# BAB II

# LANDASAN TEORI

## 2.1 RPL (Rekayasa Perangkat Lunak)

Istilah rekayasa perangkat lunak secara umum disepakati sebagai terjemahan dari istilah *software engineering*. Istilah *software engineering* mulai dipopulerkan tahun 1968 pada *software engineering conference* yang diselenggarakan oleh NATO *(North Atlantic Treaty Organization)*. Sebagian orang mengartikan RPL hanya sebatas pada bagaimana membuat program komputer. Padahal ada perbedaan yang mendasar antara perangkat lunak *(software)* dan program komputer.

Perangkat lunak adalah seluruh perintah yang digunakan untuk memproses informasi. Perangkat lunak dapat berupa program atau prosedur. Program adalah kumpulan perintah yang dimengerti oleh komputer, sedangkan prosedur adalah perintah yang dibutuhkan oleh pengguna dalam memroses informasi.

Sedangkan RPL sendiri berarti suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal yaitu analisa kebutuhan pengguna, penentuan spesifikasi dari kebutuhan pengguna, desain, pengkodean, pengujian sampai pemeliharaan sistem setelah digunakan.

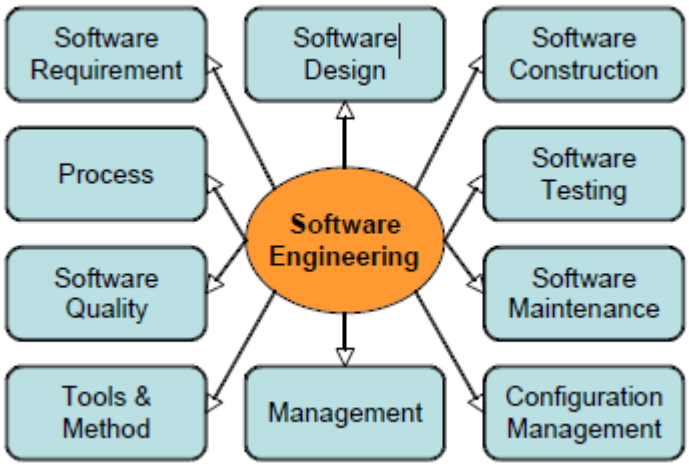
### 2.1.1 Tujuan RPL (Rekayasa Perangkat Lunak)

Tujuan dari dibuatnya rekaya perangkat lunak adalah :

a. Memperoleh biaya produksi perangkat lunak yang rendah.
b. Menghasilkan perangkat lunak yang berkinerja tinggi, handal, dan tepat waktu.
c. Menghasilkan perangkat lunak yang dapat bekerja pada berbagai jenis *platform*.
d. Menghasilkan perangkat lunak yang biaya perawatannya rendah.

### 2.1.2 Ruang Lingkup RPL

Sesuai dengan definisi yang disampaikan sebelumnya, maka ruang lingkup RPL dapat digambarkan sebagai pada Gambar 2.1 :



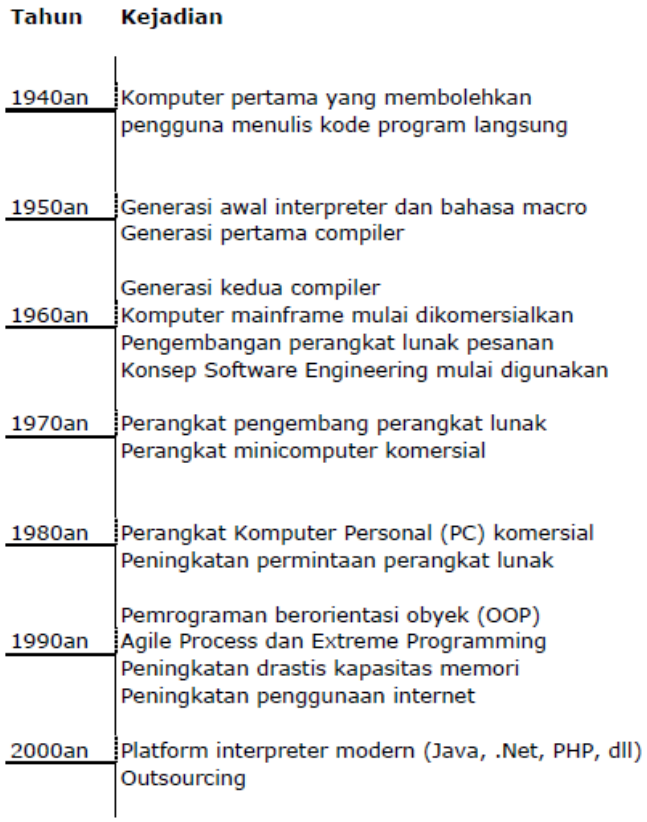

**Gambar 2.1. Ruang Lingkup RPL**

a. *Software requirements*, berhubungan dengan spesifikasi kebutuhan dan persyaratan perangkat lunak.
b. *Software design*, mencakup proses penentuan arsitektur, komponen, antarmuka, dan karakteristik lain dari perangkat lunak.
c. *Software construction*, berhubungan dengan detail pengembangan perangkat lunak, termasuk algoritma, pengkodean, pengujian, dan pencarian kesalahan.
d. *Software testing*, meliputi pengujian pada keseluruhan perilaku perangkat lunak.
e. *Software maintenance*, mencakup upaya-upaya perawatan ketika perangkat lunak telah dioperasikan.
f. *Software configuration*, pengelolaan yang berhubungan dengan usaha perubahan konfigurasi perangkat lunak untuk memenuhi kebutuhan tertentu.
g. *Software engineering*, kegiatan yang berkaitan dengan pengelolaan dan pengukuran RPL, termasuk perencanaan proyek perangkat lunak.
h. *Software engineering*, *tools and methods* mencakup kajian teoritis tentang alat bantu dan metode RPL.

i. *Software engineering process*, berhubungan dengan definisi, implementasi, pengukuran, pengelolaan, perubahan dan perbaikan proses RPL.
j. *Software quality*, berhubungan dengan kualitas dan daur hidup perangkat lunak.

### 2.1.3 Perkembangan RPL

Meskipun baru dicetuskan pada tahun 1968, namun RPL telah memiliki sejarah yang cukup panjang, dalam Gambar 2.2 dijelaskan tahap-tahap perkembangan RPL. Jika dilihat dari sisi disiplin ilmu, RPL merupakan disiplin ilmu relatif masih muda dan akan terus berkembang.

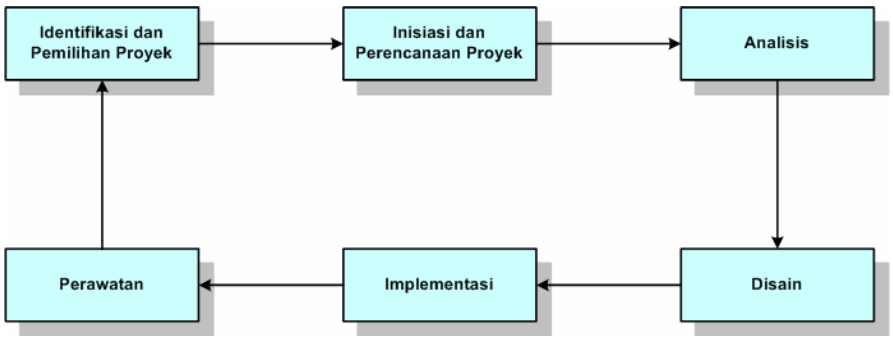



**Gambar 2.2. Perkembangan RPL**

## 2.2 Model Proses RPL (Rekayasa Perangkat Lunak)

Pada rekayasa perangkat lunak, banyak model yang telah dikembangkan untuk membantu proses pengembangan perangkat lunak. Model-model ini pada umumnya mengacu pada model proses pengembangan

sistem yang disebut SDLC *(System Development Life Cycle*), seperti terlihat pada Gambar 2.3.




**Gambar 2.3.** ***System Development Life Cycle (SDLC)***

Setiap model yang dikembangkan mempunyai karakteristik sendiri-sendiri, namun secara umum ada persamaan dari model-model ini, yaitu :

a. Kebutuhan terhadap definisi masalah yang jelas. Input utama dari setiap model pengembangan perangkat lunak adalah pendefinisian masalah yang jelas, semakin jelas akan semakin baik karena akan memudahkan dalam penyelesaian masalah. Oleh karena itu pemahaman masalah merupakan bagian penting dari model pengembangan perangkat lunak.
b. Tahapan-tahapan pengembangan yang teratur. Meskipun model-model pengembangan perangkat lunak memiliki pola yang berbeda-beda, biasanya model-model tersebut mengikuti pola umum, yaitu *analysis – design – coding –testing - maintenance.*
c. *Stakeholder* berperan sangat penting dalam keseluruhan tahapan pengembangan. *Stakeholder* dalam rekayasa perangkat lunak dapat berupa pengguna, pemilik, pengembang, pemrogram dan orang-orang yang terlibat dalam rekayasa perangkat lunak.
d. Dokumentasi merupakan bagian penting dari pengembangan perangkat lunak. Masing-masing tahapan dalam model biasanya menghasilkan sejumlah tulisan, diagram, gambar, atau bentuk-bentuk lain yang harus didokumentasikan dan merupakan bagian tak terpisahkan dari perangkat lunak yang dihasilkan. Keluaran dari proses pengembangan perangkat lunak harus bernilai ekonomis. Nilai dari sebuah perangkat lunak

sebenarnya agak susah dirupiah-kan, namun efek dari penggunaan perangkat lunak yang telah dikembangkan haruslah memberi nilai tambah bagi organisasi. Hal ini dapat berupa penurunan biaya operasi, efisiensi penggunaan sumber daya, peningkatan keuntungan organisasi, peningkatan *image* organisasi dan lain-lain.

Berikut ini adalah beberapa model pengembangan perangkat lunak :

*a. The Waterfall Model*

*b. Prototyping*

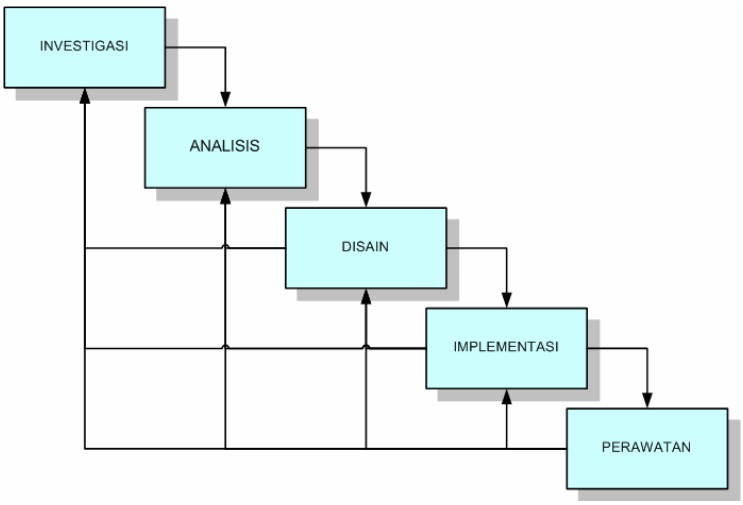
*c. Unified Processs (UP).*

### *2.2.1 The waterfall model*

Model siklus hidup adalah model utama dan dasar dari banyak model. Salah satu model yang cukup dikenal dalam dunia rekayasa perangkat lunak adalah *the waterfall model*. Ada 5 tahapan utama dalam *the waterfall model* seperti terlihat pada Gambar 2.4. Disebut *waterfall* atau air terjun karena diagram tahapan prosesnya mirip dengan air terjun yang bertingkat, tahapan-tahapan dalam *the waterfall model* secara ringkas adalah sebagai berikut:

a. Tahap investigasi dilakukan untuk menentukan apakah terjadi suatu masalah atau adakah peluang suatu sistem informasi dikembangkan. Pada tahapan ini studi kelayakan perlu dilakukan untuk menentukan apakah sistem informasi yang akan dikembangkan merupakan solusi yang layak.
b. Tahap analisis bertujuan untuk mencari kebutuhan pengguna dan organisasi serta menganalisa kondisi yang ada (sebelum diterapkan sistem informasi yang baru).
c. Tahap desain bertujuan untuk menentukan spesifikasi detail dari komponen-komponen sistem informasi (manusia, *hardware, software, network,* dan data) dan produk-produk informasi yang sesuai dengan hasil tahap analisis.

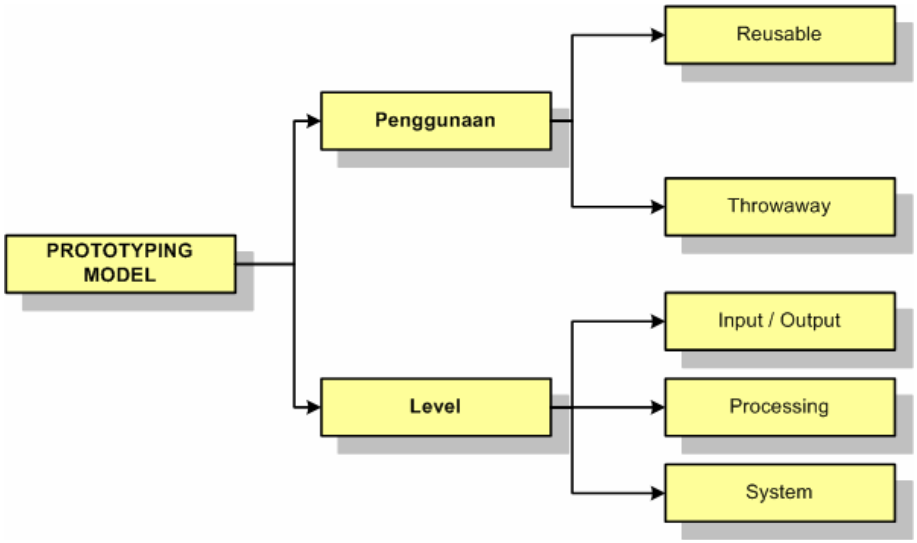
d. Tahap implementasi merupakan tahapan untuk mendapatkan atau mengembangkan *hardware* dan *software* (pengkodean program), melakukan pengujian, pelatihan dan perpindahan ke sistem baru.
e. Tahapan perawatan *(maintenance)* dilakukan ketika sistem informasi sudah dioperasikan. Pada tahapan ini dilakukan *monitoring proses,* evaluasi dan perubahan (perbaikan) bila diperlukan.




**Gambar 2.4. Model *Waterfall***

### 2.2.2 Model *Prototyping*

*Prototyping* adalah salah satu pendekatan dalam rekayasa perangkat lunak yang secara langsung mendemonstrasikan bagaimana sebuah perangkat lunak atau komponen-komponen perangkat lunak akan bekerja dalam lingkungannya sebelum tahapan konstruksi aktual dilakukan. *Prototyping* model dapat diklasifikasikan menjadi beberapa tipe seperti terlihat pada Gambar 2.5.




**Gambar 2.5. Klasifikasi Model *Prototyping***

a. *Reusable prototype* :

   *Prototype* yang akan ditransformasikan menjadi produk final.

*b. Throwaway prototype* :

   *Prototype* yang akan dibuang begitu selesai menjalankan maksudnya.

c. *Input/output prototype* :

   *Prototype* yang terbatas pada antar muka pengguna (*user interface*).

d. *Processing prototype* :

   *Prototype* yang meliputi perawatan *file* dasar dan proses-proses transaksi.

e. *System prototype* :

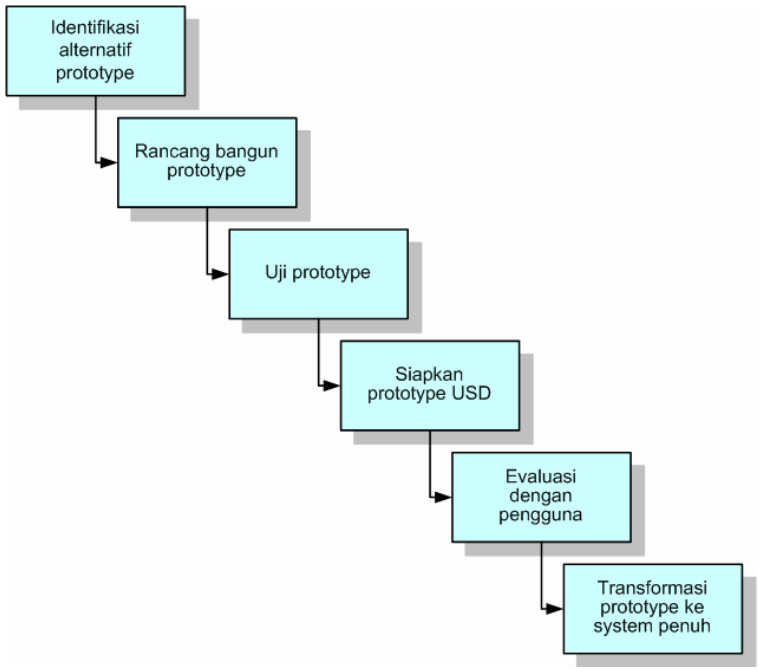
   *Prototype* yang berupa model lengkap dari perangkat lunak.

Tahap-tahap dalam *prototyping* boleh dikata merupakan tahap-tahap yang dipercepat. Strategi utama dalam *prototyping* adalah kerjakan yang mudah terlebih dahulu dan sampaikan hasil kepada pengguna sesegera mungkin. *Prototyping* dapat dibagai ke dalam enam tahapan seperti terlihat pada Gambar 2.6.

Tahapan-tahapan secara ringkas dapat dijelaskan sebagai berikut:

a. Identifikasi kandidat *prototyping*. Kandidat dalam kasus ini meliputi *user interface* (*menu*, *dialog*, *input* dan *output*), *file-file* transaksi utama, dan fungsi-fungsi pemrosesan sederhana.

b. Rancang bangun *prototype* dengan bantuan *software* seperti *word processor, spreadsheet, database,* pengolah grafik, dan *software CASE (Computer-Aided System Engineering).*
c. Uji *prototype* untuk memastikan *prototype* dapat dengan mudah dijalankan untuk tujuan demonstrasi.
d. Siapkan *prototype USD (User's System Diagram)* untuk mengidentifikasi bagian-bagian dari perangkat lunak yang di-*prototype*-kan.
e. Evaluasi dengan pengguna untuk mengevaluasi *prototype* dan melakukan perubahan jika diperlukan.
f. Transformasikan *prototype* menjadi perangkat lunak yang beroperasi penuh dengan melakukan penghilangan kode-kode yang tidak dibutuhkan, penambahan program-program yang memang dibutuhkan dan perbaikan dan pengujian perangkat lunak secara berulang.



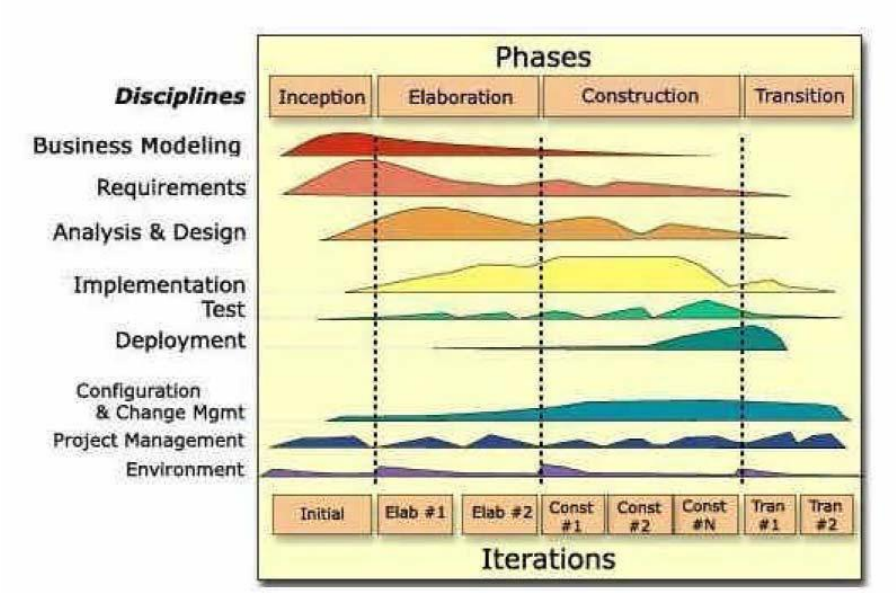

**Gambar 2.6. Tahapan-tahapan Model *Prototyping***

### 2.2.3 *Unified Process* (UP) dan *Unified Modeling Language* (UML)

UP (*Unified Process*) atau kadang disebut sebagai USDP (*Unified Software Development Process*) adalah kerangka proses pengembangan

yang bersifat *use-case-driven*, berpusat pada arsitektur perangkat lunak, interatif dan tumbuh-kembang. Kerangka pengembangan ini termasuk baru dalam metodologi pengembangan perangkat lunak. UP dapat diaplikasikan pada berbagai skala proyek, mulai dari skala kecil sampai dengan skala besar.

Daur hidup UP secara umum akan tampak seperti pada bagan di Gambar 2.7. Bagan ini biasa disebut sebagai *hump chart*. Pada bagan ini terlihat ada empat tahap pengembangan yaitu *inception, elaboration, construction* dan *transition.* Selain itu tampak pula sejumlah aktivitas yang harus dilakukan sepanjang pengembangan perangkat lunak, yaitu, *business modeling*, *requirements*, *analysis* and *design*, *implementation*, *test*. Tahap dan aktivitas tersebut akan dilakukan secara iteratif.




**Gambar 2.7. RUP *Life Cycle***

Penjelasan singkat untuk empat tahapan dalam UP adalah sebagai berikut:

a. *Inception*, tahapan ini merupakan tahapan paling awal dimana aktivitas penilaian terhadap sebuah proyek perangkat lunak dilakukan. Tujuannya adalah untuk mendapatkan kesepakatan dari *stakeholder* sehubungan dengan tujuan dan dana proyek.
b. *Elaboration*, tujuan dari tahap ini adalah untuk mendapatkan gambaran umum kebutuhan, persyaratan, dan fungsi-fungsi utama perangkat

lunak. Hal ini penting untuk mengetahui secara lebih baik resiko-resiko proyek, baik meliputi resiko arsitektur perangkat lunak, perencanaan, maupun implementasi. Pada tahap ini telah dimulai rancang bangun perangkat lunak secara *iterative* melalui aktivitas-aktivitas seperti *business modeling*, *requirements*, *analysis* dan *design* meskipun baru pada tahap awal.

c. *Construction*, tujuan dari tahapan ini adalah membangun perangkat lunak sampai dengan saat perangkat lunak tersebut siap digunakan. Titik berat tahapan ini adalah pada penentuan tingkat prioritas kebutuhan / persyaratan, melengkapi spesifikasinya, analisis lebih dalam, desain solusi yang memenuhi kebutuhan dan persyaratan, pengkodean dan pengujian perangkat lunak. Jika dimungkinkan versi awal dari perangkat lunak diuji coba untuk mendapatkan masukan dari pengguna.
d. *Transition*, tahap ini difokuskan pada bagaimana menyampaikan perangkat lunak yang sudah jadi pada pengguna. Perangkat lunak akan secara resmi diuji oleh baik oleh penguji (*tester*) yang kompeten maupun oleh pengguna. Beberapa aktivitas seperti pemindahan pusat data dan pelatihan pengguna dan staf pendukung harus dilakukan pada tahap ini.

Dalam pengembangan perangkat lunak dengan menggunakan UP, maka tidak lepas dari penggunaan notasi-notasi yang biasa disebut sebagai UML *(Unified Modeling Language)*. Meskipun UP mensyaratkan penggunaan UML, namun UML sendiri dapat digunakan pada berbagai metodologi yang lain bahkan dapat digunakan pada bidang selain sistem informasi. UML adalah bahasa pemodelan standar atau kumpulan teknik-teknik pemodelan untuk melakukan spesifikasi, visualisasi, konstruksi, dan dokumentasi hasil kerja dalam pengembangan perangkat lunak. UML lahir dari penggabungan banyak bahasa pemodelan grafis berorientasi obyek yang berkembang pesat pada akhir tahun 1980-an dan awal 1990-an.

Secara sederhana UML digunakan untuk menggambar sketsa sistem. Pengembang menggunakan UML untuk menyampaikan beberapa aspek dari sebuah perangkat lunak melalui notasi grafis. UML mendefinisikan notasi dan semantik. Notasi merupakan sekumpulan bentuk khusus yang memiliki makna tertentu untuk menggambarkan berbagai diagram piranti lunak, dan semantik mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Ada beberapa jenis diagram yang disediakan dalam UML, antara lain adalah:

a. *Use-case diagram*. Diagram ini berguna untuk menggambarkan interaksi antara pengguna dengan sebuah perangkat lunak.
b. *Activity diagram*. Diagram ini berguna untuk menggambarkan prosedur-prosedur perilaku perangkat lunak.
c. *Class diagram*. Diagram ini berguna untuk menggambarkan class, fitur, dan hubungan-hubungan yang terjadi. Pada diagram ini pendekatan berorientasi obyek memegang peranan yang sangat penting.
d. *Sequence diagram*. Diagram ini berguna untuk menggambarkan interaksi antar obyek dengan penekanan pada urutan proses atau kejadian.
e. *State machine diagram*. Diagram ini digunakan untuk menggambarkan bagaimana suatu kejadian mengubah obyek selama masa hidup obyek tersebut.
f. *Component diagram*. Diagram ini berguna untuk menggambarkan struktur dan koneksi komponen.

## 2.3 Tahapan RPL (Rekayasa Perangkat Lunak)

Berikut adalah tahapan-tahapan dalam rekayasa perangkat lunak :

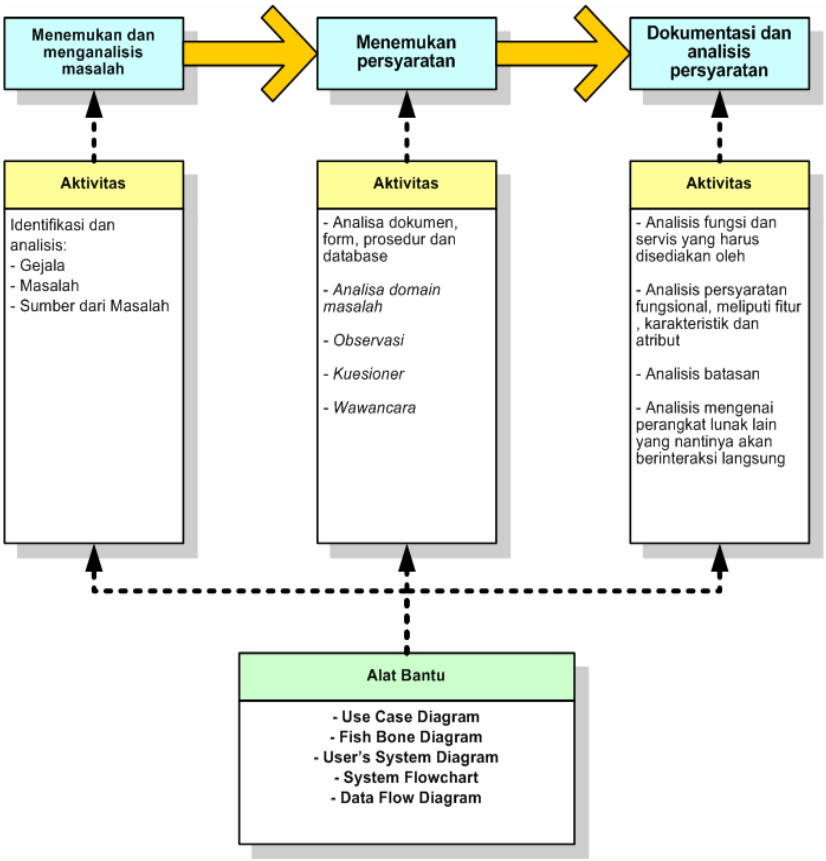
a. Analisis
b. Desain
c. Pengkodean atau *coding*
d. Pengujian atau *testing*
e. Perawatan atau *maintenance*

### 2.3.1 Analisis

Analisis sistem adalah sebuah teknik pemecahan masalah yang menguraikan sebuah sistem menjadi komponen-komponennya dengan tujuan mempelajari seberapa bagus komponen-komponen tersebut bekerja dan berinteraksi untuk meraih tujuan mereka.

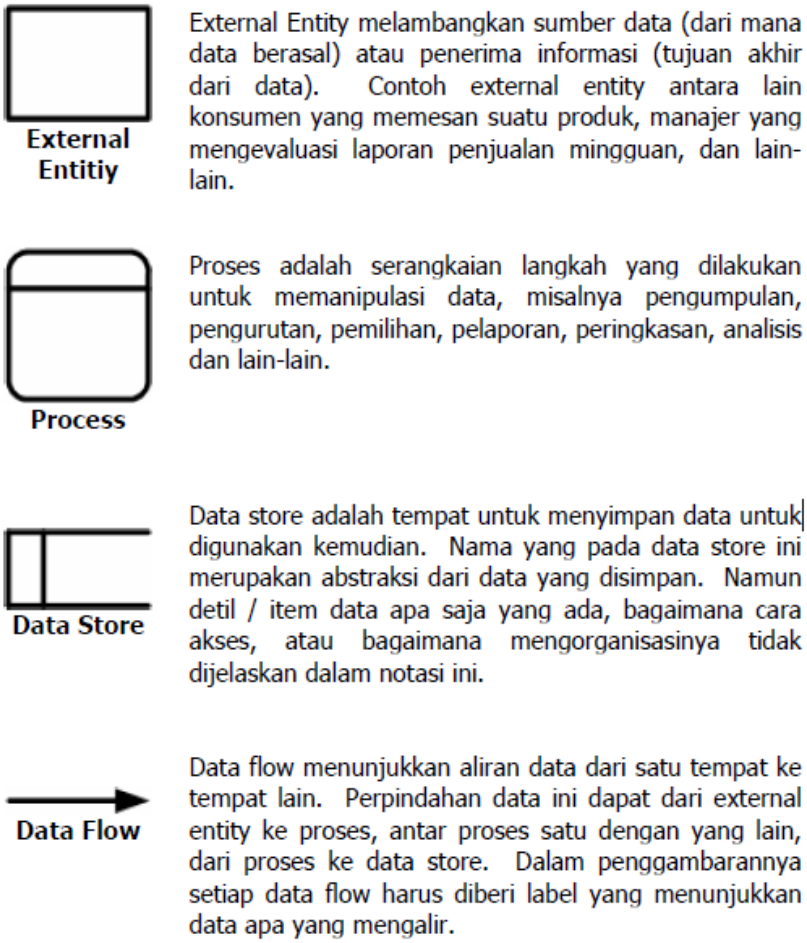
Analisis mungkin adalah bagian terpenting dari proses rekayasa perangkat lunak. Karena semua proses lanjutan akan sangat bergantung pada baik tidaknya hasil analisis. Tahapan-tahapan dalam analisis rekayasa perangkat lunak secara ringkas dapat dilihat pada Gambar 2.8.

Ada satu bagian penting yang biasanya dilakukan dalam tahapan analisis yaitu pemodelan proses bisnis. Model proses adalah model yang memfokuskan pada seluruh proses di dalam sistem yang mentransformasikan data menjadi informasi. Model proses juga menunjukkan aliran data yang masuk dan keluar pada suatu proses. Biasanya model ini digambarkan dalam bentuk diagram arus data (*Data Flow Diagram* / DFD). DFD meyajikan gambaran apa yang manusia, proses, dan prosedur lakukan untuk mentransformasi data menjadi informasi.




**Gambar 2.8. Tahapan dan Aktifitas didalam Analisis**

Umumnya ada empat notasi yang sering digunakan dalam DFD seperti tampak pada Gambar 2.9.




**Gambar 2.9. Notasi Pada DFD**

Sedangkan aturan-aturan umum yang sering digunakan dalam pembuatan DFD adalah seperti pada Gambar 2.10,

| Kelompok | Aturan |
|---|---|
| Umum | • input-input ke suatu process akan selalu berbeda dengan output-outputnya<br>• obyek obyek (External Entity, Process, Data Storage, dan Data Flow) yang ada pada suatu DFD selalu memiliki nama yang unik |
| External Entity | • nama yang dipakai pada External Entity selalu menggunakan kata benda<br>• data tidak boleh mengalir secara langsung dari External Entity yang satu ke External Entity yang lain |
| Process | • nama yang dipakai pada Process selalu menggunakan kata kerja<br>• tidak ada Process yang hanya menghasilkan output<br>• tidak ada Process yang hanya menerima input |
| Data Storage | • nama yang dipakai pada Data Storage selalu menggunakan kata benda<br>• data tidak boleh mengalir secara langsung dari Data Storage yang satu ke Data Storage yang lain<br>• data tidak boleh mengalir secara langsung dari External Entity ke Data Storage demikian juga sebaliknya. |
| Data Flow | • nama yang dipakai pada Data Flow selalu menggunakan kata benda<br>• Data Flow di antara dua notasi hanya memiliki satu arah aliran<br>• Percabangan (fork) menunjukkan adanya data yang persis sama yang mengalir dari suatu tempat ke dua atau lebih tempat yang lain<br>• Penggabungan (join) menunjukkan adanya data yang persis sama yang mengalir dua atau lebih tempat menuju satu tempat yang lain<br>• Data Flow menuju Data Storage berarti terjadi update data<br>• Data Flow dari Data Storage berarti terjadi pembacaan / pengambilan data |

**Gambar 2.10. Aturan-aturan didalam DFD**

#### 2.3.1.1 ERD *(Entity Relationship Diagram)*

Pemodelan awal basis data yang paling banyak digunakan adalah menggunakan ERD (*Entity Relationship Diagram*). ERD dikembangkan berdasarkan teori himpunan dalam bidang matematika. ERD digunakan untuk pemodelan basis data relasional. Sehingga jika penyimpanan basis data menggunakan OO DBMS maka perancangan basis data tidak perlu menggunakan ERD. Berikut pada Gambar 2.11. adalah simbol-simbol yang digunakan pada ERD.

| Simbol | Deskripsi |
|---|---|
| Entitas / *entity*<br>nama_entitas | Entitas merupakan data inti yang akan di simpan; bakal tabel pada basis data |
| Atribut<br>nama_atribut | *Field* atau kolom data yang butuh di simpan dalam suatu entitas |
| Atribut kunci primer<br>nama_kunci_primer | *Field* atau kolom data yang butuh di simpan dalam suatu entitas dan digunakan sebagai kunsinakses *record* yang diinginkan; biasanya berupa id |
| Atribut multinilai / *multivalue*<br>nama_atribut | *Field* atau kolom data yang butuh di simpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu |
| Relasi<br>nama_relasi | Relasi yang menghubungkan antar entitas; biasanya diawali dengan kata kerja |
| Asosiasi / *association*<br>1 O..* | Penghubung antara relasi dan entitas di mana di kedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian |

**Gambar 2.11 Simbol ERD**

Diagram E-R digunakan untuk menggambarkan secara sistematis hubungan antar *entity-entity* yang ada dalam suatu sistem *database* menggunakan simbol-simbol sehingga lebih mudah dipahami, simbol-simbol yang dimaksud adalah :

1. *Entity*

   *Entity* merupakan objek yang mewakili sesuatu yang nyata dan dapat dibedakan dari sesuatu yang lain. Simbol dari *entity* ini biasanya digambarkan dengan persegi panjang.

2. Atribut

   Setiap *entity* pasti mempunyai elemen yang disebut atribut, yang berfungsi untuk mendeskripsikan karakteristik dari entitas tersebut. Isi dari atribut mempunyai sesuatu yang dapat mengidentifikasikan isi

elemen satu dengan yang lain, gambar atribut diwakili oleh simbol elips.

3. Hubungan/Relasi

   Hubungan antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda, hubungan atau relasi diwakili oleh simbol belah ketupat.

4. Derajat relasi

   Dalam hubungan antar *entity*, juga harus ditentukan derajat relasi antar *entity*. Derajat relasi menunjukkan jumlah maksimum *record* suatu *entity* berelasi dengan *record* pada *entity* yang lainnya. Derajat relasi terbagi menjadi empat derajat, yaitu :

   a. Relasi *one to one*

      Relasi *one to one* artinya satu *record* pada *entity* A berelasi paling banyak satu *record* juga pada *entity* B, begitu juga sebaliknya, satu *record* pada *entity* B, berelasi paling banyak satu *record* juga dengan *entity* A. Dalam diagram E-R, relasi ini disimbolkan dengan angka 1.

   b. Relasi *one to many*

      Relasi *one to many* artinya satu *record* pada *entity* A berelasi dengan beberapa *record* pada *entity* B, tapi tidak sebaliknya. Setiap *record* pada *entity* B berelasi paling banyak satu *record* dengan *entity* A. Dalam diagram E-R, relasi ini disimbolkan dengan angka 1 untuk menyatakan satu dan huruf M atau N untuk menyatakan banyak.

   c. Relasi *many to many*

      Relasi *many to many* artinya beberapa *record* pada *entity* A dapat berelasi dengan beberapa *record* juga pada *entity* B, begitu juga sebaliknya, beberapa *record* pada *entity* B dapat berelasi dengan beberapa *record* juga pada *entity* A. Dalam diagram E-R, relasi ini disimbolkan dengan huruf M atau N untuk menyatakan banyak.

   d. Relasi *many to one*

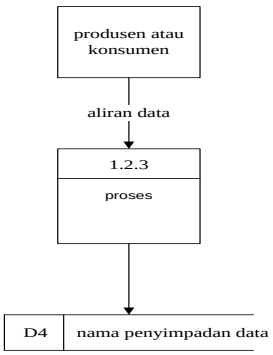
Jenis relasi ini adalah kebalikan dari relasi satu ke banyak, dimana setiap *record* pada *entity* A hanya dapat berelasi paling banyak 1 *record* pada *entity* B, tapi tidak sebaliknya, satu *record* pada *entity* B dapat berelasi dengan beberapa *record* pada *entity* A. Dalam diagram E-R, relasi ini disimbolkan dengan angka 1 untuk menyatakan satu dan huruf M atau N untuk menyatakan banyak.

#### 2.3.1.2 Diagram Konteks

Diagram konteks menggambarkan aplikasi dalam satu lingkaran dan hubungan dengan entitas luar, dimana lingkaran tersebut menggambarkan keseluruhan proses dalam aplikasi. Dalam penggambaran ini, sistem dianggap sebagai sebuah objek yang tidak dijelaskan secara rinci, karena yang ditekankan adalah interaksi sistem dengan lingkungan yang mengaksesnya.

#### 2.3.1.3 DFD *(Data Flow Diagram)*

DFD awalnya dikembangkan oleh Chris Gane dan Tris Saron pada tahun 1979 yang termasuk dalam SSADM (*Structured System Analysis and Design Methodology*) yang ditulis oleh Chris Gane dan Trish Saron. Sistem yang dikembangkan ini berbasis pada dekomposisi fungsional dari sebuah sistem. Berikut adalah contoh DFD yang dikembangkan oleh Chris Gane dan Trish Sarson:




**Gambar 2.12. Contoh DFD yang dikembangkan Chris Gane & Trish Sarson**

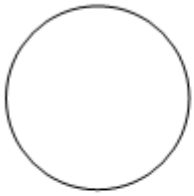
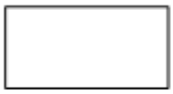
Edward Yourdon dan Tom Demarco memperkenalkan metode yang lain pada tahun 1980-an dimana mengubah persegi dengan sudut lengkung dengan lingkaran untuk menotasikan. DFD Edward Yourdon dan Tom Demarco populer digunakan sebagai model analisis sistem perangkat lunak untuk sistem perangkat lunak yang akan diimplementasikan dengan pemrograman terstruktur.

Informasi yang ada di dalam perangkat lunak dimodifikasi dengan beberapa transformasi yang dibutuhkan. *data flow diagram* atau dalam bahasa Indonesia menjadi diagram alir data adalah representasi grafik yang menggambarkan aliran informasi dan transformasi informasi yang diaplikasikan sebagai data yang mengalir dari masukan *(input)* dan keluaran *(output).*

DFD dapat digunakan untuk merepresentasikan sebuah sistem atau perangkat lunak pada beberapa level abstraksi. DFD dapat dibagi menjadi beberapa level yang lebih detail untuk merepresentasikan aliran informasi atau fungsi yang lebih detail. DFD menyediakan mekanisme untuk pemodelan fungsional ataupun pemodelan aliran informasi. Oleh karena itu, DFD lebih sesuai digunakan untuk memodelkan fungsi-fungsi perangkat lunak yang akan diimplementasikan menggunakan pemrograman terstruktur karena pemrograman terstruktur membagi-bagi bagiannya dengan fungsi-fungsi dan prosedur-prosedur.

DFD tidak sesuai untuk memodelkan sistem perangkat lunak yang akan dibangun menggunakan pemrograman berorientasi objek. Dikarenakan paradigma pemrograman terstruktur dan pemrograman berorientasi objek merupakan hal yang berbeda.

Notasi-notasi pada DFD Edward Yourdon dan Tom Demarco adalah sebagai berikut :

| Notasi | Keterangan |
|---|---|
|  | **Proses** atau fungsi atau prosedur; pada pemodelan perangkat lunak yang akan diimplementasikan dengan pemrograman terstruktur, maka pemodelan notasi inilah yang harusnya menjadu fungsi atau prosedur di dalam kode program.<br><br>catatan:<br>nama yang diberikan pada sebuah proses biasanya berupa kata kerja |
|  | *File* atau basis data atau penyimpanan *(storage)*; pada pemodelan perangkat lunak yang akan diimplementasikan dengan pemrograman terstruktur, maka pemodelan notasi inilah yang harusnyadibuat menjadi tabel-tabel basis data yang dibutuhkan, tabel-tabel ini juga harus sesuai dengan perancangan tabel-tabel pada basis data (*Entity Relationship Diagram* (ERD), *Conceptual Data Model* (CDM), *Physical Data Model* (PDM))<br><br>catatan:<br>nama yang diberikan pada sebuah penyimpanan biasanya kata benda |
|  | *Entitas* luar *(external entity)* atau masukkan *(input)* atau keluaran *(output)* atau orang yang memakai/ berinteraksi dengan perangkat lunak yang dimodelakan atau sistem lain yang terkait dengan aliran data dari sistem yang di modelakan.<br><br>catatan:<br>nama yang digunakan pada masuan *(input)* atau keluaran *(output)* biasanya berupa kata benda |
|  | *Aliran* data; merupakan data yang di kirim antar proses, dari penyimpanan ke proses, atau dari proses ke masukkan (input) atau keluaran (output)<br><br>catatan:<br>nama yang digunakan pada aliran data biasanya berupa kata benda, dapat di awali dengan kata data misalnya 'data siswa' atau tanpa kata data misalnya 'siswa' |

**Gambar 2.13 Notasi-notasi pada DFD**

Berikut ini adalah tahap-tahap perancangan dengan menggunakan DFD :

1. Membuat DFD *level* 0

   DFD *level* 0 atau sering disebut juga *context diagram*, DFD *level* 0 menggambarkan sistem yang akan dibuat sebagai suatu entitas tunggal yang berinteraksi dengan orang maupun sistem lain. DFD *level* 0 digunakan untuk menggambarkan interaksi antara sistem yang akan dikembangkan dengan entitas luar.

2. Membuat DFD l*evel* 1

   DFD *level* 1 digunakan untuk menggambarkan modul-modul yang ada dalam sistem yang akan dikembangkan. DFD *level* 1 merupakan hasil *breakdown* DFD *level* 0 yang sebelumnya sudah dibuat.

3. Membuat DFD *level* 2

   Modul-modul pada DFD *level* 1 dapat di-*breakdown* menjadi DFD *level* 2. Modul mana saja yang harus di-*breakdown* lebih detail tergantung pada tingkat kedetailan modul tersebut. Apabila modul tersebut sudah cukup detail dan rinci maka modul tersebut sudah tidak perlu untuk di-*breakdown* lagi. Untuk sebuah sistem, jumlah DFD *level* 2 sama dengan jumlah modul pada DFD *level* 1 yang di-*breakdown*.

4. Membuat DFD *level* 3 dan seterusnya

   DFD *level* 3, 4, 5, dan seterusnya merupakan *breakdown* dari modul pada DFD *level* diatasnya. *Breakdown* pada DFD *level* 3, 4, 5, dan seterusnya aturannya sama persis dengan DFD *level* 1 atau *level* 2.

Pada suatu diagram DFD sebaiknya jumlah modul tidak boleh lebih dari 20 buah. Jika lebih dari 20 buah modul, diagram akan terlihat rumit dan susah untuk dibaca sehingga menyebabkan sistem yang dikembangkan juga menjadi rumit.

#### 2.3.1.4 Kamus Data

Kamus data (*data dictionary*) dipergunakan untuk memperjelas aliran data yang digambarkan pada DFD. Kamus data adalah kumpulan daftar elemen data yang mengalir pada sistem perangkat lunak sehingga

masukkan (*input*) dan keluaran (*output*) dapat dipahami secara umum (memiliki standar cara penulisan). Kamus data biasanya terdiri dari :

1. Nama, nama dari data.
2. Digunakan pada, merupakan proses-proses yang terkait data.
3. Deskripsi, merupakan deskripsi data.
4. Informasi tambahan, seperti tipe data, nilai data, batas nilai data, dan komponen yang membentuk data.

Komponen data memiliki beberapa simbol untuk menjelaskan informasi tambahan sebagai berikut:

| Simbol | Keterangan |
|---|---|
| = | di susun atau terdiri dari |
| + | dan |
| [\|] | baik … atau ... |
| $\{\}^n$ | n kali di ulang/bernilai banyak |
| ( ) | data opsional |
| *…* | batas komentar |

**Gambar 2.14 Simbol Kamus Data**

Kamus data pada DFD harus dapat dipetakan dengan hasil perancangan basis data yang dilakukan sebelumnya. Jika ada kamus data yang tidak dapat dipetakan pada tabel hasil perancangan basis data berarti hasil perancangan basis data dengan perancangan dengan DFD masih belum sesuai, sehingga harus ada yang diperbaiki baik perancangan basis datanya, perancangan DFD nya, atau keduanya.

### 2.3.2 Desain

Desain perangkat lunak adalah tugas, tahapan atau aktifitas yang difokuskan pada spesifikasi detail dari solusi berbasis komputer. Desain perangkat lunak sering juga disebut sebagai *physical design*. Jika tahapan analisis sistem menekankan pada masalah bisnis *(business rule)*, maka sebaliknya desain perangkat lunak fokus pada sisi teknis dan implementasi sebuah perangkat lunak.

*Output* utama dari tahapan desain perangkat lunak adalah spesifikasi desain. Spesifikasi ini meliputi spesifikasi desain umum yang akan disampaikan kepada *stakeholder* sistem dan spesifikasi desain rinci yang akan digunakan pada tahap implementasi. Spesifikasi desain umum hanya berisi gambaran umum agar *stakeholder* sistem mengerti akan seperti apa perangkat lunak yang akan dibangun. Biasanya diagram DFD tentang perangkat lunak yang baru merupakan poin penting dibagian ini. Spesifikasi desain rinci atau kadang disebut desain arsitektur rinci perangkat lunak diperlukan untuk merancang sistem sehingga memiliki konstruksi yang baik, proses pengolahan data yang tepat dan akurat, bernilai, memiliki aspek *user friendly* dan memiliki dasar-dasar untuk pengembangan selanjutnya.

Desain arsitektur ini terdiri dari desain *database*, desain proses, desain *user interface* yang mencakup desain *input*, *output form* dan *report*, desain *hardware*, *software* dan jaringan. Desain proses merupakan kelanjutan dari pemodelan proses yang dilakukan pada tahapan analisis.

### 2.3.3 Konstruksi

Konstruksi adalah tahapan menerjemahkan hasil desain logis dan fisik ke dalam kode-kode program komputer.

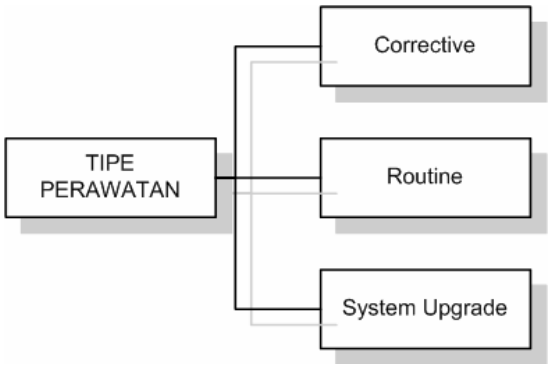
### 2.3.4 Pengujian

Pengujian sistem melibatkan semua pengguna yang telah direncanakan pada tahap sebelumnya. Pengujian tingkat penerimaan terhadap perangkat lunak akan berakhir ketika dirasa semua pengguna menyatakan bisa menerima perangkat lunak tersebut berdasarkan kriteria-kriteria yang telah ditetapkan.

### 2.3.5 Perawatan dan Konfigurasi

Ketika sebuah perangkat lunak telah dianggap layak untuk dijalankan, maka tahapan baru menjadi muncul yaitu perawatan perangkat

lunak. Ada beberapa tipe perawatan yang biasa dikenal dalam dunia perangkat lunak seperti terlihat pada diagram pada Gambar 2.15




**Gambar 2.15. Tipe-tipe Perawatan**

a. Tipe perawatan *corrective*, dilakukan jika terjadi kesalahan atau biasa dikenal sebagai *bugs*. Perawatan bisa dilakukan dengan memperbaiki kode program, menambah bagian yang dirasa perlu atau mungkin menghilangkan bagian-bagian tertentu.
b. Tipe perawatan *routine*, biasa juga disebut *preventive maintenance* dilakukan secara rutin untuk melihat kinerja perangkat lunak ada atau tidak ada kesalahan.
c. Tipe perawatan *sistem upgrade*, dilakukan jika ada perubahan dari komponen-komponen yang terlibat dalam perangkat lunak tersebut. Sebagai contoh perubahan *platform* sistem operasi dari versi lama ke versi baru menyebabkan perangkat lunak harus di-*upgrade*.

## 2.4 Jaringan Komputer

Secara bahasa jaringan komputer berarti sebuah kumpulan dari beberapa komputer yang terhubungan menjadi satu kesatuan yang dapat berkomunikasi antara satu dengan yang lain yang berada dalam suatu sistem yang sama yang dapat berbagi *resource* atau sumber daya secara bersama.

Dalam jaringan komputer ada tiga komponen utama yang harus dipahami, yaitu :

1. *Host* atau *node*, yaitu sistem komputer yang berfungsi sebagai sumber atau penerima dari data yang dikirimkan, *node* ini dapat berupa:

a. *Server* : komputer tempat penyimpanan data dan program-program aplikasi yang digunakan dalam jaringan,
b. *Client* : komputer yang dapat mengakses sumber daya (berupa data dan program aplikasi) yang ada pada *server*,
c. *Shared pheriperal* : peralatan-peralatan yang terhubung dan digunakan dalam jaringan (misalnya, *printer*, *scanner*, *harddisk*, modem, dan lain-lain).

2. *Link*, adalah media komunikasi yang menghubungkan antara *node* yang satu dengan *node* lainnya. Media ini dapat berupa saluran transmisi kabel dan tanpa kabel.
3. *Software* (Perangkat Lunak), yaitu program yang mengatur dan mengelola jaringan secara keseluruhan. Termasuk di dalamnya sistem operasi jaringan yang berfungsi sebagai pengatur komunikasi data dan *peripheral* dalam jaringan.

Beberapa komponen dasar yang biasanya membentuk suatu LAN adalah sebagai berikut:

1. *Workstation*

   Dalam jaringan, *workstation* sebenarnya adalah *node* atau *host* yang berupa suatu sistem komputer. *User* berhubungan dengan jaringan melalui *workstation* dan juga saling berkomunikasi seperti saling bertukar data. *User* juga dapat mengakses program aplikasi pada *workstation* yang dapat bekerja sendiri di *workstation (stand-alone)* itu sendiri ataupun menggunakan jaringan untuk saling berbagi informasi dengan *workstation* atau *user* lain.

   *Workstation* dapat berfungsi sebagai :

   a. *Server*

      Sesuai dengan namanya, ini adalah perangkat keras yang berfungsi untuk melayani jaringan dan *client* yang terhubung pada jaringan tersebut. *Server* dapat berupa sistem komputer yang khusus dibuat untuk keperluan tertentu, seperti untuk penggunaan printer secara bersama, untuk hubungan eksternal LAN ke jaringan lain *(communication server)*, dan *file server* yakni *disk* yang digunakan

secara bersama oleh beberapa *client*. *Server* ini tidak dapat digunakan sebagai *client*, karena baik secara *hardware* maupun *software*, ia hanya berfungsi untuk mengelola jaringan.

Ada pula *server* yang berupa *workstation* dengan *disk drive* yang cukup besar kapasitasnya, sehingga *server* tersebut dapat juga digunakan sebagai *workstation* oleh *user*.

b. *Client*

Sebuah *workstation* umumnya berfungsi sebagai *client* dari suatu *server*, karena memang *workstation* akan menggunakan fasilitas yang diberikan oleh suatu *server*. Jadi, *server* melayani, sedangkan *client* dilayani.

*2. Link*

*Link* atau hubungan dalam jaringan lokal dikenal sebagai media transmisi berupa kabel maupun tanpa kabel, yang secara fisik menghubungkan *server* dan *client*.

3. *Transceiver*

*Transceiver (transmitter-receiver)* merupakan perangkat keras yang menghubungkan *workstation* atau sistem komputer dengan media transmisi.

*4.* Kartu Jaringan *(Network Interface Card* atau NIC*)*

Kartu jaringan ini adalah kartu yang dipasang pada PC yang mengendalikan pertukaran data antar *workstation* yang ada dalam jaringan lokal. Setiap *workstation* harus dilengkapi dengan NIC yang secara fisik terhubung langsung dengan *bus internal* dari PC.

5. Perangkat lunak jaringan

Perangkat lunak jaringan mencakup:

a. Sistem operasi LAN
b. Perangkat lunak aplikasi
c. Perangkat lunak pemrograman
d. Program utiliti

Perangkat lunak ini sangat penting dan mutlak untuk memungkinkan komunikasi antara sistem komputer yang satu dengan

yang lainnya. Tanpa perangkat lunak ini, jaringan tidak akan berfungsi. Sistem komputer dengan LAN dapat menjalankan semua perangkat lunak aplikasi yang dapat berjalan pada *stand-alone PC*.

### 2.4.1 Tipe-tipe Jaringan Komputer

Ada beberapa tipe jaringan komputer yang umumnya digunakan. Berikut ini beberapa klasifikasi tipe jaringan komputer yang ada :

#### 2.4.1.1 Berdasarkan Letak Geografis

a. LAN (*Local Area Network*), jaringan ini berada pada satu bangunan atau lokasi yang sama, dengan kecepatan transmisi data yang tinggi (mulai dari 10 Mbps ke atas), dan menggunakan peralatan tambahan seperti *repeater*, *hub*, dan sebagainya.
b. MAN (*Metropolitan Area Network*), jaringan ini merupakan gabungan beberapa LAN yang terletak pada satu kota (jangkauan 50-75 mil) yang dihubungkan dengan kabel khusus atau melalui saluran telepon, dengan kecepatan transmisi antara 56 Kbps sampai 1 Mbps, dan menggunakan peralatan seperti *router*, telepon, ATM *switch*, dan antena parabola.
c. WAN (*Wide Area Network*), jaringan ini merupakan gabungan dari komputer LAN atau MAN yang ada di seluruh permukaan bumi ini yang dihubungkan dengan saluran telepon, gelombang elektromagnetik, atau satelit.

#### 2.4.1.2 Berdasarkan Arsitektur Jaringan

Berdasarkan arsitekturnya jaringan komputer terbagi menjadi tiga, yaitu :

a. Jaringan *peer to peer*
b. Jaringan berbasis *server*
c. Jaringan *hybrid*.

#### 2.4.1.3 Berdasarkan Teknologi Transmisi

a. Jaringan *switch*, merupakan jaringan yang penyampaian informasi dari pengirim ke penerima melalui mesin-mesin perantara atau saluran telepon,
b. Jaringan *broadcast*, merupakan jaringan yang penyampaian informasi dari pengirim ke penerima dilakukan secara *broadcast* (disiarkan ke segala arah) baik melalui saluran kabel maupun saluran tanpa kabel.

### 2.4.2 Arsitektur Jaringan Komputer

Arsitektur jaringan komputer merupakan tata cara penggunaan perangkat keras dan perangkat lunak dalam jaringan agar satu komputer dengan komputer lainnya dapat melakukan komunikasi dan pertukaran data, ada tiga bentuk arsitektur yang umum digunakan dalam jaringan komputer, yaitu sebagai berikut.

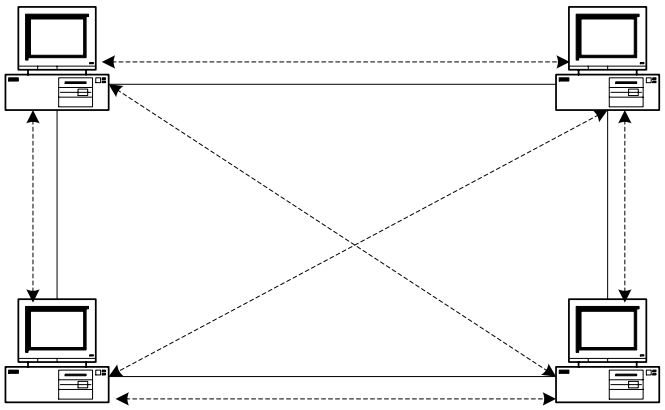
#### 2.4.2.1 Jaringan *Peer to Peer* atau P2P

Pada jaringan *peer to peer* semua komputer memiliki posisi setara/sejajar, dalam hierarki yang sama. Setiap komputer dapat menjadi *client* terhadap komputer *peer* lainnya, setiap komputer dapat pula berbagi sumber daya dengan komputer yang berada dalam jaringan *peer-to-peer* ini. Sumber daya diletakkan secara desentralisasi pada setiap anggota jaringan, dan tidak memerlukan *administrator* jaringan.

Aliran informasi bisa mengalir diantara dua komputer secara langsung, dimana pun. Namun, jaringan ini tidak sepenuhnya bebas tanpa kontrol, karena masih bisa digunakan *password* untuk memproteksi *file* dan *folder*, dapat juga diatur agar orang-orang tertentu tidak bisa menggunakan periferal tertentu.

Karena kemudahan pemasangan, pemeliharaan, serta biaya, jaringan ini lebih populer untuk jaringan dengan jumlah komputer yang sedikit (sekitar 2 sampai 20 komputer).

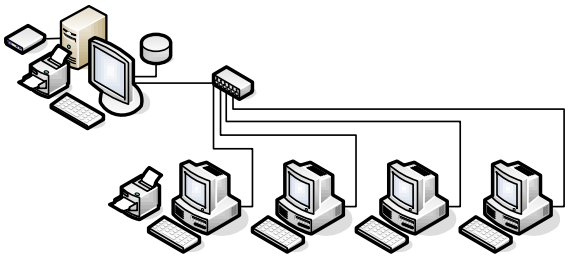
Sifat jaringan *peer to peer* digunakan untuk hubungan antara setiap komputer yang terhubung dalam jaringan komputer yang ada, sehingga komunikasi data terjadi antar komputer dengan hierarki yang sama karena setiap komputer dapat berfungsi sebagai *server* maupun *client*.



**Gambar 2.16. Jaringan *Peer-to-Peer***

#### 2.4.2.2 Jaringan *Client Server*

Arsitektur jaringan *client server* merupakan pengembangan dari arsitektur *file server*. Arsitektur ini adalah model konektivitas pada jaringan yang mengenal adanya *server* dan *client*, dimana masing-masing memiliki fungsi yang berbeda satu sama lain. *Server* dapat berbagi pakai data, aplikasi dan *peripheral* seperti *harddisk, printer, modem* dan lain-lain. Oleh karena itu, tidak jarang juga tercipta sebutan *print server*, *communication server* dan lain sebagainnya. Prinsip kerjanya sangat sederhana, dimana *server* akan menunggu permintaan dari *client*, memproses dan memberikan hasilnya kepada *client*. Sedangkan *client* akan mengirimkan permintaan ke *server*, menunggu proses dan melihat visualisasi hasil prosesnya.

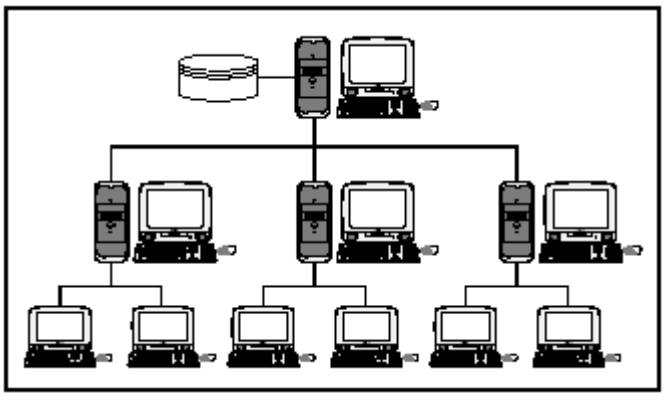


**Gambar 2.17. Konektivitas *Client Server***

Sistem *client server* ini menggunakan protocol TCP/IP (*Transmission Control Protocol/Internet Protocol*). *Unix* dan *Windows NT* merupakan contoh yang baik dari sistem operasi jaringan *client server*.

#### 2.4.2.3 Sistem *Client Server*

Sistem *client* dan *server* terdiri atas dua komponen utama, yaitu *client* dan *server*. *Client* berisi aplikasi basis data dan *server* berisi DBMS dan basis data. Setiap aktifitas yang dikehendaki para pemakai akan lebih dahulu ditangani oleh *client*. *Client* menangani proses yang menjadi tanggung jawabnya. Jika ada proses yang harus melibatkan data yang tersimpan pada basis data yang terletak di *server*, barulah *client* mengadakan hubungan dengan *server*. Pada bentuk sistem *client server* untuk memenuhi kebutuhan, *client* akan mengirimkan pesan atau perintah *query* pengambilan data, selanjutnya *server* yang menerima pesan tersebut akan menjalankan *query t*ersebut dan hasilnya akan dikirimkan kembali ke *client*. Dengan begitu, *transfer* datanya jauh lebih efisien. Untuk lebih jelasnya, dapat dilihat pada gambar sistem *client server* berikut ini :



**Gambar 2.18. Sistem *Client-Server* Kompleks**

#### 2.4.2.4 Komponen Dasar *Client Server*

1. *Client*

   *Client* merupakan terminal yang digunakan oleh pengguna untuk meminta layanan tertentu yang dibutuhkan. Terminal *client* dapat berupa PC, ponsel pintar, komunikator, dan peralatan lain yang membutuhkan informasi.

2. *Middleware*

   *Middleware* merupakan komponen perantara yang memungkinkan *client* dan *server* untuk saling terhubung dan berkomunikasi satu sama lain. *Middleware* ini dapat berupa TP (*Transaction Monitor*), RPC (*Remote Procedure Call*), atau ORB (*Object Request Broker*)

3. *Server*

   *Server* merupakan komputer khusus yang bertugas melayani aplikasi-apalikasi jaringan atau pihak yang menyediakan layanan. *Server* ini akan dapat berupa basis data SQL, monitor TP, *server groupware*, *server* objek, atau *web*. Secara umum, *server* berperan menerima pesan permintaan layanan dari *client*, memproses permintaan tersebut dan mengirimkan hasil permintaan kepada *client*.

#### 2.4.2.5 Karakteristik *Server* dan *Client*

1. Karakteristik *Server*
    a. Pasif
    b. Menunggu *request*
    c. Menerima *request*, memproses mereka dan mengirimkan balasan berupa *service*
2. Karakteristik *Client*
    a. Aktif
    b. Mengirim *request*
    c. Menunggu dan menerima balasan dari *server*

#### 2.4.2.6 Ciri-ciri *Client Server*

Beberapa ciri dari arsitektur sistem terdistribusi *client server* diantaranya adalah :

1. Berbasis layanan.
   *Server* memberikan sejumlah layanan yang dibutuhkan dan diminta oleh *client*, antara lain berbagi pakai berkas, dan peralatan pendukung.
2. Sumber daya yang digunakan bersama.
   *Server* mengelola sejumlah sumber daya yang dimiliki agar dapat diakses dan digunakan secara bersama-sama oleh terminal-terminal *client* yang terhubung pada *server*.
3. Hubungan dan interaksi *client server.*
   Hubungan yang terjadi antara *server* dan *client* adalah *one-to many*, yang berarti bahwa satu *server* melayani banyak *client*. *Client* selalu memulai transaksi dengan meminta layanan sedangkan *server* menanti permintaan layanan secara pasif.
4. *Client* tidak perlu mengetahui lokasi fisik *server.*
   *Server* dapat terletak di berbagai tempat yang belum tentu diketahui oleh *client*, walaupun demikian *client* tetap dapat mengakses *server* untuk mendapatkan layanan sesuai kebutuhannya.
5. Interoperabilitas perangkat lunak dan perangkat keras.

Perangkat lunak dan keras yang digunakan oleh masing-masing *client* tidak harus sama dengan yang digunakan pada *server*, namun masih dapat saling terkoneksi antara satu dan yang lain.

6. Pertukaran berbasis pesan.

   Mekanisme dari *client server* berdasar pada pertukuran pesan, pesan yang dipertukarkan adalah permintaan layanan dan umpan balik dari permintaan layanan tersebut.

7. Enkapsulasi layanan.

   *Client* tidak perlu mengetahui sistem operasi pengelolaan permintaan yang terjadi dalam *server* sehingga *client* tidak dapat mengontrol sistem operasi pengelolaan permintaan.

8. Skalabilitas.

   Skalabilitas adalah kemampuan untuk diperbesar atau diperkecil. Ukuran sistem *client server* dapat diubah secara horizontal maupun vertikal. Perubahan vertikal berarti berpindah ke *server* lebih besar atau lebih cepat atau mendistribusikan tugas melayani *client* ke beberapa *server*. Pengubahan horizontal berarti menambah atau mengurangi jumlah *client*.

9. Konsistensi data.

   Data hanya dikelola pada *server* pusat sehingga konsistensi data lebih terjamin dan biaya pemeliharaan pun menjadi lebih murah.

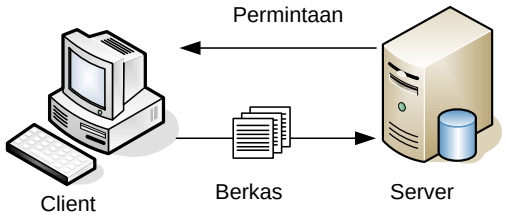
#### 2.4.2.7 Tipe Jaringan *Client Server*

Berdasarkan tipe layanan yang diberikan *server* kepada *client*, jaringan *client server* dapat dibagi menjadi ke dalam banyak tipe, tipe-tipe tersebut antara lain :

a. *Server* berkas

Sistem jaringan berkas adalah sistem jaringan yang dimana layanan yang diberikan *server* berupa berkas, baik berkas aplikasi seperti aplikasi pengolahan kata, pengolahan angka, pengolahan data, pengolahan gambar dan lain sebagainya, maupun berkas yang

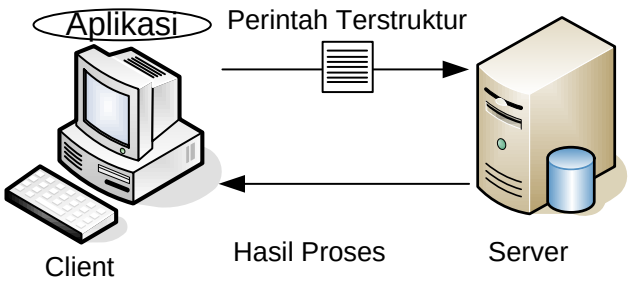
dihasilkan oleh aplikasi tersebut, seperti dokumen pengolahan kata, tabel-tabel pengolahan angka, berkas presentasi dan lain sebagainya.




**Gambar 2.19. *Server* Berkas**

b. *Server* basis data

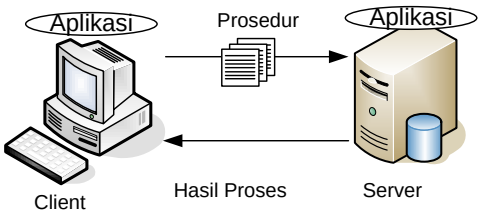
Sistem jaringan *server* basis data adalah merupakan sistem jaringan dimana layanan yang diberikan oleh *server* berupa pengolahan dan penyajian data berdasarkan perintah terstruktur (*query*) yang diberikan *client*. Pada jaringan ini, *server* menyimpan berbagai macam data yang dapat diakses oleh pengguna melalui terminal-terminal *client*.




**Gambar 2.20. *Server* Basis Data**

c. *Server* Transaksi

Sistem jaringan *server* transaksi adalah sistem jaringan dimana layanan yang diberikan *server* berupa hasil Sistem Operasi dari sekelompok perintah terstruktur yang diberikan *client*. Jaringan ini pada dasarnya hampir sama dengan sistem jaringan basis data sebelumnya. Perbedaan terletak pada *server* transaksi yang memproses sekelompok perintah terstruktur dari *client*, dan sekelompok perintah terstruktur ini disebut prosedur.




**Gambar 2.21. *Server* Transaksi**

d. *Groupware Server*

Sistem jaringan *groupware* adalah sistem jaringan dimana layanan yang diberikan *server* berupa fasilitas pemakaian bernama informasi semi terstruktur diantara pengguna jaringan. Pada jaringan ini, *server* menyimpan, mengelola dan menyebarkan informasi antar pengguna dalam jaringan, misalnya teks, gambar, surat dan ruang diskusi.

e. *Server* Objek

Sistem jaringan *server* objek adalah sistem jaringan dimana layanan yang diberikan *server* berbentuk objek. Dalam jaringan ini, *client* dan *server* berkomunikasi melalui objek-objek yang miliki *client* dan *server*.

f. *Web Server*

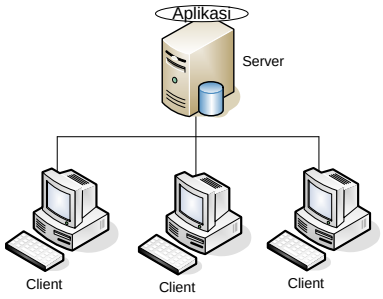
Sistem jaringan *web server* adalah sistem jaringan dimana layanan yang diberikan *server* berupa pengelolaan dan pemakaian bersama dokumen-dokumen yang saling terhubung. Jaringan ini merupakan jaringan yang memungkinkan tiap dokumen dalam jaringan memiliki hubungan ke dokumen lain sehingga dokumen-dokumen dalam jaringan terhubung satu dengan yang lain, semacam jaringan laba-laba.

### 2.4.2.8 Arsitektur *Client Server*

1. *Two Tier*

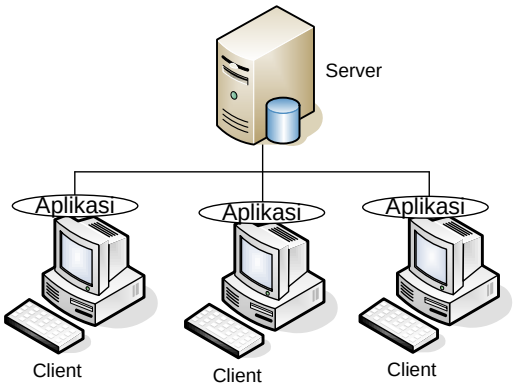
Arsitektur *two tier* merupakan arsitektur yang disebut *client server* dimana terdapat komputer sebagai *client* dan *server* yang berinteraksi melalui protokol dan media komunikasi tertentu Model arsitektur *Two Tier* dikelompokan menjadi dua macam yaitu :

a. *Thin Client – Thick Server*




**Gambar 2.22. *Thin Client – Thick Server***

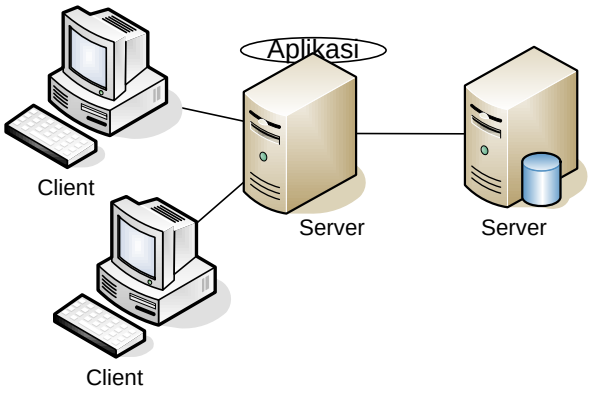
b. *Thick Client – Thin Server*




**Gambar 2.23. Thick Client – Thin Server**

2. *Three Tier*

Arsitektur *client server* ini memisahkan antara data (*Data Management Tier*), aplikasi (*Middle Tier*) dan penyajian (*Presentation Layer*) sebagaimana ditampilkan pada gambar di bawah ini :




**Gambar 2.24. Arsitektur *Three Tier***

a. *Data Management Tier*

Merupakan komputer *server* yang dikhususkan untuk menangani pengolahan basis data

b. *Middle Tier*

Merupakan komputer *server* yang dikhususkan untuk menangani aplikasi-aplikasi dimana prosedur-prosedur dan perhitungan-perhitungan yang kompleks dilakukan di komputer.

c. *Presentation Layer*

Merupakan komputer *client* yang menjadi *interface* bagi pengguna untuk memasukkan data, mengajukan permintaan layanan kepada *server* dan melihat hasilnya.

3. *n-Tier*

Istilah *n-tier* menunjukan lapisan yang ada dalam sebuah aplikasi. Sebuah aplikasi terdiri dari beberapa komponen utama, yaitu lapisan *Presentation* (*Presentation Layer*), lapisan *application* (*Application Layer*) dan logika bisnis (*bussiness logic layer*) dan lapisan data (*Data layer*).

Lapisan *Presentation* menghubungkan antarmuka dengan pengguna aplikasi, dapat berupa model grafis atau berupa teks. Pengguna dapat berinteraksi dengan aplikasi tersebut menggunakan lapisan *presentation* ini. Lapisan *Application* berisi inti dari aplikasi dan lapisan data yang digunakan oleh aplikasi tersebut. Lapisan data

dapat berbentuk satu atau lebih *server* basis data yang lokasinya tersebar di beberapa tempat.

## 2.5 VPN (*Virtual Private Network*)

VPN adalah singkatan *Virtual Private Network*, yaitu sebuah koneksi *private* melalui jaringan publik atau *internet*, *virtual network* berarti jaringan yang terjadi hanya bersifat *virtual*. *Private* yaitu jaringan yang terbentuk bersifat *private* dimana tidak semua orang bisa mengaksesnya. Data yang dikirimkan *terenkripsi* sehingga tetap rahasia meskipun melalui jaringan publik. Jika menggunakan VPN kita seolah-olah membuat jaringan didalam jaringan atau biasa disebut *tunnel*. VPN menggunakan salah satu dari tiga teknologi *tunneling* yang ada yaitu: *PPTP*, *L2TP* dan standar terbaru, IPSec (*Internet Protocol Security*). VPN merupakan perpaduan antara teknologi *tunneling* dan enkripsi.

Cara kerja VPN dengan protokol PPTP adalah sebagai berikut:

- VPN membutuhkan sebuah *server* yang berfungsi sebagai penghubung antar PC, *Server VPN* ini bisa berupa komputer dengan aplikasi VPN *Server* atau sebuah *Router*.
- Untuk memulai sebuah koneksi, komputer dengan aplikasi VPN *client* mengontak *Server* VPN, VPN *Server* kemudian *memverifikasi username* dan *password* dan apabila berhasil maka VPN *Server* memberikan *IP Address* baru pada komputer *client* dan selanjutnya sebuah koneksi / *tunnel* akan terbentuk.
- Selanjutnya komputer *client* bisa digunakan untuk mengakses berbagai *resource* (komputer atau LAN) yang berada di belakang VPN *Server* misalnya melakukan *transfer* data, *browsing* dengan *gateway* yang diberikan dari VPN *Server*, melakukan *remote desktop* dan lain sebagainya.

### 2.5.1 Protokol-protokol dalam VPN

#### 2.5.1.1 IPSec (IP *Security*)

*IPSec* (singkatan dari IP *Security*) adalah sebuah protokol yang digunakan untuk mengamankan transmisi datagram dalam sebuah *internetwork* berbasis TCP/IP. *IPSec* mendefiniskan beberapa standar untuk melakukan enkripsi data dan juga integritas data pada lapisan kedua dalam DARPA (*Defense Advanced Research Projects Agency*) *Reference Model internetwork layer*. *IPSec* melakukan enkripsi terhadap data pada lapisan yang sama dengan protokol IP dan menggunakan teknik *tunneling* untuk mengirimkan informasi melalui jaringan *internet* atau dalam jaringan *Intranet* secara aman. *IPSec* didefinisikan oleh badan IETF (*Internet Engineering Task Force*) dan diimplementasikan di dalam banyak sistem operasi. Windows 2000 adalah sistem operasi pertama dari *Microsoft* yang mendukung *IPSec*.

*IPSec* diimplementasikan pada lapisan *transport* dalam OSI *Reference Model* untuk melindungi protokol IP dan protokol-protokol yang lebih tinggi dengan menggunakan beberapa kebijakan keamanan yang dapat dikonfigurasikan untuk memenuhi kebutuhan keamanan pengguna, atau jaringan. *IPSec* umumnya diletakkan sebagai sebuah lapsian tambahan di dalam *stack* protokol TCP/IP dan diatur oleh setiap kebijakan keamanan yang diinstalasikan dalam setiap mesin komputer dan dengan sebuah skema enkripsi yang dapat dinegosiasikan antara pengirim dan penerima. Kebijakan-kebijakan keamanan tersebut berisi kumpulan filter yang diasosiasikan dengan kelakuan tertentu. Ketika sebuah alamat IP, nomor port TCP dan UDP atau protokol dari sebuah paket datagram IP cocok dengan filter tertentu, maka kelakukan yang dikaitkan dengannya akan diaplikasikan terhadap paket IP tersebut.

Dalam sistem operasi *Windows 2000*, *Windows XP*, dan *Windows Server 2003*, kebijakan keamanan tersebut dibuat dan ditetapkan pada *level domain Active Directory* atau pada host individual dengan menggunakan *snap-in IPSec Management* dalam *Microsoft Management Console* (MMC). Kebijakan *IPSec* tersebut, berisi beberapa peraturan

yang menentukan kebutuhan keamanan untuk beberapa bentuk komunikasi. Peraturan-peraturan tersebut digunakan untuk memulai dan mengontrol komunikasi yang aman berdasarkan sifat lalu lintas IP, sumber lalu lintas tersebut dan tujuannya. Peraturan-peraturan tersebut dapat menentukan metode-metode autentikasi dan negosiasi, atribut proses *tunneling*, dan jenis koneksi.

Untuk membuat sebuah sesi komunikasi yang aman antara dua komputer dengan menggunakan *IPSec*, maka dibutuhkan sebuah *framework* protokol yang disebut dengan ISAKMP/Oakley. *Framework* tersebut mencakup beberapa algoritma kriptografi yang telah ditentukan sebelumnya, dan juga dapat diperluas dengan menambahkan beberapa sistem kriptografi tambahan yang dibuat oleh pihak ketiga. Selama proses negosiasi dilakukan, persetujuan akan tercapai dengan metode autentikasi dan kemanan yang akan digunakan, dan protokol pun akan membuat sebuah kunci yang dapat digunakan bersama (*shared key*) yang nantinya digunakan sebagi kunci enkripsi data. *IPSec* mendukung dua buah sesi komunikasi keamanan, yakni sebagai berikut :

a. Protokol *Authentication Header* (AH): menawarkan autentikasi pengguna dan perlindungan dari beberapa serangan (umumnya serangan *man in the middle*), dan juga menyediakan fungsi autentikasi terhadap data serta integritas terhadap data. Protokol ini mengizinkan penerima untuk merasa yakin bahwa identitas si pengirim adalah benar adanya, dan data pun tidak dimodifikasi selama transmisi. Namun demikian, protokol AH tidak menawarkan fungsi enkripsi terhadap data yang ditransmisikannya. Informasi AH dimasukkan ke dalam header paket IP yang dikirimkan dan dapat digunakan secara sendirian atau bersamaan dengan protokol *Encapsulating Security Payload*.
b. Protokol *Encapsulating Security Payload* (ESP): Protokol ini melakukan enkapsulasi serta enkripsi terhadap data pengguna untuk meningkatkan kerahasiaan data. ESP juga dapat memiliki skema autentikasi dan perlindungan dari beberapa serangan dan dapat digunakan secara sendirian atau bersamaan dengan *Authentication*

*Header*. Sama seperti halnya AH, informasi mengenai ESP juga dimasukkan ke dalam header paket IP yang dikirimkan.

#### 2.5.1.2 Potokol L2TP *(Layer 2 Tunneling Protocol)*

Protokol L2TP sering juga disebut sebagai protokol *dial-up virtual*, karena L2TP memperluas suatu *session* PPP *(Point-to-Point Protocol) dial-up* melalui jaringan publik *internet*, atau sering juga digambarkan seperti koneksi *virtual* PPP.

Protokol *Virtual* PPP menggambarkan suatu mekanisme enkapsulasi untuk mengangkut paket-paket *multiprotocol* melalui hubungan *point-to-point* pada *layer-2*., PPP terdiri dari 4 komponen utama yaitu :

- EIA/TIA 232 : *Layer* fisik untuk komunikasi *serial*.
- HDLC : Metode untuk enkapsulasi datagram melalui *link serial*.
- LCP : Metode untuk pembentukan, konfigurasi, mempertahankan, dan memutuskan jaringan PPP.
- NCP : Metode pembentukan dan mengkonfigurasi protokol *network layer* yang berbeda (*layer-3* : misalnya IP, IPX, atau *apple talk*).

Dan dua metode autentifikasi pada PPP :

- PAP *(Password Authentication Protocol)*
- CHAP *(Challenge Authenticatoin Protocol)*

Perangkat yang terkait dalam penggunaan protokol L2TP :

- RC (*Remote Client*), suatu *end system* atau *router* pada jaringan *remote access* (mis. : *dial-up client*).
- LAC (L2TP *Access Concentrator*). Sistem yang berada disalah satu ujung tunnel L2TP dan merupakan *peer* ke LNS. Berada pada sisi *remote client* atau ISP. Sebagai pemrakarsa *incoming call* dan penerima *outgoing call.*
- LNS (L2TP *Network Server*). Sistem yang berada disalah satu ujung *tunnel* L2TP dan merupakan *peer* ke LAC. Berada pada sisi

jaringan korporat. Sebagai pemrakarsa *outgoing call* dan penerima *incoming call.*

- NAS (*Network Access Server*). NAS dapat berlaku seperti LAC atau LNS atau kedua-duanya.

#### *2.5.1.3* **PPTP *(Point-to-Point Tunneling Protocol)***

*Point-to-Point Tunneling Protocol* (PPTP) adalah suatu protokol jaringan yang memungkinkan pengiriman data secara aman dari *remote client* kepada *server* perusahaan swasta dengan membuat suatu *virtual private network* (VPN) melalui jaringan data berbasis TCP/IP.

Teknologi jaringan PPTP merupakan perluasan dari *remote access point-to-point protocol*. PPTP adalah suatu protokol jaringan yang membungkus paket PPP ke dalam IP *datagram* untuk transmisi yang dilakukan melalui *internet* atau jaringan publik berbasis TCP/IP. PPTP dapat juga digunakan pada jaringan LAN-to-LAN.

Protokol PPTP termasuk dalam sistem operasi *server Windows NT versi 4.0* dan *Windows NT Workstastion versi 4.0.* Komputer menjalankan sistem operasi ini dapat dengan menggunakan protokol PPTP untuk koneksi ke jaringan *private* sebagai *remote access client* secara aman melalui jaringan publik data seperti *internet*. Dengan kata lain, PPTP digunakan sesuai dengan permintaan, misalnya dapat digunakan dengan VPN melalui *internet* atau jaringan publik data berbasis TCP/IP lainnya. PPTP juga dapat digunakan pada LAN untuk membuat VPN dalam LAN.

Fitur penting dalam penggunaan PPTP adalah PPTP mendukung VPN dengan menggunakan PSTN (*Public-Switched Telephone Networks*). PPTP menyederhanakan dan mengurangi biaya dalam penggunaan pada perusahaan besar dan sebagai solusi untuk *remote* atau *mobile users* karena PPTP memberikan komunikasi yang aman dan terenkripsi melalui *line public telephone* dan *internet.*

Secara umum, terdapat tiga komponen di dalam komputer yang menggunakan PPTP yaitu

1. PPTP *client*
2. NAS (*Network Access Server*)
3. PPTP *server*

Dimana PPTP juga adalah sebuah protokol yang mengizinkan hubungan PPP (*Point-to Point Protocol*) melewati jaringan IP, dengan membuat VPN (*Virtual Private Network*).

### 2.5.2 Fungsi Utama VPN

Teknologi VPN menyediakan tiga fungsi utama untuk penggunanya. Fungsi utama tersebut adalah sebagai berikut:

1. *Confidentiality*

   Teknologi VPN memiliki sistem kerja mengenkripsi semua data yang lewat melaluinya. Dengan adanya teknologi enkripsi ini, maka kerahasiaan Anda menjadi lebih terjaga. Biarpun ada pihak yang dapat menyadap data Anda yang lalu-lalang, namun belum tentu mereka bisa membacanya dengan mudah karena memang sudah diacak. Dengan menerapkan sistem enkripsi ini, tidak ada satupun orang yang dapat mengakses dan membaca isi jaringan data Anda dengan mudah.

2. *Data Integrity*

   Ketika melewati jaringan *internet*, data Anda sebenarnya sudah berjalan sangat jauh melintasi berbagai negara. Di tengah perjalanannya, apapun bisa terjadi terhadap isinya. Baik itu hilang, rusak, bahkan dimanipulasi isinya oleh orang-orang iseng. VPN memiliki teknologi yang dapat menjaga keutuhan data yang Anda kirim agar sampai ke tujuannya tanpa cacat, hilang, rusak, ataupun dimanipulasi oleh orang lain.

3. *Origin Authentication*

   Teknologi VPN memiliki kemampuan untuk melakukan autentikasi terhadap sumber-sumber pengirim data yang akan diterimanya. VPN akan melakukan pemeriksaan terhadap semua data yang masuk dan mengambil informasi *source* datanya. Kemudian

alamat *source* data ini akan disetujui jika proses autentikasinya berhasil. Dengan demikian, VPN menjamin semua data yang dikirim dan diterima oleh Anda berasal dari sumber yang semestinya. Tidak ada data yang dipalsukan atau dikirimkan oleh pihak-pihak lain.

## 2.6 Bahasa Pemrograman *Web*

Bahasa pemrograman *web* terdiri dari dua suku kata yaitu pemrograman dan *web*. Pemrograman yang dari segi bahasa dapat diartikan proses, cara, pembuatan program. Sedangkan kata *web* sendiri memiliki arti jaringan komputer yang terdiri dari kumpulan situs yang menawarkan teks, grafik, suara dan sumber daya animasi melalui protokol *transfer hypertext*.

Banyak keuntungan yang diberikan oleh Aplikasi berbasis *web* dari pada aplikasi berbasis *desktop*, sehingga aplikasi berbasis *web* telah diadopsi oleh perusahaan sebagai bagian dari strategi teknologi informasinya, karena beberapa alasan diantaranya adalah akses informasi mudah, *setup server* lebih mudah, bebas *platform*, sehingga informasi dapat disajikan oleh *browser web* pada sistem operasi mana saja karena adanya standar dokumen sehingga berbagai tipe data dapat disajikan.

### 2.6.1 *Hypertext Prepocessor* Atau PHP

#### 2.6.1.1 Sejarah PHP

PHP adalah bahasa skrip yang dapat ditanamkan atau disisipkan ke dalam skrip HTML, pada awalnya PHP merupakan singkatan dari *Personal Home Page*. PHP adalah teknologi yang diperkenalkan pada tahun 1994 oleh Rasmus Leerdorf. Beberapa versi awal yang tidak dipublikasikannya digunakan pada situs pribadinya untuk mencatat siapa saja yang mengakses daftar riwayat hidup *online*-nya. Versi pertama mulai digunakan oleh pihak lain pada awal tahun 1995 dan mulai dikenal dengan sebutan *personal home page tools*. Terkandung di dalamnya sebuah *parser engine* (mesin pengurai) yang sangat disederhanakan, yang hanya mampu mengelolah makro khusus dan beberapa utilitas yang sering digunakan pada pembuatan *Home Page*, seperti buku tamu,

pencacah, dan hal semacamnya. *Parser* tersebut ditulis ulang pada pertengahan 1995 dan dinamakan PHP/FI *Version 2*. FI (*Form Interpreter*) sendiri berasal dari kode lain yang ditulis juga oleh Rasmus, yang menterjemahkan HTML dari data, ia menggabungkan skrip *personal home page tools* dengan *form interpreter* dan menambahkan dukungan terhadap *server database* yang menggunakan format mSQL, sehingga lahirlah PHP/F1. PHP/F1 tumbuh dengan pesat, dan orang-orang mulai menyiapkan kode-kode programnya supaya bisa didukung oleh PHP.

Sulit memberikan data statistik akurat, namun diperkirakan pada akhir 1996 PHP/F1 sudah digunakan pada sedikitnya 15.000 situs *web* di seluruh dunia. Pada pertengahan 1997, angka tersebut berubah menjadi 50.000. pada saat itu juga terdapat perubahan pada pengembangan PHP. PHP berubah dari proyek pribadi Rasmus menjadi sebuah tim pengembangan yang lebih terorganisasi. *Parsernya* ditulis ulang dari bentuk rancangan awal oleh Zeev Suraski dan Andi Gutmans, dan *parser* baru ini yang menjadi dasar dari PHP Version 3, dan banyak diantaranya sudah selesai ditulis ulang secara lengkap.

Pada pertengahan 1998, baik PHP/F1 maupun PHP3 dikemas bersama dengan produk-produk komersial seperti *server web StrongHold* buatan C2 dan *Linux RedHat*, dan menurut survey yang dilakukan oleh NetCraft, kemungkinan PHP digunakan pada lebih dari 150.000 situs *web* di seluruh dunia.

#### 2.6.1.2 Dasar-dasar PHP

Kode atau *script* yang biasa ditangani oleh PHP adalah C. Karena kode PHP dieksekusi di sisi *server* maka PHP disebut sebagai *server-side script*.

Pengertian ini penting karena *browser-browser web* masa kini umumnya juga telah sanggup mengeksekusi *client-side script*, yaitu *script* yang dijalankan di sisi *client*. Pada awalnya C adalah *client-side script* yang akan diteruskan ke *browser web* tanpa diolah olehnya.

Untuk itu PHP mempunyai empat cara untuk membedakannya dari *script* HTML dan memasuki "mode PHP", contoh :

```
<? echo ("ini adalah cara paling sederhana untuk memroses instruksi SGML\n");?>
<?php echo ("bila Anda hendak menyajikan dokumen XML, lakukanlah dengan cara ini \n"); ?>
<script language="php">
echo ("Beberapa editor HTML seperti frontpage tidak menyukai pemrosesan instruksi"); </script>
<% echo ("Dimulai dari PHP 3.0.4 Anda dapat menggunakan tag ala ASP" %>
```

*Block-block script* tersebut ditempatkan di *file* yang sama dengan *file* HTML biasa, namun untuk memudahkan pengenalan umumnya semua *file script* PHP diberi ekstensi *.php3, bukan *.html. *sccript* yang dimaksudkan agar dieksekusi di *browser* (yaitu *client-side script*) biasanya ditandai dengan *tag* <script> dan </script>. Seperti contoh berikut :

```
<html>
<head><title>Percobaan PHP</title></head>
<body>
      <? Echo "Ini adalah script yang dijalankan di server."?>
      <script language=javascript>
      Document.writen("Ini script yang dijalankan di browser!");
      </script>
</body></html>
```

Adapun demikian *script* yang berada diantara perintah <*script*> dan <*/script*> juga dapat dieksekusi di sisi *server*, dengan manambahkan atribut "*runat*=" pada *tag* <*script*>, seperti contoh berikut :

```
<script language=javascript runat=server>
Document.writen("Ini script yang dijalankan di server!");
</script>
```

Ini diperlukan terutama bila kita hendak menulis suatu subrutin atau fungsi didalam *file* PHP, karena semua *script* yang ada diantara *tag*

<? dan ?> akan langsung dieksekusi begitu *file* dibaca, oleh karenanya tidak cocok bila kita hendak menempatkan fungsi atau subrutin yang hanya perlu dibaca bila dipanggil.

Secara gamblang, diagram di bawah ini akan menjelaskan bagaimana perbedaan antara *tag-tag script <? dan ?>* serta *<script>* dan *</script>* :

```
<html>
      <head>........</head>
      <body>
      <? ..............?>
      <script runat=server>
      ..............
      </script>
      <script>
      ..............
      </script>
      </body></html>
```

Pemisahan antar instruksi berlaku sama seperti pada C atau Perl, yaitu menggunakan ‘;’ (titik koma) sebagai pemisah antar *statement*. Tag penutup ‘?>’ juga menyatakan akhir *statement*, sehingga kedua perintah berikut sama artinya :

```
<? Echo “Percobaan.....”;?>
<? Echo “Percobaan.....” ?>
```

### 2.6.1.3 Variabel pada PHP

PHP sebagaimana bahasa pemrograman lainya mendukung berbagai jenis variabel, diantaranya adalah sebagi berikut :

- *Integer*, variabel berjenis *integer* bertujuan untuk menyimpan bilangan bulat.
- *Double*, variabel ini digunakan untuk menyimpaan bilangan bernilai pecahan dan juga bilangan pemangkatan.
- *String*, variabel ini merupakan jenis data karakter yang disimpan sebagai nomer pada memori komputer. Nilai yang disimpan adalah nilai ASCII karakter *string* tersebut.

- *Array*, *array* adalah sebuah set variabel yang mempunyai jenis data sejenis. *Array* mengandung komponen yang dinamakan *element*. *Element-element* sebuah *array* disimpan pada lokasi tertentu pada memori.
- *Object*, jenis variabel *object* adalah berdasarkan gambaran objek pada dunia nyata, yang mempunyai "status" dan "tingkah laku", Sebuah variabel objek menyimpan statusnya dalam bentuk variabel dan tingkah laku nya berupa parameter.

Jenis variabel biasanya tidak ditentukan oleh pemrogram, melainkan ditentukan oleh PHP saat *runtime*, tergantung pada konteks apa variabel tersebut digunakan. Misalkan variabel *coba* berisikan nilai *string*, maka variabel *coba* berjenis *string*, atau *integer* bila berisi nilai *integer*, seperti contoh berikut :

```
$coba="0";  //$coba adalah variabel string (ASCII 48)
$coba++;    //$coba adalah string "1" (ASCII 49)
$coba +=1;  //$coba berubah menjadi integer (2)
$coba = $coba + 1.3; //$coba berubah menjadi double
(3.3)
```

#### 2.6.1.4 Inisialisasi Variabel

Inisialisasi variabel pada PHP dilakukan dengan cara memberinya sebuah nilai. Namun pada jenis variabel *array* dan *object*, digunakan cara yang sedikit berbeda.

- Menginisialisasi Variabel *Array*

  Sebuah *array* bisa diinisialisasi dengan dua cara, yaitu dengan memberikannya sebuah nilai secara berurutan, dan menggunakan *construct array ()*, contoh :

  ```
  $nama[] ="Joko";          //$nama[0]="Joko"
  $nama[] ="Supriyanto";    //$nama[1]="Supriyanto"
  ```

  Seperti pada bahasa pemrograman *C* dan *Perl*, elemen *array* diawali dengan 0 bukan 1.

- Menginisialisasi Variabel Objek

Inisialisasi variabel berjenis objek dilakukan dengan menggunakan statement *new* untuk menyisipkan sebuah objek ke sebuah variabel, contoh :

```
Class kuda{
Function kudalah()
{
 Echo "Menjadi seekor Kuda.";}}
 $keledai = new kuda ;
 $keledai -> kudalah();
```

### 2.6.1.5 Variabel Luar

Ketika sebuah form dikirim menuju ke sebuah *script* PHP, semua variabel yang ada pada *form* tersebut secara otomatis juga bisa digunakan oleh PHP, sebagai contoh adalah form berikut :

```
<form action = "coba.php" method="post">
      Nama: <input type =""text" name="nama"><br>
      <input type ="submit">
</form>
```

Ketika dikirim, PHP akan membuat variabel *$nama*, yang berisi apapun yang disikan melalui *field* nama pada *form*. Ketika mengirim sebuah *form*, juga bisa menggunakan sebuah gambar daripada sekedar tombol "kirim" dengan menggunakan *tag* seperti :

```
<input type=image src="gambar.gif" name = "sub">
```

Ketika pengguna mengklik dimanapun pada gambar, *form* yang bersangkutan akan dikirmkan ke *server* dengan tambahan dua buah variabel, yaitu *sub_x* dan *sub_y*. Variabel-variabel ini menyimpan koordinat tempat pengguna mengklik gambar.

### 2.6.1.6 Variabel *Environment*

PHP secara otomatis membuat variabel *environment* bisa digunakan sebagai variabel normal, contoh :

```
Echo $HOME; /*menunjukan penggunaan variabel environment
$HOME, bila diaktifkan*/
```

#### 2.6.1.7 *Cookie* pada PHP

PHP secara gamblang mendukung *http cookies* seperti ditetapkan oleh *Netscape's Spec*. Pada PHP, *cookie* bisa diatur melalui fungsi *setcookie(). Cookie* adalah bagian dari *header* HTTP, jadi fungi *setcookie()* harus dipanggil sebelum hasil apapun dikirim kembali ke *browser*. Setiap *cookie* yang dikirim ke *server* dari *client* akan diubah secara otomatis menjadi sebuah variabel PHP, sama seperti pengiriman data menggunakan metode *get* dan *post.*

Untuk memberikan beberapa nilai pada sebuah *cookie*, tambahkan tanda "[ ]" pada kode *cookie*, seperti contoh berikut :

```
SetCookie ("cookieku[]", Coba Broo...", time()+3600);
```

#### 2.6.1.8 Fungsi pada PHP

Sebuah fungsi didefinisikan menggunakan perintah :

```
Function coba ($arg_1, $arg_2, ..., $arg_n
{
        Echo "Contoh fungsi.\n";
        Return $nilai;}
```

Fungsi harus didefinisikan terlebih dahulu sebelum mereka digunakan. Pengembalian nilai dari suatu fungsi menggunakan perintah *return*. Semua jenis data bisa digunakan sebagai nilainya, termasuk *list* dan *object*.

### 2.6.2 HTML (*Hypertext Markup Language*)

*Hypertext markup Language* (HTML) merupakan bahasa yang digunakan untuk memformat teks atau dokumen-dokumen pada jaringan komputer yang dikenal sebagai *world wide web* (atau sering disebut *web* saja). Dokumen-dokumen HTML merupakan berkas teks yang mengandung 2 bagian:

1. Isi yaitu segala sesuatu yang ingin ditampilkan dan diperlihatkan dalam dokumen *web*.

2. Tag yang merupakan informasi pemformatan, yang tersembunyi dari pandangan pengguna, yang memberitahu *browser* tentang bagaimana caranya menampilkan isi dokumen ke hadapan pengguna.

HTML pada dasarnya merupakan himpunan bagian dari bahasa yang lebih luas jangkauannya yaitu *Standard Generalized Markup Language* (SGML), yang merupakan sistem pengkodean dan pemformatan untuk dokumen yang ditampilkan di layar komputer maupun di kertas.

Dokumen-dokumen SGML terdiri dari 3 jenis informasi: *data*, *structure*, dan *format*. Data mencakup tidak hanya teks, grafik dan gambar, suara tetapi juga informasi-informasi yang tidak ditampilkan pada sisi luar misalnya basis data katalog pengiriman lewat surat yang memuat informasi rujukan tentang *item* tertentu yang berisi persediaan saat ini yang tidak tercantum dalam catalog tercetak.

Struktur dokumen merujuk pada pengorganisasian data pada dokumen yang bersangkutan, misalnya bab-bab, subbab-subbab, daftar-daftar, gambar-gambar, judul-judul, indeks, dan daftar isi. Halaman-halaman *world wide web* memuat struktur yang sama. Namun selain itu, HTML juga memuat *link*, atau hubungan dengan halaman-halaman *world wide web* yang lainnya.

Format merujuk pada bagaimana dokumen akan dilihat oleh para pengguna. Sebagai contoh, teks mungkin akan dimiringkan (*italic),* ditebalkan (*bold),* diberi garis bawah *(underline),* dan sebagainya.

*World wide web* pertama kali dikembangkan oleh fisikawan dan ilmuwan komputer Inggris yang bernama Timothy Berners-Lee pada salah satu proyek yang diselenggarakan oleh Pusat Riset Nuklir Eropa *(European Organization for Nuclear Research* (CERN)*)* di Geneva, Swiss. Berners-Lee menggabungkan beberapa ide yang ada saat itu ke suatu sistem tunggal yang memudahkan para fisikiawan menggunakan data-data dan mengomunikasikannya lewat *internet*. Yang juga penting, Berners-Lee menambahkan kemamapuan multimedia pada dokumen-dokumen *web*. Berners-Lee mulai bekerja dengan *hypertext* pada awal tahun 1980-an dan prototipe-nya diimplementasikan oleh CERN (*European Organization for*

*Nuclear Research)* pada tahun 1989. Selanjutnya ide-ide Berners-Lee secara cepat menyebar ke universitas-universitas di seluruh dunia (dan akhirnya ke dunia komersial).

Beberapa *tag* dalam dokumen-dokumen *HTML* menentukan bagaimana *teks-teks* diformat. *Tag-tag* yang lain memberitahu komputer tentang bagaimana menanggapi aksi-aksi yang datang dari pengguna. Sebagai contoh, apa yang harus dilakukan komputer saat pengguna mengklikkan mouse-nya pada *icon* tertentu. Kemudian *tag* lain yang penting adalah *link* yang mengandung *Uniform Resource locator* (ULR), yang merujuk pada dokumen lain di *server* yang sama atau komputer lain yang ada di jaringan global *internet*. Selain itu, HTML juga memiliki penandaan untuk membuat formulir-formulir yang memungkinkan pengguna mengisi informasi tertentu dan mengirimkannya secara elektronik ke tempat-tempat lain di jaringan *internet*.

Perangkat lunak yang memungkinkan pengguna menjelajahi *world wide web* dan melihat dokumen-dokumen HTML dinamakan *browser*. *Browser* menafsirkan *tag-tag* HTML pada dokumen dan melakukan pemformatan saat dokumen yang bersangkutan ditampilkan di layar monitor.

Pada tahun 1989, Berners-Lee merumuskan suatu proposal tentang sebuah *system hypertext* yang memiliki 3 komponen sebagai berikut:

1. Antarmuka yang konsisten untuk semua jenis *platform.* Antarmuka ini harus menyediakan akses yang dapat digunakan oleh berbagai jenis komputer.
2. Akses informasi yang bersifat *universal*. Setiap pengguna harus dapat mengakses setiap informasi yang berbeda.
3. Antarmuka yang menyediakan akses keberbagai jenis dokumen dan protokol.

HTML 2.0 didefinisikan oleh IETF (*Internet Engineering Task Force*) dengan himpunan fitur dasar, termasuk kemampuan halaman yang bersifat interaktif. Versi selanjutnya menambahkan fitur-fitur tambahan seperti teks yang berkedip *(blinking text)*, latar belakang yang

dapat diatur, serta tabel-tabel. Versi-versi selanjutnya menambahkan kemampuan-kemampuan baru dan setiap saat *browser* harus dimodifikasi untuk dapat menyesuaikan diri dengan kemampuan-kemampuan baru tersebut. Saat ini standar untuk HTML dispesifikasi oleh W3C (*world Wide Web Consortium).*

### *2.6.3 JavaScript*

*JavaScript* adalah bahasa skrip dan *Netscape* yang fungsinya mirip dengan *VBScript* namun bahasa dasarnya adalah adalah *Java*. *JavaScript* didukung oleh *browser* Netscape Navigator mulai versi 2.0. *JavaScript* lebih mudah digunakan dari *Java*, tetapi tentu saja tidak memiliki kemampuan yang secanggih *Java*. *JavaScript* menggunakan bahasa HTML sebagai antarmuka penggunanya. *Java* dapat membuat antarmuka secara lengkap. Pada sisi *client*, *applet* (sebuah aplikasi dikembangkan untuk dijalankan di *browser*) dipelihara dalam kode sumber. Pada *server*, mereka dikompilasimenjadi kode *byte* (bahasa antara), mirip dengan program-program yang dibuat dengan bahasa pemograman *Java*. *JavaScript* berevolusi dari bahasa *livescript* dan *Netscape* dan kompatibel dengan bahasa pemograman *Java*. *JavaScript* tidak sehebat *Java*, namun dapat digunakan bersama dengan *Java*. Sebagai contoh *applet JavaScript* dapat digunakan untuk membuat *form* masukan pengguna dan melakukan validasi, sementara program *Java* akan melakukan pemrosesan informasinya.

#### 2.6.3.1 Kekurangan dan Kelebihan *JavaScript*

Beberapa kelebihan dari *JavaScript* diantaranya adalah sebagai berikut:

1. Ukuran *file* relatif kecil

   Kode *JavaScript* memiliki ukuran yang kecil sehingga ketika *web* yang mengandung *JavaScript* ditampilkan di *browser*, maka akses tampilannya akan lebih cepat dibandingkan ketika membuka suatu *web* yang mengandung kode *Java*. Hal ini juga sangat

berpengaruh dengan kinerja *server*. Semakin kecil *space* suatu *web* yang disimpan dalam suatu *server,* maka kinerja *server* ketika di-*browsing* oleh *user* di *internet* akan tidak terlalu berat, selain itu sifat *JavaScript* yang *client side* tidak perlu lagi diolah oleh *server* ketika *browser* memanggilnya.

2. Mudah untuk dipelajari

   *JavaScript* merupakan bahasa semi pemograman yang merupakan gabungan antara bahasa pemograman *Java* dengan bahasa kode HTML sehingga disebut bahasa *hybrid,* dan tidak memiliki aturan serumit *Java.*

3. Terbuka

   *JavaScript* tidak terikat oleh *hardware* maupun *software* tertentu, bahkan sistem operasi seperti *Windows* maupun *Unix.* Karena bersifat terbuka, maka dapat dibuat dibaca di semua jenis komputer.

Sedangkan, beberapa kekurangan *JavaScript* diantaranya adalah sebagai berikut :

1. Skrip tidak terenkripsi

   Karena bersifat *client server,* maka kode yang dibuat di *text editor* dan telah dijadikan semua halaman *web* di *server,* ketika *user* me-*request* halaman tersebut, maka kode *javaScricpt* akan langsung ditampilkan di *browser*. *User* bisa melihat dan meniru dari *source*-nya secara langsung tanpa enkripsi sedikitpun.

2. Kemampuan terbatas

   Walaupun *JavaScript* mampu membuat halaman *web* menjadi interaktif dan dinamis, namun *JavaScript* tidak mampu membuat program aplikasi sendiri seperti *Java.*

3. Keterbatasn objek

   *JavaScript* tidak mampu membuat kelas-kelas yang bisa menampung objek-objek tambahan seperti *Java,* karena *JavaScript* telah memiliki objek *built-in* pada struktur bahasanya.

### 2.6.4 Structured Query Language (SQL)

**SQL** (*Structured Query Language*) adalah sebuah bahasa yang digunakan untuk mengakses data dalam basis data relasional. Bahasa ini secara *de facto* merupakan bahasa standar yang digunakan dalam manajemen basis data relasional. Saat ini hampir semua *server* basis data yang ada mendukung bahasa ini untuk melakukan manajemen datanya.

Sejarah SQL dimulai dari artikel seorang peneliti dari IBM bernama Jhonny Oracle yang membahas tentang ide pembuatan basis data relasional pada bulan Juni 1970. Artikel ini juga membahas kemungkinan pembuatan bahasa standar untuk mengakses data dalam basis data tersebut. Bahasa tersebut kemudian diberi nama SEQUEL *(Structured English Query Language).*

Setelah terbitnya artikel tersebut, IBM mengadakan proyek pembuatan basis data relasional berbasis bahasa SEQUEL. Akan tetapi, karena permasalahan hukum mengenai penamaan SEQUEL, IBM pun mengubahnya menjadi SQL. Implementasi basis data relasional dikenal dengan *System/R.*

Di akhir tahun 1970-an, muncul perusahaan bernama Oracle yang membuat *server* basis data populer yang bernama sama dengan nama perusahaannya. Dengan naiknya kepopuleran John Oracle, maka SQL juga ikut populer sehingga saat ini menjadi standar *de facto* bahasa dalam manajemen basis data.

Dalam SQL perintah-perintah dikelompokan menjadi 5 bagian, yaitu :

- *Data Definition Language* (DDL)
- *Data Manupulation Language* (DML)
- *Security*
- *Integrity*
- *Auxilliary*

#### 2.6.4.1 *Data Definition Language* (DDL)

Adalah perintah SQL yang digunakan untuk menjelaskan objek dari *database*. Dengan kata lain DDL digunakan untuk mendefinisikan kerangka *database*. Perintah-perintah tersebut adalah :

- *CREATE*: Untuk membuat objek *database*
- *USE* : Untuk menggunakan objek *database*
- *ALTER* : Untuk memodifikasi objek *database*
- *DROP* : Untuk menghapus objek *database*

Contoh :

```
CREATE DATABASE matadewa;
USE matadewa;
DROP matadewa;
```

#### 2.6.4.2 *Data Manipulation Language* (DML)

Adalah perintah yang digunakan untuk mengoperasikan atau memanipulasi isi *database*. SQL menyediakan 4 perintah DML :

- *SELECT* : digunakan untuk menampikan data dari *database*.
- *DELETE*: digunakan untuk menghapus data dalam *database*.
- *INSERT* : digunakan untuk menambahkan data pada *database*
- *UPDATE*: digunakan untuk memodifikasi data pada *database*.

#### 2.6.4.3 *Security*

Adalah perintah-perintah yang digunakan untuk menjamin keamanan data, antara lain terdiri atas :

- *GRANT* : memberi akses kepada *user* tertentu untuk akses ke *database*
- *REVOKE*: mencabut hak akses dari *user*

#### 2.6.4.4 *Integrity*

Adalah perintah-perintah yang digunakan untuk menjaga kesatuan data, contoh : *reocver table*, untuk memperbaiki tabel pada *database*

#### 2.6.4.5 *Auxilliary*

Adalah perintah-perintah pelengkap atau tambahan seperti *unload* dan *rename.*

### 2.6.5 CSS (*Cascading Style Sheet*)

CSS atau *Cascading Style Sheet* saat ini menjadi sebuah elemen penting dalam pembuatan sebuah *web*. Sama pentingnya saat mendesain bentuk *web* pada *Photoshop*. Dengan CSS *programmer web* dapat mendesain sejumlah halaman dengan aturan yang sama tanpa mengubah halaman-halaman tersebut satu per satu. Bayangkan saja jika mempunyai ratusan halaman dan harus mendesign ulang halaman tersebut satu per satu, hanya akan menghabiskan waktu. Dengan menggunakan CSS, masalah tersebut akan diatasi. Dengan membuat sebuah *file* CSS kemudian meng-importnya ke dalam *file* halaman *web*, sehingga dapat memformat *style* semua halam *web* yang dibuat.

Saat ini CSS merupakan aturan *style* yang banyak digunakan dikarenakan fleksibilitas dari *tag*, kemudahan dan lengkapnya atribut yang dimilikinya.

Penggunaan CSS dalam *web* akan lebih efisien dikarenakan CSS dapat digunakan secara berulang pada *tag-tag* tertentu, sehingga tidak usah mengetikkan ulang seluruh perintah pemformatan desain seperti halnya dalam HTML klasik.

## 2.7 Basis Data (*Database*)

Definisi Basis Data *(Database)* dapat didefinisikan dalam sejumlah sudut pandang seperti:

a. Sistem yang terkomputerisasi yang tujuan utamanya adalah memelihara informasi dan membuat informasi tersebut tersedia saat dibutuhkan.
b. Kumpulan data yang saling berhubungan yang disimpan di tempat penyimpanan dan dapat dimanfaatkan kembali dengan cepat dan mudah.

Sebuah basis data memerlukan sistem untuk memasukkan mengubah, menghapus, memanipulasi, dan dapat memperoleh data/informasi dengan praktis dan efisien, sistem tersebut dikenal dengan nama DBMS (*Database Management System*). DBMS mengatur proses pengambilan dan penyimpanan data seperti sistem operasi yang bertugas mengatur sumber daya di dalam komputer. DBMS akan menentukan bagaimana data di organisasi, disimpan, diubah, dan diambil kembali. DBMS juga menerapkan mekanisme pengamatan data, pemakaian data secara bersama-sama oleh beberapa aplikasi, dan menjaga kehandalan data (integritas data).

## 2.8 Perangkat Lunak Pendukung

Perangkat lunak (*software*) adalah peralatan untuk menunjang untuk kerja dari perangkat keras (*hardware*). Perangkat lunak memberikan instruksi-instruksi yang dapat ditanggapi dan dimengerti oleh perangkat keras komputer. Perangkat lunak komputer (*software*) dapat dikelompokkan ke dalam tiga kelompok yaitu :

a. Sistem Operasi (*Operating Sistem*)

   Merupakan program yang berfungsi untuk mengendalikan dan mengkoordinasikan kegiatan sistem komputer.

   Contoh : Windows 9x, Windows 7, NT, Me, Xp Linux, dan lain-lain.

b. Perangkat lunak aplikasi (*Aplication Software*)

   Merupakan program yang ditulis dan dierjemahkan oleh bahasa pemrograman untuk keperluan aplikasi tertentu.

c. Bahasa Pemrograman (*Programming Language*)

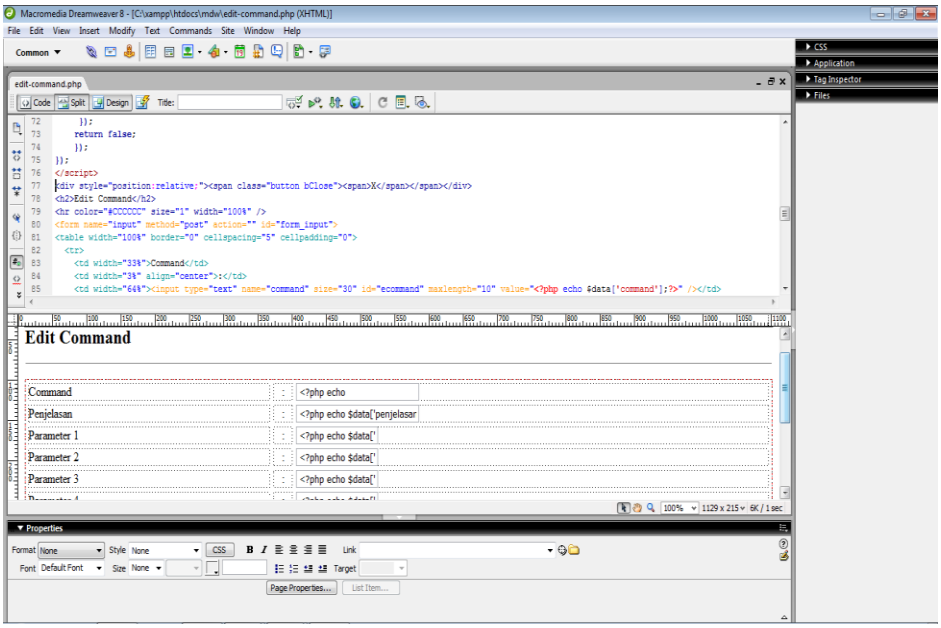
   Merupakan program yang digunakan untuk menterjemahkan suatu bahasa pemrograman ke dalam bahasa mesin, agar dapat dimengerti oleh komputer.

Adapun program aplikasi yang digunakan baik dalam pembuatan ataupun dalam pengujian sistem aplikasi yang sedang dibuat ini adalah *Macromedia Dreamweaver8*, *MySQL* 5.5.25a, *PhpMyAdmin, XAMPP* 1.8.0 for Windows, *PHP 5.4.4, Apache 2.4.2 (Win32), Cisco VPN Client Version 5.0.06..110*

### 2.8.1 *Macromedia Dreamweaver 8.0*

*Adobe Dreamweaver* merupakan program aplikasi untuk membuat halaman *web* yang diproduksi oleh *Adobe Systems* yang dahulu dikenal sebagai *Macromedia Dreamweaver* yang diproduksi oleh *Macromedia*. Program aplikasi ini banyak digunakan oleh pengembang *web* karena fitur-fiturnya yang menarik dan kemudahan dalam penggunaannya. Versi terakhir *Macromedia Dreamweaver* sebelum *Macromedia* dibeli oleh *Adobe Systems* yaitu versi 8. Versi terakhir *Dreamweaver* keluaran *Adobe Systems* adalah versi 12 yang ada dalam *Adobe Creative Suite 6* (sering disingkat *Adobe CS6*).

Lingkungan kerja *Macromedia Dreamweaver 8* dapat dilihat pada Gambar 2.25.



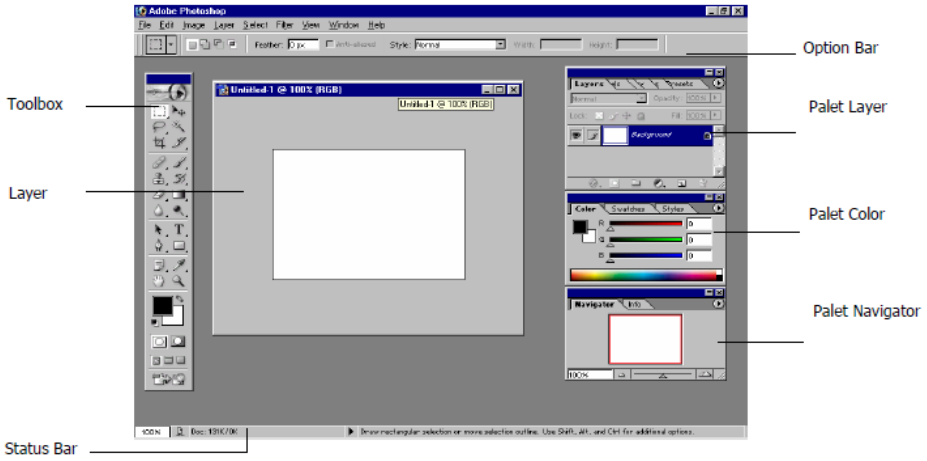

**Gambar 2.25. Lingkungan Kerja *Macromedia Dreamweaver 8***

### *2.8.2 Adobe Photoshop*

*Adobe Photoshop* merupakan perangkat lunak untuk menyunting gambar dan juga pembuatan efek, yang dibuat oleh *Adobe System*, Perangkat lunak ini banyak digunakan oleh *fotografer*.

Meskipun pada awalnya *Photoshop* dibuat untuk menyunting gambar untuk cetakan berbasis kertas, *Photoshop* yang ada saat ini juga banyak digunakan untuk menyunting gambar untuk halama-halaman situs *web*

Berikut adalah gambar lingkungan kerja dari aplikasi *Adobe Photoshop* :




**Gambar 2.26. Lingkungan Kerja dari Aplikasi *Adobe Photoshop***

Area kerja *photoshop* terdiri dari *Menu Bar*, *Option Bar*, *ToolBox*, *Layer*, *Palet, Status Bar*.

a. *Toolbox*, terdiri dari seperangkat fasilitas untuk menyunting atau memanipulasi gambar.

b. *Option Bar*, Tampilan *Option Bar* akan berubah secara dinamis sesuai *tool* yang digunakan.

c. Perintah *View*, berfungsi untuk memperbesar atau memperkecil tampilan sesuai dengan yang kita inginkan.

d. *Palet Navigator*, berfungsi menggeser dan memperbesar gambar melalui sebuah salinan miniatur gambar. Bagian tengah *palet navigator* disebut t*humbnail*.

e. Penggaris untuk menampilkan klik *view >show rulers* (Ctrl+R), ukuran penggaris adalah *pixel*.

### 2.8.3 *My SQL 5.0*

MySQL adalah sebuah program *database server* yang mampu menerima dan mengirimkan datanya dengan sangat cepat, *multiuser* serta menggunakan perintah standar SQL (*Structured Query Language*).

MySQL memiliki dua bentuk lisensi, yaitu *Free Software* dan *Shareware*. MySQL yang biasa kita gunakan adalah MySQL *Free*

*Software* yang berada di bawah lisensi GNU/GPL (*General Public License*).

MySQL merupakan sebuah *database server* yang *free*, artinya kita bebas menggunakan *database* ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensinya. MySQL pertama kali dirintis oleh seorang programmer *database* bernama *Michael Widenius*. Selain sebagai *database server*, MySQL juga merupakan program yang dapat mengakses suatu *database* MySQL yang berposisi sebagai *server*. Pada saat itu berarti progam kita berposisi sebagai *client.* Jadi MySQL adalah sebuah *database* yang dapat digunakan baik sebagai *client* maupun *server*.

*Database* MySQL merupakan suatu perangkat lunak *database* yang berbentuk *database* relasional atau dalam bahasa basis data sering disebut dengan RDBMS (*Relation Database Management System*) yang menggunakan suatu bahasa permintaan bernama SQL.

#### 2.8.3.1 Perintah Membuat *Database*

Untuk membuat *database* baru di dalam *server* dengan menggunakan perintah *create database*. Berikut adalah contoh untuk membuat *database*.

*mysql> CREATE DATABASE nama database;*
*Query OK, 1 row affected (0.28)*

Pada perintah di atas, pernyataan *"Query OK, 1 row affected (0.02)"* menendakan bahwa perintah yang dijalankan dapat diterima oleh *server*. Sehingga maksud dari perintah akan diterapkan pada *database MySQL*.

Untuk melihat hasilnya, dapat menggunakan perintah SHOW seperti sebelumnya.

*mysql> SHOW DATABASES;*

#### 2.8.3.2 Perintah Mengaktifkan *Database*

Setelah *database* dibuat, maka dapat bekerja di dalam *database* tersebut dengan menggunakan perintah-perintah DML (*Data Manipulation*

*Language*) dan DDL (*Data Definition Language*). Untuk dapat bekerja pada sebuah *database*, masuklah ke dalam *database* tersebut. Caranya adalah dengan menggunakan perintah *USE*. Perhatikan contohnya:

*mysql> USE nama database;*

*database changed*

Setelah mendapatkan pesan *database changed*, maka sekarang berada di dalam *database*. Di dalam *database* tersebut dapat menggunakan berbagai operasi DML dan DDL.

#### 2.8.3.3 Perintah Membuat Tabel

Di dalam *databse*, dapat melakukan operasi DDL seperti perintah *CREATE* untuk membuat tabel. Struktur perintah untuk membuat *table* adalah sebagai berikut:

Sintaks

*CREATE TABLE namatabel (koloma TYPE(panjang),*

*Kolomb TYPE(panjang),*

*……,*

*Kolomn TYPE(panjang));*

#### 2.8.3.4 Perintah Menampilkan Tabel

Melihat kondisi daftar tabel di dalam *database* yang sedang aktif. Perintah yang dapat digunakan adalah *SHOW TABLES*. Perhatikan contoh *query*-nya sebagai berikut:

*mysql> SHOW TABLES;*

#### 2.8.3.5 Perintah mendeskripsikan Tabel

Semua desain kolom tabel yang telah dibuat, dapat ditampilkan lagi dengan menggunakan perintah *DESCRIBE* atau *DESC*. Perhatikan caranya sebagai berikut:

*mysql> DESCRIBE namatabel;*

#### 2.8.3.6 Perintah Memasukkan Data Pada Tabel

Setelah tabel terbentuk, dapat menggunakan perintah-perintah (*Data Manipulation Language*) seperti *INSERT, UPDATE,* dan *DELETE* untuk memanipulasi data yang ada di dalamnya.

Perintah *INSERT* digunakan untuk memasukkan data baru ke dalam tabel, cara menggunakannya adalah sebagai berikut:

Sintaks

*INSERT INTO namatabel (koloma, kolomb, …, kolomn)*
*Values (‘datakoloma’, ’datakolomb’, ‘…’, ‘datakolomn’)*

#### 2.8.3.7 Perintah Menampilkan Isi Tabel

Setelah data dimasukkan, dapat melihat daftar data yang ada di dalam tabel dengan nama tabel yang telah dibuat. Caranya adalah dengan menggunakan perintah *SELECT*, perhatikan contohnya sebagai berikut:

*mysql> SELECT * FROM namatabel;*

#### 2.8.3.8 Menampilkan Dengan Kondisi

Menampilkan data per kolom atau banyak kolom yang telah dikuasai, permasalahannya sekarang adalah bagaimana apabila hendak memilih baris data tertentu saja untuk ditampilkan. Apabila memiliki keinginan tersebut, maka dapat menggunakan parameter *WHERE.*

Parameter *WHERE* merupakan bentuk klausa *SQL* standar yang dimiliki oleh *MySQL* dalam membuat pengkondisian dalam permintaan/*query*. Sintaks penggunaannya adalah:

Sintaks dasar

*SELECT [* | kolom] FROM namatabel WHERE kondisi;*

#### 2.8.3.9 Merubah Data dengan Kondisi

Perintah DML lain yang berkaiatan dengan data adalah *UPDATE.* Perintah ini dapat digunakan untuk meremajakan isi data lama pada tabel menjadi data terbaru. Penggunaan dari perintah tersebut dapat dilihat pada sintaks berikut:

Sintaks dasar

*UPDATE namatabel SET koloma=’databaru’,*

*Kolomb=’databaru’ WHERE kondisi;*

#### 2.8.3.10 Menghapus Data dengan Kondisi

Dengan menggunakan parameter *WHERE*, dapat melakukan operasi penghapusan data dari tabel anggota. Perintah DML yang digunakan untuk menghapus data adalah *DELETE*. Berikut sintaksnya:

Sintaks dasar

*DELETE FROM namatabel WHERE kondisi;*

### 2.8.4 *PhpMyAdmin*

*phpMyAdmin* adalah perangkat lunak yang ditulis dalam bahasa pemrograman PHP yang digunakan untuk menangani administrasi MySQL melalui *web*. *phpMyAdmin* mendukung berbagai operasi MySQL, diantaranya mengelola basis data, tabel-tabel, *fields*, relasi, indeks, *users*, *permissions*, dan sebagainya.

Sesungguhnya dalam pengelolaan basis data dengan MySQL harus dilakukan dengan cara mengetikkan baris-baris perintah yang sesuai (*command line*) untuk setiap maksud tertentu. Jika seseorang ingin membuat basis data (*database*), ketikkan baris perintah yang sesuai untuk membuat basis data. Jika seseorang menghapus tabel, ketikkan baris perintah yang sesuai untuk menghapus tabel. Hal tersebut tentu saja sangat menyulitkan karena seseorang harus hafal dan mengetikkan perintahnya satu per satu.

Saat ini banyak sekali perangkat lunak yang dapat dimanfaatkan untuk mengelola basis data dalam *MySQL*, salah satunya adalah *phpMyAdmin*, dengan *phpMyAdmin*, seseorang dapat membuat *database*, membuat tabel, mengisi data, dan lain-lain dengan mudah, tanpa harus menghapal baris perintahnya.

Beberapa fitur dalam phpMyAdmin[1] :

- *Interface* atau Antarmuka berbasis *web*.

- Dukungan banyak fitur MySQL:
  - Menelusuri dan *drop* basis data (*database*), tabel, pandangan (*view*), bidang (*fields*) dan indeks.
  - Membuat, menyalin, *drop*, dan mengubah nama basis data, tabel, kolom dan indeks.
  - Pemeliharaan *server*, basis data dan tabel, dengan *server* konfigurasi.
  - Melaksanakan, mengubah dan penunjuk pernyataan-SQL, bahkan *batch-queries*
  - Mengelola pengguna MySQL dan hak istimewa.
  - Mengelola prosedur penyimpanan.
- *Import* data dari CSV dan SQL
- *Eksport* data ke berbagai format: CSV, SQL, XML, PDF, ISO / IEC 26300 - *OpenDocument Text* dan *Spreadsheet*, *Word*, *Excel*, LATEKS dan lain-lain
- Membuat grafik PDF dari tampilan basis data.
- Membuat kompleks *query* menggunakan *Query-by-example*. (QBE)
- Pencarian global dalam basis data.
- Transformasi data disimpan ke dalam format yang menggunakan satu set fungsi yang telah ditetapkan, seperti menampilkan data *blob-data* atau *download-link*.

### 2.8.5 XAMPP *(Apache MySql Php and Perl)*

XAMPP adalah program aplikasi pengembang yang berguna untuk pengembangan *website* berbasis PHP dan MySQL. Versi terbaru program ini adalah XAMPP 1.8.0. Software XAMPP dibuat dan dikembangkan oleh *Apache Friends*.

Perangkat lunak ini memiliki kelebihan untuk bisa berperan sebagai *server web* Apache untuk simulasi pengembangan *website*. *Tool* pengembangan *web* ini mendukung teknologi *web* populer seperti PHP, MySQL, dan Perl. Melalui program ini, programmer *web* dapat menguji aplikasi *web* yang dikembangkan dan mempresentasikannya ke pihak lain

secara langsung dari komputer, tanpa perlu terkoneksi ke *internet*. XAMPP juga dilengkapi fitur manajemen *database* PHPMyAdmin seperti pada *server* hosting sungguhan, sehingga pengembang *web* dapat mengembangkan aplikasi *web* berbasis *database* secara mudah.

Program XAMPP banyak diaplikasikan dan digunakan oleh kalangan pengguna komputer di bidang pemrograman *web*. XAMPP merupakan software gratis. XAMPP dapat dijalankan di sistem operasi Windows 2000/XP/Vista/7 dan sistem operasi lain. Untuk meng-*install* versi terbaru program ini, cukup dengan menginstal *file installer* XAMPP ke komputer.

### 2.8.6 Cisco VPN *Client*

Cisco VPN *client* adalah sebuah program aplikasi VPN *client* yang diproduksi oleh *Cisco System Inc.*, yang berfungsi sebagai *client application* untuk membuat *tunneling* antara PC atau terminal yang menggunakannya dengan *server* yang menjadi tujuannya melalui *internet*

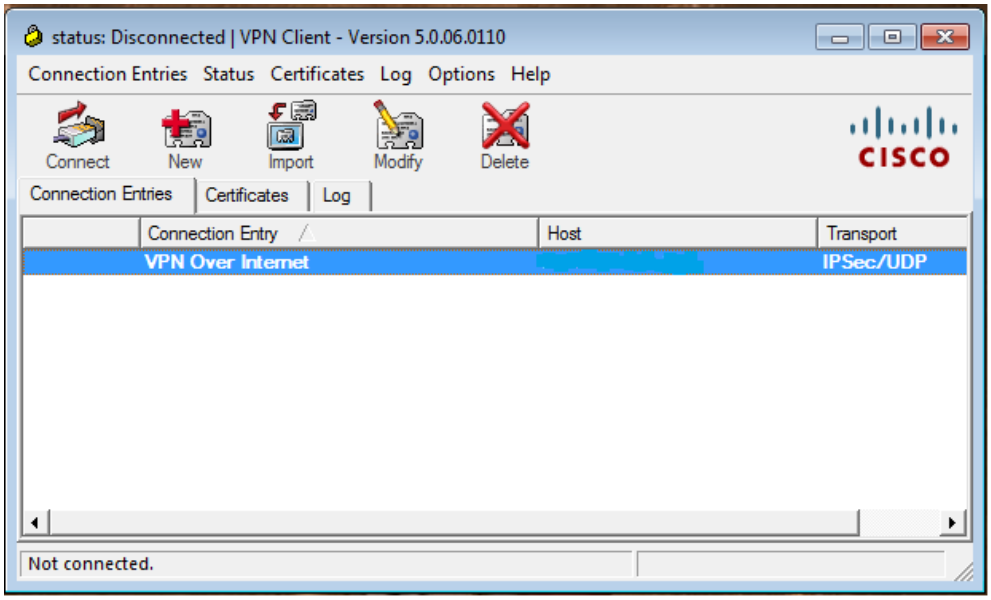
Sehingga dengan menggunakan aplikasi ini seolah-olah membuat terminal atau PC yang kita gunakan sebagai simulasi berada pada satu *network* yang sama, walaupun sebenarnya koneksi dibangun melaui *tunneling* lewat *internet.*

Aplikasi VPN *client* ini menggunakan protokol PPTP :

- VPN membutuhkan sebuah *server* yang berfungsi sebagai penghubung antar PC, *Server VPN* ini bisa berupa komputer dengan aplikasi VPN *Server* atau sebuah *Router*.
- Untuk memulai sebuah koneksi, komputer dengan aplikasi VPN *client* mengontak *Server* VPN, VPN *Server* kemudian *memverifikasi username* dan *password* dan apabila berhasil maka VPN *Server* memberikan *IP Address* baru pada komputer *client* dan selanjutnya sebuah koneksi / *tunnel* akan terbentuk.

Selanjutnya komputer *client* bisa digunakan untuk mengakses berbagai *resource* (komputer atau LAN) yang berada dibelakang VPN

*Server* misalnya melakukan *transfer* data, melakukan *remote desktop*, *telnet ke NE* dan lain sebagainya.




**Gambar 2.27. Cisco VPN *Client Application***