BAB II
DASAR TEORI

2.1 Landasan Teori

Landasan Teori adalah hal-hal atau teori yang berkaitan dengan permasalahan dan ruang lingkup permasalahan sebagai salahsatu landasan dalam penyusunan laporan ini.

2.1.1 Sistem

Sistem dapat dikatakan sebagai sebuah rangkaian jaringan kerja dari berbagai elemen - elemen yang saling berhubungan guna untuk mencapai tujuan tertentu. Selain itu pengertian sistem menurut ahli yang lain adalah jaringan proses kerja yang saling terkait dan berkumpul guna untuk mencapai sebuah tujuan serta melakukan suatu kegiatan. Dari beberapa pernyataan di atas mengenai pengertian sistem maka dapat disimpulkan bahwa sistem adalah gabungan dari kumpulan elemen, komponen atau variabel yang saling berhubungan satu sama lainnya guna untuk mencapai suatu tujuan tertentu[2].

2.1.2 Reservasi

Reservasi adalah rencana persetujuan berbentuk pemesanan suatu produk dalam bentuk jasa serta barang yang pada waktu itu telah mencapai kesalingpahaman antara produsen dengan konsumen berkenaan dengan produk yang sudah disebutkan, namun belum diselesaikan oleh suatu persetujuan jual beli antara dua pihak. Kegiatan reservasi lazimnya dikenali dengan adanya aktivitas pertukaran informasi antara konsumen dan produsen supaya dapat mencapai kesepahaman mengenai sebuah produk[3].

2.1.3 Basis Data

Database atau basis data adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Perangkat lunak yang digunakan untuk mengelola dan memanggil kueri (*query*) basis data disebut sistem manajemen basis data (*database management system*, DBMS). Sistem basis data dipelajari dalam ilmu informasi[4].

Salah Satu kebutuhan basis data dalam sistem informasi yaitu memasukkan, menyimpan, mengambil kembali data dan membuat laporan berdasarkan data yang telah disimpan. Dikarenakan hal tersebut, perlu merancang tabel-tabel yang nantinya akan dirancang maka diperlukan sebuah pola pikir penyimpanan data yang nantinya jika dalam bentuk berupa baris-baris data (*record*) dimana setiap baris terdari beberapa kolom.

MySQL adalah *Relational Database Management System (RDBMS)* yang didistribusikan secara gratis dibawah lisensi *GPL (General Public License*). Dimana setiap orang bebas untuk menggunakannya, tapi tidak boleh dijadikan produk turunan yang bersifat *Closed Source* atau komersial. *MySQL* sebenarnya merupakan turunansalah satu konsep utama dalam database sejak lama, yaitu *SQL (Structur Query Language). SQL* adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis. Sebagai *database server*, *MySQL* dapat dikatakan lebih unggul dibanding *database server* lainnya dalam *query* data. Hal ini terbukti untuk *query* yang dilakukan oleh *single user*, kecepatan *query MySQL* bisa sepuluh kali lipat lebih cepat dari *Postgre SQL* dan lima kali lebih cepat dibanding *Interbase*[5].

Sebuah DBMS memiliki tiga komponen utama. Komponen-komponen ini adalah Bahasa Definisi Data (DDL), Bahasa Manipulasi Data dan Fasilitas *Query* (DML/SQL), dan perangkat lunak untuk mengontrol akses *Database* .

1) DDL (*Data Definition Language*) adalah bagian dari bahasa *query* database yang digunakan untuk mendefinisikan struktur dan skema *database*. DDL digunakan untuk membuat, mengubah, dan menghapus objek *database* seperti tabel, indeks, tampilan, dan konstrain. Perintah-perintah DDL memungkinkan pengguna untuk mendefinisikan entitas dan hubungan antara entitas dalam *database*[6]. Beberapa perintah DDL umum meliputi:
   a) *CREATE*: Digunakan untuk membuat objek database baru, seperti tabel, indeks, atau tampilan.
      Contoh : - *CREATE DATABASE* PENGELOLA;
      CREATE TABLE PENGELOLA (id_pengelola *INTEGER*(5), nm_pengelola *VARCHAR*(45), level *INTEGER*(5), username *VARCHAR*(45), password *VARCHAR*(45), *PRIMARY KEY*(id_pengelola));
   b) *ALTER*: Digunakan untuk mengubah struktur objek *database* yang sudah ada, misalnya menambahkan kolom baru ke dalam tabel.
      Contoh : *ALTER TABLE* pengelola *ADD* jenis_kelamin *varchar*(45);
   c) *DROP*: Digunakan untuk menghapus objek *database*, seperti menghapus tabel atau indeks dari database.
      Contoh : *DROP TABLE PENGELOLA*;

2) DML (*Data Manipulation Language*) adalah bagian dari bahasa *query database* yang digunakan untuk memanipulasi data dalam tabel. DML digunakan untuk menyisipkan (*insert*), memperbarui (*update*), menghapus (*delete*), dan mengambil (*select*) data dari tabel. DML

memungkinkan pengguna untuk memanipulasi data dengan mengubah nilai-nilai dalam tabel, mengambil *subset* data yang diinginkan, dan melakukan operasi lain terkait data. Beberapa perintah DML umum meliputi:

a) *INSERT*: Digunakan untuk menyisipkan data baru ke dalam tabel.
   Contoh: *INSERT INTO TABLE* PENGELOLA (id_pengelola, nm_pengelola, *level*, *username, password*)
   VALUES (‘10’,’Galuh’,’1’,’12345’,’12345’);
b) *UPDATE*: Digunakan untuk memperbarui data yang sudah ada dalam tabel.
   Contoh: UPDATE Pengelola
   SET Username = ‘112233’
   WHERE id_pengelola = ‘10’;
c) *DELETE*: Digunakan untuk menghapus data dari tabel.
   DELETE FROM PENGELOLA
   WHERE id_pengelola = ‘10’;
d) *SELECT*: Digunakan untuk mengambil data yang spesifik dari tabel menggunakan kriteria tertentu.
   SELECT nm_pengelola
   FROM Pengelola
   WHERE level > 2;

1) *Software Controller* (pengontrol perangkat lunak) dalam konteks basis data dapat merujuk pada komponen perangkat lunak yang bertanggung jawab untuk mengendalikan atau mengatur perilaku sistem basis data secara keseluruhan. *Software controller* dalam basis data dapat memiliki fungsi seperti:
   a) Manajemen koneksi: Mengelola dan mengontrol koneksi pengguna ke basis data.
   b) Manajemen transaksi: Memastikan integritas data dan konsistensi transaksi dengan mengelola komitmen (*commit*) dan pembatalan (*rollback*) transaksi.
   c) Pengelolaan keamanan: Mengendalikan akses pengguna ke objek-objek dalam basis data dan menerapkan kebijakan keamanan.
   d) Optimisasi kinerja: Mengoptimalkan kinerja basis data dengan melakukan tuning, caching, dan indeksasi data.
   e) Pemantauan dan pemeliharaan: Memantau aktivitas basis data, memantau performa, dan melakukan pemeliharaan rutin seperti backup dan pemulihan.

2.1.4 Pemrograman Berorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan projek yang berisi data dan operasi yang diperlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis [7].

Sistem berorientasi objek merupakan sebuah sistem yang dibangun dengan berdasarkan metode berorientasi objek adalah sebuah sistem yang komponennya dibungkus menjadi kelompok data dan fungsi. Setiap komponen dalam sistem tersebut dapat mewarisi atribut, sifat, komponen lainnya, dan dapat berinteraksi satu sama lain.

Pemrograman berorientasi objek berbeda dengan pemrograman prosedural yang hanya menggunakan satu halaman kebawah untuk mengerjakan banyak perintah atau statement. Penggunaan pemrograman berorientasi objek sangat benyak sekali, contoh *: java, php, perl, c#, cobol*, dan lainnya.

Objek dalam OOP adalah unit terkecil pemrograman yag masih memiliki data (sifat karakteristik) dan fungsi. Setiap object dapat menerima pesan, memproses data, mengirim, menyimpan, dan memanipulasi data. *Class* adalah wadah berisi pemodelan suatu objek, mendeskripsikan karakteristik dan fungsi objek tersebut. Karena class merupakan wadah yang akan digunakan untuk menciptakan objek tersebut, maka Class harus diciptakan terlebih dahulu.

OOP memberikan kemudahan dalam pembuatan sebuah program, keuntungan yang didapat apabila membuat program berorientasi objek atau *Object Oriented Programming* (OOP) antara lain :

1) *Reusability*, kode yang dibuat dapat digunakan kembali
2) *Extensibility* , pemrogram dapat membuat methode baru atau mengubah yang sudah ada sesuai yang diinginkan tanpa harus membuat kode dari awal
3) *Maintainability*, kode yang sudah dibuat lebih mudah untuk dikelola apabila aplikasi yang dibuat berskala besar yang memungkinkan adanya *error* dalam pengembangannya hal tersebut dapat diatasi dengan OOP karena pemrograman OOP sudah menggunakan konsep modularitas.

Terdapat beberapa cara untuk menentukan karakteristik dalam pendekatak berorientasi objek, tetapi secara umum mencakup empat hal, yaitu identifikasi, klasifikasi, *polymorphism* (polimorfisme) dan *inheritance* (pewarisan). Metodologi pengembanagn sistem berorientasu objek mempunyai tiga karakteristik utama, yaitu :

1) *Encapsulation*

   Encapsulation (pengkapsulan) merupakan dasar untuk pembahasan ruang lingkup program terhadap data yang diproses.

2) *Inheritance*

   Inheritance (Pewarisan) adalah tehnik yang menyatakan bahwa anak dari objek akan mewarisi data / atribut dan metode dari induknya langsung. Sifat yang dimiliki oleh kelas induknya tidak perlu diulang dalam setiap subkelasnya.

3) *Polymorphism*

   Polymorphism (polimorfisme) yaitu konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku berbeda.

2.1.5 Rekayasa Perangkat Lunak

Rekayasa Perangkat Lunak adalah Suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal requirement *capturing* (analisa kebutuhan pengguna), *specification* (menentukan spesifikasi dari kebutuhan pengguna), *desain*, *coding*, *testing* sampai pemeliharaan sistem setelah digunakan. Kalimat “seluruh aspek produksi perangkat lunak” membawa implikasi bahwa Rekayasa Perangkat Lunak tidak hanya berhubungan dengan masalah teknis pengembangan perangkat lunak tetapi juga kegiatan strategis seperti manajemen proyek perangkat lunak, penentuan metode dan proses pengembangan, serta aspek teoritis, yang kesemuanya untuk mendukung terjadinya produksi perangkat lunak[8].

A. Metode Pengembangan Sistem

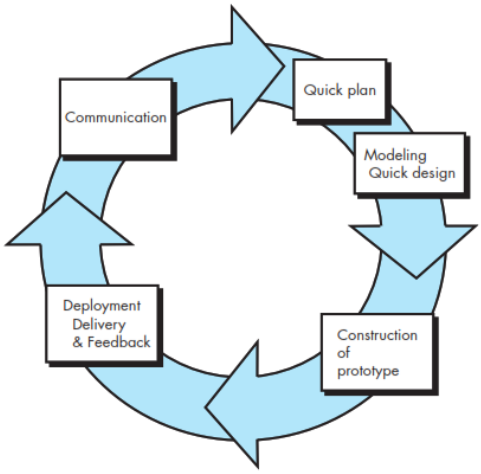
Perancangan sistem yang nantinya akan dikembangan menggunakan metode *prototype*. Metode ini akan cocok digunakan dalam mengembangkan sebuah perangkat yang nantinya akan dikembangkan kembali. Dalam metode ini nantinya kebutuhan pengguna akan dikumpulkan yang kemudian akan dibuatkan rancangan yang akan dievaluasi kembali sebelum diproses dengan benar.

Metode penelitian yang digunakan adalah *metode prototype*, *metode prototype* merupakan sebuah model proses yang diterapkan saat menjalankan komunikasi dengan *client* untuk membuat sebuah aplikasi, tidak menyajikan bentuk asli sistem secara lengkap akan tetapi

metode prototype berperan penting dalam penelitian untuk memberikan gambaran aplikasi yang akurat terhadap *client*[9].

Metode Prototype bukanlah metode yang sempurna, tetapi metode ini juga perlu dievaluasi dan dimodifikasi kembali. Segala perubahan yang nantinya akan dimodifikasi untuk memenuhi kebutuhan pengguna. Pada model *Prototype* ini *developer* dan *client* akan sangat diuntungkan dalam pembuatan sebuah aplikasi karena model *prototype* ini memberikan sebuah pendekatan antara *developr* dengan *client* untuk terus berkomunikasi selama pembuatan aplikasi berlangsung dan *developer* akan mendapatkan *feedback* dari *client* yang akan digunakan untuk memperbaiki aplikasi yang dibuat[10]. sehingga memungkinkan pengembang memahami secara lebih baik kebutuhan yang dibutuhkan pengguna kedepannya.

Metode *prototype* ini memiliki beberapa tahapan yang memiliki perannya masing - masing selama proses perancangan perangkat lunak yang bisa dijelaskan masing -masing tahapan tersebut pada penjelasan dibawah ini pada Gambar 2.1:



Gambar 2. 1 Tahapan *Metode Prototype* (Pressman, 2010)

1. *Communication*

Dimulai dengan tahap *communication*, tahapan ini bertujuan untuk mengidentifikasikan berbagai kebutuhan aplikasi yang akan dirancang nantinya dengan melibatkan para *client* yang bersangkutan agar selama proses perancangan bisa memberikan hasil yang tepat sesuai keinginan *client* yang bersangkutan.

2. *Quick Plan*

Pada tahap *quick plan* ini perancang perangkat lunak akan melakukan perencanaan cepat sesuai dengan spesifikasi kebutuhan user berdasarkan data yang telah dikumpulkan pada tahap

communication dengan merancang desain antarmuka yang dibutuhkan dan kebutuhan pendukung pada proses ini.

3. *Modeling Quick Design*

Pada tahap ini tim perancang akan membuat model design UML ataupun pemodelan yang dibutuhkan lainnya dengan waktu perancangan yang efektif untuk mendeskripsikan kebutuhan client berdasarkan analisis yang telah dilakukan sebelumnya.

4. Construction of Prototype

Selanjutnya pada tahap ini perancang akan memulai membangunan perangkat lunak berdasarkan data yang telah dikumpulkan sebelumnya, proses pembangunan ini lebih berfokus terhadap aspek utama perangkat lunak dengan maksud pada proses selanjutnya perancang bisa dengan cepat mendapatkan *feedback* dari *client* tentang perangkat lunak yang dibuat.

5. Deployment Delivery & Feedback

Dalam tahap ini *prototype* akan diserahkan kepada client untuk mendapatkan *feedback* dari hasil prototype tersebut, *feedback* tersebut akan digunakan sebagai landasan untuk memperbaiki *prototype* agar sesuai dengan spesifikasi kebutuhan *client*.

B. *Tools* atau Alat Bantu Penelitian

1. Flowchart

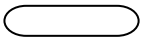
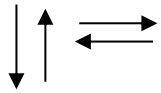
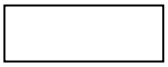
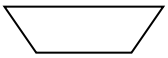
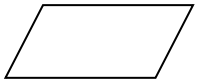
Flowchart adalah Teknik analitis bergambar yang di gunakan untuk menjelaskan beberapa aspek dari sistem informasi secara jelas, ringkas, dan logis Bagan air mencatat cara proses bisnis dilakukan dan cara dokumen mengalir melalui organisasi. *Flowchart* adalah gambar alir akan sistem dan prosedur serta pengendalian intern yang telah dijalankan oleh perusahaan. *flowchart* merupakan penggambaran secara grafik dari langkah-langkah dan urutan prosedur suatu program. Biasanya mempermudah penyelesaian masalah yang khususnya perlu dipelajari dan dievaluasi lebih lanjut[11]

Ada beberapa jenis-jenis *flowchart* diantaranya :

a. Bagan alir sistem (*system flowchart*)
b. Bagan alir dokumen (*document flowchart*)
c. Bagan alir skematik (*schematic flowchart*)
d. Bagan alir program (*program flowchart*)
e. Bagan alir proses (*process flowchart*).

Berikut adalah Simbol-simbol yang ada dalam *flowchart* Tabel 2.1

Table 2. 1 Simbol Flowchart

No	Simbol	Keterangan
1	 ***Terminator Symbol***	Yaitu simbol untuk permulaan (*start*) atau akhir (*end*) dari suatu kegiatan.
2	 ***Flow Direction Symbol***	Yaitu simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga connecting line.
3	 ***Processing Symbol***	Yaitu simbol yang menunjukan pengolahan yang dilakukan oleh komputer.
4	 ***Manual Operation Symbol***	Yaitu simbol yang menunjukan pengolahan data yang tidak dilakukan oleh komputer.
6	 ***Decision Symbol***	Yaitu simbol untuk pemilihan proses berdasarkan kondisi yang ada.
7	 ***Input-Output Symbol***	Yaitu simbol yang menyatakan proses input dan output tanpa tergantung dari jenis peralatannya.
6	 ***Document Symbol***	Yaitu simbol yang menyatakan *input* berasal dari dokumen dalam bentuk kertas atau *output* dicetak ke kertas.

2. Unified Modeling Language (UML)

UML (*Unfield Modelling Language*) ialah bahasa yang menjadi standar dalam industri untuk visualisasi perancangan dan dokumentasi sistem perangkat lunak dengan menggabungkan elemen yang tidak praktis ditambah dengan elemen dari metode lain yang lebih efektif dan elemen baru yang belum ada dalam metode terdahulu, sehingga UML (*Unfield Modelling Language)* lebih eksresif dan seragam daripada *metode* lainnya[12]. UML (*Unfield Modelling Language*) digunakan untuk tujuan tertentu, yaitu:

a. Merancang *software* (perangkat lunak).

b. Alat komunikasi bagi perangkat lunak dengan proses bisnis.

c. Menjabbarkan secara rinci untuk menganalisa dan mencari apa yang diperlukan oleh sistem.

d. Dokumentasi untuk sistem yang ada dan proses-proses yang terjadi dan organisasinya.

Macam-macam dari *Unified Modeling Language* (UML) antara lain *use case* diagram, *sequence* diagram dan *class* diagram.

a. Use Case Diagram

Use case diagram ialah pola atau gambaran berbentuk diagram yang menggambarkan hubungan suatu sistem yang tengah dibuat. *Use case diagram* merupakan sebuah teknik yang digunakan dalam pengembangan sebuah *softwar*e atau sistem informasi untuk menangkap kebutuhan fungsional dari sistem yang bersangkutan, yang ditekankan adalah "apa" yang diperbuat sistem dan bukan "bagaimana". Sebuah *use case* mempresentasikan sebuah interaksi antara aktor dan sistem, sebuah use case direpresentasikan dengan urutan langkah yang sederhana. Use case merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng-*create* sebuah daftar belanja dan sebagainya. *Actor* adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan tertentu.

Use case diagram dapat sangat membantu bila kita sedang menyusun *requitment* sebuah sistem. Mengkomunikasikan rancangan dengan klien dan merancang *test case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsional dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extand use case* lain dengan *behavior*-nya sendiri. Sementara hubungan *generalitation* antar *use case* menunjukan bahwa *use case* yang satu merupakan spesialisasi dari yang lain[12]. Simbol *Use Case* dapat dilihat pada Tabel 2.2

Table 2. 2 Simbol *Use Case Diagram*

No	Simbol	Keterangan
1.	 ***Actor***	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan use case
3.	 ***Generalization***	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk

4.	 ***Include***	Menspesifikasikan bahwa use case sumber secara eksplisit
5.	 ***Extend***	Menspesifikasikan bahwa use case target memperluas perilaku dari use case sumber pada suatu titik yang diberikan.
6.	***Associtation***	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7.	 ***System***	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8.	 ***UseCase***	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil terukur bagi suatu aktor.

3. *Entity Relationship Diagram* (ERD)

Entity Relationship adalah suatu metode yang diaman pemodelan basis data yang digunakan merupakan skema konseptual yang dimana jenis dari model data semantic sistem. Dimana sistem yang digunakan pada *entity- relationship* merupakan basis data relasional yang memiliki sifat *top-down*. Diagram yang digunakan ialah suatu gambaran model *entity-relationship* yang disebut dengan entity-relationship diagram, ER diagram atau ERD. Entity ini adalah suatu objek yang dibedakan dari didetivikasikan secara unik dengan relationship yang dimana menghubungkan antara satu sama lainnya, sedangkan atribut yang akan membentuk karakteristik setiap entitras dengan jumlah konvensi.

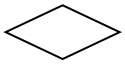
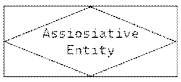
Entitas adalah suatu penjelasan yang dapat digambarkan oleh suatu data, yang dapat diartikan sebagai kata yang secara individu yang akan mewakili sesuatu yang nyata yang mewakili sesuatu yang lainnya. Entitas ini adalah suatu model data yang apabila entitas tersebut kuat dan memiliki ketergantungan antara entitas lainnya maka entitas tersebut akan dihubungan dengan tanda garis pada sistem untuk memperjelas kata bagian yang ada didalam sistem database management system tersebut. Salah satu contoh dari entitas ini ialah menujukkan keanggotaan yang bergantung pada entitas lemah atau entitas lainnya, dimana keberadaannya ada pada relasi data.

Entity Relationship Diagram (ERD) adalah diagram yang berbentuk notasi grafis yang berada dalam pembuatan database menghubungkan antara data satu dan data yang lain. Fungsi

ERD adalah sebagai alat bantu dalam pembuatan database dan memberikan gambaran bagaimana kerja database yang akan dibuat[13]. Di dalam ERD terdapat 3 elemen dasar, yaitu entitas, atribut, dan relasi.

1. Entitas adalah objek dalam suatu database. Entitas dapat berupa manusia, tempat, benda, atau kondisi mengenai data yang dibutuhkan. Simbol dari entitas berbentuk persegi panjang.
2. Atribut adalah informasi yang terdapat dalam entitas. Sebuah entitas harus memiliki primary key sebagai ciri khas entitas dan atribut deskriptif. Atribut biasanya terletak dalam tabel entitas atau dapat juga terpisah dari tabel. Simbol dari atribut berbentuk elips.
3. Relasi di dalam ERD merupakan hubungan antara dua atau lebih entitas. Simbol dari relasi berbentuk belah ketupat. Relasi yang dapat dimiliki oleh ERD ada beberapa macam, yaitu : *One to One* (Satu anggota entitas dapat berelasi dengan satu anggota entitas lain), *One to Many* (Satu anggota entitas dapat berelasi dengan beberapa anggota entitas lain), *Many to Many* (Beberapa anggota entitas dapat berelasi dengan beberapa anggota entitas lain).

Table 2. 3 Simbol *Entity Relationship Diagram* (ERD)

No.	Simbol	Keterangan Fungsi
1.	 **Entitas**	Persegi panjang menyatakan himpunan entitas adalah orang, kejadian, atau berada.
2.	 **Atribut**	Atribut merupakan informasi yang diambil tentang sebuah entitas.
3.	 **Relasi**	Belah ketupat menyatakan himpunan relasi merupakan hubungan antar entitas.
4.	**Link**	Garis sebagai penghubung antara himpunan, relasi, dan himpunan entitas dengan atributnya
5.	 **Entitas Assosiatif**	Entitas yang digunakan untuk menghubungkan dua atau lebih entitas lain dalam hubungan banyak-ke-banyak (many-to-many). Entitas ini dapat memiliki atribut sendiri selain atribut yang mewakili hubungan antara entitas-entitas tersebut.