2. TEORI DASAR

2.1. Sistem Informasi Manajemen

Sistem Informasi Manajemen (SIM) merupakan sistem manusia dan mesin yang terintegrasi yang menyajikan informasi guna mendukung fungsi operasi, manajemen, dan pengambilan keputusan dalam sebuah organisasi.[1] Sistem Informasi Manajemen (SIM) dalam suatu perusahaan selalu mencakup dua hal penting, yaitu: sistem pengolahan data dan sistem pengolahan informasi. Dalam pelaksanaan proses pengolahan data dan informasi, Sistem Informasi Manajemen (SIM) menggunakan beberapa alat bantu diantaranya:

- Perangkat keras (*hardware*)
- Perangkat lunak (*software*)
- Prosedur pedoman
- Model manajemen dan keputusan
- *Database*

Sistem Informasi Manajemen (SIM) sebagai perpaduan konsep sistem manusia dan mesin memiliki artian, bahwa:

- Perancang sebuah Sistem Informasi Manajemen (SIM) harus memahami kemampuan manusia sebagai pengolah informasi dan perilaku manusia dalam mengambil keputusan.
- Sebagian tugas sebaiknya dilaksanakan oleh manusia dan lainnya sebaiknya dilakukan oleh mesin.
- Sebuah sistem gabungan dengan hasil yang diperoleh melalui serangkaian dialog dan interaksi antara komputer dan seorang manusia pengolah.
- Merupakan paduan terencana dari berbagai penerapan yang layak dan efektif.

Jika ditinjau dari segi struktur kalimatnya, Sistem Informasi Manajemen merupakan penggabungan dari kata: Sistem, Informasi, dan Manajemen. Dimana masing-masing kata memiliki pengertian yang lebih mendalam sebagai satu

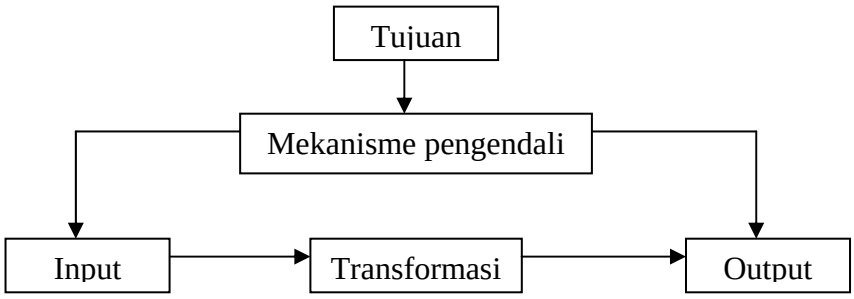
[1] Gordon B. Davis. (1998). Management Information System : Conceptual Foundation, Structures, and Development. International Student Edition. Tokyo : McGraw Hill Kogakusha. p. 5.

kesatuan yang menyusun elemen-elemen pembangunan sistem organisasi suatu perusahaan.

a. Sistem

Suatu sistem dapat dijelaskan dengan sederhana sebagai sekelompok elemen yang terintegrasi dengan maksud yang sama untuk mencapai suatu tujuan. Sistem sendiri memiliki konfigurasi dasar, antara lain:

- Elemen *Input*
- Elemen Transformasi
- Elemen *Output*
- Mekanisme pengendali dan tujuan
- *Feedback loop*



Gambar 2.1. Konfigurasi Sistem

Sumber daya *input* diubah menjadi sumber daya *output*. Sumber daya mengalir dari elemen *input* melalui elemen transformasi kepada elemen *output.* Suatu mekanisme pengendali memantau proses transformasi untuk meyakinkan bahwa sistem tersebut memenuhi tujuannya. Mekanisme pengendali ini dihubungkan pada arus sumber daya dengan memakai suatu lingkaran umpan balik (*feedback loop*) yang mendapatkan informasi dari *output* sistem dan menyediakan informasi bagi mekanisme pengendali. Mekanisme pengendali membandingkan sinyal-sinyal umpan balik dengan tujuan, dan mengarahkan sinyal pada elemen *input* jika sistem operasi memang perlu diubah.

b. Informasi

Informasi merupakan produk pengolahan data, yang memiliki arti penting dan ini menunjukkan bahwa data merupakan bahan baku dari informasi yang berasal dari gambaran kejadian yang berwujud karakter, angka, dan simbol yang digunakan untuk tujuan normatif atau kesimpulan, argumentasi, atau sebagai dasar untuk peramalan dan pengambilan keputusan. Kualitas dari suatu informasi diukur berdasarkan keakuratannya. Wujud dari informasi sendiri adalah data, gambar, naskah, dan dokumen.

c. Manajemen

Manajemen merupakan suatu proses tertentu yang terdiri dari perencanaan, pengarahan, dan pengawasan yang dilakukan untuk menentukan dan mencapai tujuan yang telah ditetapkan dengan menggunakan manusia dan sumber daya lainnya.[2] Manajemen membutuhkan sistem informasi untuk mendukung pengambilan keputusan yang akan diterapkan. Sumber informasi untuk pengambilan keputusan manajemen dapat diperoleh dari informasi internal maupun informasi eksternal.

2.2. *E-Commerce*

2.2.1. Pengertian *E-commerce*

E-commerce adalah singkatan dari *electronic commerce* yang merupakan konsep yang timbul yang menggambarkan proses pembelian, penjualan atau pertukaran produk, pelayanan dan informasi dengan menggunakan jaringan komputer seperti internet.[3] Klasifikasi *e-commerce* berdasarkan karakteristik transaksi ada enam jenis (Turban et al.2000):

- *Business-to-Business* (B2B)

  Transaksi antara organisasi yang satu dengan organisasi yang lain.

  Kebanyakan *e-commerce* saat ini, menggunakan kategori B2B.

- *Business-to-Consumer* (B2C)

  Penjual menawarkan produk/jasa langsung ke pembeli.

[2] George R. Terry, Ph D. (1997). Principles of Management Seven Edition. Rihard D. Irwin. Inc. Homewood. Illinois. p. 4.
[3] Turban, Efraim; King, David; Lee, Jae; Warkentin, Merrill; Chung, H. Michael. (2002). Electonic Commerce : A Managerial Perspective (International Edition). p. 4.

- *Consumer-to-Consumer* (C2C)

  Konsumen menjual produk/jasa langsung ke konsumen lain.

  Biasanya individu mengiklankan produk, jasa, pengetahuan maupu keahlian di salah satu situs lelang atau *classified adjective*. Interaksi semacam ini membutuhkan biaya yang cukup tinggi dengan nilai transaksi yang sifatnya kurang menguntungkan (*profitable*).
- *Consumer-to- Business* (C2B)

  Perorangan menjual produk/jasa pada organisasi. Perorangan mencari penjual, berinteraksi dengan mereka baru kemudian mengadakan transaksi.
- *Government-to-Citizens* (G2C)

  Pemerintah menjual/membeli produk, jasa, atau informasi kepada organisasi atau masyarakat.
- *Non-Business E-commerce*

  Terdiri dari institusi *non-business* seperti lembaga pendidikan, organisasi nirlaba, organisasi keagamaan, sosial, dan instansi pemerintah.

2.2.2. Langkah-Langkah Dalam Pembuatan *E-commerce*

Terdapat enam langkah dalam proses pembuatan *e-commerce* yang diharapkan dapat membantu untuk mengevaluasi apakah pembuatan *e-commerce* tersebut sudah sesuai dengan kebutuhan atau belum. Menurut Koontz (2000) ada enam langkah pembuatan *e-commerce* yang disarankan yaitu:

- Menetapkan tujuan dan visi organisasi..

  Tujuan dan visi organisasi sangat penting diketahui, sebagai langkah awal untuk menentukan proses dalam membangun arsitektur *e-commerce* selanjutnya.
- Mengumpulkan informasi mengenai rancangan *web site* tersebut.

  Informasi merupakan hal yang penting dalam memenuhi tujuan pembuatan *web site*.
- Mengumpulkan data mengenai rancangan *web site*.

  Setelah mengetahui informasi apa yang harus diproses oleh aplikasi *web site*, perlu ditentukan data dan informasi apa yang ingin diperoleh dari konsumen.

- Menetapkan aplikasi yang cocok untuk pembuatan *web site*.
  Dalam langkah ini, perlu ditentukan komponen atau modul aplikasi yang cocok dengan data yang dibutuhkan.
- Menetapkan susunan secara teknis dari web yang akan dibuat.
  Dalam langkah ini, perlu menguji *hardware* yang spesifik dan *software* yang dibutuhkan untuk mendukung analisis pada langkah sebelumnya.
- Menetapkan susunan organisasi.
  Susunan organisasi berhubungan dengan sumber daya manusia dan prosedur yang dibutuhkan dari langkah ke-1 sampai langkah ke-5.

2.3. *System Development Life Cycle* (SDLC)

2.3.1. Pengertian SDLC

System Development Life Cycle (SDLC) atau siklus hidup pengembangan sistem adalah proses evolusioner yang diikuti dalam menerapkan sistem atau subsistem informasi berbasis komputer.[4] Sistem ini terdiri dari serangkaian tugas yang erat mengikuti langkah-langkah pendekatan sistem karena tugas-tugas tersebut mengikuti suatu pola yang teratur dan dilakukan secara *top down*, sistem ini sering disebut sebagai pendekatan air terjun (*waterfall approach*) bagi pengembangan dan penggunaan sistem.

2.3.2. Langkah-Langkah Dalam SDLC

Terdapat 5 langkah SDLC yaitu:

1. *Planning*
   Tahap *planning* merupakan tahap pendeskripsian masalah yang dihadapi dan perencanaan terhadap lingkup sistem yang akan dibuat untuk memenuhi kebutuhan.
2. *Analysis*
   Tahap *analysis* merupakan langkah awal dalam proses pengembangan sistem. Analisa sistem berfungsi untuk mengumpulkan seluruh informasi yang diperlukan dalam membuat sistem baru.

[4] Mc Leod, R. (1996). Sistem Informasi Manajemen. p. 214.

3. *Design*

   Tahap *design* merupakan tahap dimana kita mempertimbangkan bagaimana sistem yang akan dikembangkan dapat sesuai dengan kebutuhan pengguna.

4. *Implementation*

   Tahap *implementation* merupakan tahap pembuatan sistem dari desain sistem yang telah dibuat dan pengkodean *software*.

5. *Maintenance*

   Tahap *maintenance* merupakan tahap pemeliharaan sistem yang telah dibuat. *Maintenance* bertujuan untuk menjaga sistem dari kerusakan, memudahkan programmer bila ingin memodifikasi sistem, dan menjaga agar *performance* sistem berjalan dengan baik.

2.4. *Flowchart*

Flowchart adalah suatu gambar grafik dari urutan-urutan dan langkah-langkah logikal yang terlibat dalam suatu prosedur atau suatu program. *Flowchart* membantu analis atau *programmer* untuk memecahkan masalah yang besar menjadi lebih kecil, lebih menjadi segmen-segmen yang dapat dikerjakan dengan mudah, dan membantu dalam menganalisa jalan-jalan alternatif dalam suatu operasi. *Flowchart* dapat menambah keefektifan waktu bekerja dan waktu pikiran. *Flowchart* terdiri dari tiga elemen grafik sederhana yang dapat dikombinasikan untuk menggambarkan berbagai tipe aliran informasi dan proses secara *physical*. Ketiga elemen tersebut adalah sebagai berikut:

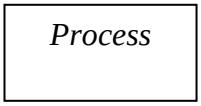
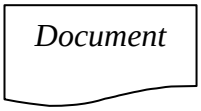
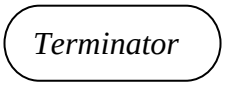
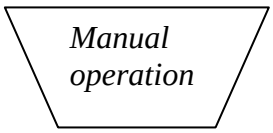
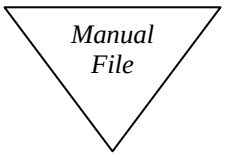
1. *Symbol*

   Banyak variasi simbol yang dapat digunakan untuk menggambarkan aspek *physical* aliran dokumen atau data dan proses sistem informasi.

   Tabel 2.1. di bawah ini merupakan beberapa contoh simbol *flowchart*.[5]

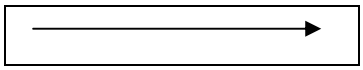
[5] Kristanto, Andi. (2003). Perancangan Sistem Informasi dan Aplikasinya. Yogyakarta : Penerbit Gava Media. P.68.

Tabel 2.1. Beberapa Simbol Dalam *Flowchart*

Simbol	Keterangan
	Menggambarkan suatu operasi, aktivitas, atau tugas.
	*Output* dari suatu operasi
	*Terminator* dalam suatu *flowchart*, misalnya *start* dan *stop*.
	Menggambarkan operasi yang dilakukan secara manual
	Tempat penyimpanan data (arsip manual)
	Data penyimpanan (data *storage*)
	Pengambilan keputusan

2. *Flow Lines*

   Flow Lines digunakan untuk menghubungkan simbol-simbol *flowchart*. Garis penuh menggambarkan aliran dokumen *physical* secara nyata, seperti gambar 2.2.di bawah ini, sedangkan garis putus-putus menggambarkan aliran informasi bukan dokumen *physical*.



Gambar 2.2. Simbol *Flow Lines*

3. *Areas of responsibility*

   Areas of responsibility memungkinkan user untuk mengidentifikasikan perubahan aliran data atau aliran informasi dengan lebih jelas. *Areas of responsibility* ini dibagi ke dalam segmen-segmen dimana tiap segmen dapat menunjukkan sebuah departemen atau pekerja individual, seperti gambar 2.3. di bawah ini.

Dept A	Dept B	Dept C

Gambar 2.3. *Areas of responsibility*

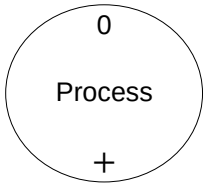
2.5. *Data Flow Diagram* (DFD)

2.5.1. Pengertian DFD

Data Flow Diagram (DFD) adalah gambaran grafis dari suatu sistem yang secara grafis digambarkan dengan sejumlah simbol-simbol untuk menunjukkan perpindahan data dalam proses-proses suatu sistem. Dalam DFD ada empat komponen utama yaitu:

1. *Process*

   Process menunjukkan kegiatan atau kerja yang dilakukan dari suatu arus data masuk untuk menghasilkan arus data ke luar. Simbol *process* dapat dilihat pada gambar 2.4.



Gambar 2.4. Simbol *Process*

2. *Data Flow*

   Data Flow menunjukkan aliran data yang menunjukkan perpindahan data dari satu bagian ke bagian yang lain dalam suatu sistem. Simbol *data flow* dapat dilihat pada gambar 2.5.

Gambar 2.5. Simbol *Data Flow*

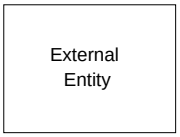
3. *Data Store*

   Data Store menunjukkan tempat penyimpanan data, baik manual maupun secara elektronik. *Data Store* dapat berupa *file*, arsip, *table*, dan lain lain. Simbol *data store* dapat dilihat pada gambar 2.6.

Gambar 2.6. Simbol *Data Store*

4. *External Entity*

   External Entity menunjukkan orang, organisasi, atau sebuah sistem lain yang memberi *input* atau menerima *output* dari suatu sistem.Simbol *external entity* dapat dilihat pada gambar 2.7.



Gambar 2.7. Simbol *External Entity*

2.5.2. Peraturan Penting Dalam DFD

Dalam penggambaran simbol DFD, ada beberapa peraturan yang harus diperhatikan sehingga dalam penggambarannya tidak terjadi kesalahan.
Peraturan-peraturannya adalah sebagai berikut:

1. Antar *entity* luar tidak diijinkan terjadi hubungan atau relasi.
2. Dalam DFD tidak dibedakan antara data dan informasi, semua dianggap data.
3. Nama proses dalam *context diagram* harus sama dengan nama sistem tersebut.
4. Setiap simbol harus diberi nama yang unik, namun harus konstan atau tidak boleh berubah jika berpindah ke level berikutnya.
5. Hindari garis yang berpotongan jika mungkin.
6. Setiap proses dalam DFD harus mempunyai *input* dan *output*.
7. Suatu *external entity* (kesatuan luar) hanya boleh mempunyai *input* atau *output*, tapi tidak boleh kedua-duanya.
8. Setiap *data store* (simpanan data) hanya boleh menerima *input* dari proses dan juga memberikan *output* ke proses saja.

2.6. *Entity Relationship Diagram* (ERD)

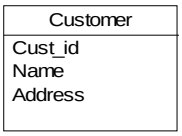
2.6.1. Pengertian ERD

ERD adalah diagram yang dipakai untuk mendokumentasikan data perusahaan dengan mengidentifikasi jenis entitas dan hubungannya. ERD menyediakan suatu konsep yang bermanfaat yang dapat mengubah deskripsi informal dari apa yang diinginkan oleh user menjadi hal yang lebih detail, presisi, dan deskripsi detail tersebut dapat diimplementasikan ke dalam DBMS. Dengan menggunakan ERD dapat dilihat dengan jelas hubungan antar *file-file* dalam *database* dan melalui ERD ini diharapkan *programmer* dapat menentukan seperti apakah program yang akan dibuat. Hal ini sangat membantu dalam merevisi program atau dalam pengembangan program tersebut nantinya.

2.6.2. Simbol-simbol ERD

1. *Entity*

   Entity menggambarkan tabel yang akan dipakai dengan nama *entity* di bagian atas dan atribut di bagian bawah. Suatu *entity* bisa berupa objek yang memiliki keberadaaan fisik (manusia, rumah, dan lain sebagainya) atau bisa berupa suatu objek yang memiliki keberadaaan konseptual (organisasi, pekerjaan, dan lain sebagainya). Contoh dari *entity* dengan *attribute* seperti ditunjukkan pada gambar 2.8. di bawah ini.



Gambar 2.8. Simbol *Entity* dengan *Attribute*

2. *Attribute*

   Attribute adalah pengelompokkan isi dari suatu *entity* yang berupa informasi tentang *entity* tersebut.

   Misalnya *entity customer* memiliki *attribute* Cust_id, Name, Address.

3. *Key*

   - *Candidate key*

     Candidate key adalah deskripsi dari suatu atribut atau *field* yang mengidentifikasikan secara unik *rows* dari suatu tabel dalam *relational database*.Contoh entitas *employees* yang memiliki atribute *employno*, *name*, *address*. *Employno* adalah *candidate key* karena mengidentifikasi secara unik entitas *employee*.

   - *Primary key*

     Primary Key adalah *candidate key* (bisa lebih dari satu) yang dipilih oleh database designer untuk mengidentifikasi suatu entitas.

   - *Foreign Key*

     Foreign Key adalah atribut yang digunakan untuk menghubungkan tabel. Contoh: idgroup pada tabel komedian merupakan *foreign key* dari tabel komedian.

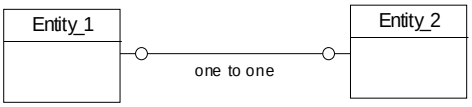
4. *Relationship*

Relatioship menggambarkan relasi atau hubungan antara dua *entity* atau lebih dengan arti tertentu.

Ada empat macam relasi yang ada pada ERD, yaitu:

- *One to One* (1:1)

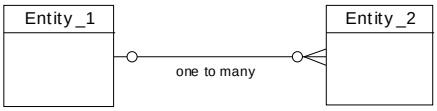
  Merupakan relasi dimana anggota entitas A berhubungan dengan paling banyak dengan satu anggota entitas B, dan begitu juga sebaliknya anggota entitas B berhubungan dengan paling banyak satu anggota entitas A. Notasi *one to one relationship* dapat dilihat pada gambar 2.9.



Gambar 2.9. *One to One Relationship*

- *One to Many*

  Merupakan relasi dimana anggota entitas A dapat berhubungan dengan banyak anggota entitas B, tetapi tidak sebaliknya, setiap anggota entitas B berhubungan dengan hanya satu anggota entitas A. Notasi *one to many relationship* dapat dilihat pada gambar 2.10.



Gambar 2.10. *One to Many Relationship*

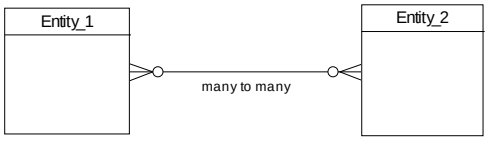
- *Many to One*

  Merupakan relasi dimana anggota entitas A berhubungan dengan hanya satu anggota entitas B, tetapi tidak sebaliknya, setiap anggota entitas B dapat berhubungan dengan banyak anggota entitas A. Notasi *many to one relationship* dapat dilihat pada gambar 2.11.

Gambar 2.11. *Many to One Relationship*

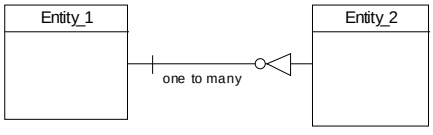
- *Many to Many*

  Merupakan relasi dimana setiap anggota entitas A dapat berhubungan dengan banyak anggota entitas B, dan demikian sebaliknya, setiap anggota entitas B dapat berhubungan dengan bnayak anggota entitas A. *Relationship* ini jarang terjadi. *Relational model* tidak dapat mendukung M:N *relationship*, sehingga mengubah *relationship* M:N menjadi dua 1:M dengan membuat satu tipe entitas ketiga. Notasi *many to many relationship* dapat dilihat pada gambar 2.12.



Gambar 2.12. *Many to Many Relationship*

Pada keempat macam relasi yang telah disebutkan di atas, dapat terjadi sebuah relasi yang *dependent.* Yang dimaksud dengan *dependent* adalah tabel yang satu tergantung dengan tabel yang lain.
Contohnya tabel A bergantung dengan tabel B.



Gambar 2.13. *One to Many* yang *dependent*

2.6.3. Cara Merancang ERD

Langkah-langkah yang harus dilakukan dalam pembentukan sebuah ERD adalah :

1. Mengidentifikasi entitas.
2. mengidentifikasi hubungan.
3. Menyiapkan ERD kasar.
4. Memindahkan elemen elemen data ke dalam *entity*.
5. Melakukan analisis data, dengan proses yang disebut normalisasi.
6. Menyiapkan ERD yang telah dimodifikasi.
7. Me-*review* ERD bersama-sama dengan *user* dan melakukan perbaikan jika perlu.

(Mc Leod, 1996)

2.6.4. Normalisasi

Adalah suatu proses dimana elemen-elemen data dikelompokkan menjadi tabel-tabel, dimana dalam tabel tersebut terdapat entity-entity dan relasi antar entity tersebut.[6]

Dalam proses normalisasi, *key* memegang peranan yang penting dalam pembuatan tabel yang berisi *entity* dan relasinya. Hasil dari proses normalisasi adalah himpunan-himpunan data dalam bentuk normal (normal *form*).

Kegunaan normalisasi:

- Meminimalisasi pengulangan informasi.
- Memudahkan identifikasi *entity*/objek.

Tahap-tahap normalisasi adalah:

- Bentuk normal pertama (1NF)

  Yaitu suatu bentuk relasi jika dan hanya jika setiap atribut dari relasi tersebut hanya memiliki nilai tunggal dalam satu baris atau *record*.
- Bentuk normal kedua (2NF)

  Bentuk normal kedua adalah suatu bentuk relasi yang memenuhi syarat-syarat yaitu:

[6] Kristanto, Andi. (2003). Perancangan Sistem Informasi dan Aplikasinya. Yogyakarta : Penerbit Gava Media. p.76.

a. Sudah memenuhi kriteria sebagai bentuk 1NF.
b. Setiap atribut yang bukan kunci utama tergantung secara fungsional terhadap semua atribut kunci dan bukan hanya sebagian atribut.

- Bentuk normal ketiga (3NF).

  Bentuk normal ketiga adalah suatu bentuk relasi yang memenuhi syarat-syarat yaitu:

  a. Relasi antar *file* sudah merupakan bentuk normal kedua
  b. Setiap atribut bukan kunci tidak tergantung secara fungsional kepada atribut bukan kunci yang lain dalam relasi tersebut.

2.7. PHP

PHP diciptakan oleh Rasmus Lerdorf, yang pada awalnya digunakan untuk mencatat jumlah pengunjung pada *web site*-nya. Secara resmi PHP merupakan kependekan dari PHP:*Hypertext Preprocessor*, merupakan bahasa pemrograman *server-side scripting* yang disisipkan pada HTML untuk membuat situs dinamis (Weiling and Thomson, 2003). Ini berarti bahwa data akan diekstrak dari *database* dan sebuah halaman akan diproses terlebih dahulu sebelum dikirim ke *client* (ditampilkan di halaman *web*). Dengan menggunakan PHP maka *maintenance* sebuah situs *web* menjadi lebih mudah. Nama file PHP diberi *ekstension* php.
Kelebihan PHP jika dibandingkan dengan ASP (*Active Page Server*), JSP (*Java Server Page*) dan *Allaire Coldfusion,* yaitu:

- PHP bersifat *open source*, *source code* PHP dapat diakses dan di-*edit* dengan gratis (Weiling, Thomson, 2003).
- PHP bersifat *multiplatform* yaitu dapat berjalan dalam *web server* yang berbeda dan dalam sistem operasi yang berbeda pula. PHP dapat berjalan di sistem operasi UNIX, Windows 98, Windows NT dan Macintosh.
- Mudah untuk mempelajari PHP karena sintaks yang dipakai berdasarkan bahasa pemrograman yang ada khususnya C dan Perl.
- PHP dapat di-*download* dengan gratis. Situs *web* untuk *download* PHP: http://www.php.net/downloads.php.

- PHP mempunyai *built in libraries* yang didesain untuk mendukung pembuatan web. Adapun fasilitas tersebut adalah untuk menghasilkan gambar GIF, koneksi ke *network services* yang lain, bekerja dengan *cookies* dan menghasilkan dokumen PDF, dan fasilitas *send email*.
- PHP mempunyai hubungan-hubungan yang alami ke banyak sistem *database*, seperti dBase, MySQL, MSQL, Sybase, dan masih banyak yang lain.
- *Error Handling* (www.php.net).

  Seluruh PHP3 *expression* dapat dipanggil dengan *prefix "@"*, yang akan mematikan *error reporting* untuk *expression* itu. Jika terjadi *error* pada *expression* itu, dan fitur *track error* di-*enabled*, maka dapat ditemukan *error message* pada variabel global $php_errormsg.
- *Script* asli PHP tidak dapat dilihat oleh *browser* sehingga keamanan lebih terjamin.
- *Server side*, artinya tidak diperlukan adanya kompatibilitas atau harus menggunakan browser tertentu. *Server* yang akan mengerjakan *script* tersebut dan hasil yang akan dikirimkan kembali ke *browser* biasanya dalam bentuk teks/gambar sehingga dapat dikenali oleh *browser* apapun.

2.7.1. Sintaks PHP

Script PHP diawali dengan *tag* <? dan diakhiri dengan *tag* ?>. Setiap statement atau perintah harus diakhiri dengan tanda titik koma, dan pada umumnya satu *statement* dituliskan dalam satu baris.

Contoh penulisan script PHP:

```
<HTML>
        <HEAD>
                <TITLE> Contoh penulisan script PHP </TITLE>
        </HEAD>
        <BODY>
                <?
                   Echo "Contoh penulisan script PHP";
                ?>

        </BODY>
</HTML>
```

2.7.2. Struktur Kontrol PHP

2.7.2.1. Struktur Kontrol *if…else…*

Struktur kontrol *if…else…* merupakan struktur kontrol pencabangan yang digunakan untuk menjalankan satu atau lebih perintah yang menyatakan keadaan. Perintah dalam blok *if* akan dikerjakan apabila nilai dari ekspresi di dalam *if* bernilai *true*, jika ekspresi bernilai *false* maka akan menjalankan perintah dalam blok *else*.

Standar penulisannya adalah:

```
if (kondisi )
{       pernyataan 1;
        pernyataan2;
}
else
{       pernyataan a;
        pernyataan b;
}
```

2.7.2.2. Struktur Kontrol *switch…case…*

Struktur kontrol *switch…case…* merupakan struktur kontrol pencabangan yang digunakan untuk mengevaluasi suatu ekspresi dengan kemungkinan banyak nilai dan banyak perintah yang harus dieksekusi berdasarkan nilai dan ekspresi tersebut. Perintah ini digunakan sebagai alternative pengganti dari struktur kontrol *if…else…*

Standar penulisannya adalah:

```
 switch (kondisi)
{       case konstanta1:
              pernyataan 1;
              break;
        case konstanta2:
              pernyataan 2;
             break;
        default:
               pernyataan default;
}
```

2.7.2.3. Struktur Kontrol *while…*

Struktur kontrol *while*…merupakan struktur kontrol pengulangan yang digunakan untuk mengulangi sebuah perintah sampai jumlah pengulangan yang ditentukan oleh nilai dari suatu ekspresi.

Perintah dalam blok *while* akan dikerjakan apabila ekspresi bernilai *true*, dalam blok perintah ini harus ada perubahan nilai sehingga ekspresi yang diperiksa dapat bernilai *false*.

Standar penulisannya adalah:

```
while (kondisi)
{       pernyataan yang akan dijalankan }
```

2.7.2.4. Struktur Kontrol *for*…

Struktur Kontrol *for*…merupakan struktur kontrol pengulangan yang digunakan untuk mengulangi perintah dengan jumlah pengulangan yang sudah diketahui.

Standar penulisannya adalah:

```
for ($c=nilai_awal; $c<=batas_akhir; c++)
{       pernyataan1;
        pernyataan2;
}
```

2.8. MySQL

My SQL adalah *database* yang cepat dan *reliable* yang menggabungkan dengan baik dengan PHP dan cocok untuk aplikasi yang berbasis *internet* yang *dynamic* (Welling and Thomson, 2003). MySQL merupakan *software database* yang menggunakan SQL (*Structured Query Language),* yang merupakan bahasa standar pemrograman *database*. Dengan menggunakan SQL, proses pengaksesan *database* menjadi lebih *user friendly* karena mirip dengan bahasa Inggris standar. MySQL dapat dijalankan pada personal komputer dan juga pada sistem operasi komersial seperti Solaris, Windows, IRIX, MySQL dapat menangani *database* yang besar dengan jutaan *record*.

Kelebihan-kelebihan MySQL adalah:

- *Security*

  MySQL memiliki beberapa lapisan sekuritas pada level *subnetmask,* nama *host* dan ijin akses *user* dengan sistem perijinan yang mendetail serta *password* terinkripsi.
- *Multiuser*

MySQL dapat digunakan untuk beberapa *user* dalam jangka waktu bersamaan tanpa mengalami masalah.

- *High performance*

  Para *developer* berpendapat bahwa MySQL merupakan *database* tercepat dan sedikit terjadi *crash* (jarang bermasalah).
- *Portability*

  MySQL dapat digunakan pada banyak sistem UNIX yang berbeda-beda sebaik menggunakan Microsoft Windows.
- *Source code* tersedia gratis sehingga dapat diperoleh dan dimodifikasi.

2.8.1. Koneksi PHP dengan MySQL

Perintah-perintah yang digunakan untuk menghubungkan antara bahasa pemrograman PHP dengan MySQL, yaitu:

1. Pembuatan koneksi antara *server* dari MySQL dengan *web server* tempat menyimpan halaman *web*.

   <? mysql_connect (“nama server mysql”, login, password); ?>
2. Setelah terbentuk koneksi, kemudian dilakukan pemilihan *database* yang digunakan.

   <? mysql_select_db (“nama database”); ?>
3. Jika belum pernah dibentuk suatu *database* maka harus dibuat database yang baru.

   <? mysql_create_db (“nama database”); ?>
4. Setelah itu, perintah-perintah MySQL yang lain seperti *select, update, delete, query,* dan lain sebagainya dapat dilakukan.

   <? mysql_query (“perintah query”); ?>

2.8.2. Tipe-tipe tabel dalam MySQL

Dalam MySQL ada lima tipe tabel, yaitu:

- ISAM

Merupakan tipe *not transaction-safe tables.* Menggunakan *B-Tree index*, dan indeks disimpan dalam *file* dengan ekstensi .ISM dan data disimpan dalam *file* dengan ekstensi .ISD. Fitur-fitur yang terdapat pada ISAM anatara lain:

- *Compressed* dan panjang *key* yang pasti.
- Panjang *record* dinamis dan pasti.
- 16 *keys* dengan 16 *key* / bagian *key*.
- Panjang maksimum *key* 256 (*default).*
- Data disimpan dalam format OS yang dependen.

- HEAP

  Merupakan tipe *not transaction-safe tables.* Menggunakan *hashed indexes* dan disimpan dalam memori. Ini menyebabkan HEAP lebih cepat namun jika terjadi *crash* pada MySQL maka semua data akan hilang, sehingga HEAP berguna untuk tabel sementara.

- MyISAM

  Merupakan tipe *not transaction-safe tables* dan merupakan *default* tabel yang digunakan pada MySQL. Indeks disimpan dalam *file* dengan ekstensi .MYI dan data disimpan dalam *file* dengan ekstensi .MYD. MyISAM berdasarkan pada kode-kode ISAM dan memiliki banyak ekstensi yang berguna. MyISAM mendukung beberapa hal, antara lain:

  - Mendukung tipe *varchar* dengan panjang *record* yang pasti atau dinamis.
  - Terdapat *flag* yang mengidentifikasikan apakah tabel sudah ditutup dengan baik. Apabila tabel tidak tertutup dengan baik maka dilakukan perbaikan penutupan tabel.
  - Semua data disimpan dengan *byte* terendah.
  - *Internal handling* terhadap auto_increment.

  Dengan menggunakan MyISAM, *file* indeks lebih kecil dibandingkan dengan menggunakan ISAM, yang berarti lebih sedikit menggunakan *system resources*, tetapi memerlukan CPU *time* yang lebih besar ketika meng-*insert*-kan data ke dalam *compressed index*.

- InnoDB

Merupakan tipe *transaction-safe tables* yang memiliki kemampuan *commit, rollback,* dan *crash recovery,* dan merupakan *database backend* yang lengkap yang memiliki *buffer* sendiri untuk menyimpan data dan indeks dalam *main memory*. Didesain untuk *performance* yang maksimum ketika memproses data yang besar.

- BDB atau BerkeleyDB

  Merupakan tipe *transaction-safe tables* yang memiliki kemampuan *commit* dan *rollback* pada *transaction.*