field* membentuk suatu *record*. Ada 3 hal yang penting dalam suatu *field*, yaitu :

a. Nama dari *field* (*field name*)

Field harus diberi nama untuk membedakan *field* yang satu dengan *field* yang lainnya.

b. Representasi dari *field* (*field representation*)

Representasi dari *field* menunjukkan tipe dari *field* (*field type*) serta lebar dari *field* (*field width*). *Field* dapat bertipe *numeric*, *text*, *boolean*, *data/time*, dan sebagainya. Lebar dari *field* menunjukkan ruang maksimum dari *field* yang dapat diisi dengan karakter – karakter data.

c. Nilai dari *field* (*field value*)

Nilai dari *field* menunjukkan isi dari *field* untuk masing- masing *record*.

3. *Record*

Kumpulan dari *field* membentuk suatu *record*. *Record* menggambarkan suatu unit data individu yang tertentu. Kumpulan dari *record* membentuk suatu *file*. Misalnya *file* anggota, tiap – tiap *record* dapat mewakili data – data tiap anggotanya.

4. *File*

File terdiri dari *record* – *record* yang menggambarkan satu kesatuan data yang sejenisnya. Misalnya *file* berisi data tentang semua buku yang ada.

5. *Database*

Kumpulan dari *file* membentuk suatu *database*.



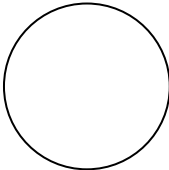
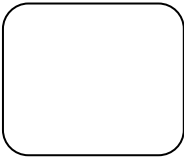


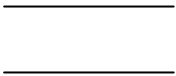
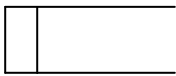
3.6 ***DFD (Data Flow Diagram)***

Data Flow Diagram (DFD) adalah representasi grafik dari sebuah sistem. DFD menggambarkan komponen-komponen sebuah sistem, aliran-aliran data di mana komponen-komponen tersebut, dan asal, tujuan, dan penyimpanan dari data tersebut.

Menggunakan DFD untuk dua hal utama, yaitu untuk membuat dokumentasi dari sistem informasi yang ada, atau untuk menyusun dokumentasi untuk sistem informasi yang baru.

Empat simbol yang digunakan :

Tabel 3.1. Simbol DFD

<i>Notasi Yourdon/ DeMarco</i>	<i>Notasi Gane & Sarson</i>	Keterangan
		Simbol <i>Entitas eksternal/Terminator</i> menggambarkan asal atau tujuan data di luar sistem
		Simbol lingkaran menggambarkan entitas atau proses dimana aliran data masuk ditransformasikan ke aliran data keluar
		Simbol aliran data menggambarkan aliran data
		Simbol <i>file</i> menggambarkan tempat data di simpan

Ada 3 (tiga) jenis DFD, yaitu ;

1. *Context* Diagram (CD)
2. DFD Fisik
3. DFD Logis

3.6.1 *DFD Level*

DFD dapat digambarkan dalam Diagram *Context* dan Level *n*. Huruf *n* dapat menggambarkan level dan proses di setiap lingkaran.

1. Diagram *Context*
2. Diagram Level *n*
 1. DFD Logis
 2. DFD Fisik

3.6.2 *Context Diagram (CD)*

Jenis pertama *Context Diagram*, adalah data *flow* diagram tingkat atas (DFD *Top Level*), yaitu diagram yang paling tidak *detail*, dari sebuah sistem informasi yang menggambarkan aliran-aliran data ke dalam dan ke luar sistem dan ke dalam dan ke luar entitas-entitas eksternal. (CD menggambarkan sistem dalam satu lingkaran dan hubungan dengan entitas luar. Lingkaran tersebut menggambarkan keseluruhan proses dalam sistem).

Beberapa hal yang harus diperhatikan dalam menggambar CD :

1. Terminologi sistem :
 1. Batas *Sistem* adalah batas antara “daerah kepentingan sistem”.
 2. Lingkungan Sistem adalah segala sesuatu yang berhubungan atau mempengaruhi sistem tersebut.
 3. *Interface* adalah aliran yang menghubungkan sebuah sistem dengan lingkungan sistem tersebut.

Sebagai contoh, dalam gambar 1.2.

- a. Menggunakan satu simbol proses,

Catatan:

Yang masuk didalam lingkaran konteks (simbol proses) adalah kegiatan pemrosesan informasi (Batas Sistem). Kegiatan informasi adalah mengambil data

- dari file, mentransformasikan data, atau melakukan filing data, misalnya mempersiapkan dokumen, memasukkan, memeriksa, mengklasifikasi, mengatur, menyortir, menghitung, meringkas data, dan melakukan filing data (baik yang melakukan secara manual maupun yang dilakukan secara terotomasi).
- b. Nama/keterangan di simbol proses tersebut sesuai dengan fungsi sistem tersebut,
 - c. Antara Entitas Eksternal/Terminator tidak diperbolehkan komunikasi langsung
 - d. Jika terdapat termintor yang mempunyai banyak masukan dan keluaran, diperbolehkan untuk digambarkan lebih dari satu sehingga mencegah penggambaran yang terlalu rumit, dengan memberikan tanda asterik (*) atau garis silang (#).
 - e. Jika Terminator mewakili individu (personil) sebaiknya diwakili oleh peran yang dipermainkan personil tersebut.
 - f. Aliran data ke proses dan keluar sebagai output keterangan aliran data berbeda.

3.6.3 *Diagram Level n / Data Flow Diagram Levelled*

Dalam diagram n DFD dapat digunakan untuk menggambarkan diagram fisik maupun diagram diagram logis.

Dimana Diagram Level n merupakan hasil pengembangan dari *Context* Diagram ke dalam komponen yang lebih detail tersebut disebut dengan *top-down* partitioning. Jika melakukan pengembangan dengan benar akan mendapatkan DFD-DFD yang seimbang. Sebagai contoh, gambar 3.1, gambar 3.2, gambar 3.3, gambar 3.4 dan gambar 3.5.

Beberapa hal yang harus diperhatikan dalam membuat DFD ialah:

1. Pemberian Nomor pada diagram level n dengan ketentuan sebagai berikut:
 - a. Setiap penurunan ke level yang lebih rendah harus mampu merepresentasikan proses tersebut dalam spesifikasi proses yang jelas. Sehingga seandainya belum cukup jelas maka seharusnya diturunkan ke level yang lebih rendah.
 - b. Setiap penurunan harus dilakukan hanya jika perlu.
 - c. Tidak semua bagian dari sistem harus diturunkan dengan jumlah level yang sama karena yang kompleks bisa saja diturunkan, dan yang sederhana mungkin tidak perlu diturunkan. Selain itu, karena tidak semua proses dalam level yang sama punya derajat kompleksitas yang sama juga.
 - d. Konfirmasikan DFD yang telah dibuat pada pemakai dengan cara *top-down*.

- e. Aliran data yang masuk dan keluar pada suatu proses di level n harus berhubungan dengan aliran data yang masuk dan keluar pada level $n+1$. Dimana level $n+1$ tersebut mendefinisikan sub-proses pada level n tersebut.
 - f. Penyimpanan yang muncul pada level n harus didefinisikan kembali pada level $n+1$, sedangkan penyimpanan yang muncul pada level n tidak harus muncul pada level $n-1$ karena penyimpanan tersebut bersifat lokal.
 - g. Ketika mulai menurunkan DFD dari level tertinggi, cobalah untuk mengidentifikasi external *events* dimana sistem harus memberikan respon. External *events* dalam hal ini berarti suatu kejadian yang berkaitan dengan pengolahan data di luar sistem, dan menyebabkan sistem memberikan respon.
2. Jangan menghubungkan langsung antara satu penyimpanan dengan penyimpanan lainnya (harus melalui proses).
 3. Jangan menghubungkan langsung dengan tempat penyimpanan data dengan entitas eksternal / *terminator* (harus melalui proses), atau sebaliknya.
 4. Jangan membuat suatu proses menerima input tetapi tidak pernah mengeluarkan output yang disebut dengan istilah *black hole*.

5. Jangan membuat suatu tempat penyimpanan menerima input tetapi tidak pernah digunakan untuk proses.
6. Jangan membuat suatu hasil proses yang lengkap dengan data yang terbatas yang disebut dengan istilah *magic process*.
7. Jika terdapat *terminator* yang mempunyai banyak masukan dan keluaran, diperbolehkan untuk digambarkan lebih dari satu sehingga mencegah penggambaran yang terlalu rumit, dengan memberikan tanda asterik (*) atau garis silang (#), begitu dengan bentuk penyimpanan.
8. Aliran data ke proses dan keluar sebagai *output* keterangan aliran data berbeda.

3.6.4 DFD Fisik

Adalah representasi grafik dari sebuah sistem yang menunjukkan entitas-entitas internal dan eksternal dari sistem tersebut, dan aliran-aliran data ke dalam dan keluar dari entitas-entitas tersebut. Entitas-entitas internal adalah personel, tempat (sebuah bagian), atau mesin (misalnya, sebuah komputer) dalam sistem tersebut yang mentransformasikan data. Maka DFD fisik tidak menunjukkan apa yang dilakukan, tetapi menunjukkan dimana, bagaimana, dan oleh siapa proses-proses dalam sebuah sistem dilakukan. (Tidak Bahas).

Perlu diperhatikan didalam memberikan keterangan di lingkaran-lingkaran (simbol proses) dan aliran-aliran data (simbol

aliran data) dalam DFD fisik menggunakan label/keterangan dari kata benda untuk menunjukkan bagaimana sistem mentransmisikan data antara lingkaran-lingkaran tersebut.

Misal :

Aliran Data : Kas, Formulir 66W, Slip Setoran

Proses : Cleck Penjualan, Kasir, Pembukuan.

3.6.5 DFD Logis

Adalah representasi grafik dari sebuah sistem yang menunjukkan proses-proses dalam sistem tersebut dan aliran-aliran data ke dalam dan ke luar dari proses-proses tersebut. Menggunakan DFD logis untuk membuat dokumentasi sebuah sistem informasi karena DFD logis dapat mewakili logika tersebut, yaitu apa yang dilakukan oleh sistem tersebut, tanpa perlu menspesifikasi dimana, bagaimana, dan oleh siapa proses-proses dalam sistem tersebut dilakukan.

Keuntungan dari DFD logis dibandingkan dengan DFD fisik adalah dapat memusatkan perhatian pada fungsi-fungsi yang dilakukan sistem.

Perlu diperhatikan di dalam pemberian Keterangan/ Label;

1. Lingkaran-lingkaran (simbol proses) menjelaskan apa yang dilakukan sistem

Misal : Menerima Pembayaran, Mencatat Penjualan, Membandingkan kas dan Daftar Penerimaan, Mempersiapkan Setoran.

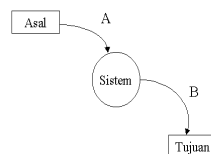
- Aliran-aliran data (simbol aliran data) menggambarkan sifat data.

Misal : Pembayaran (bukan “Cek”, “Kas”, “Kartu Kredit”

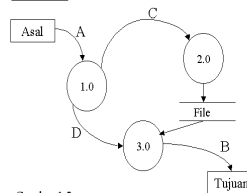
Jurnal Penjualan (bukan “Buku Penjualan”),

Usulan dari analisis (berupa DFD diatas), beberapa hal yang umum yang mendapat perhatian dalam mendesain baru tersebut ialah:

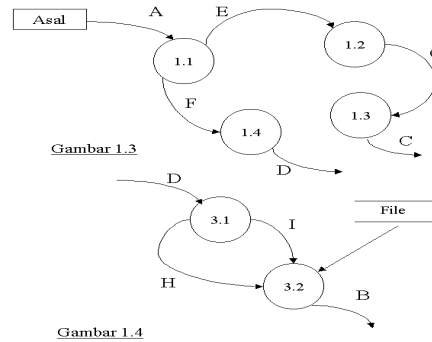
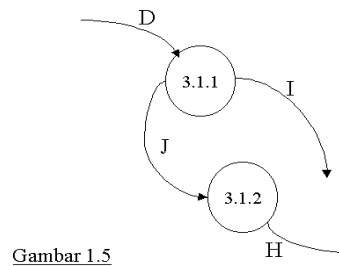
- Menggabungkan beberapa tugas menjadi Satu
- Master Detail Update*
- Meminimalkan tugas-tugas yang tidak penting
- Menghilangkan tugas-tugas yang duplikat
- Menambahkan proses baru
- Meminimalkan proses input
- Menetapkan bagian mana yang harus dikerjakan komputer dan bagian mana yang harus dikerjakan manual.



Gambar 1.1



Gambar 1.2

Gambar 3.9. *Context Diagram*Gambar 3.10. *Context Diagram*Gambar 3.11. *Context Diagram*

3.7 *Entity Relationship Diagram (ERD)*

Entity Relationship Diagram merupakan jaringan yang menggunakan susunan data yang disimpan dari sistem secara abstrak. Diagram *Entity Relationship* ini ditemukan oleh Chen tahun 1976.

Tujuan dari *Entity Relationship* adalah untuk menunjukkan objek data dan *relationship* yang ada pada objek tersebut. Disamping itu Model ER ini merupakan salah satu alat untuk perancangan dalam basis data.

Komponen (Simbol) ERD

3.7.1 *Entity*

Adalah suatu objek yang dapat dibedakan atau dapat diidentifikasi secara unik dengan objek lainnya, dimana semua informasi yang berkaitan dengannya dikumpulkan. Kumpulan dari *entity* yang sejenis dinamakan *Entity Set*.

Contoh : Proyek Penjualan

Langganan Kendaraan

Peralatan Pegawai

Pasien Obat.

Simbol dari *Entity* :

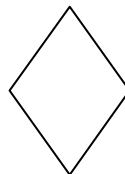


Gambar 3.12. Simbol *Entity*

3.7.2 *Relationship*

Adalah hubungan yang terjadi antara satu *entity* dengan *entity* lainnya. *Relationship* tidak mempunyai keberadaan fisik atau konseptual kecuali yang sejenis dinamakan dengan *Relationship Diagram*.

Simbol dari *Relationship* adalah :



Gambar 3.13. Simbol *Relationship*

Contoh :



Gambar 3.14. Contoh *Relationship*

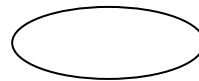
Keterangan :

Memiliki adalah *relationship set* yang terbentuk antara *entity* Pegawai dengan *entity* Kendaraan.

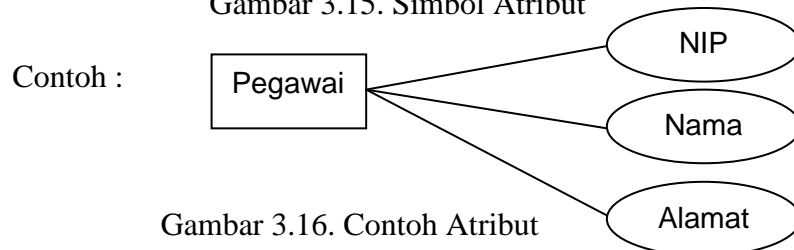
3.7.3 Atribut

Adalah karakteristik dari *entity* atau *relationship* yang menyediakan penjelasan detail tentang *entity* atau *relationship* tersebut.

Simbol dari Atribut adalah :



Gambar 3.15. Simbol Atribut



Gambar 3.16. Contoh Atribut

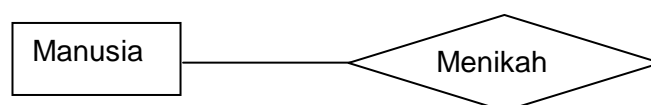
3.8 Derajat *Relationship*

Derajat *Relationship* adalah :

1. *Unary* (Derajat Satu)

Adalah satu buah *relationship* menghubungkan satu buah *entity*.

Contoh :



Gambar 3.17. Contoh *Unary*

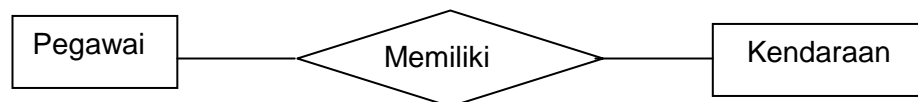
Keterangan :

Manusia menikah dengan manusia, *relationship* menikah hanya menghubungkan *entity* manusia.

2. *Binary* (Derajat Dua)

Adalah satu buah *relationship* yang menghubungkan dua buah *entity*.

Contoh :

Gambar 3.18. Contoh *Binary*

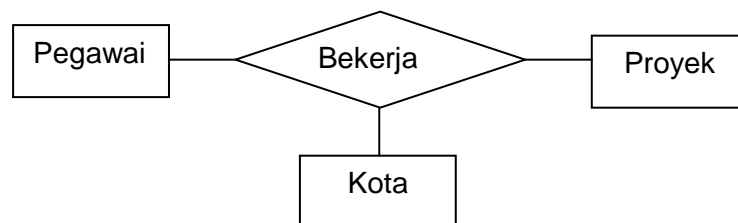
Keterangan :

Pegawai memiliki kendaraan, sebuah *relationship* memiliki menghubungkan *entity* Pegawai dan *entity* Kendaraan.

3. *Ternary* (Derajat Tiga)

Adalah satu buah *relationship* menghubungkan tiga buah *entity*.

Contoh :

Gambar 3.19. Contoh *Ternary*

Keterangan :

Pegawai pada kota tertentu mempunyai suatu Proyek.

Entity Bekerja menghubungkan *Entity* Pegawai, Proyek dan Kota

3.9 Cardinality Rasio

Yaitu menjelaskan batasan pada jumlah *entity* yang berhubungan melalui suatu *relationship*.

Jenis-jenis *Cardinality* Rasio :

1. *One To One* (1 : 1)

Yaitu perbandingan antara *entity* pertama dengan *entity* kedua berbanding satu berbanding satu.

Contoh :

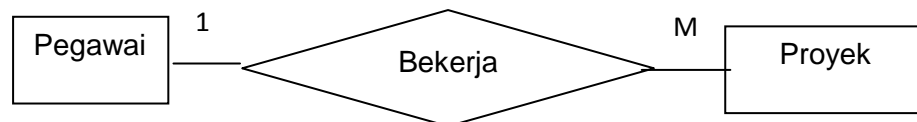


Gambar 3.20. Contoh *One To One*

2. *One To Many* (1 : M)

Yaitu perbandingan antara *entity* pertama dengan *entity* kedua berbanding satu berbanding banyak.

Contoh :

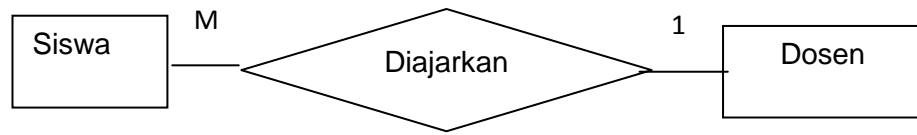


Gambar 3.21. Contoh *One To Many*

3. *Many To One* (M : 1)

Yaitu perbandingan antara *entity* pertama dengan *entity* kedua berbanding banyak berbanding satu.

Contoh :

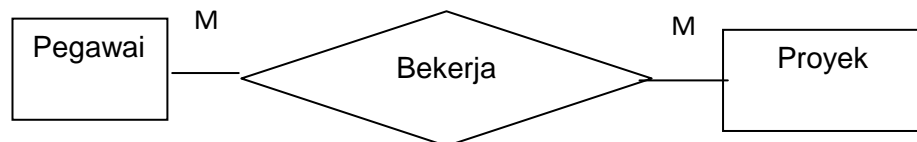


Gambar 3.22. Contoh *Many To One*

4. *Many To Many* (M : M)

Yaitu perbandingan antara *entity* pertama dengan *entity* kedua berbanding banyak berbanding banyak.

Contoh :



Gambar 3.23. Contoh *Many To Many*

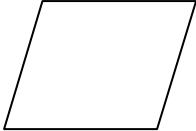


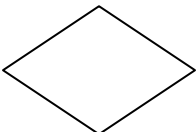
3.10 Bagan Alir (*flowchart*)


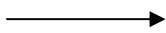
Flowchart adalah bagan alir yang menggambarkan tentang urutan langkah jalannya suatu program dalam sebuah bagan dengan simbol-simbol bagan yang sudah ditentukan.

Adapun simbol-simbol flowchart program adalah sebagai berikut :

Tabel 3.2 Simbol-simbol Flowchart Program

Simbol	Keterangan
	Terminator / Terminal Merupakan simbol yang digunakan untuk menentukan state awal dan state akhir suatu flowchart program.
	Preparation / Persiapan Merupakan simbol yang digunakan untuk mengidentifikasi variabel-variabel yang akan digunakan dalam program. Bisa berupa

Simbol	Keterangan
	pemberian harga awal, yang ditandai dengan nama variabel sama dengan ('') untuk tipe string, (0) untuk tipe numeric, (.F./T.) untuk tipe Boolean dan ({//}) untuk tipe tanggal.
	<p>Input output / Masukan keluaran</p> <p>Merupakan simbol yang digunakan untuk memasukkan nilai dan untuk menampilkan nilai dari suatu variabel. Ciri dari simbol ini adalah tidak ada operator baik operator aritmatika hingga operator perbandingan. Yang membedakan antara masukan dan keluaran adalah jika Masukan cirinya adalah variabel yang ada didalamnya belum mendapatkan operasi dari operator tertentu, apakah pemberian nilai tertentu atau penambahan nilai tertentu. Adapun ciri untuk keluaran adalah biasanya variabelnya sudah pernah dilakukan pemberian nilai atau sudah dilakukan operasi dengan menggunakan operator tertentu.</p>
	<p>Process / Proses</p> <p>Merupakan simbol yang digunakan untuk memberikan nilai tertentu, apakah berupa rumus, perhitungan counter atau hanya pemberian nilai tertentu terhadap suatu variabel.</p>
	<p>Predefined Process / Proses Terdefinisi</p> <p>Merupakan simbol yang penggunaannya seperti link atau menu. Jadi proses yang ada di dalam simbol ini harus di buatkan penjelasan flowchart programnya secara tersendiri yang terdiri dari terminator dan diakhiri dengan terminator.</p>
	<p>Decision / simbol Keputusan</p> <p>Digunakan untuk menentukan pilihan suatu kondisi (Ya atau tidak). Ciri simbol ini dibandingkan dengan simbol-simbol flowchart program yang lain adalah simbol keputusan ini minimal keluaran arusnya 2 (dua), jadi Jika hanya satu keluaran maka penulisan simbol ini adalah salah, jadi diberikan pilihan jika kondisi bernilai benar (true) atau salah (false). Sehingga jika nanti keluaran dari simbol ini adalah lebih dari dua</p>

Simbol	Keterangan
	bisa dituliskan. Khusus untuk yang keluarannya dua, harus diberikan keterangan Ya dan Tidaknya pada arus yang keluar.
	Connector Konektor dalam satu halaman merupakan penghubung dari simbol yang satu ke simbol yang lain. Tanpa harus menuliskan arus yang panjang. Sehingga akan lebih menyederhanakan dalam penggambaran aliran programnya, simbol konektornya adalah lingkaran, sedangkan Konektor untuk menghubungkan antara simbol yang satu dengan simbol yang lainnya yang berbeda halaman, maka menggunakan simbol konektor yang segi lima, dengan diberikan identitasnya, bisa berupa karakter alfabet A – Z atau a – z atau angka 1 sampai dengan 9.
	Arrow / Arus Merupakan simbol yang digunakan untuk menentukan aliran dari sebuah flowchart program. Karena berupa arus, maka dalam menggambarkan arus data harus diberi simbol panah.

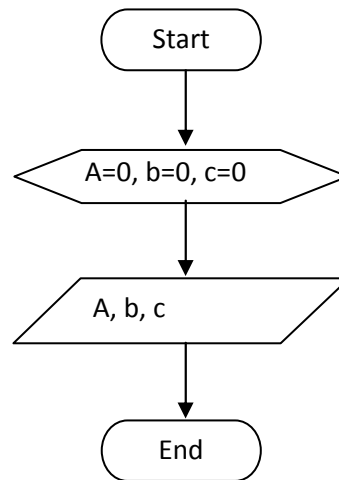
Ketentuan Menuliskan *Flowchart* Program adalah sebagai berikut:

1. *Flowchart* dituliskan dari atas ke bawah
2. Jika tidak cukup dan akan dituliskan ke samping, maka *Flowchart* dituliskan dari kiri ke kanan.
3. Tiap-tiap simbol harus memberikan keterangan yang jelas.
4. Untuk simbol terminal / terminator, keterangan yang bisa dituliskan di dalamnya adalah [Mulai | Selesai | Start | End] → atau yang menjelaskan tentang state awal dan state akhir.
5. Untuk simbol Proses terdapat operator aritmatika
6. Untuk simbol Keputusan boleh terdapat operator pembandingan

7. Untuk penggunaan konektor dalam satu halaman menggunakan simbol konektor dengan bentuk lingkaran, dan untuk konektir dari satu simbol ke simbol yang lain dengan simbol yang berbentuk segi lima.

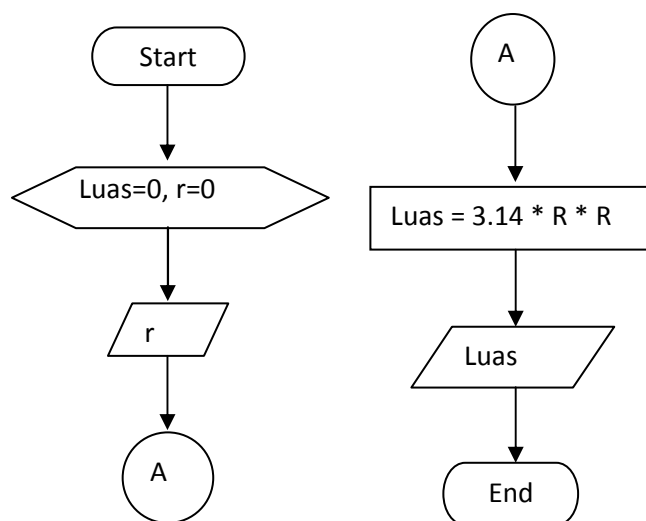
Contoh :

Flowchart program untuk memasukkan beberapa bilangan.



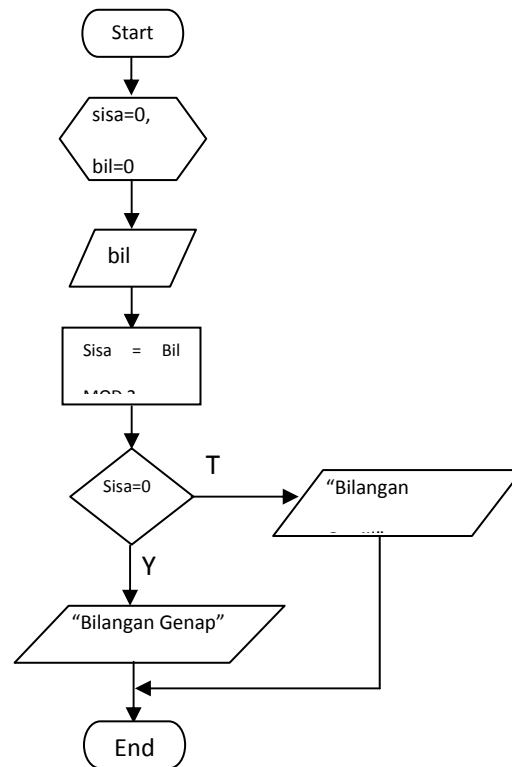
Gambar 3.24 Contoh *flowchart*

Flowchart program untuk menghitung luas lingkaran dengan masukan sebuah jari-jari.



Gambar 3.25. Contoh *Flowchart*

Flowchart program untuk menentukan sebuah bilangan yang dimasukkan apakah termasuk bilangan ganjil atau bilangan genap.



Gambar 3.26. Contoh *Flowchart*