BAB II

LANDASAN TEORI

2.1 Konsep Dasar Program

A. Program

Menurut Fakhri (2013:4) “Program adalah algoritma yang ditulis dalam satu bahasa komputer yang dapat dijalankan pada komputer”.

Program merupakan kata ekpresi, atau pernyataan yang disusun dan di rangkai menjadi satu kesatuan prosedur, yang berupa urutan langkah, untuk menyelesaikan masalah yang di implementasikan dengan menggunakan bahasa pemrograman sehingga dapat di eksekusi oleh komputer.

B. Bahasa Pemrograman

Menurut Setiawan (2014:1) mengemukakan bahwa “Bahasa pemrograman adalah teknik komando/intruksi standar untuk memerintah komputer yang merupakan suatu himpunan dari aturan sintaks dan sistematik yang dipakai untuk mendefinisikan program komputer”.

1. Java

   Menurut Mardiani dkk (2017:27) menyimpulkan bahwa “Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer, termasuk telpon genggam. Dikembangkan oleh Sun Microsystem dan rilis tahun 1995. Java berbeda dengan JavaScript. JavaScript adalah bahasa *scripting* yang digunakan oleh *web*”.

   Java merupakan bahasa pemrograman yang berbasis objek. Berikut ini konsep-konsep dasar pemrograman berbasis objek:

a. Objek

b. Kelas

c. Abstraksi Data

d. Enkapsulasi Data

e. Pewarisan

f. Polimorfisme

2. NetBeans

Menurut Jubilee Interprise (2014:25) menyimpukan bahwa, "NetBeans merupakan IDE(*Integrated Development Environment*) untuk membuat aplikasi dengan java, PHP, C, C++, dan HTML 5".

3. iReport

Report / laporan sangat diperlukan dalam suatu aplikasi sistem informasi. Tools yang cukup dikenal untuk membuat laporan, yaitu Crystal Report dan biasanya digabungkan dengan Visual Basic, namun untuk menggunakan Crystal Report harus mengeluarkan sejumlah uang untuk lisensinya.

Menurut Mardiani dkk (2017:54) menyimpulkan bahwa, "iReport adalah report designer visual yang dibangun pada JasperReports yang mingisi kekurangan itu. Ini adalah intuitif dan mudah digunakan membangun laporan visual / desainer untuk JasperReports, tertulis dalam kitab Java".

Sebagai alternatif, terdapat tools iReport (dengan *library* JasperReport) yang dapat pula membantu kita dalam pembuatan laporan. *Library* JasperReport sendiri merupakan Java Library (JAR) yang bersifat open

dan dirancang untuk menambahkan kemampuan pelaporan (reporting capabilities) pada aplikasi java.

JasperReport memiliki sejumlah fitur antara lain:

a. Layout dan desain laporan fleksibel.
b. Dapat menampilkan laporan dalam bentuk teks maupun gambar (*chart*).
c. Dapat menghasilkan report dalam berbagai format html,pdf,rtf,xls,csv.
d. Dapat menerima data dari berbagai sumber data JDBC, BeanCollection, ResultSet, CSV, XML, Hibernate.

C. Basis Data

Menurut Fathansyah (2015:3) menyimpulkan bahwa, "Basis Data (*Database)* adalah Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan".

Basis Data dan lemari arsip sesungguhnya memiliki prinsip kerja dan tujuan yang sama. Prinsip utamanya adalah pengaturan data/arsip. Dan tujuan utamanya adalah kemudahan dan kecepatan dalam pengambilan kembali data/arsip. Perbedaannya hanya terletak pada penyimpanan yang digunakan. Jika lemari arsip menggunakan lemari dari besi atau kayu sebagai media penyimpanan, maka basis data menggunakan media penyimpanan elektronis seperti cakram magnetis (*magnetic disk* atau disingkat sebagai *disk* saja). Hal ini merupakan konsekuensi yang logis karena lemari arsip langsung dikelola oleh manusia, sementara basis data dikelola melalui perantara mesin pintar elektronis (yang kita

kenal sebagai komputer). Perbedaan media ini yang selanjutnya melahirkan perbedaan-perbedaan lain yang menyangkut jumlah dan jenis metode yang dapat digunakan dalam upaya penyimpanan.

Satu hal yang juga harus diperhatikan, bahwa basis data bukan hanya sekedar penyimpanan data secara elektronis (dengan bantuan komputer). Artinya, tidak semua bentuk penyimpanan data secara elektronis bisa disebut basis data. Kita dapat menyimpan dokumen berisi data dalam *file* teks (dengan program pengolahan kata), *file spread sheet,* dan lain-lain, tetapi tidak bisa disebut sebagai basis data. Hal ini, karena di dalamnya tidak ada pemilahan dan pengelompokkan data sesuai jenis data. Kelak ketika *file-file* tersebut sudah cukup banyak, maka situasi ini tentu akan menyulitkan pencarian data tertentu. Yang sangat ditonjolkan dalam basis data adalah pengaturan, pemilihan, pengelompokkan, pengorganisasian data yang akan kita simpan sesuai fungsi/jenisnya. Pemilhan, pengelompokkan, pengorganisasian ini dapat berbentuk sejumlah tabel terpisah atau dalam bentuk pendefinisian kolom-kolom (*field*) data dalam setiap tabel.

Untuk mengolah *database* diperlukan suatu perangkat lunak yang disebut DBMS (*Database* Management System). DBMS merupakan suatu system perangkat lunak yang memunginkan user (pengguna) untuk membuat, memelihara, mengontrol, dan mengakses *database* secara praktis dan efisien. Dengan DBMS, user akan lebih mudah mengontrol dan memanipulasi data yang ada.

1. Operasi Dasar Basis Data

   Didalam sebuah *disk*, basis data dapat diciptakan dan dapat pula ditiadakan. Dapat pula menempatkan beberapa (lebih dari satu) basis data.

Sementara dalam sebuah basis data, dapat ditempatkan satu atau lebih dari tabel. Pada tabel inilah sesungguhnya data disimpan. Setiap basis data umumnya dibuat untuk mewakili sebuah semesta data yang spesifik. Misalnya, ada basis data kepegawaian, basis data akademik, basis data inventory (pergudangan), dan sebagainya. Sementara dalam basis data akademik, misalnya dapat menempatkan tabel mahasiswa, tabel mata_kuliah, tabel dosen, tabel jadwal, tabel kehadiran, tabel nilai, dan seterusnya.

Operasi-operasi dasar basis data dapat meliputi :

a. Pembuatan basis data baru (*create database*), yang identik dengan pembuatan lemari arsip yang baru.
b. Penghapusan basis data (*drop database*), yang indentik dengan perusakan lemari arsip (sekaligus beserta isinya, jika ada)
c. Pembuatan tabel baru ke suatu basi data (*create table*), yang identik dengan penambahan map arsip baru ke sebuah lemari arsip yang telah ada.
d. Penghapusan tabel dari sebuah basis data (*drop table*), yang identik dengan perusakan map arsip lama yang ada di sebuah lemari arsip.
e. Penambahan/pengisian data baru ke sebuah tabel (*query*), yang identik dengan pencairan lembaran arsip dari sebuah map arsip.
f. Pengubahan data dari sebuah tabel (*update*), yang identik dengan perbaikan isi lembaran arsip yang ada di sebuah map arsip.
g. Penghapusan data dari sebuah tabel (*delete*), yang identik dengan penghapusan sebuah lembaran arsip yang ada disebuah map arsip.

Operasi yang berkenaan dengan pembuatan objek (basis data dan tabel) merupakan operasi awal yang hanya dilakukan sekali dan berlaku seterusnya. Sedang operasi-operasi yang berkaitan dengan isi tabel (data) merupakan operasi rutin yang akan berlangsung berulang-ulang dank arena itu operasi-operasi inilah yang lebih tepat mewakili aktivitas pengolahan (*management*) dan pengolahan (*processing*) data dalam basis data.

2. Objektif Basis Data

Telah disebutkan di awal bahwa tujuan dan utama dalam pengelolaa data dalam sebuah basis data adalah agar kita dapat memperoleh menemukan kembali data (yang kita cari) dengan mudah dan cepat. Di samping itu, pemanfaatan basis data untuk pengolahan data, juga memiliki tujuan-tujuan lain. Secara lebih lengkap, pemanfataan basis data dilakukan untuk memenuhi sejumlah tujuan (objektif) seperti berikut ini:

a. Kecepatan dan kemudahan (*speed*)

   Pemanfaatan basis data memungkinkan kita untuk dapat menyimpan data atau melakukan perubahan/manipulasi terhadap data atau menampilkan kembali data tersebut dengan lebih cepat dan mudah, dari pada jika kita menyimpan data secara manual (non-elektronis) atau secara elektronis (tetapi tidak dalam bentuk penerapan basis data, misalnya dalam bentuk *spread sheet* atau dokumen teks biasa).

b. Efisiensi Ruang Penyimpanan (*space*)

   Karean keterkaitan yang erat antarkelompok data dalam sebuah basis data, maka redudansi (pengulangan) data pasti akan selalu ada.

Banyaknya redudansi ini tentu akan memperbesar ruang penyimpanan (baik di memori utama maupun memori sekunder) yang harus disediakan. Dengan basis data, efisiensi/optimalisasi penggunaan ruang penyimpanan dapat dilakukan, karena kita dapat melakukan penekanan jumlah redudansi data, baik dengan menerapkan sejumlah pengodean atau dengan mempuat relasi-relasi (dalam bentuk tabel) antarkelompok data yang saling berhubungan.

c. Keakuratan (*Accuracy*)

Pemanfaatan pengkodean atau pembentukan relasi antar data bersama dengan penerapan aturan/batasan (*constraint*) tipe data, *domain* data, keunikan data, dan sebagainya, yang secara ketat dapat diterapkan dalam sebuah basis data, sangat berguna untuk menekan ketidakakuratan penyimpanan data.

d. Ketersediaan (*Availability*)

Pertumbuhan data (baik dari sisi jumlah maupun jenisnya) sejalan dengan waktu akan semakin membutuhkan ruang penyimpanan yang besar. Padahal tidak semua data itu selalu kita butuhkan. Karena itu kita dapat memilah adanya data utama/master/referensi, data transaksi, data histori hingga data yang kadaluarsa. Data yang sudah jarang atau bahkan tidak pernah lagi kita gunakan, dapat kita atur untuk dilepaska dari sistem basis data yang sedang aktif (menjadi *off-line*) baik dengan cara penghapusa atau dengan memindahkannya ke media penyimpanan off-line (seperti *removable disk*, atau *tape*). Di sisi lain, karena kepentingan pemakaian data, sebuah basis data dapat memiliki

data yang disebar di banyak lokasi geografis. Data nasabah sebuah bank misalnya, dipisah-pisah dan disimpan dilokasi yang sesuai dengan keberadaan nasabah. Dengan pemanfaatan teknologi jaringan computer, data yang berada disuatu cabang, dapat diakses (menjadi tersedia/*available*) bagi cabang lain.

e. Kelengkapan (*Completeness*)

Lengkap atau tidaknya data yang kita kelola dalam sebuah basis data bersifat relative (baik terhadap kebutuhan maupun terhadap waktu). Seorang pemakai mungkin sudah menganggap bahwa data yang dikelola sudah lengkap, tetapi pemakai yang lain belum tentu berpendapat sama. Atau, yang sekarang dianggap sudah lengkap, belum tentu dimasa yang akan datang juga demikian. Dalam sebuah basis data, di samping data kita juga harus menyimpan struktur yang baik (baik yang mendefinikan objek-objek dalam basis data maupun definisi detail dari tiap objek, seperti struktur *file*/tabel dan indeks). Untuk mengakomodasikan kebutuhan kelengkapan data yang semakin berkembang, maka kita tidak hanya dapat menambah *record-record* data, tetapi juga dapat melakukan perubahan struktur dalam basis data, baik dalam bentuk penambahan objek baru (tabel) atau dengan penambahan *field-field* baru pada suatu tabel.

f. Keamanan (*Security*)

Memang ada sejumlah sistem (aplikasi) pengolahan basis data yang tidak menerapkan aspek keamanan dalam penggunaan basis data. Akan tetapi untuk sistem yang besar dan serius, aspek keamanan juga

dapat diterapkan dengan ketat. Dengan begitu, kita dapat menentukan siapa-siapa (pemakai) yang boleh menggunakan jenis-jenis operasi apa saja yang boleh dilakukannya.

g. Kebersamaan Pemakaian (*Sharability*)

Pemakaian basis data seringkali tidak terbatas pada suatu pemakai saja, atau satu lokasi saja atau oleh satu sistem/aplikasi. Data pegawai dalam basis data kepegawaian, misalnya, dapat digunakan oleh bnyak pemakai, dari sejumlah departemen dalam perusahaan atau oleh banyak sistem (sistem penggajian, sistem akuntansi, sistem inventory, dan sebagainya). Basis data yang dikelola oleh sistem (aplikasi) yang mendukung lingkungan *multi-user*, akan dapat memenuhi kebutuhan ini, tetapi tetap dengan menjaga/menghindari munculnya persoalan baru seperti inkonsistensi data (karena data yang sama diubah oleh banyak pemakai pada saat yang bersamaan) atau kondisi *deadlock* (karena ada banyak pemakai yang saling menunggu untuk menggunakan data).

E. MySQL

Menurut Mardiani dkk (2017:37) menyimpulkan bahwa “SQL merupakan kependekan dari kata “*Structured Query Language*”. SQL merupakan suatu bahasa permintaan yang terstruktur. Dikatakan terstruktur karena pada penggunanya, SQL memiliki beberapa aturan yang telah distandarkan oleh asosiasi yang bernama ANSI”.

SQL juga dapat diartikan sebagai Antar muka standar untuk sistem manajemen basis data relasional, termasuk sistem yang beroperasi pada computer pribadi. SQL memungkinkan seorang pengguna untuk mengakses informasi tanpa mengetahui di mana lokasinya atau bagaimana informasi tersebut disusun. SQL lebih mudah digunakan dibandingkan dengan bahasa pemrograman, tetapi lebih rumit dibanding software lembar kerja dan pengolah kata. Sebuah pernyataan SQL yang sederhana dapat menghasilkan set permintaan untuk informasi tersimpan pada komputer yang berbeda di berbagai lokasi yang tersebar sehingga membutuhkan waktu dan sumber daya komputasi yang banyak. SQL dapat digunakan untuk investigasi interatif atau pembuatan lapiran *ad hoc* atau disisipkan dalam program aplikasi.

1. Perintah Dasar SQL

   Telah dikatakan sebelumnya bahwa SQL merupakan sebuah bahasa permintaan yang melekat pada suatu SMBD termasuk MySQL. Perintahnya dapat kita sebut dengan query. Dalam penggunaanya, perintah SQL dikategorikan menjadi tiga sub perintah DDL (*Data Definition Language*), DML (*Data Manipulation Language*), dan DCL(*Data Control Language*)

2. *Data Definition Language* (DDL)

   Data Definition Language (DDL) merupakan sub bahasa SQL yang digunakan untuk membangun kerangka database. Ada tiga perintah yang termasuk dalam DDL, yaitu sebgai berikut:

   a. CREATE : Perintah ini digunakan untuk membuat,termasuk diantaranya membuat database baru, tabel baru, view baru, dan kolom.

b. ALTER : Perintah ini digunakan untuk mengubah struktur tabel yang telah dibuat. Pekerjaannya mencakup mengganti nama tabel, menambah kolom, mengubah kolom, menghapus kolom, maupun memberikan attribut pada kolom.

c. DROP : Perintah ini digunakan untuk menghapus database dan tabel.

3. *Data Manipulation Language* (DML)

Data Manipulation Language (DML) merupakan sub bahasa SQL yang digunakan untuk memanipulasi data dalam database yang telah dibuat. Perintah yang digunakan di antaranya sebagai berikut:

a. INSERT : Perintah ini digunakan untuk menyisipkan atau memasukan data baru ke dalam tabel. Penggunaanya setelah database dan tabel selesai dibuat.

b. SELECT : Perintah ini digunakan untuk mengambil data atau menampilkan data dari satu tabel atau beberapa tabel dalam relasi. Data yang diambil dapat kita tampilkan dalam layar prompt MySQL secara langsung maupun ditampilkan pada tampilan aplikasi.

c. UPDATE : Perintah ini digunakan untuk memperbaharui data lama menjadi data terkini. Jika memiliki data yang salah atau kurang *Up To Date* dengan kondisi sekarang maka dapat di ubah isi datanya dengan menggunakan perintah UPDATE.

d. DELETE : Perintah ini digunakan untuk menghapus data dari tabel. Biasanya data yang di hapus adalah data yang tidak diperlukan lagi. Pada saat menghapus data, perintah yang telah di jalankan tidak dapat

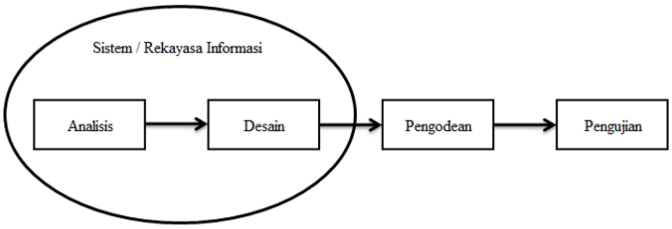
digagalkan sehingga data yang telah hilang tidak dapat dikembalikan lagi.

Selain untuk mengambil informasi dari database, Anda juga dapat menggunakan perintah SQL untuk memanipulasi data. Proses tersebut meliputi menambah, menghapus, dan mengedit.

Perintah manipulasi data sangat sering digunakan dalam aplikasi database bahkan dapat dikatakan menjadi inti sebuah aplikasi. Sebuah tabel dapat diisi dengan datam dihapus, maupun di edit datanya. Perintah tersebut dilaksanakan berdasarkan criteria tertentu menggunaan Keyword WHERE, BETWEEN, maupun LIKE.

F. Metode Pengembangan Perangkat Lunak

Menurut Sukamto dan M. Shalaludin (2016:28) mengemukakan bahwa "Model SDLC air terjun (*waterfall*) sering juga disebut model sekuensial linear (*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, pengodean, pengujian, dan tahapan pendukung (*support*)".



Sumber : (Dermawan & Hartini, 2017)

Gambar II.1.

Ilustrasi Model *Waterfall*

Berikut adalah penjelasan dari tahapan-tahapan tersebut:

1. Analisa kebutuhan perangkat lunak

   Proses pengumpulan kebutuhan dilakukan secara intensif untuk mespesifikasikan kebutuhan perangkat lunak seperti apa yang di butuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

2. Desain

   Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan program perangkat lunak, representasi antarmuka, dan prosedur pengodean. 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 di buat pada tahap desain.

4. Pengujian

   Pengujian fokus pada perangkat lunak dari 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.

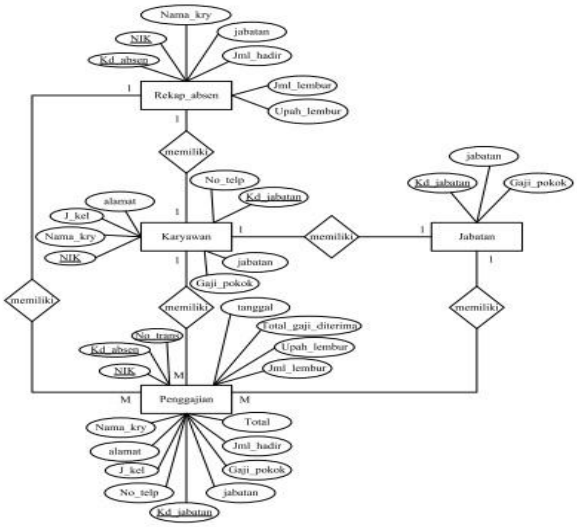
5. Pendukung (*support*) atau pemeliharaan(*maintenance*)

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

2.2 Tools Program

A. *ERD* (*Entity Relationship Diagram*)

Menurut Pratama (2014:49) menyimpulkan bahwa, "ERD (*Entity Relationship Diagram*) adalah diagram yang menggambarkan keterkaitan antar tabel beserta dengan *field-field* didalamnyam pada suatu database sistem". Contoh ERD adalah:



Sumber : (Entas & Alawiah, 2015)

Gambar II.2.

Contoh *Entity Relationship Diagram*

Sebuah database memuat minimal sebuah tabel dengan sebuah atau beberapa buah *field* (kolom) di dalamnya. Namun pada kenyataanya, databaselebih sering memilliki dari satu buah tabel (dengan beberapa *field* di dalamnya). Setiap tabel umumnya memiliki keterkaitan hubungan. Keterkaitan antartabel ini biasa disebut dengan relasi. Terdapat tiga buah jenis relasi antar tabel di dalam bagan ERD. Ketiga relasi tersebut yaitu:

1. *One to One* (satu ke satu)

   Relasi ini menggambarkan hubungan satu *field* pada tabel pertama ke satu *field* pada tabel kedua. Relasi ini paling sederhana. Sebagai contoh, pada sistem informasi perpustakaan terdapat tabel Buku (dengan *field* Kode_Buku, Kode_Kategori, Kode_Penulis, Nama_Penulis, Judul, Penerbit) dan tabel Kategori (Kode_Kategori, Nama_Kategori, Alamat). *Field* Kode_Kategori memiliki keterkaitan (relasi) satu ke satu pada tabel Buku dan tabel Kategori.

2. *One to Many* (satu ke banyak)

   Relasi ini menggambarkan hubungan satu *field* pada tabel pertama ke dua atau beberapa buah *field* di tabel kedua.

3. *Many to Many* (banyak ke banyak)

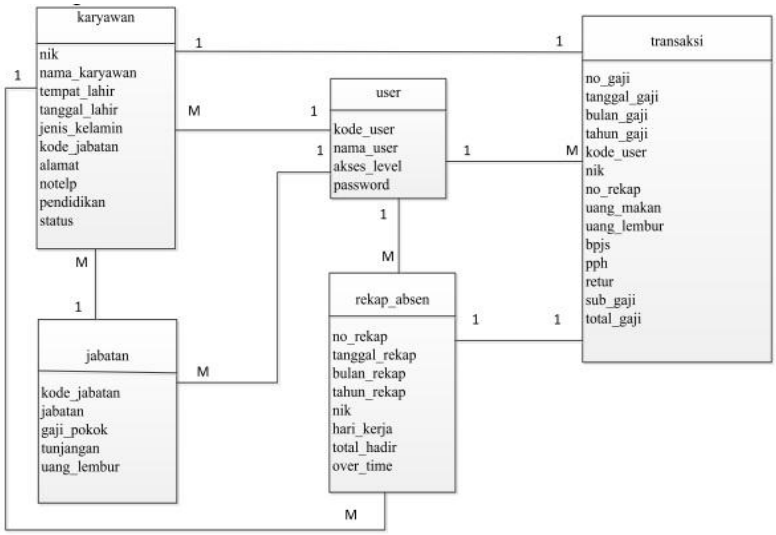
   Relasi ini menggambarkan hubungan satu *field* pada tabel pertama ke *Many to Many* (banyak ke banyak).

   Sebagai contoh, sebuah sistem informasi sekolah memiliki pengguna guru dan siswa di dalamnya. Sistem informasi ini memiliki sebuah database bernama sisfosekolah dengan tiga buah tabel di dalamnya. Ketiga tabel tersebut adalah tabel Guru (memuat *field* NIP, Nama_Guru, Jabatan,

Pangkat_Golongan, Alamat), tabel Mata Pelajaran (memuat *field* Kode_Mata_Pelajaran, Nama_Mata_Pelajaran), dan tabel Mengajar (memuat *field* NIP, Kode_Mata_Pelajaran, Kelas).

B. *LRS (Logical Record Structure)*

LRS merupakan hasil dari transformasi dalam tahapan kardinalitas dari ERD ke LRS dan menghasilkan attribute-attribut yang saling berelasi. Contoh LRS adalah:



Sumber : (Hidayatun, 2016)

Gambar II.3.

Contoh *Logical Record Structure*

Menurut Ladjamudin (2013:159) "*Logical Record Structure* (LRS) merupakan hasil transformasi ERD ke LRS yang melalui proses kardinalitas dan menghasilkan attribute-attribut yang saling berelasi".

Aturan pokok dalam melakukan transformasi E-R Diagram ke *Logical Record Structure* sangat dipengaruhui oleh elemen yang menjadi titik perhatian

utama pada langkah transformasi dengan proses kardinalitas, yang terdiri dari tiga kardinalitas yaitu sebagai berikut Ladjamudin (2013:159):

1. *One to one*

   Yaitu proses kardinalitas yang panahnya lebih diarahkan di Entity dengan jumlah attribute yang lebih sedikit

2. *One to Many*

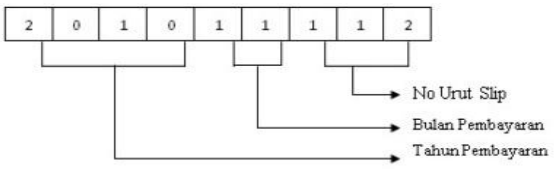
   Relasi harus digabungkan dengan *entity* pada pihak *many*, dan tidak perlu melihat banyak sedikitnya pada *entity* tersebut.

3. *Many to Many*

   Yaitu proses kardinalitas pada *relationship* berubah status menjadi file konektor, sehingga baik *entity* maupun relasi akan menjadi struktur *record* sendiri.

C. Pengkodean

Struktur kode digunakan untuk perancangan kode unik agar setiap data yang masuk ke dalam basis data memiliki identifikasi masing-masing dan mencegah terjadinya redundansi data dimana kode yang dibuat tersusun dari aturan-aturan yang di rancang berdasarkan elemen-elemen tertentu yang digunakan oleh perancang. Contoh pengkodean adalah:



Sumber : (Muspiawati & Arifin, 2015)

Gambar II.4.

Contoh Pengkodean No Slip Gaji

Menurut Shatu (2016:106) mengemukakan bahwa “kode memudahkan proses pengolahan data karena dengan kode, data akan lebih mudah di identifikasi”.

Dapat disimpulkan bahwa struktur kode merupakan teknik untuk menyusun kode setiap data agar data tersebut bersifat unik yang terdiri dari himpunan karakter dan simbol yang digunakan untuk mengidentifikasi objek tertentu agar data lebih mudah untuk diidentifikasi.

Dalam pembuatan sebuah kode yang baik memiliki persyaratan-persyaratan tertentu atau faktor-faktor yang perlu di pertimbangkan. Adapun faktor-faktor pertimbangan dalam pembuatan kode yaitu:

1. Kode yang disusun perlu disesuaikan dengan metode proses data.
2. Setiap kode harus mewakili hanya satu *item* sehingga tidak membingungkan.
3. Kode yang disusun harus memudahkan pemakai untuk mengingatnya.
4. Kode yang disusun harus fleksibel, dalam arti memungkinkan dilakukan perluasan tanpa perubahan menyeluruh.
5. Setiap kode harus menggunakan jumlah angka dan huruf yang sama.

Kode dapat dibuat dalam berbagai struktur kode yang berbeda. Setiap struktur mempunyai kelebihan dan kelemahan. Oleh karena itu perlu suatu struktur kode yang sesuai sehingga tujuan pemberian kode dapat tercapai. Berikut ini adalah macam-macam kode yang dapat digunakan:

1. Kode urut nomor

   Kode yang terbentuk dari susunan angka/nomor. Setiap kode memiliki jumlah angka yang sama (digit).

2. Kode kelompok

   Tiap kelompok akan diberi kode dengan angka atau huruf tertentu, sehingga masing-masing posisi angka/huruf dari kode mempunyai arti.

3. Kode blok

   Setiap kelompok data diberi kode dalam blok nomor tertentu. Kode blok mirip dengan kode kelompok.

4. Kode Decimal

   Setiap kelompok data akan diberi kode 0 sampai dengan 9. Oleh karena itu pengelompokan data harus dilakukan maksimum dalam sepuluh kelompok.

5. Kode mnemonic

   Kode mnemonic merupakan kode singkatan data yang digunakan untuk membantu pengguna kode ini dalam membaca maksud dari singkatan tersebut.

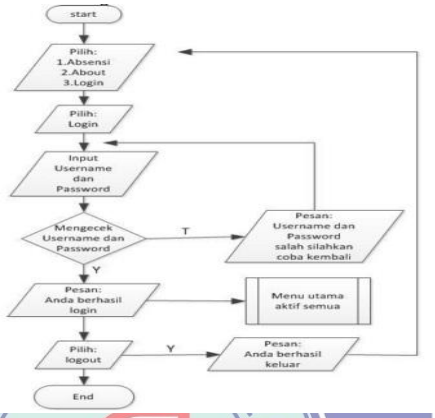
D. HIPO (*Hierarchy Plus Input Proses Output*)

Pendokumentasian rancangan program wajib dilakukan untuk mengkomunikasikan spesifikasi sistem kepada para programmer melalui perancangan HIPO (*Hierarchy Plus Input Proses Output*).

> Menurut Ladjamudin (2013:211) “HIPO dikembangkan oleh personil IMB yang percaya bahwa dokumentasi sistem pemrograman yang dibentuk dengan menekankan pada fungsi-fungsi sistem yang akan mempercepat pencarian prosedur yang akan di modifikasi, karena HIPO menyediakan fasilitas lokasi dalam bentuk kode dari tiap prosedur dalam suatu sistem”.

E. Diagram Alir Program (*Flowchart*)

Menurut Indrajani (2015:36) mengatakan bahwa "*Flowchart* merupakan penggambaran secara grafik langkah-langkah dan urutan prosedur suatu program." Biasanya mempermudah penyelesaian masalah khusunya yang perlu dipelajari dan di evaluasi lebih lanjut. Contoh *flowchart* adalah:



Sumber : (Widiarina, 2014)

Gambar II.5.

Contoh *Flowchart Login*

Jenis-jenis *flowchart* terdiri atas:

1. *System Flowchart*
2. *Document Flowchart*
3. *Schematic Flowchart*
4. Program *Flowchart*
5. *Process Flowchart*

F. Implementasi dan Pengujian Unit

Pengujian unit focus pada usaha verifikasi pada unit terkecil pada desain perangkat lunak (komponen atau modul perangkat lunak). Setiap unit perangkart

lunak diuji agar dapat diperiksa apakah aliran masukan (*input*) dan keluaran (*output*) dari unit sudah sesuai dengan yang diinginkan. Pengujian unit biasanya dilakukan saat kode program dibuat. Karena dalam sebuah perangkat lunak banyak memiliki unit-unit kecil maka untuk menguji unit-unit kecil ini biasanya dibuat program kecil (*main program*) untuk mengkaji unit-unit perangkat lunak.

Menurut Sukamto dan M. Shalaludin (2016:275) mengatakan bahwa "*Black-Box Testing* yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dank ode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan".

Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian kotak hitam harus dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus proses login maka kasus uji yang dibuat adalah:

1. Jika *user* memasukan nama pemakai (*username*) dan kata sandi (*password*) yang benar.
2. Jika *user* memasukan nama pemakai (*username*) dan kata sandi (*password*) yang salah, misalnya nama pemakai benar tapi kata sandi salah, atau sebaliknya, atau keduanya salah.