BAB II

LANDASAN TEORI

2.1 Konsep Dasar program

Konsep pemrograman memegang peranan penting dalam merancang, menyusun, memelihara, dan mengembangkan suatu program, khususnya program aplikasi yang besar dan kompleks. Proses pemrograman komputer bukan hanya sekedar menulis suatu urutan instruksi yang harus dikerjakan oleh komputer, akan tetapi bertujuan untuk memecahkan suatu masalah serta membuat mudah pekerjaan yang diinginkan oleh pemakai (user). Yang menjadi alasan utama belajar bahasa pemrograman komputer adalah karena ingin memanfaatkan komputer sebagi alat bantu untuk menyelesaikan masalah.

2.1.1 Pengertian Program

Menurut Binanto (2009:1) Program adalah "himpunan atau kumpulan instruksi yang dibuat oleh *programmer* atau suatu bagian *executable* dari suatu *software".*

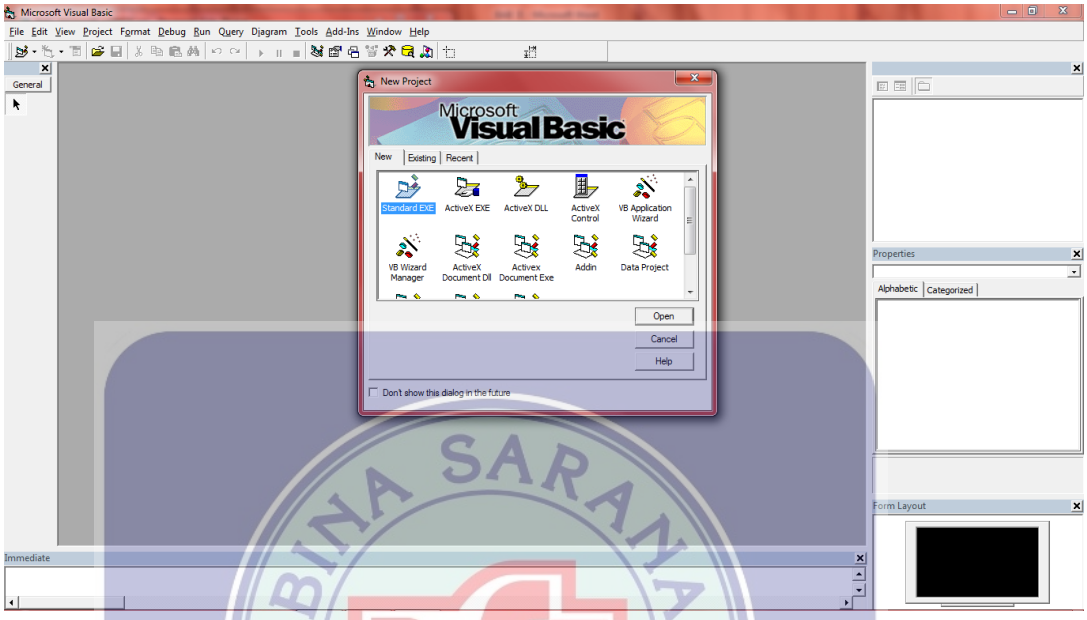
Menurut Kadir (2010:2) Program adalah "kumpulan perintah yang ditunjukan kepada komputer dapat melakukan tindakan sesuai yang kehendaki oleh pembuat perintah".

Adapun program yang digunakan antara lain:

a. Microsoft Visual Basic

Menurut Kurniadi (2011:5) Visual basic adalah "sebuah sarana pembuat

program yang lengkap dan namun mudah, siapapun yang bisa menggunakan windows, ia pasti bisa membuat program *Visual Basic*".



Sumber : Kurniadi (2011:5)

Gambar II.1
Tampilan awal Microsoft Visual Basic 6.0

Dalam pengembangan aplikasi, *Visual Basic* menggunakan pendekatan visual untuk merancang *user interface* dalam bentuk *form*, sedangkan untuk kodingnya menggunakan dialog bahasa basic yang cenderung mudah dipelajari. Layar Visual Basic adalah salah satu lingkungan besar yang terdiri dari beberapa bagian-bagian kecil yang memiliki sifat :

1. Floating : dapat di geser kemana saja
2. Sizeabkle : ukuran dapat diubah-ubah
3. Dockabel : dapat menempel pada bagian lain yang berdekatan

Pada *visual basic* versi 6.0 yang dipakai oleh penulis sekarang ini banyak keistimewaan, di bidang-bidang yang sangat berguna untuk programmer ataupun

pemakai, keistimewaan itu berupa :

1. Sarana akses data yang begitu cepat untuk membuat aplikasi *database* yang berkemampuan tinggi.
2. Memiliki *compiler* yang menghasilkan *file* eksekusi (*executable*) yang lebih cepat dan efisien.
3. Perlengkapan untuk mernacang aplikasi web juga tersedia sehingga pemakai dapat lebih mudah menyusun aplikasi internet yang bisa membuka, mentransfer *file*, menukar data situs web, *download* otomatis, perangkat lunak dan lain-lain.
4. Perangkat otomatis yang memungkinkan apabila *programmer* melupakan kode program dan sintaks prosedur, maka secara otomatis *Intellegent IDE* akan membantu.
5. Memiliki beberapa sarana *wizard* yang baru. *Wizard* adalah sarana yang mempermudah dalam mengotomatisasi tugas tertentu.

Bahasa pemrograman berbasis *visual*, khusunya V*isual Basic* kini tidak lagi mengguanakan orientasi linier dalam pembuatan programnya, melainkan dengan berorientasi objek –objek yang terpisah-pisah (*object oriented*), yang disebut juga *Object-Oriented Programming* (OOP), *Visual Basic* juga memiliki konsep *moduler programming*, dimana kode-kode program letak nya tersebar didalam modul-modul (objek) yang terpisah-pisah.

b. *Crystal Report*

Menurut Madcoms (2010:234) "*Crystal Report* merupakan program yang terpisah dengan program *Microsoft Visual Basic 6.0*, tetapi keduanya dapat dihubungkan (*Linkpage*)". Membuat laporan dengan *Crystal Report* hasilnya lebih

baik dan lebih mudah, karena pada *Crystal Report* banyak tersedia objek-objek maupun komponen yang mudah digunakan.

c. XAMPP

Menurut WK (2015:55) "XAMPP merupakan singkatan dari X (empat sistem operasi apapun), *Apache*, *MySQL*, *PHP*, Perl". XAMPP adalah tool yang menyediakan paket perangkat lunak dalam satu buah paket.

Pada paket XAMPP telah terdapat *Apache* (*web server*), *MySQL* (*database*), PHP (*server side scripting*), Perl, FTP server, PhpMyAdmin, dan berbagai pustaka bantu lainnya. Jika anda memiliki XAMPP, maka anda tidak perlu lagi melakukan instalasi dan melakukan konfigurasi web server Apache, PHP, dan MySQL secara manual.

2.1.2 Bahasa Pemrograman

Menurut kurniadi (2011:3) bahasa pemrograman adalah "perintah-perintah atau instruksi yang dimengerti oleh komputer untuk melakukan tugas-tugas tertentu". Bahasa pemrograman berbasis visual, khususnya visual basic kini tidak lagi menggunakan orientasi linier dalam pembuatan programnya, melainkan berorientasi objek-objek yang terpisah (object oriental), yang disebut juga dengan *Object-Oriented Programming* (OOP), visual basic juga memiliki konsep *moduler programming*, dimana kode-kode program letaknya tersebar didalam modul-modul (objek) yang terpisah-pisah.

Bahasa pemrograman komputer yang dikelompokkan berdasarkan perkembangannya, yaitu sebagai berikut :

1. Bahasa mesin (*Machine Language*)

   Bahasa pemrograman yang hanya dapat dimengerti oleh mesin (komputer)

yang didalamnya terdapat CPU (Central Prosesing Unit) yang hanya mengenal dua keadaan yang berlawanan, yaitu bernilai 1 bila terjadi kontak (ada arus) dan akan bernilai 0 bila kontak terputus (tidak ada arus).

2. Bahasa Tingkat Rendah (*Low Level Language*)

   Bahasa tingkat rendah sebenarnya sama dengan bahasa mesin sebelumnya yaitu bahasa pemrograman yang berorientasi pada mesin. Pemrograman yang digunakan bahasa ini dapat berpikir logika mesin komputer, sehingga bahasa ini dinilai kurang fleksibel dan sulit oleh dipahami oleh pemula.

3. Bahasa Tingkat Menengah (*Middle Level Language*)

   Bahasa pemrograman yang menggunakan aturan-aturan gramatikal dalam penulisan pernyataan, mudah untuk dipahami dan memiliki instruksi-instruksi tertentu yang dapat langsung diakses oleh komputer. Contohnya adalah bahas pemrograman C.

4. Bahasa Tingkat Tinggi (*High Level Language)*

   Bahasa pemrograman yang ada didalam penulisan pernyataannya mudah dipahami secara langsung.

5. Bahasa Berorientasi Objek (*Object Oriented Language*)

6. Bahasa pemrograman yang berorientasi pada obyek mengandung fungsi-fungsi untuk menyelesaikan suatu permasalahan dan program tidak harus menulis secara detail semua pernyatannya, tetapi cukup memasukan kriteria-kriteria yang dikehendaki saja. Contoh : *Visual foxpro, delphi, Visual C++ dan lain-lain.*

Adapun aplikasi yang digunakan antara lain:

Structured Query Languange (*SQL)*

Menurut WK (2015:56) “SQL merupakan bahasa terstruktur yang khusus digunakan untuk mengolah database.” SQL pertama kali didefinisikan oleh American National Standards Institute (ANSI) pada tahun 1986.

2.1.3 Basis Data

Menurut Sukamto dan M. Shalahuddin (2013:43) menyimpulkan bahwa :

Sistem basis data adalah sistem yang terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan. Pada intinya basis data adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat.

Menurut Fathansyah (2012:2) mengemukakan bahwa :
Basis data terdiri dari dua kata, yaitu basis dan data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang/berkumpul. Sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya, yang diwujudkan dalambentuk angka, huruf, simbol, teks, gambar, bunyi atau kombinasinya.

Adapun aplikasi yang digunakan adalah:

MySQL

Menurut WK (2015:56) “MySQL adalah sebuah sistem manajemen database yang bersifat open source.” MySQL merupakan pasangan serasi dari PHP. MySQL dapat digunakan untuk membuat danmengolah database beserta isinya. MySQL merupakan sistem manajemen database yang bersifat relational.

2.1.4 Model Pengembangan Perangkat Lunak

Menurut Sukamto dan M. Shalahudin (2013:25) menjelaskan bahwa “pada awal pengembangan perangkat lunak, para pembuat program (*programmer*) langsung melakukan pengkodean perangkat lunak tanpa menggunakan prosedur

atau tahapan pengembangan perangkat lunak". Dan ditemuilah kendala-kendala seiring dengan perkembangan skala sistem-sistem perangkat yang semakin besar.

1. SDLC

   SDLC atau *Software Development Life Cycle* atau sering disebut juga System *Development Life Cycle* adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sitem perangkat lunak sebelumnya (berdasarkan *best pratice* atau cara-cara yang sudah teruji baik). Seperti halnya proses metamorfosis pada kupu-kupu, untuk menjadi kupu-kupu yang indah maka dibutuhkan beberapa tahap untuk dilalui, sama halnya dengan membuat perangkat lunak, memiliki daur tahapan yang dilalui agar menghasilkan lunak yang berkualitas.

2. Waterfall

   Dalam perancangan aplikasi pada tugas akhir ini penulis menggunakan SDLC model *Water fall*. Menurut Rosa A.S dan M. Shalahuddin (2013:28) menjelaskan bahwa "model SDLC air terjun (*water fall*) sering juga disebut model sekuensial linier (*sequential linear*) 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 penjelasannya:

   a. Analisis kebutuhan perangkat lunak

      Proses pengumpulan kebutuhan dilakukan secara intensif untuk mespesifikasikan 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 pengodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desai 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 perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahapdesain.

d. Pengujian

Pengujian fokus pada perangkat lunak secara segi lojik 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 dapat mengulangi proses pengembangan mulai dari tahap analisis spesifikasi untuk perubahan perangkat lunak baru.

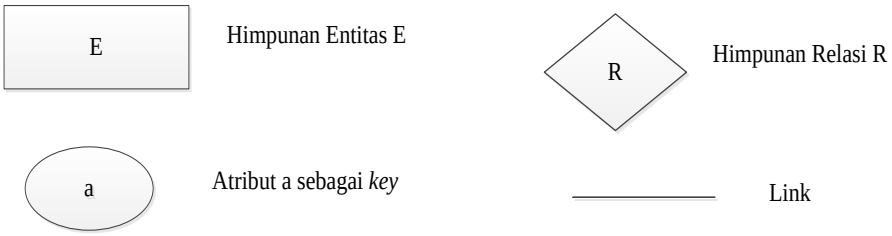
2.2 Tools Program

Dengan adanya tools program akan mempermudah proses pembuatan program. Penulis menggunakan peralatan pendukung untuk merancang model program yang akan dibuat tujuannya agar program yang dihasilkan menjadi lebih mudah diketahui.

2.2.1 ERD (*Entity Relationship Diagram)*

Menurut Fathansyah (2012:81) "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 E-R)". Notasi-notasi simbolik di dalam Diagram E-R yang dapat kita gunakan adalah:

1. Persegi panjang, menyatakan Himpunan Entitas.
2. Lingkaran/Elip, menyatakan Atribut (Atribut yang berfungsi sebagai *key* digarisbawahi).
3. Belah ketupat, menyatakan Himpunan Relasi.
4. Garis, sebagai penghubung antara Himpunan Relasi dengan Himpunan Entitas dan Himpunan Entitas dengan Atributnya.
5. 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).



Sumber : Fathansyah, 2012:82

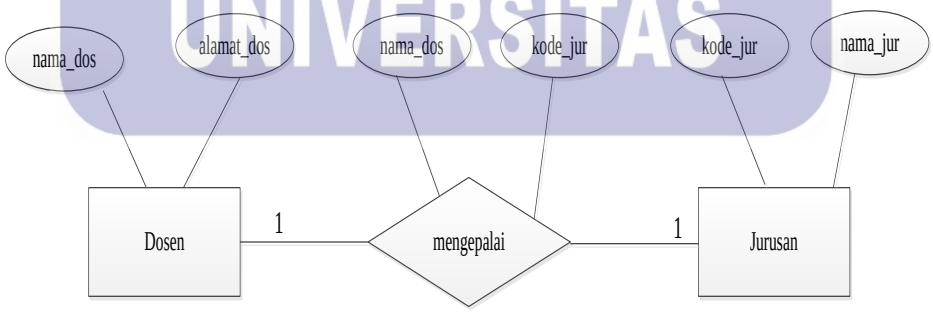
Gambar II.2.
Simbolik Diagram E-R

Berikut adalah contoh penggambaran relasi antar himpunan entitas lengkap dengan kardinalitas relasi dan atribut-atributnya.

1. Relasi satu-ke-satu (*one-to-one*)

   Contoh:

   Adanya relasi antara himpunan entitas Dosen dengan himpunan entitas Jurusan. Himpunan relasinya kita beri nama 'Mengepalai'. Para relasi ini, setiap dosen paling banyak mengepalai satu jurusan (walaupun memang tidak semua dosen yang menjadi ketua jurusan). Dan setiap jurusan pasti dikepalai oleh paling banyak satu orang dosen. Maka penggambarannya adalah:



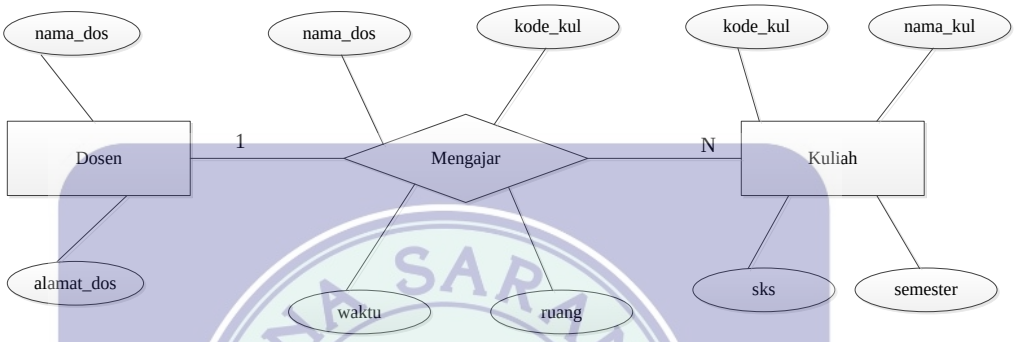
Sumber : Fathansyah, 2012:82

Gambar II.3.
Diagram E-R untuk Relasi Satu ke Satu

2. Relasi satu-ke-banyak (*one-to-many*)

   Contoh:

Adanya relasi antara himpunan entitas Dosen dengan himpunan entitas Kuliah. Himpunan relasinya kita beri nama ‘Mengajar’. Pada relasi ini, setiap dosen dapat mengajar lebih dari satu mata kuliah, sedang setiap mata kuliah diajar hanya oleh paling banyak satu orang dosen. Maka penggambarannya adalah:



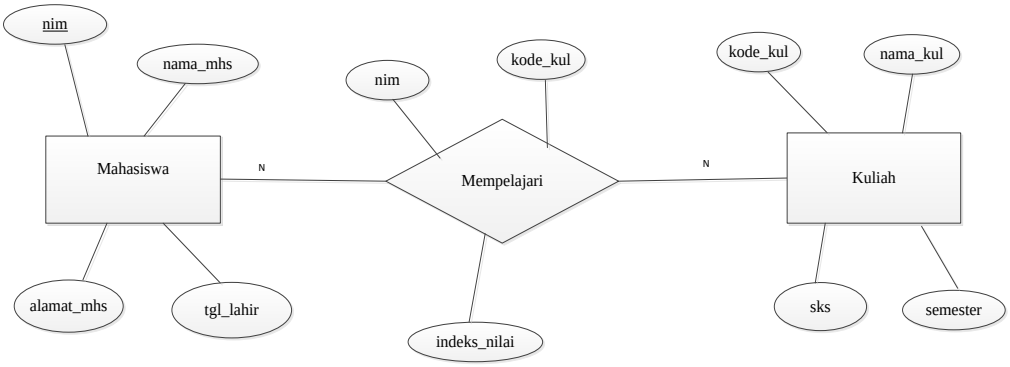
Sumber : Fathansyah, 2012:83

Gambar II.4.
Diagram E-R untuk Relasi Satu ke Banyak

3. Relasi banyak-ke-banyak (*many-to-many*)

   Contoh:

   Adanya relasi antara himpunan entitas Mahasiswa dengan himpunan entitas Kuliah. Himpunan relasinya kita beri nama ‘Mempelajari’. Pada relasi ini, setiap mahasiswa dapat mempelajari lebih dari satu mata kuliah. Demikian juga sebaliknya, setiap mata kuliah dapat dipelajari oleh lebih dari satu orang mahasiswa. Maka penggambarannya adalah:



Sumber : Fathansyah, 2012:85

Gambar II.5.
Diagram E-R untuk Relasi Banyak ke Banyak

2.2.2 LRS (*logical Record Structure*)

Menurut Ramakarishnan and Gehrke dalam aunur R. Mulyanto (2008:267) menyebutkan bahwa "logical Record Structure adalah konsep relationship pada model E-R berbeda dengan konsep relation didalam metode data relational. Relationship adalah mekanisme yang menghubungkan entitas".

logical Record Structure dibentuk dengan nomor dari tipe record. logical Record Structure terdiri dari link-link diantara tipe record. Link ini menunjukan 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 lainnya dimulai dengan ER-diagram dan langsung dikonversikan ke LRS. Menentukan kardinalitas, jumlah table dan foreign key.

Terdapat tiga kardinalitas didalam Logical Record Structure yaitu :

1. One To One (1-1)

Tingkat hubungan satu ke satu, dinyatakan dengan satu kejadian pada entitas

pertama, hanya mempunyai satu hubungan dengan satu kejadian pada entitas yang kedua sebaliknya.

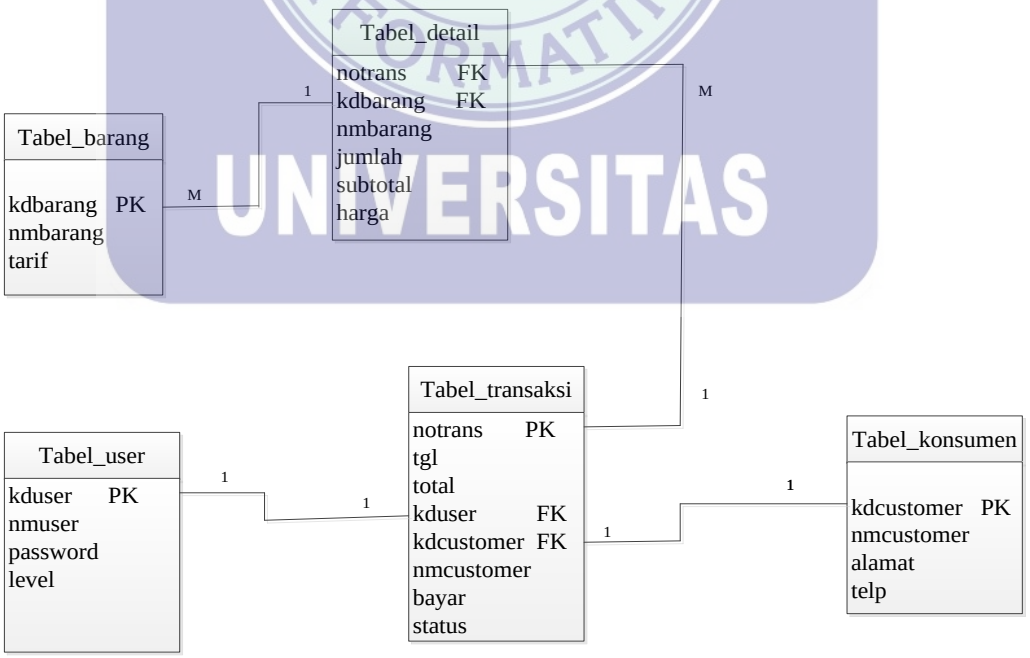
2. One To Many (1-M)

   Tingkat hubungan satu kebanyak adalah sama banyak ke satu. Tergantung dari arah mana hubungan tersebut dilihat. Untuk satu kejadian pada entitas yang pertama dapat mempunyai dapat hubungan dengan kejadian pada entitas yang kedua. Sebaliknya satu kejadian pada entitas yang kedua hanya dapat mempunyai satu hubungan dengan satu kejadian pada entitas yang pertama.

3. Many To Many (M-M)

   Tingkat hubungan kebanyakan terjadi jika tiap kejadian pada sebuah entitas akan mempunyai banyak hubungan dengan kejadian pada entitas lainnya. Baik dilihat dari sisi entitas yang pertama, maupun dilihat dari sisi yang kedua.

Berikut contoh gambaran mengenai LRS (*Logical Record Structure*).



Sumber : Simarmata dan Paryudi

Gambar II.6.
Contoh gambaran LRS (*Logical Record Structure*)

2.2.3 Pengkodean

Menurut Mustakini (2008:384) “Struktur kode bertujuan untuk mengklasifikasikan data, memasukkan data ke dalam komputer untuk mengambil informasiyang berhubungan dengannya”.

Berikut beberapa petunjuk pembuatan struktur kode yang baik, antara lain:

1. Harus mudah diingat

   Agar kode mudah diingat, maka dapat dilakukan dengan cara menghubungkan kode tersebut dalam objek yang mewakili dengan kodenya.

2. Harus unik

   kode harus unik untuk masing-masing item yang diwakilinya. Unik berarti tidak ada kode yang kembar.

3. Harus fleksibel

   Kode harus fleksibel sehingga memungkinkan perubahan-perubahan atau penambahan itembaru dapat diwakili oleh kode.

4. Harus efisien

   Kode harus sependek mungkin, selain mudah diingat juga akan efesien bila direkam disimpanan luar komputer.

5. Harus konsisten

   Kode harus konsisten dengan kode yang telah dipergunakan.

6. Harus distandarisari

   Kode harus distandarisasikan untuk seluruh tingkatan dandepartemen dalam organisasi. Kode yang tidak standar akan mengakibatkan kebingungan, salah pengertian dan dapat cenderung terjadi kesalahan pemakaian yang menggunakan kode tersebut.

7. Spasi dihindari

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

8. Hindari karakter mirip

   Karakter-karakter yang hampir serupa bentuk dan bunyi pengucapannya sebaiknya tidak digunakan dalam kode.

9. Panjang kode harus sama

   Masing-masing kode sejenis harus mempunyai panjang yang sama.

2.2.4 Hieraki Input Proses Output (HIPO)

Menurut Puspitawati dan Sri Dewi Anggadini (2011:114) adalah "merupakan serangkaian diagram yang terdiri dari serangkaian level yang mengalir dari atas ke bawah yang menggambarkan sistem yang lebih detail". Diagram HIPO dirancang sebagai alat bantu danalat dokumentasi yang digunakan untuk menyelesaikan suatu masalah/problem. Selain itu diagraminijuga digunakan untuk menguraikan keseluruhan pemrosesan transaksi yang terjadi dalam aktifitas perusahaan.

2.2.5 Flowchart

Menurut Sitorus (2015:14) "Flowchart adalah menggambarkan urutan logika dari suatu prosedur pemecahan masalah, sehingga flowchart merupakan langkah-langkah penyelesaian masalah yang dituliskan dalam simbol-simbol tertentu".

Flowchart menggunakan simbol yang berbeda yang berisi informasi tentang langkah-langkah atau urutan kejadian. Masing-masing dari simbol-simbol ini terkait dengan panah untuk menggambarkan arah aliran proses. Bagan alur

(*flowchart*) adalah bagan (*chart*) yang menunjukan hasil (*flow*) didalam program atau prosedur sistem secara logika. Bagan alir digunakan terutama untuk alat bantu komunikasi dan untuk dokumentasi.

Ada beberapa jenis flowchart diantaranya:

1. Bagan alir sistem (*system flowchart*)

   System flowchart dapat didefinisikan sebagai bagan yang menunjukkan arus pekerjaan secara keseluruhan dari sistem. Bagan ini menjelaskan urut-urutan dari prosedur-prosedur yang ada didalam sistem. Bagan alir sistem menunjukkan apa yang dikerjakan di sistem.

2. Bagan alir dokumen (*document flowchart*)

   Bagan alir dokumen (*document flowchart*) atau disebut juga bagan alir formulir (*form flowchart*) atau paperwork flowchart merupakan bagan alir yang menunjukkan arus dari laporan dan formulir termasuk tembusan-tembusannya.

3. Bagan alir skematik (*schematic flowchart*)

   Bagan alir skematik (*schematic flowchart*) merupakan bagan alir yang mirip dengan bagan alir sistem, yaitu untuk menggambarkanprosedur didalam sistem. Perbedaannya adalah, bagan air skematik selain menggunakan simbol-simbol bagan alir sistem, juga menggunakan gambar-gambar komputer dan peralatan lainnya yang digunakan.

4. Bagan alir program (*program flowchart*)

   Bagan alir program (*program flowchart*) merupakan bagan yang menjelaskan secara rinci langkah-langkah dari proses program. Bagan alir program dibuat dari derivikasi bagan alir sistem.bagan alir program dapat terdiri dari dua

macam, yaitu bagan alir logika program (*program logic flowchart*) dan bagan alir program komputer terinci (*detail computer program flowchart*).

5. Bagan alir proses (*process flowchart*)

   Bagan alir proses (*process flowchart*) merupakan bagan alir yang banyak digunakan diteknik industri. Bagan alir ini juga berguna bagi analis sistem untuk menggambarkan proses dalam suatu prosedur.