# BAB III

# LANDASAN TEORI

## 3.1 Sistem Informasi

Menurut Robert A. Leitch dan K. Roscoe Davis (1983) sistem informasi merupakan suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat menajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan–laporan yang diperlukan untuk proses pengambilan keputusan. Jadi dapat ditarik kesimpulan bahwa sistem informasi adalah suatu sistem yang terintegrasi dengan berbagai elemen pendukungnya untuk menyediakan suatu informasi dari data-data yang ada bagi penggunanya.

### 3.1.1 Sistem

Sistem dapat didefinisikan sebagai kumpulan komponen yang saling terkait yang bekerjasama untuk mencapai tujuan bersama. Fungsi sistem adalah untuk menerima masukan menjadi output (Bojic, 2008).

### 3.1.2 Informasi dan Data

Data adalah sebuah kebenaran, atau kenyataan, contoh nama pegawai pesanan penjualan, nomor penjualan. (Stair dan George, 2006). Informasi adalah sekumpulan kebenaran atau kenyataan yang terorganisir sedimikian rupa yang menyebabkan mereka memilki nilai tambah dari pada kumpulan kebenaran itu sendiri. (Stair dan George, 2006).

### 3.2 *System Development Life Cycle* (SDLC)

SDLC (*System Development Life Cycle*) menurut Kendall & Kendall (2006) adalah fase-fase pendekatan yang dilakukan untuk menganalisis dan merancang sistem informasi. Kegiatan-kegiatan yang dilakukan pada SDLC antara lain:

1. Mengidentifikasi Masalah, Kesempatan, dan Tujuan

   Pada kegiatan ini, analis sistem harus mengetahui masalah apa saja yang terjadi, kesempatan, dan tujuan pembuatan sistem tersebut. Mengidentifikasi tujuan merupakan salah satu komponen yang penting karena analis sistem harus mengetahui apa yang ingin dicapai oleh perusahaan tersebut. Selain itu, analis sistem juga perlu mengetahui bila ada aspek-aspek aplikasi sistem informasi yang dapat digunakan untuk membantu mencapai tujuan perusahaan dengan merumuskan masalah yang spesifik atau kesempatan. Keluaran dari tahap ini adalah laporan kelayakan yang berisikan definisi masalah dan rangkuman dari tujuan. Orang-orang yang terlibat dalam fase ini adalah pengguna, analis sistem dan manajer.

2. Menentukan Kebutuhan Informasi

   Fase berikutnya adalah menentukan informasi apa saja yang dibutuhkan. Cara-cara yang digunakan untuk mendapatkan informasi yang dibutuhkan adalah wawancara, membuat sampel dan menginvestigasi *hard data*, dan kuesioner. Orang-orang yang ikut serta dalam tahap ini adalah analis sistem, pengguna (manajer operasi dan pegawai). Analis sistem harus mengetahui detail dari sistem yang telah berjalan saat ini, siapa saja orang yang terlibat (*who*), apa jenis aktivitas bisnisnya (*what*), dimana pekerjaan ini akan dilakukan (*where*),

waktunya (*timing*), dan bagaimana prosedur yang sekarang dijalankan (*how*). Inti dalam fase ini adalah analis sistem harus dapat mengetahui bagaimana fungsi bisnis yang telah berjalan dan mempunyai informasi yang lengkap atas orang, tujuan, dan prosedur yang bersangkutan.

3. Menganalisis Kebutuhan Sistem

   Dalam fase ini, analis sistem harus mengetahui kebutuhan sistem yang akan dibuat. Alat yang digunakan dalam fase ini adalah *data flow diagram* untuk menggambarkan masukan, proses dan keluaran fungsi bisnis dalam bentuk grafis yang terstruktur. Dari *data flow diagram*, dapat dibuat *data dictionary* yang berisikan daftar-daftar data yang akan digunakan dalam sistem dan spesifikasinya.

4. Mendesain Sistem yang Direkomendasikan

   Dalam fase ini, analis sistem menggunakan informasi yang telah dikumpulkan sebelumnya untuk membuat desain sistem informasi. Bagian desain sistem informasi ini adalah pembuatan *user interface*, pembuatan sistem basis data/*database system*, perancangan desain hasil keluaran dari sistem. Terakhir analis sistem harus mendesain pengendalian dan prosedur *backup* untuk melindungi sistem dan data.

5. Pengembangan dan Pendokumentasian Piranti Lunak

   Pada fase kelima ini, analis sistem bekerja dengan *programmer* untuk membuat piranti lunak yang dibutuhkan. Pada fase ini, analis sistem juga harus bekerja dengan *user* untuk mengembangkan pendokumentasian *software* efektif yang meliputi prosedur manual, buatan langsung melalui internet, situs *web*. Dokumentasi memberikan informasi kepada pengguna tentang

bagaimana cara menggunakan piranti lunak tersebut dan apa yang harus dilakukan apabila piranti lunak tersebut mengalami masalah.

6. Testing dan Perawatan Sistem

   Sebelum sistem dapat digunakan, sistem yang telah dibuat harus dites terlebih dahulu. Hal ini dilakukan untuk mengurangi biaya yang akan dikeluarkan sebelum sistem tersebut diimplementasikan kepada pengguna. Perawatan kepada sistem dimulai dari fase ini. Perawatan sistem bisa berupa *update program* yang bisa dilakukan melalui *web*.

7. Implementasi dan Evaluasi Sistem

   Fase terakhir dari pengembangan sistem. Analis sistem harus membantu mengimplementasikan sistem tersebut. Fase ini meliputi pelatihan *user* untuk bisa menggunakan sistem. Selain itu, analis sistem harus melakukan migrasi dari sistem yang lama ke sistem yang baru. Hal ini meliputi data dari format yang lama ke format yang baru, pembuatan sistem basis data memenuhi kebutuhan sistem.

### 3.3. Aplikasi Berbasis Web

Pada awalnya aplikasi *web* dibangun dengan hanya menggunakan bahasa yang disebut *Hypertext Markup Language* (HTML). Pada perkembangan berikutnya, sejumlah skrip dan objek dikembangkan untuk memperluas kemampuan HTML seperti PHP dan ASP pada skrip dan *Applet* pada objek. Aplikasi *Web* dapat dibagi enjadi dua jenis yaitu aplikasi *web* statis dan dinamis. Pengertian aplikasi berbasis *web* adalah “Aplikasi sisi *server* (*server side*) yang menggunakan standar HTTP dan menggnakan browser untuk menggunakan

aplikasi. Termasuk didalamnya teknologi PHP, ASP dan lainnya" (Jogiyanto, 2003).

Arsitektur aplikasi *web* meliputi klien, *web server*, *middleware* dan basis data. Klien berinteraksi dengan *web server*. Secara internal, *web server* berkomunikasi dengan *middleware* dan *middleware* yang berkomunikasi dengan basis data. Contoh *middleware* adalah PHP dan ASP. Pada mekanisme aplikasi *web* dinamis, terjadi tambahan proses yaitu *server* menerjemahkan kode PHP menjadi kode HTML. Kode PHP yang diterjemahkan oleh mesin PHP yang akan diterima oleh klien (Herlambang, 2005).

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

Menurut Sutarman (2003), HTML (*Hypertext Markup Languange*) adalah suatu bahasa yang digunakan untuk menulis halaman *web*, HTML dirancang untuk digunakan tanpa bergantung pada suatu *platform* tertentu. Dokumen HTML adalah suatu dokumen teks biasa, dan disebut sebagai *markup language* karena mengandung tanda-tanda (*tag*) tertentu yang digunakan untuk menentukan tampilan suatu teks dan tingkat kepentingan dari teks tersebut dalam suatu dokumen.

HTML (*Hypertext Markup Languange*) adalah bahasa yang digunakan untuk menulis halaman *web*. Ciri utama dokumen HTML adalah adanya *tag* dan elemen. Elemen dalam dokumen HTML dikategorikan menjadi dua yaitu elemen <*HEAD*> yang berfungsi memberikan informasi tentang dokumen tersebut dan elemen <*BODY*> yang menentukan bagaimana isi suatu dokumen ditampilkan oleh *browser*, seperti paragraf, *list* (daftar), tabel dan lain-lain. Sedangkan *tag* dinyatakan dengan tanda lebih kecil " < " ( *tag* awal ) " > " ( *tag* akhir ).

Dokumen HTML mempunyai tiga buah *tag* utama yang membentuk struktur dari dokumen HTML yaitu HTML, *HEAD*, dan *BODY*. *Tag* HTML digunakan untuk menyatakan dokumen HTML, *tag HEAD* berfungsi untuk memberikan informasi tentang dokumen HTML dan *tag BODY* berfungsi untuk menyimpan informasi atau data yang akan ditampilkan dalam dokumen HTML.

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

Menurut Saputra & Agustin (2011), CSS atau yang memiliki kepanjangan *Cascading Style Sheet* merupakan suatu bahasa pemrograman *web* yang digunakan untuk mengendalikan dan membangun berbagai komponen dalam *web* sehingga tampilan *web* akan lebih rapi, terstruktur, dan seragam.

CSS merupakan pemrograman wajib yang harus dikuasai oleh setiap pembuat program *(Web Programmer)*, terlebih lagi itu adalah pendesain *web (web designer)*.

Ada dua sifat CSS, yaitu internal dan eksternal.

1. Internal, Jika kode CSS yang akan dibuat tersebut dimasukkan atau disisipkan ke dalam file kode HTML.
2. Eksternal, pembuatan kode CSS dan HTML terpisah. Artinya, kita membuatkan satu file CSS untuk kemudian file CSS tersebut dapat dipanggil berulang-ulang guna dihubungkan dengan file HTML (melalui *Linked*).

### 3.6 PHP

PHP merupakan singkatan dari *Hypertext Preprocessor*, yakni instruksi atau perintah pemrograman berbasis *web* yang biasa disisipkan dalam dokumen HTML, sebagai skrip pendukung yang ada di lingkungan server (*server side* HTML

*embedded scripting*) (Musyawarah, 2005). Pada dasarnya PHP dapat mengerjakan semua yang dikerjakan oleh program CGI (*Common Gateway Interface*), seperti menyimpan data yang dimasukkan melalui sebuah *form* dalam sebuah *website*, menampilkan isi *website* yang dinamis, serta menerima *cookies.*

Menurut Kadir (2008), PHP dirancang untuk membentuk aplikasi *web* dinamis. Artinya, ia dapat membentuk suatu penampilan berdasarkan permintaan terkini. Misalnya, bisa menampilkan *database* ke halaman *web*. Pada prinsip PHP mempunyai fungsi yang sama dengan skrip-skrip seperti *Active Server Page* (ASP), *Cold Fusion*, atau *pearl*. Namun, perlu diketahui bahwa PHP sebenarnya bisa dipakai secara *command line.* Artinya, Skrip PHP dapat dijalankan tanpa melibatkan *web server* maupun *browser*.

Selain itu, kemampuan PHP yang paling menonjol adalah dukungan ke banyak ke *database.* Karena PHP bersifat *server side*, maka untuk dapat menjalankan PHP pada browser, maka terlebih dahulu meng-*install* Apache, PHP Triad, PWS, Wammp, Xampp, dan sebagainya. Beberapa *database* yang dapat diakses melalui skrip PHP yaitu dBase, FilePro, MySql, ODBC, Oracle, Postgres, Sybase, Velocis.

### 3.7 Basis Data (*Database*)

Menurut Chendramata (2009), *database* adalah sebuah perangkat lunak yang dirancang dan diperuntukkan sebagai media untuk menyimpan data-data transaksi yang dihasilkan pada sebuah proses bisnis. *Database* minimal terdiri dari satu file yang cukup untuk dimanipulasi oleh komputer sedemikian rupa.

Sedangkan menurut Agung Nugroho (2005). *Database* adalah sebuah bentuk media yang digunakan untuk menyimpan sebuah data. *Database* dapat

diilustrasikan sebagai rumah atau gudang yang akan dijadikan tempat menyimpan berbagai macam barang. Dalam *database*, barang tersebut adalah data. Dari kedua pengertian tersebut dapat disimpulkan bahwa basis data/*database* merupakan perangkat lunak yang digunakan untuk menyimpan data.

### 3.8 MySQL

MySQL adalah *database server* relasional yang gratis di bawah lisensi *General Public License.* Dengan sifatnya yang *open source*, memungkinkan *user* untuk melakukan modifikasi pada *source code* untuk memenuhi kebutuhan spesifik mereka sendiri. MySQL merupakan *database server multi-user* dan *multi-threaded* yang tangguh (*robust*) yang memungkinkan *backend* yang berbeda, sejumlah *program client* dan *library* yang berbeda, *tool administrative*, dan beberapa antarmuka pemrograman. MySQL juga tersedia sebagai *library* yang bisa digabungkan ke aplikasi. Dengan memiliki banyak fitur, MySQL bisa bersaing dengan *database* komersial sekalipun (Utdirartatmo, 2002:1).

Menurut Prasetyo (2003:3) mengemukakan beberapa keistimewaan yang dimiliki oleh MySQL, diantaranya adalah:

1. *Portability*: dapat berjalan stabil pada berbagai sistem operasi (Windows, Linux, Mac OS dan lain-lain).
2. *Open Source*: didistribusikan secara gratis, di bawah lisensi GPL sehingga dapat digunakan secara cuma-cuma tanpa dipungut biaya.
3. *Multi-user*: dapat digunakan oleh beberapa *user* dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.
4. *Performance Tuning*: memiliki kecepatan yang menakjubkan dalam menangani *query* sederhana.

5. *Column Types*: memiliki tipe kolom yang sangat kompleks, seperti *signed/unsigned integer, float, double, char, varchar, text, blob, date, time, datetime, timestamp, year, set,* serta *enum.*
6. *Commands and Functions*: memiliki *operator* dan fungsi secara penuh yang mendukung perintah *SELECT* dan *WHERE* dalam *query.*
7. *Security:* memiliki beberapa lapisan sekuritas seperti *level subnetmark,* nama *host,* dan izin akses *user* dengan sistem perizinan yang mendetil serta *password* terenkripsi.
8. *Scalability* dan *Limit*: mampu menangani *database* dalam skala besar, dengan jumlah *records* lebih dari 50 juta dan 60 ribu tabel serta 5 miliar baris. Selain itu, batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.
9. *Connectivity*: dapat melakukan koneksi dengan *client* menggunakan *protocol* TCP/IP, Unix *socket* (Unix), atau *Named Pipes*.
10. *Localization*: dapat mendeteksi pesan kesalahan (*error code*) pada *client* dengan mengunakan lebih dari dua puluh bahasa.
11. *Interface*: memiliki *interface* terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (*Application Programming Interface*).
12. *Lients and Tools*: dilengkapi dengan berbagai *tool* atau perangkat yang dapat digunakan untuk administrasi *database,* dan pada setiap *tool* yang ada juga disertakan petunjuk *online*.

13. Struktur Tabel: memiliki struktur tabel yang lebih fleksibel dalam menangani *ALTER TABLE*, dibandingkan *database* lainnya semacam PostgreSQL dan Oracle.

### 3.9 Alur Sistem (*System Flow)*

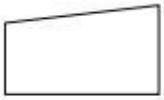
*System flow* adalah suatu bagan yang menunjukkan arus pekerjaan secara menyeluruh dari suatu sistem dimana bagian ini menjelaskan urutan prosedur yang ada di dalam sistem dan biasanya dalam membuat *system flow* sebaiknya ditentukan pada fungsi-fungsi yang melaksanakan atau bertanggungjawab terhadap sub gambar sistem (Jogiyanto, 2003).

### 3.10 Diagram Alir Dokumen (*Document Flowchart*)

Menurut Jogiyanto (2005), diagram alir dokumen atau *paperwork flowchart* merupakan diagram alir yang menunjukkan arus laporan dan formulir beserta tembusannya. Berdasarkan pengertian di atas dapat disimpulkan bahwa diagram alir dokumen adalah diagram yang menggambarkan aliran seluruh dokumen. Diagram alir dokumen ini menggunakan simbol-simbol yang sama dengan diagram alir sistem. *Document flowchart* digambar dengan menggunakan simbol-simbol diagram alir dokumen yang dijelaskan pada tabel 3.1.

Table 3.1. Simbol - simbol Diagram Alir Dokumen

| No | Nama Simbol | Simbol | Fungsi |
|---|---|---|---|
| 1. | **Terminator** |  | Simbol ini digunakan untuk menunjukkan awal dan akhir suatu proses dokumen. |

| | | | |
|---|---|---|---|
| 2. | **Document** |  | Simbol ini digunakan sebagai *input* dan *output* baik secara manual ataupun dengan menggunakan komputer. |
| 3. | **Manual Input** |  | Simbol ini berfungsi untuk memasukkan data dengan menggunakan *online keyboard*. |
| 4. | **Manual Process** |  | Simbol ini menunjukkan kegiatan manual. |
| 5. | ***Offline Storage*** |  | Simbol ini merupakan dokumen yang diarsip dan diurutkan bedasarkan N (*numeric*), A (*alphabet*), C (*chronological*) |
| 6. | ***Flow*** |  | Simbol ini digunakan sebagai arah aliran dokumen. |

### 3.11 Diagram Alir Sistem (*System Flowchart*)

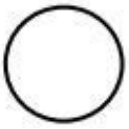
Diagram alir sistem merupakan diagram alir yang menggambarkan suatu sistem peralatan komputer yang digunakan untuk mengolah data dan menghubungkan antar peralatan tersebut (Oetomo, 2002). Diagram alir sistem ini tidak digunakan untuk menggambarkan langkah-langkah dalam memecahkan masalah tetapi hanya menggambarkan prosedur pada sistem yang dibentuk.

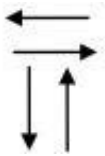
Simbol-simbol diagram alir sistem dihubungkan oleh simbol penghubung yang disebut dengan *flow direction symbols* yang dijelaskan pada poin berikut:

1. *Flow Direction Symbols*

*Flow direction symbols* digunakan untuk menghubungkan antara satu simbol dengan simbol lainnya (Ladjamudin, 2005). Simbol ini disebut *connecting line*. Simbol-simbol tersebut dijelaskan pada tabel di 3.2.

Table 3.2. Simbol - simbol Flow Direction

| No | Nama Simbol | Simbol | Fungsi |
|---|---|---|---|
| 1. | ***Offline Connector*** |  | Fungsi dari simbol ini adalah menyambungkan antara suatu proses dengan proses lainnya di halaman yang berbeda. |
| 2. | ***Connector*** |  | Fungsi dari simbol ini adalah menyambungkan antara, suatu proses dengan proses lainnya di halaman yang sama. |

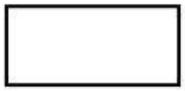
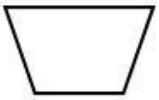
| | | | |
|---|---|---|---|
| 3. | ***Communication Link*** |  | Fungsi dari simbol ini adalah mentransisi suatu data atau informasi dari setiap lokasi. |
| 4. | ***Flow*** |  | Fungsi dari simbol ini adalah menyatakan jalannya arus suatu proses. |

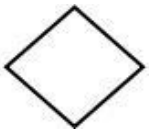
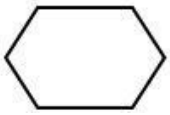
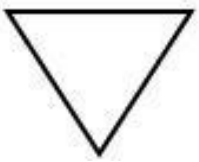
Kemudian terdapat simbol untuk proses dalam diagram alir sistem/*system flowchart* yang disebut dengan *processing symbols.*

**2. *Processing Symbols***

*Processing symbols* merupakan simbol yang menunjukkan jenis operasi pengolahan data dalam suatu proses (Ladjamudin, 2005). Simbol-simbol tersebut dijelaskan pada tabel 3.3.

**Table 3.3. Simbol - Simbol *Processing***

| No. | Nama Simbol | Simbol | Fungsi |
|---|---|---|---|
| 1. | ***Offline Conector*** |  | Simbol ini berfungsi untuk menyambungkan satu proses dengan proses lainnya di halaman yang berbeda. |
| 2. | ***Manual Process*** |  | Simbol ini berfungsi untuk melakukan prosedur atau proses tanpa menggunakan |

| | | | |
|---|---|---|---|
| | | | komputer. |
| 3. | ***Decision*** |  | Simbol ini berfungsi untuk melakukan pengecekan. Biasanya menghasilkan jawaban ya atau tidak. |
| 4. | ***Predefined Process*** |  | Simbol ini berfungsi sebagai tempat penyimpanan nilai awal. |
| 5. | ***Terminal*** |  | Simbol ini berfungsi untuk menyatakan permulaan atau penghentian suatu program. |
| 6. | ***Key Operation*** |  | Simbol ini berfungsi untuk menyatakan suatu jenis operasi yang diproses dengan menggunakan mesin yang memiliki *keyboard*. |
| 7. | ***Offline Storage*** |  | Simbol ini digunakan untuk menyimpan data ke suatu media tertentu. |

| | | | |
|---|---|---|---|
| 8. | ***Manual Input*** |  | Simbol ini berfungsi untuk memasukkan data dengan menggunakan *online keyboard*. |

### 3.12. *Data Flow Diagram* (DFD)

*Data Flow Diagram* (DFD) merupakan alat yang digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir ataupun lingkungan fisik dimana data tersebut akan disimpan (Jogiyanto, 2009).

Sedangkan menurut Whitten (2004), *Data Flow Diagram* (DFD) merupakan alat yang menggambarkan aliran data melalui sistem. Dalam pembuatan DFD, terdapat beberapa tingkatan yang bertujuan untuk menghindari aliran data yang rumit. Tingkatan tersebut dimulai dari tingkatan tertinggi ke bentuk yang lebih rinci. Tingkatan DFD terdiri atas:

1. **Diagram Konteks (*Context Diagram*)**

   Diagram konteks merupakan sebuah model proses yang digunakan untuk mendokumentasikan ruang lingkup dari sebuah sistem (Whitten, 2004).

2. **Diagram Rinci**

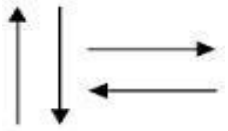
   Diagram rinci menggambarkan rincian dari proses yang ada pada tingkatan sebelumnya. Diagram ini merupakan diagram dengan tingkatan paling rendah dan tidak dapat diuraikan lagi.

### 3. Diagram Level 0

Diagram level 0 merupakan diagram aliran data yang menggambarkan sebuah *event* konteks. Diagram ini menunjukkan interaksi antara *input*, output, dan *data store* pada setiap proses yang ada.

**Table 3.4. Simbol - Simbol DFD**

| Nama Simbol | Simbol | Keterangan |
|---|---|---|
| ***External Entity*** |  | *External entity* merupakan kesatuan di lingkungan luar sistem yang dapat berupa orang, organisasi, atau sistem lainnya yang akan memberikan *input* ataupun menerima *output*. |
| ***Process*** |  | Proses adalah kegiatan yang dilakukan oleh orang atau komputer dari arus data yang masuk untuk menghasilkan arus data yang keluar. |
| ***Data Store*** |  | *Data store* merupakan tempat penyimpanan data yang berupa *file* maupun *database* di dalam sistem komputer. |

| | | |
|---|---|---|
| ***Data Flow*** |  | *Data flow* atau aliran data yang mengalir diantara proses. Aliran data dapat digambarkan dari bawah ke atas, kiri ke kanan, maupun sebaliknya. |

### 3.13. *Entity Relationship Diagram* (ERD)

Pengertian *Entity Relation Diagram* (ERD) menurut Jogiyanto (2001) adalah suatu komponen himpunan entitas dan relasi yang dilengkapi dengan atribut yang mempresentasikan seluruh fakta. ERD digunakan untuk menggambarkan model hubungan data dalam sistem yang di dalamnya terdapat hubungan entitas berserta atribut relasinya serta mendokumentasikan kebutuhan sistem untuk pemrosesan data. ERD memiliki 4 jenis objek, antara lain:

1. *Entity*

Menurut Connolly dan Begg (Whitten, 2004), Entitas adalah kelompok orang, tempat, objek, kejadian atau konsep tentang apa yang diperlukan untuk menyimpan data. Setiap entitas yang dibuat memiliki tipe untuk mengidentifikasi apakah entitas tersebut bergantung dengan entitas lainnya atau tidak. Tipe entitas merupakan kumpulan objek yang memiliki kesamaan properti yang teridentifikasi oleh perusahaan dan memiliki keberadaan yang independen. Tipe entitas terdiri atas dua jenis, yaitu:

a. *Strong Entity*

*Strong entity* adalah tipe entitas yang tidak bergantung pada keberadaan jenis entitas lainnya. Suatu entitas dikatakan kuat apabila tidak tergantung pada entitas lainnya.

b. *Weak Entity*

*Weak Entity* adalah tipe entitas yang bergantung pada keberadaan jenis entitas lain yang saling berhubungan. Karakteristik *weak entity* terletak pada entitas *occurrence* yang tidak dapat teridentifikasi secara unik. Entitas *occurrence* adalah sebuah objek yang secara unik dapat teridentifikasi dengan tipe entitas

2. *Attribute*

Menurut Connolly dan Carolyn (2002) atribut adalah deskripsi data yang mengidentifikasikan dan membedakan suatu entitas dengan entitas lainnya. Setiap atribut memiliki *domain* untuk mendefinisikan nilai-nilai potensial yang dapat menguatkan atribut. *Atrribute domain* adalah kumpulan nilai-nilai yang diperbolehkan untuk satu atau lebih atribut.

Atribut dapat dibedakan menjadi 5 jenis, yaitu:

a. *Simple Attribute*

*Simple Attribute* adalah atribut yang terdiri dari komponen tunggal. *Simple attribute* tidak dapat dibagi menjadi komponen yang lebih kecil.

b. *Composite Attribute*

*Composite Attribute* adalah atribut yang terdiri dari beberapa komponen yang bersifat independen.

c. *Single-value Attribute*

*Single-value Attribute* adalah atribut yang memegang nilai tunggal dari suatu entitas.

d. *Multi-value Attribute*

*Multi-value Attribute* adalah atribut yang dapat memegang nilai lebih dari suatu entitas.

e. *Derived Attribute*

*Derived Attribute* adalah atribut yang mewakili turunan nilai sebuah atribut yang saling berkaitan dan belum tentu dalam tipe entitas yang sama.

3. *Keys*

Menurut Connolly dan Carolyn (2002) *keys* terdiri atas beberapa jenis, yaitu:

a. *Candidate Key*

*Candidate key* merupakan *set* minimal dari suatu atribut yang secara unik mengidentifikasi setiap *occurrence* dari tipe entitas. *Candidate key* tidak boleh *null* (kosong).

b. *Primary Key*

Sebuah *candidate key* yang dipilih untuk mengidentifikasi secara unik tiap kejadian pada suatu entitas. *Primary key* harus bernilai *unique* dan tidak boleh *null* (kosong).

c. *Composite Key*

Sebuah *candidate key* yang mempunyai dua atribut atau lebih. Suatu atribut yang membentuk *composite key* bukanlah kunci sederhana karena *composite key* tidak membentuk kunci senyawa.

d. *Alternate Key*

Sebuah *candidate key* yang tidak menjadi *primary key*. *Key* ini biasa disebut dengan *secondary key*.

e. *Foreign Key*

Himpunan atribut dalam suatu relasi yang cocok dengan *candidate key* dari beberapa relasi lainnya. *Foreign key* mengacu pada *primary key* suatu tabel. Nilai *foreign key* harus sesuai dengan nilai *primary key* yang diacunya.

4. *Relationship*

Menurut Whitten (2004) *relationship* adalah asosiasi bisnis alami antara satu entitas atau lebih. Dalam suatu relasi, entitas yang saling berelasi memiliki kata kerja aktif yang menunjukan bahwa keduanya saling berelasi satu sama lain.

Relasi terdiri atas enam tipe, yaitu:

a. Relasi *one to many*

Relasi *one to many* berarti suatu entitas himpunan A dapat berhubungan dengan banyak entitas pada entitas himpunan B, namun tidak sebaliknya.

b. Relasi *one to one*

Relasi *one to one* berarti setiap entitas himpunan A hanya berhubungan dengan satu entitas himpunan B, begitu juga sebaliknya.

c. Relasi rekursif *one to one*

Relasi rekursif *one to one* adalah sebuah tipe relasi yang dimana entitasnya mempartisipasi lebih dari satu peran.

d. Relasi *superclass*/*subclass*

Untuk setiap relasi *superclass* / *subclass*, entitas *superclass* diidentifikasikan sebagai entitas induk dan entitas *subclass* sebagai anggotanya.

e. Relasi *many to many*

Relasi *many to many* berarti setiap entitas himpunan A dapat berhubungan dengan entitas pada himpunan B, begitu juga sebaliknya.

f. Relasi kompleks

Relasi kompleks adalah tipe relasi yang dimana satu entitas berhubungan dengan entitas lainnya yang dapat membentuk sirkulasi dalam relasi tersebut.

### 3.14. *Internet*

Menurut Febrian (2007), *internet* merupakan tempat terhubungnya berbagai mesin komputer yang mengolah informasi di dunia ini, baik berupa *server*, komputer pribadi, *handphone*, komputer genggam, PDA, dan lain sebagainya. Masing-masing mesin ini bekerja sesuai dengan fungsinya, baik sebagai penyedia layanan yang biasa disebut dengan *server* maupun sebagai pengguna layanan yang biasa disebut dengan *client.* Berbagai jenis komputer yang jumlahnya mencapai jutaan, terhubung melalui jaringan yang disebut dengan *internet* ini. Mereka terhubung baik melalui kabel, saluran telepon, saluran *handphone*, satelit, *fiber optic*, gelombang, listrik, cahaya, serta media apa saja yang mungkin dialiri oleh data.