BAB 2

LANDASAN TEORI

2.1 Konsep Analisis dan Perancangan Sistem

Menurut Jogiyanto (2005, p1), suatu sistem adalah suatu jaringan kerja dari prosedur – prosedur yang saling berhubungan, berkumpul bersama – sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu.

2.1.1 Pengertian Analisis Sistem

Menurut Whitten (2004, p176), analisis sistem adalah sebuah teknik pemecahan masalah yang menguraikan sebuah sistem menjadi bagian – bagian komponen dengan tujuan mempelajari seberapa bagus bagian – bagian komponen tersebut bekerja dan berinteraksi untuk meraih tujuan mereka.

Menurut Jogiyanto (1999, p129), analisis sistem dapat didefinisikan sebagai penguraian dari suatu sistem informasi yang utuh ke dalam bagian – bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan – permasalahan, kesempatan – kesempatan, hambatan – hambatan yang terjadi dan kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan – perbaikannya. Tujuan utamanya adalah untuk memahami sistem dan masalah yang ada, untuk menguraikan kebutuhan informasi dan untuk menetapkan prioritas pekerjaan sistem selanjutnya.

2.1.2 Pengertian Perancangan Sistem

Menurut Whitten (2004, p176), perancangan sistem adalah sebuah teknik pemecahan masalah yang saling melengkapi (dengan analisis sistem) yang

merangkai kembali bagian – bagian komponen menjadi sebuah sistem yang lengkap. Hal ini meliputi penambahan, penghapusan, dan perubahan bagian – bagian relatif pada sistem aslinya (awalnya).

Menurut Jogiyanto (1999, p179), perancangan sistem mempunyai 2 tujuan, yaitu : untuk memenuhi kebutuhan para pemakai sistem dan untuk memberikan gambaran yang jelas kepada pemrogram komputer dan ahli – ahli teknik lainnya yang terlibat.

2.2 *Database*

2.2.1 Pengertian *Database*

Menurut Jogiyanto (1999, p217), *database* merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan dalam perangkat keras komputer dan digunakan oleh perangkat lunak untuk memanipulasinya. *Database* merupakan salah satu komponen yang penting dalam sistem informasi, karena merupakan basis dalam menyediakan informasi bagi para pemakai.

Menurut Connolly (2005, p15), *database* adalah *"a shared collection of logically related data, and a description of this data, designed to meet the information needs of an organization"* yang mempunyai arti sekumpulan data logikal yang berelasi beserta deskripsinya, yang di desain untuk memenuhi kebutuhan informasi sebuah organisasi.

Dengan demikian definisi *database* adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis

data tersebut. Perangkat lunak yang digunakan untuk mengelola dan memanggil *query* basis data disebut *database management system (DBMS).*

2.2.2 Keuntungan *Database*

Dengan dimasukkannya data informasi ke dalam suatu sistem *database* maka data tersebut dapat diakses oleh semua orang yang memiliki hak akses terhadap data tersebut. Selain untuk fasilitas *shared data*, *database* memiliki beberapa keuntungan, antara lain yaitu :

a. Mengurangi ataupun menghilangkan duplikasi data (data redudan),
b. Meningkatkan integritas data,
c. Memelihara sifat indepedensi data,
d. Meningkatkan keamanan data,
e. Memelihara konsistensi data,
f. Data lebih mudah di manipulasi,
g. Data mudah digunakan dan diakses

2.2.3 *Database Management System (DBMS)*

Database Management System (DBMS) merupakan suatu perangkat lunak yang memungkinkan pengguna untuk mendefinisikan, merancang, memelihara, dan menentukan akses control ke dalam *database*. Beberapa fasilitas yang disediakan oleh DBMS, yaitu :

a. DBMS memungkinkan pengguna untuk mendefinisikan *database* melalui *Data Definition Language* (DDL). DDL menyediakan fasilitas kepada

pengguna untuk menentukan tipe data, struktur serta batasan aturan (*constraint*) pada data yang akan disimpan ke dalam *database*.

b. DBMS memungkinkan pengguna untuk membuat, mengubah, menghapus, dan menampilkan data dari *database* dengan menggunakan *Data Manipulation Language* (DML).

c. DBMS menyediakan akses control ke *database*, yaitu :

1. *Security system*, yaitu sistem yang dapat mencegah pengguna yang tidak memiliki otoritas untuk mengakses *database.*
2. *Integrity system*, yaitu sistem yang menjaga konsistensi penyimpanan data.
3. *Concurrency control system*, yaitu sistem yang memungkinkan pengguna untuk mengakses *database* bersamaan dengan pengguna lain.
4. *Recovery control system*, yaitu mengembalikan *database* ke kondisi sebelumnya bila terjadi kerusakan atau kesalahan pada perangkat keras atau lunak.
5. *User-accessible catalog*, yaitu adanya deskripsi data di dalam sebuah *database*.

2.2.4 Keuntungan dan Kerugian DBMS

Menurut Connoly (2005, p26), keuntungan dari DBMS adalah sebagai berikut :

a. Terdapat kontrol pengulangan data

Database berusaha untuk menghilangkan pengulangan dengan mengintegrasikan *file* sehingga berbagai *copy* dari data yang sama tidak tersimpan. Tetapi, pendekatan *database* tidak menghilangkan redudansi sepenuhnya, tetapi mengendalikan jumlah redudansi *database*.

b. Data yang konsisten

Dengan menghilangkan atau mengendalikan redudansi, dapat mengurangi resiko terjadinya ketidak-konsistenan.

c. Penggunaan data bersama

Biasanya, *file* dimiliki oleh orang atau departemen yang menggunakannya. Padahal *database* dimiliki oleh seluruh organisasi dan dapat digunakan bersama oleh pengguna yang berhak.

d. Meningkatkan integritas data

Integritas *database* mengacu pada validitas dan konsistensi data yang disimpan. Integritas biasanya menunjukkan batasan-batasan, yaitu aturan-aturan konsistensi yang tidak boleh dilanggar dalam *database*. Batasan-batasan dapat diterapkan pada data atau pada relasi antar data. Integrasi memungkinkan DBA untuk mendefinisikan, dan DBMS menerapkan batasan integritas.

e. Meningkatkan keamanan data

Keamanan *database* adalah perlindungan *database* dari pengguna yang tidak berhak. Hal ini dapat dilakukan dengan menggunakan *username* dan *password* untuk mengidentifikasikan orang yang berwenang untuk menggunakan *database*. Akses pengguna yang berwenang pada *database*

mungkin dibatasi oleh jenis operasi seperti pengambilan, *insert*, *update,* dan *delete*.

f. Kebutuhan pengguna yang kompleks dapat teratasi

Setiap kebutuhan dari pengguna di departemen mungkin mengalami konflik dengan kebutuhan pengguna lainnya. Sejak *database* dikontrol oleh DBA, DBA dapat membuat keputusan mengenai desain dan kegunaan operasional *database* yang menyediakan sumber yang terbaik untuk organisasi secara keseluruhan. Keputusan ini akan memberikan kinerja yang optimal untuk aplikasi penting, dan mungkin dengan mengorbankan yang kurang penting.

g. Informasi lebih lanjut dari jumlah data yang sama

Dengan diintegrasikannya data operasional, memungkinkan organisasi untuk mengendalikan penambahan informasi untuk data yang sama.

h. Penetapan standarisasi

Integrasi memungkinkan DBA untuk mendefinisikan dan membuat standar yang diperlukan. Standar ini termasuk standar departemen, organisasi, nasional, atau internasional dalam hal format data, untuk memfasilitasi pertukaran data antar sistem, ketetapan penamaan, standarisasi dokumentasi, prosedur *update*, dan aturan pengaksesan.

i. Pengurangan biaya

Dengan menggabungkan seluruh data operasional organisasi ke dalam suatu *database*, dan membuat serangkaian aplikasi yang bekerja pada satu sumber data dapat menghemat biaya. Dalam hal ini, anggaran yang

biasanya dialokasikan tiap departemen untuk mengembangkan dan merawat sistem berbasis *file* dapat digabungkan.

j. Meningkatkan kemampuan akses dan respon data

Dengan pengintegrasian data yang melintasi batasan departemen dapat secara langsung diakses oleh pengguna akhir. Banyak DBMS menyediakan fasilitas *query* atau pembuat laporan yang memungkinkan pengguna untuk menanyakan pertanyaan khusus dan untuk mendapatkan informasi secara cepat dari terminalnya, tanpa membutuhkan *programmer* untuk membuat program yang menghasilkan informasi dari *database*.

k. Meningkatkan produktivitas

DBMS menyediakan banyak fungsi-fungsi standar yang biasanya *programmer* harus tulis di aplikasi berbasis *file*. Perlengkapan dari fungsi-fungsi ini memungkinkan *programmer* untuk berkonsentrasi pada fungsi-fungsi khusus yang dibutuhkan pengguna tanpa harus khawatir tentang detail implementasi. Hasilnya meningkatkan produktifitas *programmer* dan mengurangi waktu pengembangan.

l. Meningkatkan pemeliharaan dengan data yang bebas

DBMS memisahkan data dengan aplikasi, sehingga membuat aplikasi tidak harus terpengaruh oleh perubahan data.

m. Meningkatkan *concurreny*

Bila dua atau lebih pengguna dapat mengakses file yang sama secara bersamaan, kemungkinan pengaksesan tersebut akan saling mempengaruhi, mengakibatkan kehilangan informasi dan integritas.

Banyak DBMS mengelola pengaksesan secara bersamaan pada *database* dan memastikan masalah diatas tidak terjadi.

n. Meningkatkan layanan *back up* dan *recovery*

Banyak sistem berbasis *file* melakukan pengamanan data terhadap gangguan pada sistem atau program aplikasi oleh pengguna. Caranya adalah dengan membuat *back up* data. Sebaliknya, DBMS menyediakan fasilitas untuk meminimalisasi pemrosesan yang hilang akibat kegagalan.

Menurut Connoly (2005, p29), kerugian dari DBMS adalah sebagai berikut :

a. Kompleksitas

Ketentuan dari fungsi yang kita harapkan dari DBMS yang baik membuat DBMS menjadi sebuah *software* yang sangat kompleks. Perancang dan pengembang *database*, DA, dan DBA, serta pengguna akhir harus memahami fungsi tersebut untuk mendapatkan banyak keuntungan dari DBMS ini.

b. Ukuran

Fungsi yang kompleks dan luas membuat DBMS menjadi *software* yang sangat besar, memerlukan banyak ruang *harddisk* dan jumlah memori yang besar untuk berjalan dengan efisien.

c. Biaya dari DBMS

Biaya dari DBMS bervariasi, tergantung pada lingkungan dan fungsi yang disediakan. Di situ juga terdapat biaya pemeliharaan tahunan yang juga dimasukkan dalam daftar harga DBMS.

d. Biaya penambahan perangkat keras

Kebutuhan tempat penyimpanan bagi DBMS dan *database* sangat memerlukan pembelian tempat penyimpanan tambahan. Lebih lanjut, untuk mencapai performa yang diperlukan, mungkin diperlukan untuk membeli mesin yang lebih besar lagi. Hal ini tentu memerlukan tambahan biaya yang tidak sedikit. Tergantung pada spesifikasi perangkat keras yang diperlukan.

e. Biaya konversi

Untuk mengkonversi sistem lama ke sistem yang baru yang memakai DBMS terkadang sangat mahal.

f. Performa

Pada dasarnya DBMS dibuat untuk menyediakan banyak aplikasi, akibatnya mungkin beberapa aplikasi akan berjalan tidak seperti biasanya.

g. Dampak yang tinggi atas kegagalan

Karena sistem yang terpusat, jika seluruh *user* dan aplikasi terakses dari DBMS maka kerusakan pada bagian manapun dari sistem, akan menyebabkan operasi terhenti.

2.2.5 Fungsi DBMS

Menurut Connolly, fungsi DBMS adalah sebagai berikut :

a. Penyimpanan, pengambilan dan pengubahan data

Sebuah DBMS harus menyediakan kemampuan menyimpan, mengambil dan mengubah data dalam *database*.

b. Katalog yang dapat diakses oleh pemakai (*user-accesible*)

DBMS menyediakan sebuah katalog yang berisi deskripsi item data yang disimpan dan diakses oleh pengguna.

c. Mendukung transaksi

DBMS menyediakan mekanisme yang akan menjamin semua pengubahan yang dilakukan sesuai dengan transaksi yang diberikan atau tidak ada kegiatan pengubahan yang dibuat untuk transaksi tersebut.

d. Layanan kendali konkurensi

Sebuah DBMS harus menyediakan mekanisme yang menjamin bahwa *database* di-*update* secara benar pada saat beberapa pemakai melakukan pengubahan terhadap *database* yang sama secara bersamaan.

e. Layanan perbaikan (*recovery*)

DBMS menyediakan mekanisme untuk mengembalikan *database* ke keadaan semula sebelum terjadinya kerusakan pada *database*.

f. Layanan authorisasi

DBMS menyediakan mekanisme untuk menjamin bahwa hanya pengguna yang berwenang yang dapat mengakses *database*. Hal ini untuk mencegah data yang tersimpan tak terlihat oleh semua pengguna dan melindungi *database* dari akses yang tidak berwenang.

g. Mendukung komunikasi data

DBMS harus mampu mengintegrasikan dengan *software* komunikasi.

h. Layanan integritas

DBMS berguna untuk menjamin semua data dalam *database* dan setiap terjadinya perubahan data harus sesuai dengan aturan yang berlaku.

i. Layanan peningkatan keterbebasan data (*data independence*)

DBMS harus mempunyai fasilitas untuk mendukung kemandirian program dari struktur *database* yang sebenarnya.

j. Layanan utilitas

DBMS harus menyediakan seperangkat layanan utilitas. Dengan adanya program utilitas dapat membantu DBA mengelola *database* secara efektif.

2.2.6 *Structured Query Language (SQL)*

2.2.6.1 Pengertian SQL

Menurut Kadir (2002, p101), *Structured Query Language (SQL)* merupakan bahasa *query* standar yang digunakan untuk mengakses basis data relasional.

Menurut Connolly (2005, p113), SQL merupakan *transform-oriented language*, atau bahasa yang dirancang untuk menggunakan hubungan untuk mengubah *input* menjadi *output* yang diperlukan. SQL terbagi menjadi dua komponen utama, yaitu *data definition language* dan *data manipulation language*.

2.2.6.2 *Data Definition Language*

Menurut Connolly (2005, p40), *Data definition language* (DDL) adalah bahasa yang memperbolehkan seorang *database administrator* (DBA) atau pengguna untuk mendeskripsikan nama dari suatu entitas, atribut, hubungan yang dibutuhkan oleh aplikasi bersamaan dengan integritas data dan keamanan datanya.

DDL merupakan kelompok perintah yang berfungsi untuk mendefinisikan atribut-atribut basis data, tabel, kolom, batasan-batasan terhadap suatu atribut serta hubungan antar tabel.

Fungsi-fungsi dalam *data definition language* adalah sebagai berikut :

a. CREATE : untuk membuat objek *database*.
b. ALTER : untuk memodifikasi objek *database*.
c. DROP : untuk menghapus objek *database*.

2.2.6.3 *Data Manipulation Language*

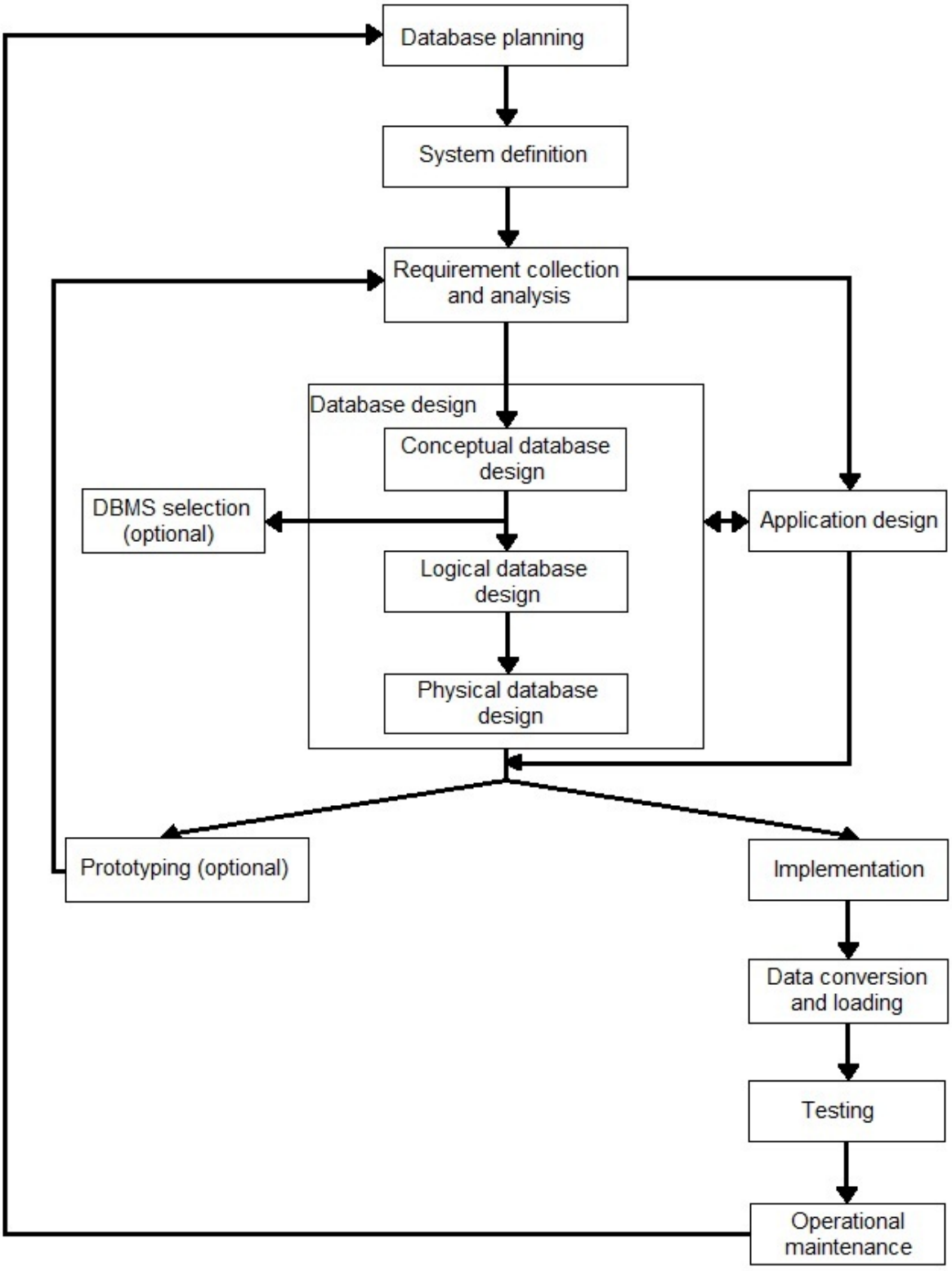
Menurut Connoly (2005, p41), *Data manipulation language* (DML) merupakan bahasa yang memberikan fasilitas pengoperasian data yang ada dalam basis data.

Fungsi-fungsi dalam *data manipulation language* adalah sebagai berikut :

a. SELECT : untuk mengambil data dari *database*.
b. DELETE : untuk menghapus data dari *database*.
c. INSERT : untuk menambahkan data pada *database*.
d. UPDATE : untuk memodifikasi data pada *database*.

2.2.7 *Database Lifecycle*

Menurut Connolly (2005, p284), Tahapan – tahapan dalam siklus hidup *database* adalah sebagai berikut : (Gambar 2.1)



Gambar 2.1 ***Database Lifecycle***

2.2.7.1 *Database planning*

Database planning merupakan kegiatan pengaturan yang memungkinkan tahapan-tahapan dari *database system development lifecycle* dapat direalisasikan se-efektif dan se-efisien mungkin.

Perencanaan *database* harus terintegrasi dengan keseluruhan strategi sistem informasi dari organisasi. Ada tiga hal utama yang berkaitan dengan strategi sistem informasi, yaitu :

a. Identifikasi rencana dan sasaran organisasi termasuk kebutuhan sistem informasi.
b. Evaluasi sistem informasi terkini untuk menentukan kelebihan dan kekurangan yang dimiliki.
c. Penaksiran kesempatan teknologi informasi yang mungkin memberikan keuntungan kompetitif.

Metodologi yang digunakan untuk mengatasi hal tersebut diatas yaitu :

a. *Mission Statement*

   Mission statement ini menjelaskan tujuan utama aplikasi *database*, juga membantu menjelaskan tujuan proyek *database*, dan menyediakan maksud yang lebih jelas dalam pembuatan aplikasi *database* secara efektif dan efisien (Connolly, 2005, p286). Dengan merumuskan apa sebenarnya yang menjadi tujuan dari proyek *database* ini diharapkan dapat lebih memfokuskan pekerjaan pada tahap selanjutnya.

b. *Mission Objectives*

Setelah *mission statement* didefinisikan, kegiatan selanjutnya yaitu identifikasi *mission objectives*. Setiap *mission objective* akan menjelaskan tugas tertentu yang harus didukung *database*, dengan asumsi jika *database* mendukung *mission objective*, maka *mission statement*-nya juga akan sesuai.

2.2.7.2 *System definition*

System definition yaitu menentukan ruang lingkup dan batasan aplikasi *database*, dan sudut pandang (*user view*) yang utama. *User view* sangat diperlukan untuk mengidentifikasi informasi yang dibutuhkan oleh pengguna. *User view* menggambarkan apa yang dibutuhkan oleh aplikasi *database* dari sudut pandang peran jabatan tertentu (seperti manajer atau pengawas) atau dari sudut pandang area aplikasi organisasi (seperti pemasaran, personalia, atau pengawasan persediaan). (Connolly, 2005, p287).

2.2.7.3 *Requirement collection and analysis*

Proses mengumpulkan dan menganalisa informasi tentang bagian dari organisasi yang didukung oleh sistem *database*, dan menggunakan informasi ini untuk mengidentifikasi kebutuhan untuk sistem yang baru. (Connolly, 2005, p288).

Teknik yang digunakan dalam mengumpulkan informasi tentang sistem dan kebutuhan-kebutuhannya disebut dengan teknik *fact-finding*. Beberapa kegiatan yang dipakai dalam teknik ini, yaitu :

a. Memeriksa dokumentasi

   Pemahaman terhadap jalannya sistem akan cepat diperoleh dengan memeriksa dokumen-dokumen, formulir, laporan, dan dokumentasi lainnya yang berkaitan dengan sistem yang sedang berjalan.

b. Wawancara

   Wawancara merupakan cara yang paling umum dipakai. Wawancara bertujuan untuk mengumpulkan fakta-fakta, memeriksa kebenaran fakta yang ada dan mengklarifikasinya, mengidentifikasi kebutuhan-kebutuhan, dan mengumpulkan ide-ide dan pendapat. Teknik ini memerlukan kemampuan komunikasi yang baik dalam menghadapi pengguna yang memiliki nilai, prioritas, pendapat, motivasi, dan kepribadian yang berbeda-beda.

c. Kuesioner

   Kuesioner adalah dokumen dengan tujuan khusus yang memungkinkan fakta-fakta dikumpulkan dari banyak orang sambil menjaga kontrol terhadap tanggapan yang diberikan. Ada dua tipe pertanyaan yang dapat ditanyakan dalam kuesioner, yaitu format bebas dan format pasti. Format bebas memungkinkan responden lebih bebas dalam memberikan

jawaban. Sedangkan format pasti menyediakan pilihan jawaban yang harus dipilih responden.

2.2.7.4 *Database Design*

Database design adalah proses yang menghasilkan sebuah desain yang dapat mendukung *mission statement* dan *mission objectives* suatu organisasi untuk kebutuhan sistem *database*. (Connolly, 2005, p291). Tujuan utamanya adalah :

a. Merepresentasikan data dan hubungan antar data yang dibutuhkan oleh seluruh area aplikasi utama dan *user group*.
b. Menyediakan model data yang mendukung segala transaksi yang diperlukan pada data.
c. Menentukan desain minimal yang secara tepat disusun untuk memenuhi kebutuhan performa yang ditetapkan pada sistem.

Dua pendekatan utama untuk menrancang sebuah *database*, yaitu :

a. *Top-down*

   Diawali dengan pembentukan model data yang berisi beberapa entitas *high-level* dan relasi, yang kemudian menggunakan pendekatan *top-down* secara berturut-turut untuk mengidentifikasikan entitas *lower-level*, relasi dan atribut lainnya.

b. *Bottom-up*

Dimulai dari atribut dasar (yaitu sifat-sifat dasar entitas dan relasi), dengan analisis dari penggabungan antar atribut, yang dikelompokkan ke dalam suatu relasi yang merepresentasikan tipe dari entitas dan relasi antar entitas.

2.2.7.5 *DBMS selection (optional)*

Pemilihan DBMS yang tepat untuk mendukung aplikasi *database*. Apabila tidak ada DBMS yang dipakai sebelumnya, maka tahap pemilihan DBMS ini dilakukan di antara tahap perancangan *database* konseptual dan perancangan *database* logikal. Pemilihan dapat dilakukan kapanpun sebelum menuju desain logikal asalkan telah terdapat cukup informasi mengenai kebutuhan sistem.

Tahapan-tahapan utama untuk memilih sebuah DBMS :

a. Mendefinisikan kriteria DBMS yang dibutuhkan.
b. Menentukan 2 atau 3 produk yang masuk dalam kriteria.
c. Evaluasi produk.
d. Rekomendasi pilihan dan laporan produk.

2.2.7.6 *Application design*

Application design merupakan tahap perancangan *user interface* dan program aplikasi yang akan digunakan dan memproses *database*. Desain *database* dan aplikasi adalah aktivitas paralel pada *database system development lifecycle*. (Connolly, 2005, p300).

Dalam melakukan perancangan aplikasi ini, harus diyakinkan bahwa semua kebutuhan pengguna sudah didefinisikan dalam perancangan aplikasi *database*, sehingga aplikasi dapat berjalan sinkron dengan *database* yang telah dirancang. Terdapat dua aspek penting dalam desain aplikasi, yaitu *Transaction Design* dan *User Interface Design.*

2.2.7.7 *Prototyping*

Membangun model kerja suatu aplikasi *database*. Tujuan utama dari *prototyping* adalah :

a. Untuk mengidentifikasi fitur dari sistem berjalan dengan baik atau tidak.
b. Untuk memberikan perbaikan-perbaikan atau penambahan fitur baru ke sistem *database*.
c. Untuk memperjelas kebutuhan pemakai dan pengembang sistem.
d. Untuk mengevaluasi kelayakan desain sistem tertentu.

Ada dua strategi *prototyping* yang umum digunakan saat ini, yaitu :

a. *Requirement prototyping* menggunakan *prototype* untuk menentukan kebutuhan suatu aplikasi *database* yang diinginkan dan ketika kebutuhan itu terpenuhi maka *prototype* akan dibuang.
b. *Evolutionary prototyping* digunakan untuk tujuan yang sama, perbedaan yang penting adalah *prototype* tidak dibuang tetapi

dengan pengembangan lebih lanjut menjadi sistem *database* yang bekerja.

2.2.7.8 *Implementation*

Merupakan realisasi fisik dari *database* dan desain aplikasi. Implementasi *database* dicapai dengan menggunakan *Data Definition Language* (DDL) atau *Graphical User Interface* (GUI), yang menyediakan fungsionalitas yang sama ketika menyembunyikan *low-level DDL statements*. *Statement* DDL digunakan untuk membuat struktur *database* dan *file database* kosong. Spesifikasi *user views* juga diimplementasikan pada tahap ini.

Program aplikasi diimplementasikan menggunakan bahasa generasi ketiga atau keempat (3GL atau 4GL). Bagian dari program aplikasi yang termasuk dalam proses transaksi diimplementasikan dengan menggunakan *Data Manipulation Language* (DML), dan mungkin tertanam dalam bahasa pemograman, seperti PHP, Java, Visual Basic, dan lain-lain.

2.2.7.9 *Data conversion*

Melakukan pemindahan data yang ada kedalam *database* baru dan mengkonversikan aplikasi yang ada agar dapat digunakan pada *database* yang baru. Tahapan ini dibutuhkan ketika sistem *database* baru menggantikan sistem yang lama.

2.2.7.10 *Testing*

Suatu proses eksekusi program aplikasi dengan tujuan untuk menemukan kesalahan. Apabila proses pengujian ini berjalan dengan baik, maka pengembang dapat melihat masalah yang masih terjadi pada aplikasi maupun struktur *database*. Pengujian ini hanya akan terlihat jika terjadi kesalahan *software*.

2.2.7.11 *Operational Maintenance*

Melakukan pengawasan dan pemeliharaan terhadap sistem setelah instalasi, meliputi :

a. Pengawasan performa sistem, jika performa menurun maka memerlukan perbaikan atau pengaturan ulang *database*.
b. Pemeliharaan dan pembaharuan aplikasi *database* (jika dibutuhkan).
c. Penggabungan kebutuhan baru kedalam aplikasi *database*.

2.2.8 *Entity Relationship Modeling*

Menurut Connoly (2005, p342), *Entity Relationship Modeling* adalah pendekatan *top-down* pada perancangan *database* yang dimulai dengan identifikasi data penting yang disebut entitas dan relasi antara data yang harus digambarkan pada model. Selanjutnya ditambahkan detail seperti informasi yang ingin disimpan mengenai entitas dan relasi yang disebut atribut dan batasan – batasan lain pada entitas, relasi, dan atribut.

Entity Relationship Diagram (ERD) merupakan suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan objek – objek dasar data yang mempunyai hubungan antar relasi.

Konsep *entity relationship modeling*, terdiri dari :

2.2.8.1 Tipe Entitas

Konsep dasar dari model ER adalah tipe entitas, yaitu sekumpulan objek yang mempunyai sifat atau properti yang sama, yang diidentifikasi oleh organisasi. Sebuah tipe entitas mempunyai eksistensi yang independen dan dapat berupa objek dengan eksistensi fisikal (atau nyata) atau objek dengan eksistensi konseptual (atau abstrak)

Entity occurrence adalah sebuah objek unik yang dapat diidentifikasikan dari sebuah tipe entitas. Setiap entitas diidentifikasi dan disertakan propertinya.

2.2.8.2 Tipe Hubungan atau Relasi

Relasi merupakan suatu set asosiasi antara satu atau lebih tipe entitas yang berpartisipasi.

Relationship occurrence adalah hubungan yang dapat diidentifikasi secara unik, yang termasuk satu kejadian dari setiap entitas yang berpartisipasi.

Derajat relasi yaitu jumlah entitas yang berpartisipasi dalam suatu hubungan. Derajat relasi terdiri dari :

a. *Binary relationship*

Yaitu keterhubungan antara dua tipe entitas.

b. *Ternary relationship*

Yaitu keterhubungan antar tiga tipe entitas.

c. *Quaternary relationship*

Yaitu keterhubungan antar empat tipe entitas.

d. *Unary relationship*

Yaitu keterhubungan antar tipe entitas yang sama, dimana tipe entitas tersebut berpartisipasi lebih dari satu kali dengan peran yang berbeda. Kadang disebut juga *recursive relationship*.

2.2.8.3 Atribut

Atribut yaitu setiap elemen yang terdapat dalam suatu entitas yang berfungsi untuk mendeskripsikan karakteristik dari entitas tersebut. Setiap atribut menyimpan nilai yang menjelaskan setiap *entity occurrence* dan menggambarkan bagian utama dari data yang disimpan dalam *database*.

Attribute domain adalah himpunan nilai yang diperbolehkan untuk satu atau lebih atribut. Beberapa macam atribut, yaitu :

a. *Simple Attribute*

Yaitu atribut yang terdiri dari satu komponen tunggal dengan keberadaan yang independen dan tidak dapat dibagi menjadi bagian yang lebih kecil lagi. Disebut juga dengan *atomic attributes*. Contohnya nim, nis, noKTP, noSIM, noKaryawan dan sebagainya.

b. *Composite Attribute*

Yaitu atribut yang terdiri dari beberapa komponen, dimana masing-masing komponen memiliki keberadaan yang independen. Misalnya atribut alamat dapat terdiri dari jalan, kota, kodePos.

c. *Single-valued Attribute*

Yaitu atribut yang mempunyai nilai tunggal untuk setiap kejadian. Misalnya entitas Cabang memiliki satu nilai untuk atribut nomorCabang pada setiap kejadian.

d. *Multi-valued Attribute*

Yaitu atribut yang mempunyai beberapa nilai untuk setiap kejadian. Misalnya entitas Cabang memiliki beberapa nilai untuk atribut noTelp pada setiap kejadian.

e. *Derived Attribute*

Yaitu atribut yang memiliki nilai yang dihasilkan dari satu atau beberapa atribut lainnya, dan tidak harus berasal dari satu entitas. Contohnya, lama pinjam dihasilkan dari perhitungan mulai pinjam dikurangi tanggal pengembalian, dan sebagainya.

2.2.8.4 *Keys*

a. *Candidate Key*

Yaitu jumlah minimal atribut-atribut dimana secara unik mengidentifikasikan setiap kejadian atau *record* dari sebuah tipe entitas.

b. *Primary Key*

Yaitu *candidate key* yang dipilih untuk mengidentifikasikan setiap kejadian atau *record* dari suatu entitas secara unik.

c. *Composite Key*

Yaitu *candidate key* yang terdiri dari dua atau lebih atribut.

2.2.8.5 *Strong and Weak Entity Types*

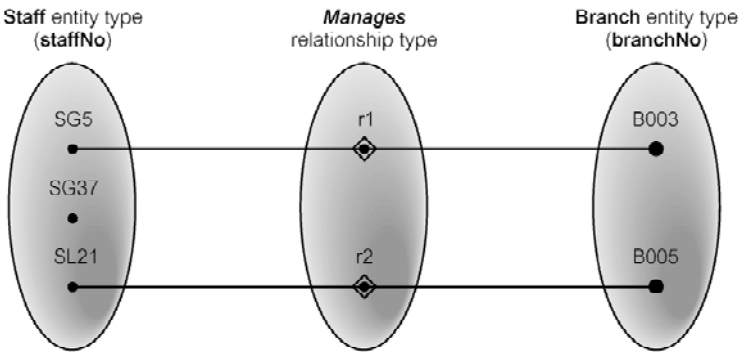
Strong entity type yaitu entitas yang keberadaannya tidak bergantung pada entitas lain. Sedangkan *Weak entity type* yaitu entitas yang keberadaanya bergantung pada entitas lain. *Strong entity type* terkadang disebut dengan entitas *parent, owner,*atau *dominant* dan *weak entity type* disebut entitas *child, dependent,* atau *subordinate*.

2.2.8.6 *Structural Constraint*

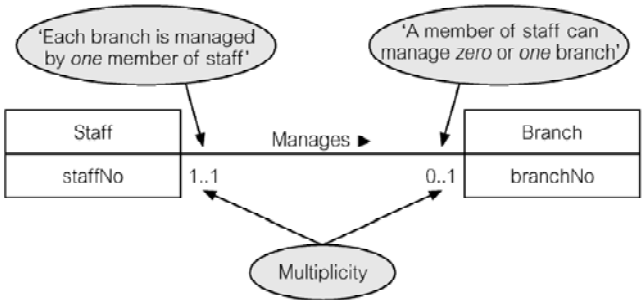
Batasan utama pada relasi adalah *multiplicity*. *Multiplicity* adalah jumlah (atau jangkauan) dari kejadian yang mungkin terjadi pada suatu entitas yang terhubung ke satu kejadian dari entitas lain yang berhubungan melalui suatu relasi.

Derajat relasi yang paling umum adalah *binary relationship*. Beberapa macam *binary relationship*, yaitu :

a. *One-to-One (1:1) Relationship*

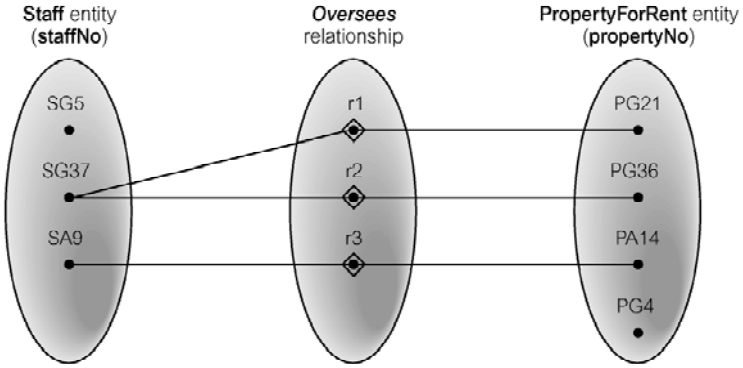


Gambar 2.2 *Semantic net* yang menunjukkan kejadian *Staff manages Branch relationship type*

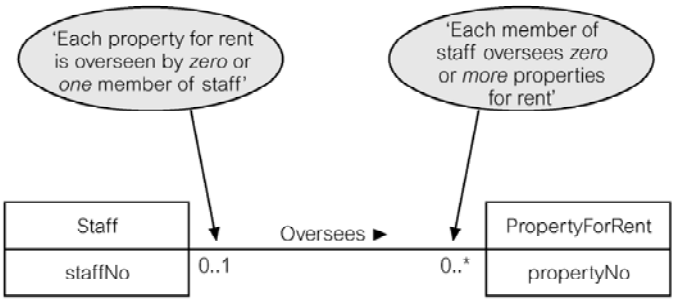


Gambar 2.3 *Multiplicity one-to-one (1:1) relationship*

b. *One-to-Many (1:*) Relationship*

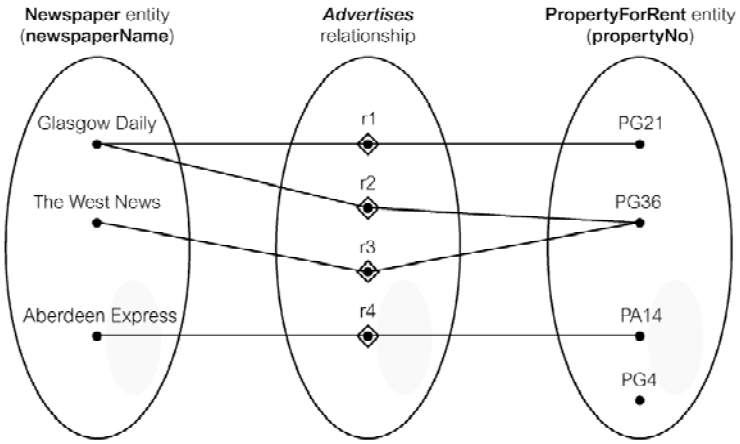


Gambar 2.4 *Semantic net* yang menunjukkan tiga kejadian dari *Staff Overseas PropertyForRent relationship type*

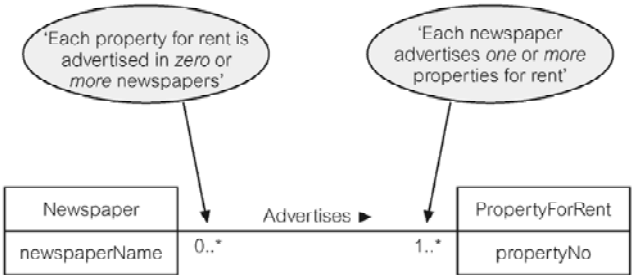


Gambar 2.5 ***Multiplicity one-to-many (1:*) relationship type***

c. *Many-to-Many (*:*) Relationship*



Gambar 2.6 ***Semantic net*** **yang menunjukkan empat kejadian dari** ***Newspaper Advertises PropertyForRent relationship type***



Gambar 2.7 ***Multiplicity many-to-many (*:*) relationship type***

2.2.9 Metodologi Perancangan *Database*

2.2.9.1 Perancangan Konseptual (*Conceptual Database Design*)

Perancangan *database* konseptual adalah suatu proses pembentukan model dari informasi yang digunakan dalam organisasi, bebas dari keseluruhan aspek fisik. Model data dibangun dengan menggunakan informasi dalam penentuan kebutuhan pengguna. Model data konseptual merupakan sumber informasi untuk fase desain logikal.

Langkah-langkah dalam metodologi perancangan *database* konseptual, yaitu :

1. Identifikasi tipe entitas

   Tahap pertama dalam membangun model data konseptual lokal adalah menentukan objek utama dimana pengguna mempunyai ketertarikan pada objek tersebut. Objek-objek ini adalah tipe entitas untuk model. Salah satu metode untuk mengidentifikasi entitas adalah dengan menguji spesifikasi kebutuhan dari pengguna. Dari spesifikasi ini kita mengidentifikasikan kata benda (*noun*) atau ungkapan kata benda (*noun phrase*) yang disebutkan. Kami juga melihat objek utama seperti orang, tempat, atau konsep dari ketertarikan, diluar kata benda lainnya yang merupakan kualitas dari objek lain.

2. Identifikasi tipe relasi

   Tahap ini bertujuan untuk mengidentifikasi relasi penting yang ada antara tipe-tipe entitas yang telah diidentifikasikan. Kita dapat menggunakan *grammar* dari spesifikasi kebutuhan untuk

mengidentifikasi relasi. Biasanya, relasi diindikasi oleh kata kerja (*verb*) atau *verbal expressions*. Relasi berada antara dua tipe entitas atau disebut *binary*. Tetapi kita juga harus berhati-hati untuk melihat relasi kompleks yang mungkin melibatkan lebih dari dua tipe entitas dan relasi rekursif yang hanya melibatkan satu tipe entitas.

3. Identifikasi dan hubungkan atribut dengan entitas atau tipe relasi

   Tahap ini bertujuan untuk menghubungkan atribut dengan tipe entitas atau relasi yang sesuai. Atribut-atribut dapat diidentifikasi dengan kata benda (*noun*) atau ungkapan kata benda (*noun phrase*) seperti properti, kualitas, *identifier*, atau karakteristik dari suatu entitas atau relasi. Ada tiga jenis atribut, yaitu atribut *simple* atau *composite*, atribut *single* atau *multi value*, atribut *derived*.

4. Menentukan domain atribut

   Tahap ini bertujuan untuk menentukan domain atribut pada model data konseptual lokal dan mendokumentasikan setiap detail dari domain. Domain merupakan sekumpulan nilai-nilai dari satu atau lebih atribut yang menggambarkan nilainya. Model data yang dibangun menspesifikasikan domain untuk setiap atribut dan meyertakan :

   a. Nilai yang diizinkan untuk atribut
   b. Ukuran dan format atribut

5. Menentukan atribut *candidate, primary, dan alternate key*

Tahap ini bertujuan untuk mengidentifikasi *candidate key* untuk setiap tipe entitas dan, jika terdapat lebih dari satu *candidate key*, pilihlah satu sebagai *primary key*, dan sisanya sebagai *alternate key*. Ketika memilih sebuah *primary key* diantara beberapa *candidate key*, gunakan petunjuk berikut untuk membantu dalam memilih :

a. *Candidate key* dengan set minimal atribut
b. *Candidate key* yang nilainya paling sedikit mengalami perubahan
c. *Candidate key* dengan karakter terkecil (untuk yang dengan atribut tekstual)
d. *Candidate key* dengan nilai maximum terkecil (untuk yang dengan atribut numerik)
e. *Candidate key* yang paling mudah digunakan pada sudut pandang pengguna

6. Mempertimbangkan penggunaan dari *enhanced modeling concepts (optional step)*

   Tahap ini bertujuan untuk mempertimbangkan penggunaan konsep model *enhanced*, seperti spesialisasi, generalisasi, penggabungan (*aggregation*), dan komposisi (*composition*).

7. Memeriksa model untuk redundasi

   Tahap ini bertujuan untuk memeriksa apakah ada redundansi dalam model. Dua aktivitas dalam tahap ini adalah menguji

ulang relasi *one-to-one* (1:1) dan menghilangkan relasi yang redundan.

8. Validasikan model konseptual lokal terhadap transaksi pengguna

   Tahap ini bertujuan untuk memastikan bahwa model data konseptual mendukung transaksi yang dibutuhkan. Diuji dua pendekatan untuk memastikan model data konseptual mendukung transaksi yang dibutuhkan, yaitu :

   a. Mendeskripsikan transaksi-transaksinya

      Memeriksa seluruh informasi (entitas, relasi, dan atribut-atributnya) yang dibutuhkan oleh setiap transaksi yang disediakan oleh model, dengan mendokumentasikan suatu deskripsi dari setiap kebutuhan-kebutuhan transaksi.

   b. Menggunakan jalur-jalur transaksi

      Untuk validasi model data terhadap transaksi yang dibutuhkan termasuk representasi diagram jalur yang digunakan oleh setiap transaksi langsung pada diagram ER.

9. Memeriksa kembali model data konseptual lokal dengan pengguna

   Tahap ini bertujuan untuk memeriksa kembali model data konseptual dengan pengguna untuk memastikan model tersebut adalah representasi sebenarnya dari *view*. Model data konseptual ini termasuk ER diagram dan dokumentasi pendukung yang mendeskripsikan model data. Bila ada kejanggalan (anomali) dalam model data, maka harus dibuat perubahan yang sesuai

yang mungkin membutuhkan pengulangan langkah-langkah sebelumnya.

2.2.9.2 Perancangan Logikal (*Logical Database Design*)

Perancangan *database* logikal adalah suatu proses pembentukan model dari informasi yang digunakan dalam organisasi berdasarkan model data tertentu (missal : relasional), tetapi tidak tergantung terhadap *Database Management System* (DBMS) tertentu dan aspek fisik lainnya. Model data konseptual yang telah dibuat sebelumnya diperbaiki dan dipetakan kedalam model data logikal.

Langkah-langkah dalam metodologi perancangan *database* logikal, yaitu :

1. Menghilangkan fitur – fitur yang tidak kompatibel dengan model relasional

   Tahap ini bertujuan untuk memperbaiki model data konseptual lokal dengan cara menghilangkan fitur – fitur yang tidak kompatibel dengan model relasional.

   Tujuan dari tahap ini, yaitu :

   a. Menghilangkan tipe relasi biner *many-to-many* (*:*)
   b. Menghilangkan tipe relasi kompleks
   c. Menghilangkan atribut *multi-valued*
2. Membuat relasi untuk model data logikal

Tahap ini bertujuan untuk membuat hubungan antara model data logikal untuk merepresentasikan entitas, relasi, atribut yang telah diidentifikasikan.

Relasi yang dapat terjadi dari model data yaitu :

a. Tipe entitas kuat
b. Tipe entitas lemah
c. Tipe relasi biner *one-to-many* (1:*)
d. Tipe relasi biner *one-to-one* (1:1)
e. Tipe relasi rekursif *one-to-one* (1:1)
f. Tipe relasi *superclass/subclass*
g. Tipe relasi biner *many-to-many* (*:*)
h. Tipe relasi kompleks
i. Atribut *multi-valued*

3. Validasikan relasi dengan normalisasi

   Tahap ini bertujuan untuk memvalidasikan hubungan dalam model data logikal dengan menggunakan normalisasi.

4. Validasikan relasi terhadap transaksi pengguna

   Tahap ini bertujuan untuk memastikan hubungan dalam model data logikal mendukung transaksi yang dibutuhkan. Dalam langkah ini akan dilakukan operasi *database* secara manual, bila semua transaksi yang dibutuhkan dapat berjalan semestinya, maka model data logikal terhadap transaksi telah divalidasi.

5. Memeriksa batas integritas

Batas integritas (*integrity constraint*) adalah batasan yang ditentukan untuk menghindari data menjadi tidak konsisten. Tipe-tipe batasan integritas, yaitu :

a. *Required data*

   Beberapa atribut harus selalu berisi data yang valid. Dengan kata lain, tidak boleh kosong.

b. *Attribute domain constraints*

   Setiap atribut mempunyai domain, batasan nilai yang legal. Contohnya jenis kelamin harus diisi dengan ‘P’ atau ‘L’.

c. *Multiplicity*

   Multiplicity menunjukkan batasan yang ditempatkan pada relasi-relasi antar data dalam *database.*

d. *Entity integrity*

   Primary key dari suatu entitas tidak boleh kosong.

e. *Referential integrity*

   Jika suatu *foreign key* memiliki nilai, maka nilai tersebut harus menunjuk ke sebuah baris yang ada pada relasi ‘*parent*’.

f. *General constraints*

   Updates ke entitas mungkin dikontrol oleh batasan dalam transaksi ‘dunia sebenarnya’. Contohnya *DreamHome* mempuyai aturan yang mencegah seorang member dari *staff* mengatur lebih dari 100 properti pada saat yang sama.

6. Memeriksa kembali model data logikal dengan pengguna

Tahap ini bertujuan untuk memeriksa kembali model data logikal dengan pengguna untuk memastikan model data logikal dan dokumentasi pendukung yang menjelaskan model telah merepresentasikan kebutuhan.

2.2.9.3 Perancangan Fisikal (*Physical Database Design*)

Perancangan *database* fisikal adalah suatu proses yang menghasilkan deskripsi implementasi *database* pada penyimpanan sekunder. Menggambarkan struktur penyimpanan dan metode akses yang digunakan untuk mencapai akses yang efisien terhadap data. Desain fisikal merupakan cara pembuatan menuju system DBMS tertentu.

Langkah-langkah dalam metodologi perancangan *database* fisikal, yaitu :

a. Menerjemahkan model data logikal global untuk DBMS target
    1. Merancang relasi dasar
    2. Merancang representasi dari data yang dihasilkan
    3. Merancang batasan-batasan umum
b. Merancang organisasi *file* dan *indexes*
    1. Menganalisis transaksi
    2. Memilih organisasi *file*
    3. Memilih *indexes*
    4. Memperkirakan kebutuhan tempat penyimpanan (*disk space*)
c. Merancang *user views*

d. Merancang mekanisme keamanan

e. Mempertimbangkan pengenalan dari redundasi terkontrol

f. Awasi dan atur sistem operasional

2.2.10 Normalisasi

Normalisasi adalah suatu teknik untuk menghasilkan sekumpulan relasi dengan sifat-sifat (*properties*) yang diinginkan berdasarkan kebutuhan-kebutuhan data pada suatu organisasi.

Ada tiga tahap normalisasi yang digunakan sebagai landasan skripsi ini, yaitu :

a. *First Normal Form* (1NF)

   Suatu relasi dikatakan 1NF jika titik temu tiap baris dan kolom pada relasi tersebut mengandung satu nilai.

b. *Second Normal Form* (2NF)

   Suatu relasi dikatakan 2NF jika relasi tersebut berada pada 1NF dan setiap atribut yang bukan *primary key* bergantung penuh (*fully functionally dependent*) terhadap *primary key*.

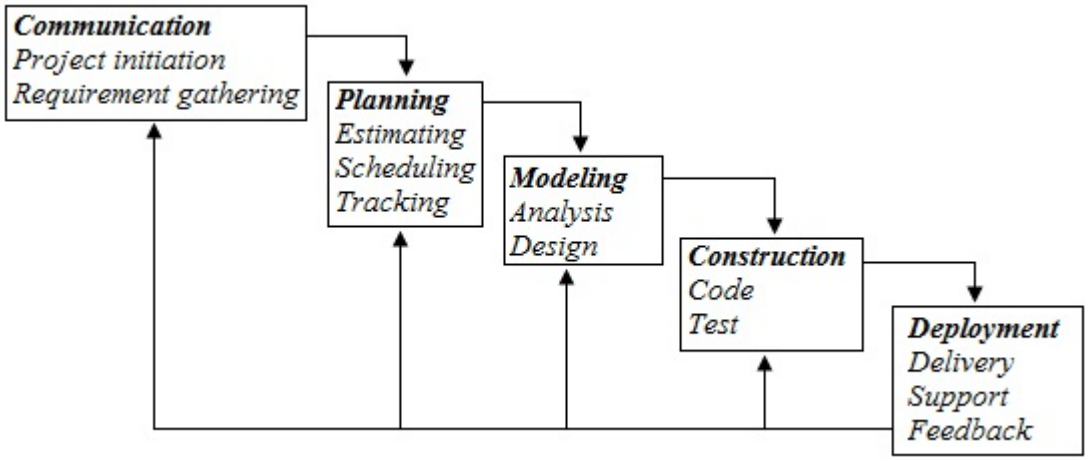
c. *Third Normal Form* (3NF)

   Suatu relasi dikatakan 3NF jika relasi tersebut berada dalam bentuk 1NF dan 2NF, dan tidak ada atribut bukan *primary key* bergantung secara transitif (*transitively dependent*) terhadap *primary key*.

2.3 *System Development Life Cycle*

System Development Life Cycle atau siklus hidup pengembangan sistem merupakan suatu metode terhadap perancangan sistem. Metode ini sangat dibutuhkan pada saat proses pembuatan atau perubahan terhadap sistem yang baru akan dibuat maupun sistem lama yang sudah berjalan. Metode SDLC yang digunakan dalam penulisan ini adalah *Waterfall Model*.

Menurut Pressman (2006, p79), model *waterfall* adalah proses pengembangan perangkat lunak sekuensial, dimana kemajuan dipandang sebagai terus mengalir ke bawah (seperti air terjun) melalui tahapan konsepsi, inisiasi, analisis, desain, konstruksi, pengujian dan pemeliharaan. Berikut adalah tahap – tahap yang dilakukan di dalam model *waterfall* :



Gambar 2.8 *Waterfall Model*

a. *Communication* meliputi :

   1. *Project initiation*

      Melakukan komunikasi dan kolaborasi dengan klien mengenai *project* yang akan dibuat dengan merumuskan masalah yang ingin dipecahkan dengan solusi yang akan dicapai.

2. *Requirement gathering*

   Merumuskan dan mencatat apa saja yang dibutuhkan dari keseluruhan sistem oleh klien.

b. *Planning* meliputi :

1. *Estimating*

   Menentukan kapan *project* akan dimulai dan kapan akan selesai.

2. *Scheduling*

   Menentukan jadwal untuk tiap tahap – tahap proses penyelesaian masalah.

3. *Tracking*

   Menentukan jalur serta *milestone* yang akan ditempuh selama proses pembangunan sistem.

c. *Modeling* meliputi :

1. *Analysis*

   Melakukan analisis terhadap kebutuhan klien dan menetukan solusi yang akan dicapai untuk menyelesaikan masalah.

2. *Design*

   Merumuskan hasil analisis ke dalam bentuk model atau diagram.

d. *Construction* meliputi :

1. *Coding*

   Melakukan implementasi koding. Tahap ini merupakan implementasi dari tahap desain yang secara teknis nantinya dikerjakan oleh *programmer*.

2. *Testing*

Melakukan uji coba terhadap semua fungsi –fungsi *software*, agar *software* bebas dari eror, dan hasilnya harus sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya.

e. *Deployment* meliputi :

1. *Delivery*

   Memberikan laporan kepada klien apa saja yang telah dilakukan dan hasil sementara *project* untuk tiap tahap proses pengerjaan.

2. *Support*

   Menyediakan bantuan berupa *training* untuk klien dalam menggunakan sistem aplikasi basis data, agar klien dapat menggunakannya dengan baik.

3. *Feedback*

   Menerima kritik dan saran dari klien yang menggunakan sistem aplikasi basis data yang telah kita buat untuk perbaikan sistem ke arah yang lebih baik lagi.

2.4 *Data Flow Diagram* (DFD)

DFD adalah suatu model logika data atau proses yang dibuat untuk menggambarkan dari mana asal data dan kemana tujuan data yang keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut dan interaksi antara data yang tersimpan dan proses yang dikenakan pada data tersebut.

DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir atau dimana data tersebut akan disimpan.

DFD merupakan alat yang digunakan pada metodologi pengembangan sistem yang terstruktur. Kelebihan utama pendekatan aliran data, yaitu :

1. Kebebasan dari menjalankan implementasi teknis sistem.
2. Pemahaman lebih jauh mengenai keterkaitan satu sama lain dalam sistem dan subsistem.
3. Mengkomunikasikan pengetahuan sistem yang ada dengan pengguna melalui diagram aliran data.
4. Menganalisis sistem yang diajukan untuk menentukan apakah data-data dan proses yang diperlukan sudah ditetapkan.

DFD terdiri dari *context diagram* dan diagram rinci (DFD *Levelled*). *Context diagram* berfungsi memetakan model lingkungan (menggambarkan hubungan antara entitas luar, masukan dan keluaran sistem), yang direpresentasikan dengan lingkaran tunggal yang mewakili keseluruhan sistem. DFD *levelled* menggambarkan sistem sebagai jaringan kerja antara fungsi yang berhubungan satu sama lain dengan aliran dan penyimpanan data, model ini hanya memodelkan sistem dari sudut pandang fungsi.

Dalam DFD *levelled* akan terjadi penurunan level dimana dalam penurunan level yang lebih rendah harus mampu merepresentasikan proses tersebut ke dalam spesifikasi proses yang jelas. Jadi dalam DFD *levelled* bisa dimulai dari DFD level 0 kemudian turun ke DFD level 1 dan seterusnya.

Simbol-simbol yang digunakan dalam DFD menurut Gane atau Sarson, yaitu :

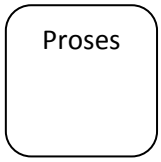
1. Entitas eksternal

   Entitas eksternal, dapat berupa orang atau unit terkait yang berinteraksi dengan sistem tetapi diluar sistem.

Gambar 2.9 Simbol Entitas Eksternal

2. Proses

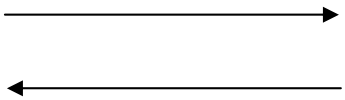
   Proses adalah sesuatu yang mengubah input menjadi output. Tiap simbol proses diidentifikasikan dengan label.



Gambar 2.10 Simbol Proses

3. Aliran data

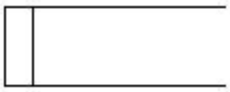
   Aliran data dengan arah khusus dari sumber ke tujuan.



Gambar 2.11 Simbol Aliran Data

4. Data store

   Penyimpanan atau tempat data direfer oleh proses.



Gambar 2.12 Simbol Data Store

2.5 *State Transition Diagram*

State Transition Diagram (STD) adalah suatu diagram yang menunjukkan atau menggambarkan suatu kejadian transisi dan perubahan keadaan (dari satu state ke state lainnya) suatu objek pada sistem sebagai akibat stimulasi yang diterimanya. *State Transition Diagram* diciptakan untuk objek-objek yang secara signifikan mempunyai sifat atau *behaviour* dinamis.

2.6 Proses Belajar Mengajar

Proses belajar mengajar merupakan inti dari proses pendidikan secara keseluruhan dengan guru sebagai pemegang peranan utama. Karena Proses belajar-mengajar mengandung serangkaian perbuatan pendidik atau guru dan siswa atas dasar hubungan timbal balik yang berlangsung dalam situasi edukatif untuk mencapai tujuan tertentu. Interaksi atau hubungan timbal balik antara guru dan siswa itu merupakan syarat utama bagi berlangsungnya proses belajar-mengajar. Interaksi dalam peristiwa belajar-mengajar ini memiliki arti yang lebih luas, tidak sekedar hubungan antara guru dengan siswa, tetapi berupa interaksi edukatif. Dalam hal ini bukan hanya penyampaian pesan berupa materi pelajaran, melainkan menanamkan sikap dan nilai pada diri siswa yang sedang belajar.

Peran guru dalam proses belajar-mengajar , guru tidak hanya tampil lagi sebagai pengajar *(teache*r), seperti fungsinya yang menonjol selama ini, melainkan beralih sebagai pelatih *(coach),* pembimbing *(counselo*r) dan manager belajar *(learning manage*r). Hal ini sudah sesuai dengan fungsi dari peran guru masa depan. Di mana sebagai pelatih, seorang guru akan berperan mendorong siswanya untuk menguasai alat belajar, memotivasi siswa untuk bekerja keras dan mencapai prestasi setinggi-tingginya.

Kehadiran guru dalam proses belajar mengajar atau pengajaran, masih tetap memegang peranan penting. Peranan guru dalam proses pengajaran belum dapat digantikan oleh mesin, radio, tape recorder ataupun oleh komputer yang paling modern sekalipun. Masih terlalu banyak unsur-unsur manusiawi seperti sikap, sistem, nilai, perasaan, motivasi, kebiasaan dan Iain-lain yang diharapkan merupakan hasil dari proses pengajaran, tidak dapat dicapai melalui alat-alat tersebut. Di sinilah kelebihan manusia dalam hal ini guru dari alat-alat atau teknologi yang diciptakan manusia untuk membantu dan mempermudah kehidupannya.

Namun harus diakui bahwa sebagai akibat dari laju pertumbuhan penduduk yang cepat (di Indonesia 2,0% atau sekitar tiga setengah juta lahir manusia baru dalam satu tahun) dan kemajuan teknologi di lain pihak, di berbagai negara maju bahkan juga di Indonesia, usaha ke arah peningkatan pendidikan terutama menyangkut aspek kuantitas berpaling kepada ilmu dan teknologi. Misalnya pengajaran melalui radio, pengajaran melalui televisi, sistem belajar jarak jauh melalui sistem modul, mesin mengajar atau komputer, atau bahkan pembelajaran yang menggunakan sistem *E-learning* (*electronic learning*) yaitu pembelajaran baik secara formal maupun informal yang dilakukan melalui media elektronik, seperti internet, DVD, TV, *handphone*. Akan tetapi, *e-learning* merupakan pembelajaran yang lebih dominan menggunakan internet (berbasis *web*).

2.7 *E – Learning*

2.7.1 Pengertian *E-Learning*

Menurut Turban (2005, p118), *e-Learning* adalah proses belajar menggunakan web, yang dapat digunakan di dalam kelas biasa ataupun kelas virtual.

Menurut Dong, *e-Learning* adalah kegiatan belajar *asynchronous* (model belajar terpisah antara guru dan dilakukan tidak dalam bentuk bersamaan sehingga siswa dapat mengatur sendiri kecepatan belajarnya) melalui perangkat elektronik komputer yang tersambungkan ke internet dimana peserta belajar berusaha memperoleh bahan belajar yang sesuai dengan kebutuhannya. (Kamarga, 2002, p52).

Jadi, *e-Learning* adalah kegiatan pembelajaran yang dilakukan melalui jaringan internet yang bertujuan untuk pendidikan, pelatihan, dan pengetahuan dengan mendapatkan dukungan layanan belajar.

2.7.2 Kelebihan *E-Learning*

Menurut DelVecchio dan Loughney (2006, p5), *e-Learning* sangat berguna bagi pendidikan dan perusahaan serta untuk semua tipe pelajar. *E-Learning* sangat terjangkau, menghemat waktu, dan memiliki hasil yang dapat diukur. *E-Learning* mempunyai beberapa kelebihan, yaitu :

a. Mengurangi biaya

   E-Learning lebih hemat dibandingkan dengan cara belajar tradisional karena hemat waktu dan uang yang dihabiskan saat dalam transportasi. *E-Learning* dapat diakses dari berbagai lokasi dan tidak ada biaya transportasi sama sekali, *e-learning* lebih hemat dibandingkan dengan cara belajar tradisional.

b. Fleksibilitas

E-Learning memiliki kelebihan dalam pengaksesan dimana saja dan kapan saja. Pendidikan tersedia kapan pun dan dimana pun dibutuhkan. *E-Learning* dapat digunakan di kantor, rumah, jalan, 24 jam dan 7 hari dalam satu minggu. *E-Learning* juga memiliki pengukuran terhadap hasil belajar yang dibuat agar instruktur dan pelajar dapat mengetahui apa saja yang telah dipelajari, kapan mereka akan menyelesaikan pelajarannya dan bagaimana hasil yang telah mereka capai.

c. Pelajar sangat menyukai *e-Learning* karena mengakomodir cara belajar yang berbeda. Pelajar bisa mengambil keuntungan belajar sesuai dengan keinginan mereka. Pelajar juga bisa menyesuaikan *e-Learning* dengan jadwal kesibukan mereka. Apabila pelajar bekerja maka ia masih dapat bekerja dengan *e-Learning*. Apabila pelajar menginginkan waktu belajar di malam hari, maka pilihannya juga tersedia.

2.7.3 Kekurangan *E-Learning*

Disamping kelebihannya, menurut DelVecchio dan Loughney (2006, p5), *E-Learning* juga mempunyai kekurangan, yaitu :

a. Pelajar harus memiliki akses ke komputer dan internet.

b. Pelajar juga harus memiliki keterampilan komputer dengan programnya, seperti program *word processing, internet browsing,* dan *e-mail*.

c. Koneksi internet yang baik, karena sangat dibutuhkan dalam pengambilan materi pelajaran.

d. Dengan tidak adanya rutinitas yang ada di kelas tradisional maka pelajar mungkin akan berhenti belajar atau bingung mengenai kegiatan belajar dan tenggang waktu tugas, yang akan membuat pelajar gagal.

e. Pelajar akan merasa sangat jauh dengan instruktur. Karena instruktur tidak selalu ada untuk membantu pelajar, sehingga pelajar harus disiplin dan mengerjakan tugas secara mandiri tanpa bantuan instruktur.

f. Pelajar juga harus memiliki kemampuan menulis dan kemampuan komunikasi yang baik. Karena pelajar dan instruktur tidak bertatap muka sehingga memungkinkan terjadinya salah pengertian dalam beberapa hal.

2.8 Internet

2.8.1 Pengertian Internet

Menurut Turban (2005, p50), internet adalah sistem jaringan komputer dan jaringan dari banyak jaringan yang meliputi seluruh dunia. Internet bersifat publik, kooperatif, dan mandiri yang memfasilitasi akses ke ratusan atau jutaan manusia di seluruh dunia.

Menurut Turban (2005, p478), internet merupakan jaringan dalam jaringan yang menghubungkan komputer individual yang dimiliki oleh pemerintah, universitas, grup non-profit dan perusahaan. Interkoneksi ini dihubungkan dengan standar protokol yang bebas dan terbuka.

2.8.2 *World Wide Web (WWW)*

Menurut Turban (2005, p50), *World Wide Web* adalah aplikasi yang digunakan dalam internet yang berfungsi sebagai transportasi data yang diterima

sebagai standar untuk menyimpan, menerima, memformat dan menampilkan informasi melalui arsitektur *client/ server*. *Web* dapat menerima semua jenis informasi digital, termasuk teks, hipermedia, grafis, dan suara. Web menggunakan antarmuka pengguna grafis, sehingga sangat mudah digunakan.

Web didasari oleh bahasa hiperteks standar yang disebut *Hypertext Markup Language* (HTML), yang memformat dokumen dan memadukan link hiperteks dinamis ke dokumen-dokumen lainnya yang disimpan di dalam komputer.

Untuk mengakses situs *Web*, pengguna harus menentukan *Uniform Resource Locator* (URL), yang mengarahkan ke alamat dari sumber tertentu di *web*. Sedangkan, standar komunikasi yang digunakan untuk mentransfer halaman di bagian WWW di internet adalah *Hypertext Transfer Protocol* (HTTP).

Menurut Ellsworth (1997, p4), WWW terdiri dari dua komponen dasar, yaitu :

a. *Server Web*

   Sebuah komputer dan perangkat lunak yang menyimpan dan mendistribusikan data ke komputer lainnya (yang meminta informasi) melalui internet.

b. *Browser Web*

   Perangkat lunak yang dijalankan pada komputer pengguna (klien) yang meminta informasi dari *server web* dan menampilkannya sesuai dengan *file* data itu sendiri.

2.9 Interaksi Manusia dan Komputer

2.9.1 Pengertian Interaksi Manusia dan Komputer

Interaksi Manusia dan Komputer (IMK) atau *Human-Computer Intercation* (HCI) adalah disiplin ilmu yang berhubungan dengan perancangan, evaluasi, dan implementasi sistem komputer interaktif untuk digunakan oleh manusia, serta studi fenomena-fenomena besar yang berhubungan dengannya. (Definisi oleh ACM SIGCHI).

2.9.2 Delapan Aturan Emas Perancangan Antar Muka

Dalam merancang tampilan suatu aplikasi ada beberapa hal yang butuh diperhatikan. Dalam bukunya, Shneiderman (2005, p74), mengemukakan ada delapan aturan yang harus diperhatikan dalam perancangan layar tatap muka pengguna atau yang sering disebut *Eight Golden Rules of Interface Design*. Aturan – aturan tersebut antara lain, yaitu :

a. Berusaha untuk konsisten

   Berusaha untuk konsisten dalam aksi-aksi pada situasi tertentu, yaitu konsisten dalam penggunaan warna, bahasa, tata letak, *font*, simbol, dan sebagainya.

b. Menyediakan usability universal

   Menyediakan *shorcut* yang universal, yang sudah biasa digunakan, jangan membuat *shorcut* yang tidak dimengerti orang banyak.

c. Memberikan umpan balik yang informatif

   Dalam setiap aksi yang dilakukan pengguna, maka harus ada umpan balik yang sesuai dengan aksi tersebut sehingga membantu pengguna untuk

mengerti sistem pada aplikasi tersebut, seperti memberikan pesan apabila user melakukan suatu kesalahan.

d. Merancang dialog yang memberikan penutupan (keadaan akhir)

   Adanya keadaan akhir yang menandakan selesainya suatu kegiatan yang diberitahukan kepada pengguna melalui umpan balik, seperti memberikan tampilan atau jendela dialog bertanda user sudah mengakhiri pemakaian aplikasi.

e. Memberikan pencegahan kesalahan dan penanganan kesalahan yang sederhana

   Merancang sistem yang memungkinkan pengguna untuk tidak melakukan kesalahan fatal. Hal ini dapat diatasi dengan memberikan solusi kepada pengguna untuk menangani kesalahan tersebut.

f. Memungkinkan pembalikan aksi yang mudah

   Memberikan kebebasan kepada pengguna untuk dapat bergerak kemana saja dimana bila terjadi kesalahan maka pengguna dapat kembali tanpa harus khawatir akan rusaknya hal yang sedang dikerjakan.

g. Mendukung pusat kendali internal

   Dengan adanya pengaturan internal, maka pengguna dapat menggunakan sistem sesuai kebutuhan. Misalnya, adanya perbedaan hak akses antara anggota dengan bukan anggota pada aplikasi tersebut.

h. Mengurangi beban ingatan jangka pendek

   Kemampuan manusia untuk memproses informasi dalam jangka waktu pendek memiliki keterbatasan, sehingga perlu dirancang tampilan yang

sederhana dan efektif, agar tidak membebani ingatan manusia terlalu berat.

2.10 PHP

2.10.1 Pengertian PHP

Menurut Luke Welling dan Laura Thomson (2001, p1), PHP adalah *server-side scripting language* yang didesain secara spesifik untuk web. Dalam *page* HTML, dapat dimasukkan *code* PHP yang akan dieksekusi setiap kali halaman dikunjungi. PHP *code* diterjemahkan di *web-server* dan diubah menjadi HTML atau *output* lain yang akan dilihat oleh pengunjung halaman.

Menurut Peranginangin (2006, p2), PHP adalah singkatan dari PHP *Hypertext Prepocessor* yang digunakan sebagai *script server-side* dalam pengembangan web yang disisipkan pada dokumen HTML. PHP dikatakan sebagai sebuah *server-side embedded script language* artinya sintaks-sintaks dan perintah yang kita berikan akan sepenuhnya dijalankan oleh *server* tetapi disertakan pada halaman HTML biasa. Aplikasi-aplikasi yang dibangun oleh PHP pada umumnya akan memberikan hasil pada *web browser*, tetapi prosesnya secara keseluruhan dijalankan di *server*.

Ketika menggunakan PHP sebagai *server-side embedded script language* maka *server* akan melakukan hal-hal sebagai berikut :

a. Membaca permintaan dari *client/browser*.
b. Mencari halaman (*page*) di *server*.
c. Melakukan instruksi yang diberikan oleh PHP untuk melakukan modifikasi pada halaman (*page*).

d. Mengirim kembali halaman tersebut kepada *client* melalui internet atau intranet.

PHP adalah bahasa *open source* yang dapat digunakan di berbagai mesin (linux, unix, windows) dan dapat dijalankan secara *runtime* melalui *console* serta juga dapat menjalankan perintah-perintah sistem.

2.10.2 Kelebihan PHP

PHP memiliki beberapa kelebihan, yaitu :

a. Bahasa pemograman PHP adalah sebuah bahasa *script* yang tidak melakukan sebuah kompilasi dalam penggunaanya.
b. Banyak w*eb server* yang mendukung PHP *script* antara lain : Apache, AOLServer, Microsoft IIS, dan sebagainya. *Web server* ini dapat dijalankan pada berbagai sistem operasi, dengan konfigurasi yang relatif mudah.
c. Dalam sisi pengembangan lebih mudah, karena banyaknya *milis-milis* dan *developer* yang siap membantu dalam pengembangan.
d. Dalam sisi pemahaman, PHP adalah bahasa *scripting* yang paling mudah karena memiliki banyak referensi.
e. PHP mendukung banyak paket *database*, baik yang komersil maupun nonkomersil, seperti Oracle, Informix, MySQL, Microsoft SQL Server, dan lain-lain.

2.11 My SQL

Menurut Sukarno (2006, p3), MySQL merupakan perangkat lunak untuk sistem manajemen *database* (*database management system*). Karena sifatnya yang *open source* dan memiliki kemampuan menampung kapasitas yang sangat besar, maka MySQL menjadi *database* yang sangat popular dikalangan *programmer* web.

Menurut Luke Welling dan Laura Thomson (2001, p1), MySQL adalah sebuah *Relational Database Management System* (RDBMS) yang sangat cepat dan kuat.

MySQL adalah sebuah perangkat lunak sistem management basis data SQL atau DBMS yang *multithread, multiuser*, dengan sekitar 6 juta instalasi diseluruh dunia. MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael “Monty” Widenius. Tersedia sebagai perangkat lunak gratis dibawah lisensi GNU General Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaanya tidak cocok dengan penggunaan GPL. Selain itu terdapat juga sebuah perangkat lunak gratis untuk administrasi basis data MySQL berbasis web yang sangat populer yaitu phpMyAdmin.

MySQL sebenarnya merupakan turunan salah satu konsep utama dalam *database* sejak lama, yaitu SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Beberapa keunggulan dari MySQL, yaitu :

a. Mampu menangani jutaan pengguna dalam waktu bersamaan.
b. Mampu menampung lebih dari 50.000.000 *record*.
c. Sangat cepat mengeksekusi perintah.

d. Memiliki *user privilege* yang mudah dan efisien.