BAB II

LANDASAN TEORI

2.1. Konsep Dasar Sistem

Terdapat dua kelompok pendekatan didalam pendefinisian sistem, yaitu kelompok yang menekankan pada prosedur dan kelompok yang menekankan pada elemen atau komponennya. Pendekatan yang menekankan pada prosedur mendefinisikan sistem sebagai suatu jaringan kerja prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu. Sedangkan pendekatan sistem yang lebih menekankan pada elemen atau komponen mendefinisikan sistem sebagai kumpulan elemen yang berinteraksi untuk mencapai suatu tujuan tertentu.

A. Pengertian Sistem

Menurut Sutabri (2012:6) "Sistem pada dasarnya adalah sekelompok unsur yang erat hubungannya satu dengan yang lainnya, yang berfungsi bersama-sama untuk mencapai tujuan tertentu".

Dari definisi ini dapat dirinci lebih lanjut pengertian sistem secara umum, yaitu sebagai berikut:

1. Setiap sistem terdiri dari berbagai unsur. Unsur-unsur suatu sistem terdiri dari subsistem yang lebih kecil, yang terdiri pula dari kelompok-kelompok unsur yang membentuk subsistem tersebut.
2. Unsur-unsur tersebut merupakan bagian yang tak terpisahkan dari sistem yang bersangkutan. Unsur-unsur sistem berhubungan erat satu sama lain

dimana sifat serta kerjasama antar unsur dalam sistem tersebut mempunyai bentuk tertentu.

3. Unsur-unsur didalam sistem tersebut bekerja sama untuk mencapai tujuan sistem. Setiap sistem mempunyai tujuan tertentu.
4. Suatu sistem merupakan bagian dari sistem lain yang lebih besar.

Suatu sistem dibuat untuk menangani sesuatu yang berulang kali atau yang secara rutin terjadi. Pendekatan sistem merupakan suatu filsafat atau persepsi tentang struktur yang mengkoordinasikan kegiatan-kegiatan dan operasi-operasi dalam suatu organisasi dengan cara yang efisien dan yang paling baik.

B. Karakteristik Sistem

Model umum sebuah sistem terdiri dari *input*, proses, dan *output*. Hal ini merupakan konsep sebuah sistem yang sangat sederhana mengingat sebuah sistem dapat mempunyai beberapa masukan dan keluaran sekaligus (Sutabri, 2012:13-14). Adapun karakteristik yang dimaksud sebagai berikut:

1. Komponen Sistem (*Components*)

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang bekerja sama membentuk satu kesatuan. Komponen- komponen sistem tersebut dapat berupa suatu bentuk subsistem. Setiap subsistem memiliki sifat-sifat sistem yang menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai sistem yang lebih besar yang disebut dengan Supra sistem.

2. Batasan Sistem (*Boundary*)

   Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem lainnya atau sistem dengan lingkungan luarnya. Batasan sistem

ini memungkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat dipisah-pisahkan.

3. Lingkungan Luar Sistem (*Environtment*)

   Bentuk apapun yang ada di luar lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut dengan lingkungan luar sistem. Lingkungan luar sistem ini dapat menguntungkan dan dapat juga merugikan sistem tersebut. Lingkungan luar yang menguntungkan merupakan energi bagi sistem tersebut, yang dengan demikian lingkungan luar tersebut harus selalu dijaga dan dipelihara. Sedangkan lingkungan luar yang merugikan harus dikendalikan. Kalau tidak maka akan mengganggu kelangsungan hidup sistem tersebut.

4. Penghubung Sistem (*Interface*)

   Media yang menghubungkan sistem dengan subsistem yang lain disebut dengan penghubung sistem atau *interface*. Penghubung ini memungkinkan sumber-sumber daya mengalir dari suatu subsistem ke subsistem yang lain. Keluaran suatu subsistem akan menjadi masukan untuk subsistem yang lain dengan melewati penghubung. Dengan demikian terjadi suatu integrasi sistem yang membentuk satu kesatuan.

5. Masukan Sistem (*Input*)

   Energi yang dimasukkan ke dalam sistem disebut masukan sistem, yang dapat berupa pemeliharaan *(maintenance input)* dan sinyal *(signal input)*. Sebagai contoh : di dalam suatu unit sistem komputer, “program” adalah *maintenance input* yang akan diolah menjadi informasi.

6. Keluaran Sistem (*Output*)

   Hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain. Seperti contoh sistem informasi, keluaran yang dihasilkan adalah informasi, dimana informasi ini dapat digunakan sebagai masukan untuk pengambilan keputusan atau hal-hal lain yang merupakan *input* bagi subsistem lainnya.

7. Pengolahan Sistem (*Process*)

   Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran. Sebagai contoh, sistem akuntansi. Sistem ini akan mengolah data transaksi menjadi laporan-laporan yang dibutuhkan oleh pihak manajemen.

8. Sasaran Sistem (*Objective*)

   Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat deterministik. Kalau suatu sistem tidak memiliki sasaran, maka operasi sistem tidak ada gunanya. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuan yang telah direncanakan.

C. Klasifikasi Sistem

Suatu sistem dapat diklasifikasikan (Mustakini, 2009:53):

1. Sistem abstrak *(abstact system)* dan sistem fisik *(phisical system)*

   Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tempak secara fisik, misalnya sistem teknologi yaitu sistem yang berupa pemikiran-pemikiran hubungan antara manusia dengan Tuhan. Sitem fisik merupakan sistem yang ada secara fisik.

2. Sistem Alami *(natural system)* dan Sistem Buatan Manusia *(human made system*) Sistem alami adalah sistem yang keberadaannya terjadi secara alami tanpa campuran tangan manusia. Sedangkan sistem buatan manusia adalah sebagai hasil kerja manusia. Contoh sistem alamiah adalah sistem tata surya yang terdiri dari atas sekumpulan planet, gugus bintang dan lainnya. Contoh sistem abstrak dapat berupa sistem komponen yang ada sebagai hasil karya teknologi yang dikembangkan manusia.
3. Sistem pasti *(deterministic system)* dan sistem tidak tentu *(probobalistic system)*

   Sistem tertentu adalah sistem yang tingkah lakunya dapat ditentukan/diperkirakan sebelumnya. Sedangkan sistem tidak tentu sistem tingkah lakunya tidak dapat ditentukan sebelumnya. Sistem aplikasi komputer merupakan contoh sistem yang tingkah lakunya dapat ditentukan sebelumnya. Program aplikasi yang dirancangdan dikembangkan oleh manusia dengan menggunakan prosedur yang jelas, terstruktur dan baku.
4. Sistem Tertutup *(closed system)* dan Sistem Terbuka *(open system)*

   Sistem tertutup merupakan sistem yang tingkah lakunya tidak dipengaruhi oleh lingkungan luarnya. Sebaliknya, sistem terbuka mempunyai prilaku yang dipengaruhi oleh lingkungannya. Sistem aplikasi komputer merupakan sistem *relative* tertutup, karena tingkah laku sistem aplikasi komputer tidak dipengaruhi oleh kondisi yang terjadi diluar sistem.

Berdasarkan beberapa pendapat yang dikemukakan di atas dapat ditarik kesimpulan bahwa “sistem adalah kumpulan bagian-bagian atau subsistem-subsistem yang disatukan dan dirancang untuk mencapai suatu tujuan”.

D. Pengertian Informasi

Menurut Mc. Leod dalam Al Fatta (2007:9) mengatakan bahwa “informasi adalah data yang telah diproses, atau data yang memiliki arti”. Sedangkan menurut Davis dalam Al Fatta (2007:9) bahwa “informasi adalah data yang telah diolah menjadi sebuah bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan saat ini atau mendatang”.Nilai dari informasi ditentukan oleh dua hal, yaitu manfaat dan biaya untuk mendapatkannya.Suatu informasi dikatakan bernilai bila manfaat yang dihasilkan lebih besar dari biaya yang dikeluarkan.

E. Basis Data

Menurut Sukamto dan Shalahuddin (2014:43), Mengemukakan bahwa “Sistem basis data adalah sistem terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan”.

1. DBMS *(Database Management System)*

DBMS *(Database Management System)* atau dalam bahasa Indonesia sering disebut sebagai sistem managemen basis data adalah suatu sistem aplikasi yang digunakan untuk menyimpan mengelola dan menampilkan data. Suatu sistem aplikasi disebut DBMS jika memenuhi persyaratan minimal sebagai berikut:

a. Menyediakan fasilitas untuk mengelola akses data
b. Mampu menangani integritas data
c. Mampu menangani akses data yang dilakukan secara terkomputerisasi
d. Mampu menangani *backup* data

Karena pentingnya data bagi suatu organisasi atau perusahaan, maka hampir sebagian besar perusahaan memanfaatkan DBMS dalam mengelola data yang mereka miliki.Pengelolaan DBMS sendiri biasanya ditangani oleh tenaga ahli yang spesialis menangani DBMS yang disebut sebagai DBA (*Database Administator).* DBMS sudah mulai berkembang sejak tahun 1960-an. Kemudian sekitar tahun 1970-an mula berkembang teknologi *Relational DBMS* yaitu DBMS berbasis relasional model. Relasional model pertama kali dikembangkan oleh Edgar J. Codd pada tahun 1970.Secara sederhana relasional model dapat dipahami sebagai suatu model yang memandang data sebagai sekumpulan tabel yang saling terkait.Hampir semua DBMS komersial dan *open source* saat ini berbasis *Relational DBMS* atau RDBMS. Pada tahun 1980-an mulai berkembang *Object Oriented DBMS (*OODBMS*).* OODBMS berkembang seiring dengan perkembangan pemograman berorientasi objek.Secara umum dapat diartikan bahwa OODBMS merupakan DBMS yang memandang data sebagai suatu objek.Saat ini OODBMS juga cukup berkembang namun belum dapat menggeser kepopuleran RDBMS. Berikut ini adalah 4 macam DBMS versi komersial yang paling banyak digunakan di dunia saat ini, yaitu:

a. Oracle
b. Microsoft SQL Server
c. IDM DB_2
d. Microsoft Acces

Sedangkan DBMS versi *open source* yang cukup berkembang dan paling banyak digunakan saat ini adalah sebagai berikut:

a. MySQL

b. PostgreSQL

c. Firebird

d. SQLite

Hampir semua DBMS mengadopsi SQL sebagai bahasa untuk mengelola data pada DBMS.

2. SQL

Menurut Sukamto dan Shalahuddin (2014:46), Mengemukakan bahwa "SQL (*Structured query language*) adalah bahasa yang digunakan untuk mengelola data pada RDBMS". SQL mulai berkembang tahun 1970-an. SQL mulai digunakan sebagai standar yang resmi pada tahun 1986 oleh ANSI (*American National Standards Institute*) dan pada tahun 1987 oleh ISO (*International Organization for Standardization*) dan disebut sebagai SQL-86. Pada perkembangannya, SQL beberapa dilakukan revisi. Berikut ini sejarah perkembangan SQL sampe saat ini:

Tabel II.1.

Perkembangan SQL

NO.	Tahun	Nama
1	1986	SQL-86
2	1989	SQL-89
3	1992	SQL-92
4	1999	SQL: 1999
5	2003	SQL: 2003
6	2006	SQL: 2006
7	2008	SQL: 2008
8	2011	SQL: 2011

Sumber: Sukamto dan Shalahuddin (2014:46)

F. Model Pengembangan Perangkat Lunak

Metode yang digunakan pada pengembangan perangkat lunak ini menggunakan model *waterfall* Sukamto dan Shalahuddin (2014:28-30) yang terbagi menjadi lima tahapan yaitu:

1. Analisis kebutuhan perangkat lunak

   Prosese pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*.Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

2. Desain

   Desain perangkat lunak adalah proses multi langkah yang focus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur

perangkat lunak, representasi antarmuka, dan prosedur pengkodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

3. Pembuatan kode program

   Desain harus ditranslasikan kedalam program perangkat lunak.Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4. Pengujian

   Fokus pada perangkat lunak secara dari segi logic dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai yang diinginkan.

5. Pendukung (*support*) atau pemeliharaan (*maintenance*)

   Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengurangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, membuat perangkat lunak baru.

2.2. Teori Pendukung

Dalam penulisan laporan Analisa Perancangan Sistem Informasi ini, penulis menggunakan beberapa teori pendukung untuk membantu penulis dalam menjelaskan sistem yang dibuat secara teoritis kepada pihak pemakai. Peralatan tersebut meliputi Diagram Alir Data (*Data Flow Diagram*), Normalisasi, dan Kamus Data (*Data Dictionary*).

A. Diagram Alir Data (DAD)

> Menurut Kendall & Kendall (2010:263) menyimpulkan bahwa:
> *Data Flow Diagram* (DFD) atau Diagram Alir Data (DAD), adalah menggambarkan pandangan sejauh mungkin mengenai masukan, proses dan keluaran sistem yang berhubungan dengan masukan, proses dan keluaran serta merepresentasikan dan menganalisis prosedur-prosedur secara mendetail dalam sistem yang lebih besar.

Melalui suatu teknik analisa dan terstruktur yang disebut dengan DAD ini, penganalisis sistem dapat merepresentasikan proses-proses data yang terdapat di dalam organisasi.

Pendekatan aliran data lebih menekankan logika yang mendasari sistem. Dengan menggunakan kombinasi dari empat simbol, penganalisis sistem dapat menciptakan suatu gambaran proses-proses yang bisa menampilkan dokumentasi yang *solid*.

Diagram Alir Data merupakan alat yang digunakan pada metodologi pengembangan sistem yang terstruktur (*Structured Analysis and Design*). Diagram Alir Data merupakan alat yang cukup populer sekarang ini, karena dapat menggambarkan arus data di dalam sistem dengan terstruktur dan jelas. Tahapan pembuatan DAD (Kendall & Kendall, 2010:267):

1. Diagram Konteks

   Diagram Konteks awal harus berupa suatu pandangan, yang mencakup masukan-masukan dasar, sistem umum dan keluaran.

   Diagram Konteks adalah tingkatan tertinggi dalam diagram aliran data dan hanya memuat satu proses, menunjukkan sistem secara keseluruhan. Proses tersebut diberi nomor NOL. Semua entitas eksternal yang ditunjukkan pada diagram konteks berikut aliran data-aliran data utama menuju dan dari sistem. Diagram tersebut tidak memuat penyimpanan data dan tampak sederhana untuk diciptakan, begitu entitas-entitas eksternal serta aliran data-aliran data menuju dan dari sistem diketahui penganalisis dari wawancara dengan pengguna dan sebagai hasil analisis dokumen.

2. Diagram NOL

   Diagram NOL adalah pengembangan diagram konteks dan bisa mencakup sampai sembilan proses. Memasukkan lebih banyak proses pada level ini akan terjadi dalam suatu diagram yang kacau yang sulit dipahami. Setiap proses diberi nomor bilangan bulat, umumnya dimulai dari sudut sebelah kiri atas diagram dan mengarah ke sudut sebelah kanan bawah. Penyimpanan data-penyimpanan data utama dari sistem (mewakili file-file master) dan semua entitas eksternal dimasukkan ke dalam Diagram 0.

3. Diagram Detail (Diagram Anak)

   Setiap proses dalam Diagram 0 bisa dikembangkan untuk menciptakan diagram anak yang lebih mendetail. Proses pada Diagram 0 yang dikembangkan itu disebut *parent process* (proses induk) dan diagram yang dihasilkan disebut *child diagram* (diagram anak). Aturan utama untuk

menciptakan diagram anak, keseimbangan vertikal, menyatakan bahwa suatu diagram anak tidak bisa menghasilkan keluaran atau menerima masukan dimana proses induknya juga tidak menghasilkan atau menerima. Semua aliran data yang menuju atau keluar dari proses induk harus ditunjukkan mengalir ke dalam atau keluar dari diagram anak. Diagram anak ditetapkan nomor yang sama seperti proses induknya di dalam Diagram 0.

Simbol-simbol yang digunakan dalam menggambarkan DAD (Kendall & Kendall, 2010:265):

1. *External Entity* (Entitas Eksternal)

   Digambarkan dengan kotak rangkap dua. Dapat mengirim data atau menerima data dari sistem. Eksternal entitas atau hanya entitas, disebut juga sumber atau tujuan data. Setiap entitas diberi label biasanya berupa kata benda. Entitas yang sama bisa digunakan lebih dari sekali atas suatu diagram aliran data tertentu untuk menghindari persilangan antara jalur-jalur data.

2. *Data Flow* (Aliran Data)

   Digambarkan dengan tanda panah. Tanda panah ini, menunjukkan adanya perpindahan dari satu titik ke titik yang lain, dengan kepala tanda panah mengarah ke tujuan data. Aliran data yang muncul secara simultan, dapat digambarkan hanya dengan menggunakan tanda panah paralel (Gunakan *double headed-arrows* hanya ketika sebuah proses membaca data dan mengupdate data pada tabel atau file yang sama.

3. *Process* (proses)

   Digambarkan bujur sangkar dengan sudut membulat. Digunakan untuk menunjukkan adanya proses transformasi. Proses Mewakili baik Keseluruhan

sistem, sebuah subsistem maupun pekerjaan yang diselesaikan, sebuah aktifitas. Aliran data yang meninggalkan suatu proses selalu diberi label yang berbeda dari aliran data yang masuk. Proses-proses yang ada didalam sistem harus diberi nama yang jelas untuk memudahkan dalam memahami proses yang sedang dilakukan, dan mengikuti format sebagai berikut:

a. Menetapkan nama sistem secara keseluruhan saat menamai proses pada level yang lebih tinggi.
b. Menggunakan format: kata kerja - kata sifat - kata benda untuk proses-proses yang mendetail.
c. Kata kerja menggambarkan jenis kegiatan seperti menghitung, menverifikasi, menyiapkan, mencetak atau menambahkan. Kata benda menunjukkan hasil utama proses seperti laporan, record. Kata Sifat mengilustrasikan keluaran yang mana seperti urutan ke belakang atau inventarisasi.

4. *Data Store* (Penyimpanan data)

Digambarkan bujur sangkar dengan ujung terbuka. Atau digambarkan bujur sangkar dengan dua garis paralel yang tertutup oleh sebuah garis pendek disisi kiri dan ujungnya terbuka disisi sebelah kanan. Penyimpanan data menunjukkan tempat penyimpanan untuk data-data yang memungkinkan adanya penambahan atau perolehan data. Karena penyimpanan data mewakili seseorang, tempat atau sesuatu maka diberi nama dengan kata benda. Memberikan nomor yang unik untuk setiap penyimpanan, seperti D1, D2, dst, yang dimaksudkan untuk mengidentifikasikan tingkatannya.

Aturan main pembuatan DAD (Kendall & Kendall, 2010:272):

1. Dalam DAD tidak boleh menghubungkan antara *External Entity* dengan *External Entity* secara langsung.
2. Dalam DAD tidak boleh menghubungkan antara *Data Store* dengan *Data Store* secara langsung.
3. Dalam DAD tidak boleh menghubungkan antara *Data Store* dengan *External Entity* secara langsung (atau sebaliknya).
4. Setiap Proses harus ada *Data Flow* yang masuk dan ada *Data Flow* yang keluar.
5. Aliran data tidak boleh terbelah menjadi dua atau lebih aliran data yang berbeda.

B. Kamus Data (*Data Dictionary*)

Menurut Kendall dan Kendall (2010:333) mengemukakan bahwa "Kamus Data adalah suatu aplikasi khusus dari jenis kamus-kamus yang digunakan sebagai referensi kehidupan setiap hari".

Kamus Data merupakan hasil referensi data mengenai data (*metadata*), suatu data yang disusun oleh penganalisis sistem untuk membimbing mereka selama melakukan analisis dan desain.

Kegunaan dari kamus data (kendall dan kendall, 2010:334), yaitu:

1. Memvalidasi diagram alir data dalam hal kelengkapan dan keakuratan.
2. Menyediakan suatu titik awal untuk mengembangkan layar dan laporan-laporan.
3. Menentukan muatan data yang disimpan dalam *file-file*.
4. Mengembangkan logika untuk proses-proses diagram alir data.

Isi kamus data harus mencerminkan keterangan yang jelas tentang data yang dicatat. Maka data harus memuat hal-hal sebagai berikut:

1. Nama Arus Data

   Karena kamus data dibuat berdasarkan arus data yang mengalir di diagram alir data, maka nama arus data juga harus dicatat di kamus data, sehingga mereka yang membaca diagram alir data dan memerlukan penjelasan lebih lanjut tentang suatu arus data tertentu di diagram alir data, dapat langsung mencarinya dengan mudah di kamus data.

2. Alias

   Atau nama lain dari data dapat dituliskan bila nama lain ini ada. Alias perlu ditulis karena data yang sama mempunyai nama yang berbeda untuk orang atau departemen satu dengan yang lainnya. Contoh: Bagian pembuat faktur menyebut bukti penjualan sebagai Faktur, sedangkan bagian gudang menyebutnya sebagai tembusan permintaan persediaan.

3. Bentuk Data

   Bentuk dari data yang mengalir dapat berupa:

   a. Dokumen dasar atau formulir.

   b. Dokumen hasil cetakan komputer.

   c. Laporan tercetak.

   d. Tampilan di layar monitor.

4. Arus Data

   Arus data menunjukkan dari mana data mengalir dan kemana data akan menuju.

5. Penjelasan

   Penjelasan digunakan untuk lebih memperjelas lagi tentang makna dari arus data yang dicatat di kamus data, maka bagian penjelasan dapat diisi dengan keterangan-keterangan tentang arus data tersebut.

6. Periode

   Periode menunjukkan kapan terjadinya arus data ini. Periode perlu dicatat di kamus data karena dapat digunakan untuk mengidentifikasikan kapan input data harus dimasukkan kedalam sistem, kapan proses dari program harus dilakukan dan kapan laporan harus dihasilkan.

7. Volume

   Volume yang perlu dicatat adalah volume rata-rata dan volume puncak. Volume rata-rata, menunjukkan banyaknya rata-rata arus data yang mengalir dalam suatu periode terntentu, sedangkan volume puncak menunjukkan volume yang terbanyak.

8. Struktur Data

   Menunjukkan arus data yang dicatat di kamus data terdiri dari *item-item* data apa saja.

Dalam pembangunan sebuah program, biasanya kamus data hanya menjelaskan tentang struktur data yang dipakai dalam program tersebut. Untuk menjelaskan informasi tentang struktur data yang dipakai maka biasanya digunakan notasi-notasi tertentu. Notasi atau simbol yang digunakan dibagi menjadi dua macam yaitu sebagai berikut:

1. Notasi Tipe Data

   Notasi ini digunakan untuk membuat spesifikasi format *input* maupun *output* suatu data. Notasi yang umum digunakan antara lain:

Tabel II.2.

Notasi Tipe Data

Notasi	Keterangan
X	Untuk setiap karakter
9	Angka *numerik*
A	Karakter *alphabet*
Z	Angka nol yang ditampikan dalam spasi kosong
.	Pemisah ribuan
,	Pemisah pecahan
-	Tanda penghubung
/	Pembagi

Sumber : Kendall & Kendall (2010:344)

2. Notasi Struktur Data

   Notasi ini digunakan untuk membuat spesifikasi elemen data. Notasi yang umum digunakan adalah sebagai berikut:

Tabel II.3.

Notasi Struktur Data

Notasi	Keterangan
=	Terdiri dari
+	Dan atau *And*
( )	Pilihan optional
{ }	*Iterasi* (Perulangan proses)
[ ]	Pilih salah satu pilihan yang ada
I	Pemisah pilihan didalam tanda [ ]
*****	Keterangan atau catatan
@	*Field* Kunci

Sumber : Kendall & Kendall (2010:338)

C. ERD *(Entity Relationship Diagram)*

Menurut Sukamto dan Shalahuddin (2014:50) "ERD adalah bentuk paling awal dalam melakukan perancangan basis data relasional. Jika menggunakan OODBMS maka perancangan ERD tidak perlu dilakukan".

Diagram *Entity-Relationship* melengkapi penggambaran grafik dari struktur logika. Dengan kata lain Diagram *Entity-Relationship* menggambarkan arti dari aspek data seperti bagaimana *entity-entity*, atribut–atribut dan *relationship-relationship*. Sebelum membuat Diagram *Entity-Relationship*, tentunya kita harus memahami betul data yang diperlukan dan ruang lingkupnya. Di dalam pembuatan diagram E-R perlu diperhatikan penentuan suatu konsep apakah merupakan suatu *entity*, *atribut* atau *relationship*.

Pemodelan awal basis data yang paling banyak digunakan adalah menggunakan *Entity Relationship Diagram* (ERD). ERD dikembangkan berdasarkan teori himpunan dalam bidang matematika. ERD digunakan untuk pemodelan basis data relasional. Sehingga penyimpanan basis data menggunakan OODBMS, maka perancangan basis data tidak perlu menggunakan ERD. ERD memiliki beberapa aliran notasi seperti notasi Chen (dikembangkan oleh Peter Chen), Barker (dikembangkan oleh Richard Barker, Ian Palmer, Harry Ellis), notasi Crow's Foot,dan beberapa notasi lain. Namun yang banyak digunakan adalah notasi dari Chen. Berikut adalah simbol-simbol yang digunakan pada ERD dengan notasi Chen:

1. Komponen dasar

   Komponen dasar *entity relationship diagram* terdiri dari:

   a. Entitas

      Entitas merupakan data inti yang akan disimpan; bakal tabel pada basis data benda yang memiliki data dan harus disimpan datanya agar dapat diakses oleh aplikasi komputer; penamaan entitas biasanya lebih ke kata benda dan belum merupakan nama tabel."

   b. Atribut

      Atribut memberikan informasi lebih rinci tentang jenis entitas. Atribut memiliki struktur internal berupa tipe data. Jenis-jenis atribut antara lain:

      1.) Atribut *Key*

         Atribut yang digunakan untuk menentukan suatu *entity* secara unik. Dalam perancangan basis data dengan model E-R (*Entity-Relationship*).

2.) Atribut *Simple*

Atribut yang bernilai tunggal.

3.) Atribut *Multivalue*

Atribut yang memiliki sekelompok nilai untuk setiap instan *entity*.

4.) Atribut *Composite*

Suatu atribut yang terdiri dari beberapa atribut yang lebih kecil yang mempunyai arti tertentu.

5.) Atribut *Derivatif*

Suatu atribut yang dihasilkan dari atribut yang lain.

c. Relasi (*Relationship*)

Relasi adalah hubungan yang terjadiantara satuatau lebih *entity*. *Relationship* tidak mempunyai keberdaan fisik, kecuali yang mewarisi hubungan antara *entity* tersebut. *Relationshipset* adalah kumpulan *relationship* yang sejenis. Relasi menunjukkan adanya hubungan diantara sejumlah entitas yang berasal dari himpunan entitas yang berbeda.

d. Asosiasi (*association)*

Penghubung antara relasi dan entitas di mana di kedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian. Kemungkinan jumlah maksimum keterhubungan antara entitas satu dengan entitas yang lain disebut dengan kardinalitas. Misalkan ada kardinalitas 1 ke N atau sering disebut dengan *one to many* menghubungkan entitas A dan entitas B maka ERD biasanya memiliki hubungan binary (satu relasi menghubungkan dua buah entitas). Beberapa metode perancangan ERD menolerasi hubungan relasi ternary (satu relasi menghubungkan tiga buah

relasi) atau N-ary (satu relasi menghubungkan banyak entitas), tapi banyak metode perancangan ERD yang tidak mengizinkan hubungan ternary atau N-ary. Berikut adalah contoh bentuk hubungan relasi dalam ERD:

1) *Indicator Type* Dalam ERD (*Entity Relationship Diagram*)

   Indicator Type dalam atribut terdiri atas:

   a) *Indicator type associative eobject*, berfungsi sebagai suatu objek dan suatu *relationship*.

   b) *Indicator super type*, terdiri dari suatu object dan satu sub kategori atau lebih yang dihubungkan dengan satu *relationship* yang tidak bernama.

2) Kardinalitas (*Cardinality*)

   Merupakan tingkat hubungan yang terdir antara *entity* didalam sebuah sistem, dari sejumlah kemungkinan banyaknya hubungan antar entitas tersebut.

3) Tahapan Pembuatan ERD (*Entity Relationship Diagram*)

   ERD (*Entity Relationship Diagram*) selalu dibuat secara bertahap. Paling tidak ada dua kelompok pertahapan yang biasa ditempuh di dalam pembuatannya, yaitu:

   a) Tahap pembuatan *entity relationship diagram* awal (*premilinary design*)

   b) Tahap optimasi diagram *entity-relationship* (*final design*)

      Pada tahap kedua ini memperhatikan aspek-aspek efisiensi, performansi dan fleksibilitas.

D. LRS *(Logical Record Structure)*

Menurut Simarmata (2007:115) *"Logical Record Structure adalah* resperentasi dari struktur record–record pada tabel–tabel yang terbentuk dari hasil relasi antar himpunan entitas". Beberapa *tipe record* digambarkan oleh kotak empat persegi panjang dan dengan nama yang unik.

LRS terdiri dari *link-link* diantara *tipe record*. *Link* ini menunjukkan arah dari satu *tipe record* lainnya. Banyak *link* dari LRS yang diberi tanda *field-field* yang kelihatan pada kedua *link tipe record*. Penggambaran LRS mulai dengan menggunakan model yang dimengerti. Dua metode yang dapat digunakan, dimulai dengan hubungan kedua model yang dapat dikonversikan ke LRS. Metode yang lain dimulai dengan *Entity Relationship Diagram* dan langsung dikonversikan ke LRS.

1. Konversi ERD ke LRS, Diagram entity relationship diagram harus diubah ke bentuk LRS (struktur record secara logik). Dari bentuk LRS inilah yang nantinya dapat ditransformasikan ke bentuk relasi (tabel).
2. Konversi ERD ke LRS Sebuah model sistem yang digambarkan dengan sebuah ERD akan mengikuti pola permodelan tertentu. Dalam kaitannya 16 dengan konversi ke LRS, untuk perubahan yang terjadi adalah mengikuti aturan - aturan berikut:
    a. Setiap entitas diubah kebentuk kotak dengan nama entitas, berada diluar kotak dan atribut berada didalam kotak.
    b. Sebuah relationship kadang disatukan, dalam sebuah kotak bersama entitas, kadang sebuah kotak bersama - sama dengan entitas, kadang disatukan dalam sebuah kotak tersendiri.

3. Konversi LRS ke relasi (tabel) adalah bentuk pernyataan data secara grafis 2 (dua) *dimensi*, yang terdiri dari kolom dan baris. Relasi adalah bentuk *visual* dari sebuah file, dan tiap tuple dalam sebuah *field*, atau yang dalam bentuk lingkaran Diagram *entity relationship* dikenal dengan sebutan atribut.
4. Konversi dari *logical record structure*. dilakukan dengan cara :
   a. Nama *logical record structure* menjadi nama relasi.
   b. Tiap atribut menjadi sebuah kolom didalam relasi.

E. Pengkodean

Menurut Kendall & Kendall (2008:267) menyimpulkan bahwa "Pengkodean adalah proses dari meletakkan data yang berarti dua macam atau data yang sulit dipakai dengan segera, lebih mudah dimasukan digital atau huruf". Jenis struktur kode (Kendall dan Kendall, 2008:273):

1. Kode mnemonik (*Mnemonic Code*)

   Digunakan untuk tujuan supaya mudah diingat dan dibuat berdasarkan singkatan atau mengambil sebagian karakter dari *item* yang diwakili dengan oleh *item* tersebut.
2. Kode urut (*Sequential Code*)

   Disebut juga kode seri (*serial code*) merupakan kode yang nilainya urut antara satu kode dengan kode berikutnya.
3. Kode blok (*Block Code*)

   Mengklasifikasikan *item* ke dalam kelompok blok tertentu yang mencerminkan satu klasifikasi tertentu atas dasar pemakaian maksimum yang diharapkan.

4. Kode group (*Group Code*)

   Merupakan kode yang berdasarkan *field-field* dan tiap-tiap *field* kode mempunyai arti.

5. Kode desimal (*Decimal Code*)

   Mengklasifikasikan kode atas dasar 10 *unit* angka desimal mulai dari angka 0 sampai dengan angka 9 atau mulai dari angka 00 sampai dengan angka 99 tergantung dari banyaknya kelompok.

Hal-hal yang perlu diperhatikan dalam pembuatan kode, antara lain:

1. Harus mudah diingat

   Supaya kode mudah diingat, maka dapat dilakukan dengan cara menghubungkan kode tersebut dengan obyek yang diwakilinya.

2. Harus unik

   Kode harus unik untuk mewakili masing-masing *item* yang diwakilinya. Unik di sini berarti tidak ada kode yang kembar.

3. Harus fleksibel

   Kode harus fleksibel yang artinya memungkinkan perubahan-perubahan atau penambahan *item* baru dapat tetap diwakili oleh kode.

4. Harus efisien

   Kode harus diusahakan dibuat sependek mungkin, selain mudah diingat juga akan efisien jika disimpan pada simpanan luar komputer.

5. Harus konsisten

   Kode harus konsisten dengan kode yang telah dipergunakan sebelumnya.

6. Harus distandarisasi

   Kode yang tidak distandarisasi akan mengakibatkan kebingungan, salah pengertian dan dapat cenderung terjadi kesalahan pemakaian bagi yang menggunakan kode tersebut.

7. Spasi dihindari

   Spasi didalam kode sebaiknya dihindari, karena dapat menyebabkan kesalahan dalam penggunaannya.

8. Hindari penggunaan karakter yang mirip.

   Karakter-karakter yang hampir sama bentuk dan pengucapannya dapat membingungkan, oleh sebab itu sebaiknya dihindari.

9. Panjang kode harus baku

   Masing-masing kode yang sejenis harus mempunyai panjang yang sama.

F. HIPO *(Hierarchy Plus Input Process Output)*

Menurut Al Fatta (2007:150) menyatakan bahwa "HIPO adalah teknik penggambaran modul-modul yang nantinya akan dikembangkan oleh programmer menjadi prosedur-prosedur dalam program sistem informasi". Mula-mula tiap fungsi utama diidentifikasi dan kemudian dibagi lagi kedalam tingkatan fungsi yang lebih.

Tujuan utama HIPO dapat diringkas sebagai berikut:

1. Untuk memberikan struktur yang memungkinkan fungsi sistem dimengerti.
2. Untuk menguraikan fungsi-fugsi yang akan dikerjakan oleh suatu program, bukan mengkhususkan pernyataan program yang dipakai untuk melaksanakan fungsi.

3. Untuk memberikan deskripsi *visual* dari *input* yang akan dipakai serta *output* yang akan dihasilkan oleh masing-masing fungsi pada tiap-tiap tingkat diagram.

Ciri-ciri paket HIPO adalah bahwa paket ini berisikan tiga jenis diagram:

1. Daftar isi *Visual Tabel of Contents* (VTOC) adalah satu atau lebih diagram hierarki.
2. Diagram ringkasan adalah suatu seri diagram fungsional, masing-masing diagram dihubungkan dengan salah satu fungsi sistem.
3. Diagram rinci adalah suatu seri diagram fungsional dan masing-masing diagram dihubungkan dengan sebuah sub fungsi sistem.

G. Microsoft Visual Basic 6.0

Menurut Winarno dkk (2013:1), "Microsoft Visual Basic 6.0 merupakan sebuah bahasa pemrograman yang menawarkan *Integrated Development Environment* (IDE) visual untuk membuat program perangkat lunak berbasis sistem operasi Microsoft *Windows* menggunakan model pemrograman (COM)".

Visual Basic merupakan turunan bahasa pemrograman BASIC dan menawarkan pengembangan perangkat lunak komputer berbasis grafik dengan cepat. Beberapa bahasa skrip seperti *Visual Basic For Application* (VBA) dan *Visual Basic Scripting Edition* (VBScript), mirip seperti halnya *Visual Basic*, tetapi cara kerjanya yang berbeda.

Visual Basic pertama adalah VB 1.0. software ini dikenalkan pada tahun 1991. Membawakan konsep pemprograman dengan metode *drag-and-drop* untuk membuat tampilan aplikasi, *visual basic* ini di adaptasi dari *prototype generator form* yang dikembangkan oleh Alan Cooper dan perusahaannya, bernama Tripod.

Microsoft kemudian mengontrak Cooper dan perusahaannya untuk mengembangkan Tripod menjadi sistem *form* yang dapat di program untuk *windows* 3.0, dibawah kode nama Ruby.

Berikut ini beberapa tahapan dari pengembangan *visual basic* hingga menjadi versi 6.0, yaitu:

1. Proyek *Thunder* dimulai.
2. *Visual Basic* 1.0 dirilis untuk *windows* pada Comdex.
3. *Visual Basic* 1.0 untuk DOS dirilis September 1992. Bahasa pemrograman nya sendiri tidak terlalu kompatibel dengan *Visual Basic* untuk *Windows,* karena sesungguhnya itu adalah versi selanjutnya dari kompiler BASIC berbasis DOS yang dikembangkan oleh Microsoft sendiri, yaitu QuickBASIC.
4. *Visual Basic* 2.0 dirilis pada November 1992, lingkungan pemrograman lebih mudah untuk digunakan, dan kecepatannya lebih ditingkatkan.
5. *Visual Basic* 3.0 dirilis pada musim semi 1993 dan hadir dalam dua versi: standar dan profesional. VB3 juga menyertakan versi 1.1 dari Microsoft jet *Database Engine* yang dapat membaca dan menulis *database* Jet / *Access* 1.x.
6. *Visual Basic 4.0* dirilis pada Agustus 1995, merupakan versi pertama yang dapat membuat program 32-bit seperti program 16-bit. VB4 juga memperkenalkan kemampuannya dalam membuat aplikasi non-GUI.
7. *Visual Basic 5.0* dirilis pada Februari 1997, Microsoft merilis *Visual Basic* eksklusif untuk versi 32-bit dari *Windows.*