BAB II

LANDASAN TEORI

2.1. Konsep Dasar Sistem

Menurut Sutabri (2012:6) terdapat dua kelompok pendekatan di dalam 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 menekankaan pada elemen atau komponen mendefinisikan sistem sebagai kumpulan elemen yang berinteraksi untuk mencapai suatu tujuan tertentu. Kedua kelompok definisi ini adalah benar dan tidak bertentangan. Yang berbeda adalah cara pendekatannya.

A. Pengertian Sistem

Menurut kendal & kendal (2010:523) "Sistem adalah serangkaian sub sistem yang saling terkait dan tergantung satu sama lainya, bekerja bersama-sama untuk mencapai tujuan sasaran yang sudah ditetapkan sebelumnya".

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

1. Karakteristik Sistem

Sistem juga memiliki karakteristik atau sifat-sifat tertentu, yang mencirikan bahwa hal tersebut bisa dikatakan sebagai suatu sistem. Adapun karakteristik sistem yang dimaksud menurut Tata Sutabri (2012:13), adalah:

a. 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 betuk 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.

b. 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 suatu kesatuan yang tidak dapat dipisahkan.

c. Lingkungan Luar Sistem *(Environtment)*

   Bentuk apapun yang ada diluar ruang 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. 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.

d. 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 dari subsistem akan menjadi masukan untuk subsistem yang lain dengan melewati penghubung. Dengan demikian terjadi suatu integrasi sistem yang membentuk satu kesatuan.

e. Masukan Sistem *(Input)*

Energi yang dimasukkan kedalam sistem disebut masukan sistem, yang dapat berupa pemeliharaan (*maintenance input)* dan sinyal (*signal input*). Sebagai contoh, didalam suatu unit sistem komputer, "program" adalah *maintenance input* yang digunakan untuk mengoperasikan komputer. Sementara "data" adalah *signal input* yang akan diolah menjadi informasi.

f. 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 pengembalian keputusan atau hal-hal lain yang merupakan input bagi subsistem lainnya.

g. Pengolah Sistem *(Process)*

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

h. 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.

B. Basis Data

Menurut Fathansyah (2007:2), "Basis data adalah himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasikan sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah"

Sedangkan menurut Fathansyah (2007:9), "Sistem basis data merupakan sistem yang terdiri atas kumpulan *file* (tabel) yang merupakan sistem terdiri atas kumpuluan *file* (tabel) yang saling berhubungan"

Sistem pengelolaan basis data secara fisik tidak dilakukan oleh pemakai secara langsung, tetapi ditangani oleh sebuah perangkat lunak (sistem) yang khusus/spesifik. Perangkat lunak yang menentukan bagaimana data diorganisasikan, disimpan, diubah, dan diambil kembali dan juga menerapkan mekanisme pengamanan.

Menurut Kadir (2008:2) MySQL merupakan *software* yang tergolong sebagai DBMS (*Database management System*) yang bersifat *open source*. *Open source* menyatakan bahwa *software* ini dilengkapi dengan *source code* (kode yang dipakai untuk MySQL), selain tentu saja bentuk sistem operasi, dan bisa diperoleh dengan cara men-*download* (mengunduh) di internet secara gratis.

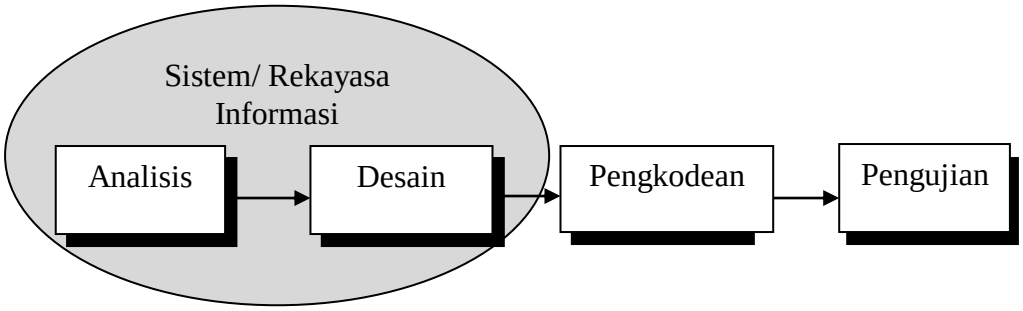
C. Model Pengembangan Perangkat Lunak

> Menurut Sukamto dan M. Shalahuddin (2013:26) :
> *Software Development Life Cycle* atau SDLC atau yang sering disebut juga *System Development Life Cycle* adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metedologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya.

Sukamto dan Shalahuddin (2013:28) mengungkapkan bahwa SDLC memiliki beberapa model dalam penerapan tahapan prosesnya. Salah satu model yang digunakan adalah sebagai berikut :

1) Model Waterfall

   Model SDLC air terjun (*waterfall)* sering juga disebut model sekuensial linier *(sequential linier)* atau alur hidup klasik *(classic life cycle).* Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengkodean, pengujian dan tahap pendukung *(support)*. Berikut adalah gambar model air terjun:



Sumber : Sukamto dan Shalahuddin (2013:28)

Gambar II.1. Ilustrasi Model *Waterfall*

a. Analisis kebutuhan perangkat lunak

   Proses 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.

b. Desain

   Desain perangkat lunak adalah proses multi langkah yang fokus 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.

c. Pembuatan kode program

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

d. Pengujian

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

e. 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 mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

2.2 Teori Pendukung

A. Diagram Alir Data *(Data Flow Diagram)*

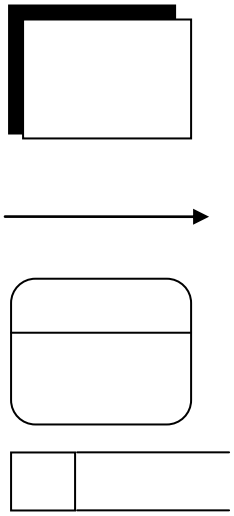
1. Konsep Dasar Diagram Alir Data *(Data Flow Diagram)*

   Kendall & Kendall (2010:263) mengungkapkan bahwa Diagram Alir Data menggambarkan pandangan sejauh mungkin mengenai masukan, proses dan keluaran dari model sistem. Serangkaian diagram aliran data berlapis juga bisa digunakan untuk merepresentasikan dan menganalisis prosedur-prosedur mendetail dalam sistem yang lebih besar tersebut.

2. Simbol-Simbol Dalam DAD

   Simbol-simbol yang digunakan menurut Kendall & Kendall (2010:265) pada diagram alir data adalah sebagai berikut :

Tabel II.1.
Simbol Diagram Alir Data (DAD)

Simbol	Keterangan
	*External Entity*/ Eksternal Entity : mengirim atau menerima data dari sistem, disebut juga sumber atau tujuan data, dan dianggap eksternal terhadap sistem yang digambarkan. *Data Flow/*Aliran Data : Menunjukan perpindahan data dari satu titik ke titik lain dengan kepala tanda panah mengarah ke tujuan data *Process/* Proses : Menunjukan adanya proses transformasi dan aliran data yang meninggalkan suatu proses selalu diberi label yang berbeda dari aliran data yang masuk. *Data Strore* : Menunjukan penyimpanan data.

Sumber: Kendall & Kendall (2010:265)

3. Aturan Main DAD

Bentuk rambu-rambu atau aturan main yang baku dan berlaku dalam penggunaan *data flow* diagram untuk membuat model sistem adalah sebagai berikut :

a. Didalam *data flow diagram* tidak boleh menghubungkan antara *external entity* dengan *external entity* lainnya secara langsung.
b. Didalam *data flow diagram* tidak boleh menghubungkan *data store* yang satu dengan *data store* yang lainnya secara langsung.
c. Didalam *data flow diagram* tidak boleh atau tidak diperkenankan menghubungkan *data store* dengan *external entity* secara langsung.
d. Setiap proses harus ada *data flow* yang masuk dan ada juga *data flow* yang keluar.

4. Langkah Langkah Mengembangkan DAD

   a. Membuat sebuah daftar tentang kegiatan bisnis dan digunakan untuk menentukan berbagai macam :

      1. Entitas eksternal
      2. Aliran data
      3. Proses-proses
      4. Penyimpanan data

   b. Menciptakan sebuah diagram yang menunjukan entitas-entitas eksternal dan aliran data menuju sistem.

   c. Menggambar diagram nol yang menunjukkan proses-proses dan penyimpanan data.

   d. Menciptakan diagram anak untuk setiap proses dalam diagram 0

   e. Mengecek kesalahan dan memastikan label-label yang ditetapkan untuk setiap proses dan aliran data.

5. Tahapan Pembuatan DAD

   (Kendall & Kendall, 2010:266) Langkah-langkah dalam membuat DAD dibagi menjadi tiga tahap yaitu sebagai berikut :

   a. Diagram Konteks

      Diagram konteks adalah tingkatan tertinggi dalam aliran data dan hanya memuat satu proses, menunjukan sistem secara keseluruhan. Proses tersebut diberi nomor nol. Semua entitas eksternal yang ditunjukan pada diagram konteks berikut aliran data utama menuju sistem. Diagram tersebut tidak memuat penyimpanan data dan tampak sederhana untuk diciptakan.

b. Diagram Nol

Diagram nol dibuat lebih mendetail dibanding diagram konteks. Masukan dan keluaran yang ditetapkan dalam diagram yang pertama tetap konstan dalam semua diagram sub urutannya. Sisa diagram asli dikembangkan kedalam gambaran terperinci yang melibatkan tiga sampai sembilan proses dan menunjukan penyimpanan data dan aliran data baru pada level yang lebih rendah.

c. Diagram Detail

Setiap proses dalam diagram nol bisa dikembangkan untuk menciptakan diagram anak (diagram detail) yang lebih mendetail. Proses pada diagram nol dikembangkan itu disebut dengan *Parent Process* (Proses Induk) dan diagram yang dihasilkan disebut *Child Diagram* (Diagram Anak). Aturan dalam mengembangkan diagram anak adalah diagram anak tidak dapat menghasilkan keluaran atau menerima masukan dimana proses induknya juga tidak menghasilkan atau menerima.

B. Kamus Data (*Data Dictionary*)

1. Konsep Dasar Kamus Data

Menurut Kendall & Kendall (2010:333), 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 (meta data), suatu data yang disusun oleh penganalisis sistem untuk membimbing mereka selama melakukan analisis dan desain.

Sebagai suatu dokumen, kamus data mengumpulkan dan mengkoordinasi istilah-istilah data tertentu, dan menjelaskan apa arti setiap istilah yang ada.

2. Hal-hal yang perlu dimuat dalam Kamus Data

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

   a. Nama Arus Data

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

   b. Alias

      Alias atau nama lain dari data yang dituliskan karena data yang sama mempunyai nama yang berbeda untuk orang atau departemen yang satu dengan yang lainnya.

   c. Tipe Data atau Bentuk Data

      Data yang mengalir dari hasil suatu proses ke proses lainnya dalam bentuk dokumen dasar atau formulir, dokumen hasil cetakan komputer, laporan terarah, tampilan layar dimonitor, variabel, parameter dan *field-field* adalah bentuk data dari arus data yang mengalir yang perlu dicatat di kamus data.

s

d. Arus Data

Arus data menunjukan dari mana data mengalir dan kemana data akan menuju. Keterangan arus data ini perlu dicatat dikamus data supaya memudahkan mencari arus data di dalam *data flow diagram*.

e. Penjelasan

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.

f. Periode

Periode ini menunjukkan kapan terjadinya arus data ini. Periode ini perlu dicatat di kamus data karena dapat digunakan untuk mendefisinikan kapan input data harus dimasukkan ke dalam sistem, kapan proses program harus dilakukan dan kapan laporan-laporan harus dihasilkan.

g. Volume

Volume yang perlu dicatat di dalam kamus data adalah tentang volume rata-rata dan volume puncak dari arus data. Volume rata-rata menunjukan banyaknya arus data yang mengalir dalam satu periode tertentu, sedangkan volume puncak menunjukkan volume yang terbanyak.

h. Struktur Data

Struktur data menunjukan arus data yang dicatat pada kamus data yang terdiri dari item-item atau elemen-elemen data.

3. Notasi Kamus Data

Notasi kamus data terbagi dalam dua bagian yaitu Notasi Tipe Data dan Notasi Struktur Data (Kendall & Kendall, 2010:344).

a. Notasi Tipe Data

Notasi Tipe Data adalah suatu bentuk untuk mempersingkat arti atau makna dari simbol yang dijelaskan . Adapun bentuk notasi sebagai berikut:

Tabel II.2.
Notasi Tipe Data

Notasi	Keterangan
X	Bisa memasukan atau menampilkan suatu karakter
9	Hanya memasukkan atau menampilkan angka
Z	Menampilkan nol-nol yang memimpin sebagai spasi
‘	Menyisipkan koma dalam suatu tampilan numerik
.	Menyisipkan periode kedalam tampilan numerik
/	Menyisipkan slash kedalam tampilan numerik
-	Menyisipkan tanda penghubung dalam tampilan numerik.
V	Menunjukan suatu posisi desimal.

Sumber: Kendall & Kendall (2010:344)

b. Notasi Struktur Data

Struktur dari data biasanya digambarkan dengan menggunakan notasi aljabar. Metode ini memungkinkan penganalisis membuat suatu gambaran mengenai elemen-elemen tersebut. Notasi aljabar mengggunakan simbol-simbol sebagai berikut :

Tabel II.3.
Notasi Struktur Data

Notasi	Keterangan
=	Terbentuk dari *(is composed)* atau terdiri dari *(consist of)* atau sama dengan *(is equivalent of)*
+	*AND*
[ ]	Salah satu dari (memilih salah satu dari elemen-elemen data di dalam kurung *bracket* ini)
I	Sama dengan simbol [ ] atau pemisah pada bentuk [ ]
N [ ] M	Iterasi (elemen data didalam kurung *bracket* beriterasi mulai minimun N kali dan maksimum M kali
( )	*Optional* (elemen data didalam kurung *parenthesis* sifatnya optional, dapat ada dan dapat tidak ada)
* atau **	Keterangan setelah tanda ini adalah komentar
@	*Identifier* data *store*
alias	Nama lain untuk data

Sumber : Kendall & Kendall (2010:388)

C. *Entity Relationship Diagram* (ERD) dan *Logical Record Structure* (LRS)

1. *Entity Relationship Diagram* (ERD)

Fathansyah (2007:79) mengungkapkan bahwa Model *Entity-Relationship* yang berisi komponen-komponen Himpunan Entitas dan Himpunan Relasi yang masing-masing dilengkapi dengan atribut-atribut yang merepresentasikan seluruh fakta dari ‘dunia nyata’ yang kita tinjau, dapat digambarkan dengan lebih sistematis dengan menggunakan Diagram *Entity-*

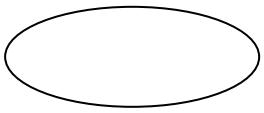
Relationship (Diagram-ER). Notasi-notasi simbolik didalam Diagram E-R yang dapat kita gunakan adalah:

a. Persegi Panjang, menyatakan Himpunan Entitas

Sumber : Fathansyah (2007:79)

Gambar II.2. Simbol Himpunan Entitas *(Entity)*

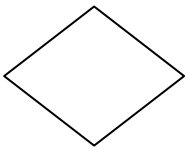
b. Lingkaran/*Elip*, menyatakan atribut (Atribut yang berfungsi sebagai *key* digaris bawahi)



Sumber : Fathansyah (2007:79)

Gambar II.3. Simbol *Lingkaran (Elip)*

c. Belah Ketupat, menyatakan Himpunan Relasi.



Sumber : Fathansyah (2007:79)

Gambar II.4. Simbol Himpunan Relasi

d. Garis, sebagai penghubung antara Himpunan Relasi dengan Himpunan Entitas dan Himpunan Entitas dengan atributnya.

Sumber : Fathansyah (2007:79)

Gambar II.5. Simbol Garis *(Link)*

e. Kardinalitas Relasi dapat dinyatakan dengan banyaknya garis cabang atau dengan pemakaian angka (1 dan 1 untuk relasi satu-ke-satu, dan N untuk relasi satu-ke-banyak atau N dan N untuk relasi banyak-ke-banyak).

Fathansyah (2007:41) mengungkapkan bahwa pada dasarnya *key* adalah satu atau gabungan dari beberapa atribut yang dapat membedakan semua baris data (*row)* dalam tabel secara unik. Artinya, jika suatu atribut dijadikan sebagai *key*, maka tidak boleh ada dua atau lebih baris data dengan nilai yang sama untuk atribut tersebut. Ada 3 (tiga) macam *key* yang dapat diterapkan pada suatu tabel, yaitu:

a. *Superkey*

   Superkey merupakan satu atau lebih atribut (kumpulan atribut) yang dapat membedakan setiap baris data dalam sebuah tabel secara unik. Bisa terjadi, ada lebih dari 1 kumpulan atribut yang bersifat seperti itu pada sebuah tabel.

b. *Candidate-Key*

   Candidate-Key merupakan kumpulan atribut minimal yang dapat membedakan setiap baris data dalam sebuah tabel secara unik. Sebuah *Candidate-Key* tidak boleh berisi atribut atau kumpulan atribut yang telah menjadi *Superkey* yang lain. Jadi, sebuah *Candidate-Key* pastilah *Superkey*, tapi belum tentu sebaliknya,

c. *Primary Key*

   Pada sebuah tabel dimungkinkan adanya lebih dari satu *Candidate-Key*. Salah satu dari *Candidate-Key* ini (jika memang ada lebih dari satu) dapat

dijadikan sebagai Key Primer *(Primary Key)*. Pemilihan Key Primer dari sejumlah *Candidate-Key* tersebut umumnya didasari oleh:

1. *Key* tersebut lebih sering (lebih natural) untuk dijadikan sebagai acuan.
2. *Key* tersebut lebih ringkas.
3. Jaminan keunikan *Key* tersebut lebih baik.

2. Logical Record Structure (LRS)

Menurut Simarmata (2007:15) "*Logical Record Structure* adalah resprentasi 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 baik yang unik. Beda LRS dengan diagram *entity relationship diagram* nama tipe *record* berada diluar kotak *field* tipe *record* ditempatkan.

LRS terdiri dari *link-link* diantara *tipe record.* Link ini menunjukkan arah dari satu *type record* lainya. Banyak link dari LRS yang diberi tanda *field-field* yang kelihatan pada kedua *link type 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.

LRS terdiri dari *link-link* diantara *tipe record. Link* ini menunjukkan arah dari satu *type record* lainya. Banyak link dari LRS yang diberi tanda *field-field* yang kelihatan pada kedua *link type 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 kaitanya 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) relasi atau tabel adalah bentuk pernyataan data secara grafis 2 (dua) dimensi, yang terdiri dari kolom dan baris. Relasi adalah bentuk *visual* dari sebuah f*ile*, dan tiap *record* dalam sebuah *field*, atau yang dalam bentuk lingkaran Diagram *entity relationship* dikenal dengan sebutan atribut.

   Konversi dari *logical record structure*. Dilakukan dengan cara:

   a. Nama *logical record structure* menjadi nama relasi.
   b. Tiap atribut menjadi sebuah kolom didalam relasi

D. Hierarki Input Proses Output (HIPO)

> Menurut Yasin (2012:280) HIPO-*Hierarchy plus Input Proses Output* adalah: Paket yang saling berisikan suatu set diagram yang secara grafis menjalankan suatu fungsi sistem dari tingkat umum ke tingkat khusus. Mula-mula tiap fungsi utama diidentifikasi dan kemudian dibagi lagi kedalam tingkatan fungsi yang lebih. Pada awalnya HIPO dikembangkan oleh *IBM* untuk mendokumentasikan fungsi program, namun pada masa sekarang HIPO dipakai sebagai alat bantu dan teknik dokumentasi pada tahap-tahap perkembangan sisterm informasi.

E. Pengkodean

Fathansyah (2015:105) mengungkapkan bahwa data/informasi yang dapat dilihat oleh pemakai awam *(user)* bisa berbeda dengan bagaimana data/informasi itu disimpan. Itulah yang dikenal dengan Abstraksi Data. Apa yang dilihat oleh pemakai awam bisa jadi merupakan hasil pengolahan yang tidak disimpan sama sekali dalam basis data. Atau bisa pula, dinyatakan dalam bentuk lain. Fathansyah mengungkapkan bahwa salah satu alasan mengapa menyatakan suatu data (atribut) dalam bentuk lain adalah untuk efisiensi ruang penyimpanan. Dan cara yang ditempuh untuk menyatakan suatu data dalam bentuk lain itu adalah melalui pengkodean *(data coding).*

Dari pemakaiannya, Fathansyah menyimpulkan bahwa dapat dibedakan adanya pengkodean eksternal *(user-define-coding)* dan pengkodean internal *(system coding).* Pengkodean eksternal mewakili pengkodean yang telah digunakan secara terbuka dan dikenal baik oleh pemakai awam *(end-user).* Ada 3 (tiga) bentuk pengkodean yang menurut Fathansyah (2015:105), yaitu:

1. Sekuensial

   Dimana pengkodean dilakukan dengan mengasosiakan data dengan kode terurut (biasanya berupa bilangan asli atau abjad), misalnya data nilai

mutu kuliah (‘Sempurna’,’Baik’,’Cukup’,’Kurang’,’Buruk’) dikodekan dengan ‘A’,’B’,’C’,’D’ dan ‘E’.

2. *Mnemonic*

   Dimana pengkodean dilakukan dengan membentuk suatu singkatan dari data yang ingin dikodekan, misalnya data jenis-kelamin (‘Laki-laki’ dan ‘Perempuan’) dikodekan dengan ‘L’ dan ‘P’.

3. Blok

   Dimana pengkodean dinyatakan dalam format tertentu, misalnya data no.induk mahasiswa dengan format XXYYYY yang terbentuk atas XX= dua digit terakhir angka tahun masuk dan YYYY= no.urut mahasiswa