BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Penelitian yang berkaitan dengan tracer study sudah banyak dilakukan oleh para peneliti terdahulu dengan tujuan dan lingkup permasalahan yang berbeda. Penelitian dengan judul "Sistem Informasi *Tracer Study* Berbasis *Website* pada SMK Dr. Soetomo Cilacap" bertujuan untuk membantu mengatasi permasalahan dalam mengelola data informasi status alumni, memantau jejak perkembangan alumni pada industri setiap tahunnya. Penelitian tersebut menggunakan metode *Software Development Life Cycle* (SDLC) dengan model *Waterfall* dan menggunakan bahasa pemrograman HTML dan PHP dengan *framework laravel* serta database MySQL. Hasil penelitian dapat disimpulkan bahwa penelitian tersebut dapat mengatasi masalah yang ada dengan memudahkan pendataan status alumni melalui jarak jauh, membantu dalam pembuatan laporan, serta dapat menampilkan pencarian informasi data status terbaru alumni. Perbedaan penelitian ini dengan penelitian yang akan dibuat yaitu terletak pada bagian kuesioner pertanyaan. Pertanyaan pada penelitian ini lebih berfokus pada status yang sedang dijalani oleh alumni. Status tersebut diantaranya bekerja, belum bekerja, melanjutkan kuliah atau berwirausaha. Hal ini menjadikan kuesioner bersifat statis dan pertanyaan tidak bisa dikembangkan atau bersifat dinamis[7].

Penelitian yang serupa dengan judul "Sistem Tracer Study dan Persebaran Alumni Berbasis Web di Universitas Islam Syekh-Yusuf Tangerang". Tujuan penelitian tersebut adalah untuk memperbaiki kualitas pembelajaran dan sinkronisasi kurikulum. Dalam perancangan sistem, model pengembangan sistem menggunakan *extreme programming*. Hasil dari penelitian adalah menyajikan informasi mengenai data alumni, informasi lowongan pekerjaan, persebaran alumni dan hasil dari sistem ini memudahkan pihak admin untuk pengelolaan data alumni, memberikan kemudahan informasi persebaran alumni serta pengisian kuesioner lulusan yang diakses secara online. Penelitian ini lebih berfokus pada persebaran keberadaan alumni serta penginformasian lowongan pekerjaan. Sedangkan penelitian yang akan dibuat lebih berfokus pada pengisian kuesioner oleh alumni[8].

Penelitian lainnya dengan judul "Analisis Dan Perancangan Sistem Informasi Pelacakan Lulusan Berbasis Website". Solusi saat ini yang dianggap dapat meningkatkan data lulusan yaitu dengan mengembangkan sistem informasi pelacakan lulusan. Metode pengumpulan data yang digunakan adalah observasi, wawancara, dokumentasi dan studi pustaka. Metode pengembangan sistem menggunakan model *waterfall*. Bahasa pemrograman adalah bahasa pemrograman web dengan menggunakan editor Sublimetext dan CSS untuk membuat dan mengedit serta mendesain. Database yang digunakan adalah PhpMyAdmin dan MySQL. Kesimpulan dari penelitian ini adalah penulis sudah berhasil menganalisis dan mengembangkan sistem informasi pelacakan lulusan di SMK Negeri 1 Pedan Klaten, sistem dibuat dua pemakai yaitu admin dan lulusan. Admin dapat melakukan manajemen sistem, sedangkan lulusan dapat memasukkan kondisinya saat ini melalui sistem yang dikembangkan ini. Sistem dapat membuat laporan hasil pelacakan lulusan, sistem dapat memberi kemudahan dalam pencarian data lulusan. Perbedaan penelitian ini dengan penelitian yang akan dibuat yaitu pada bagian kuesioner pertanyaan. Pertanyaan pada penelitian ini lebih berfokus pada status yang sedang dijalani oleh alumni tersebut. Status tersebut diantaranya bekerja, belum bekerja, melanjutkan kuliah atau berwirausaha. Hal ini menjadikan kuesioner bersifat statis dan pertanyaan tidak bisa di kembangkan atau bersifat dinamis. Perbedaan selanjutnya yaitu pada orientasi pemrograman tidak menggunakan orientasi pemrograman berbasis objek atau Pemrograman Berorientasi Objek (PBO)[1].

Berdasarkan tinjauan pustaka di atas yang membedakan penelitian yang dibuat oleh penulis dengan penelitian sebelumnya adalah penulis merancang "Aplikasi *Tracer Study* Berbasis *Website* pada SMK Negeri 1 Cilacap". Aplikasi ini menggunakan kuesioner yang bersifat dinamis, sehingga dapat menambah ataupun mengurangi jumlah pertanyaan yang akan dibagikan kepada alumni. Metode yang digunakan adalah metode *waterfall* dan bahasa pemrograman PHP serta *database* MySQL. Metode pengujian yang digunakan adalah *black-box*. Hasil dari penelitian adalah menyajikan informasi mengenai data alumni berdasarkan grafik status alumni dan memberikan kemudahan pengisian kuesioner yang dapat diakses secara *online*.

2.2 Landasan Teori

Landasan teori berisi hal-hal atau teori-teori yang berkaitan dengan permasalahan dan ruang lingkup permasalahan sebagai landasan dalam pembuatan laporan ini.

2.2.1 Aplikasi

Aplikasi adalah *software* yang dibuat oleh suatu perusahaan komputer untuk mengerjakan tugas-tugas tertentu[9]. Menurut kamus besar bahasa Indonesia, aplikasi adalah program komputer atau perangkat lunak yang didesain untuk mengerjakan tugas tertentu.

Secara garis besar, aplikasi adalah program siap pakai yang digunakan untuk menjalankan perintah-perintah dari pengguna aplikasi tersebut dengan tujuan mendapatkan hasil yang lebih akurat sesuai dengan tujuan pembuatan aplikasi tersebut.

2.2.2 Tracer Study

Tracer study atau kajian penelusuran, sering disebut juga sebagai survey alumni adalah studi mengenai alumni lembaga penyelenggara pendidikan[4]. *Tracer study* ini berguna untuk mengetahui seberapa besar alumni mampu berkiprah dalam pembangunan sesuai relevansi pendidikannya. *Tracer study* merupakan pendekatan yang dilaksanakan oleh sekolah untuk memperoleh informasi mengenai kondisi alumni khususnya dalam hal pencarian kerja, situasi saat ini, dan pemanfaatan pemerolehan kompetensi selama di sekolah.

2.2.3 Website

Website adalah suatu layanan sajian informasi yang menggunakan konsep *hyperlink*, yang memudahkan surfer atau *browser* dalam melakukan penelusuran informasi melalui internet[10]. Menurut Sibero, *website* adalah kumpulan dari berbagai halaman web dalam suatu domain yang memuat tentang berbagai informasi yang umumnya berisi mengenai konten gambar, ilustrasi, video, dan teks agar dapat dibaca dan dilihat oleh pengguna internet.

Adapun jenis-jenis *website*, antara lain :

a. *Website* statis merupakan *website* yang dapat diartikan sebagai *website* yang berisikan data dan informasi yang tidak berubah-

ubah. Dokumen web yang dikirim kepada client akan sama isinya dengan web server.

b. *Website* dinamis merupakan *website* yang memiliki data dan informasi yang berbeda-beda tergantung input yang disampaikan oleh *client*.

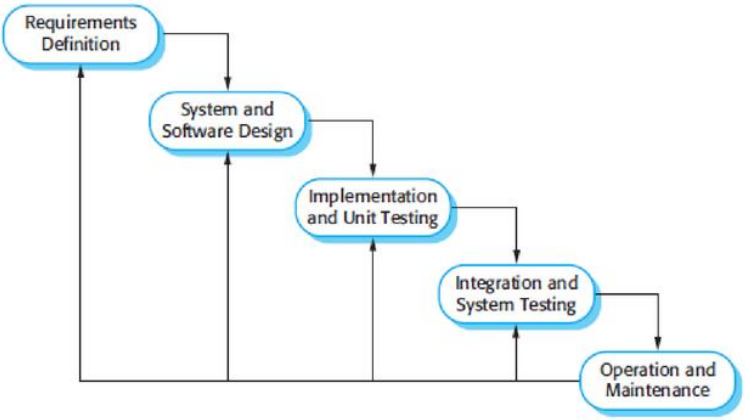
2.2.4 Rekayasa Perangkat Lunak

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 memproses informasi[11].

Rekayasa Perangkat Lunak sendiri adalah suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal yaitu analisa kebutuhan pengguna, menentukan spesifikasi dari kebutuhan pengguna, desain, pengkodean, pengujian sampai pemeliharaan sistem setelah digunakan[12]. Dari pengertian ini jelaslah bahwa Rekayasa Perangkat Lunak tidak hanya berhubungan dengan cara pembuatan program komputer. Pernyataan "semua aspek produksi" pada pengertian di atas, mempunyai arti semua hal yang berhubungan dengan proses produksi seperti manajemen proyek, penentuan personil, anggaran biaya, metode, jadwal, kualitas sampai dengan pelatihan pengguna merupakan bagian dari Rekayasa Perangkat Lunak.

A. Metode Pengembangan Aplikasi

Metode yang digunakan adalah metode *waterfall* menurut Ian Sommerville. *Waterfall* atau air terjun adalah model yang dikembangkan untuk pengembangan perangkat lunak, dan membuat perangkat lunak. Metode ini berkembang secara sistematis dari satu tahap ke tahap lain dalam mode seperti air terjun[13]. Metode ini mengusulkan sebuah pendekatan kepada pengembangan software yang sistematik dan sekuensial yang mulai dari tingkat kemajuan sistem pada seluruh analisis, desain, kode, pengujian dan pemeliharaan. Model *waterfall* dapat dilihat pada Gambar 2.1.



Gambar 2. 1 Metode *Waterfall*

Adapun penjelasan dari tahapan-tahapan metode *waterfall* sebagai berikut :

1. *Requirement Analysis and Definition*
   Langkah ini merupakan analisa terhadap kebutuhan sistem. Pengumpulan data pada tahap ini bisa melakukan sebuah penelitian, wawancara atau *study* literatur. Seorang sistem analisis akan menggali informasi sebanyak banyaknya dari *user* sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugas-tugas yang diinginkan oleh *user* tersebut. Tahapan ini akan menghasilkan dokumen *user reqruitment* atau bisa dikatakan sebagai data yang berhubungan dengan keinginan *user* dalam pembuatan sistem. Dokumen inilah yang akan menjadi acuan sistem analisis untuk menterjemahkan kedalam bahasa pemrograman.
2. *System and Software Design*
   Pada Tahap *System and Software Design* ini akan dibentuk suatu arsitektur sistem berdasarkan persyaratan yang telah ditetapkan. Selain itu juga, dilakukan identifikasi dan penggambaran terhadap abstraksi dasar sistem perangkat lunak beserta hubungan-hubungannya.
3. *Implementation and Unit Testing*
   Dalam tahapan *Implementation and Unit Testing* ini, hasil dari desain perangkat lunak akan direalisasikan sebagai satu set

program atau unit program. Setiap unit akan diuji apakah sudah memenuhi spesifikasinya.

4. *Integration and System Testing*
   Dalam tahap *Integration and System Testing* ini, setiap unit program akan diintegrasikan satu sama lain dan diuji sebagai satu sistem yang utuh untuk memastikan sistem sudah memenuhi persyaratan yang ada. Setelah itu sistem akan dikirim ke pengguna sistem.
5. *Operation and Maintenance*
   Dalam tahap *Operation and Maintenance*, sistem diinstal dan mulai digunakan. Selain itu juga memperbaiki kesalahan (*error*) yang tidak ditemukan pada tahap pembuatan. Dalam tahap ini juga dilakukan pengembangan sistem seperti penambahan fitur dan fungsi baru.

B. Metode Pengujian Aplikasi

Metode pengujian aplikasi yang digunakan adalah dengan metode *black box testing*. Metode *black box testing* merupakan pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, penguji *(tester)* dapat mendefinisikan kumpulan kondisi dan melakukan pengetesan pada spesifikasi fungsional program[8].

Ciri-ciri *black box testing* :

1. *Black box testing* berfokus pada kebutuhan fungsional pada *software*, berdasarkan pada spesifikasi kebutuhan dari *software.*
2. *Black box testing* bukan teknik alternatif daripada *white box testing.* Lebih dari itu, *black box testing* merupakan pendekatan pelengkap dalam mencakup *error* dengan kelas yang berbeda dari metode *white box testing.*
3. *Black box testing* melakukan pengujian tanpa pengetahuan detail struktur internal dari sistem atau komponen yang dites.

Kategori *error* yang akan diketahui melalui *black box testing* :

1. Fungsi yang hilang atau tak benar.
2. *Error* dari antarmuka.
3. *Error* dari struktur data atau akses *eksternal database*.
4. *Error* dari kinerja atau tingkah laku.
5. *Error* dari inisialisasi dan terminasi.

Pengujian *black-box* berusaha menemukan kesalahan dalam kategori sebagai berikut[14] :

1. Fungsi-fungsi yang tidak benar atau hilang,
2. Kesalahan *interface*,
3. Kesalahan dalam struktur data atau akses *database* eksternal,
4. Kesalahan kinerja,
5. Inisialisasi dan kesalahan terminal.

Pengujian *black-box* ini cenderung diaplikasikan selama tahap akhir pengujian. Karena pengujian *black-box* memperhatikan struktur kontrol, maka perhatian berfokus pada domain informasi. Terdapat beberapa proses pengujian *black-box*, diantaranya :

a. Menganalisa kebutuhan dan spesifikasi dari perangkat lunak.
b. Pemilihan jenis input yang memungkinkan menghasilkan *output* dengan benar serta jenis input yang memungkinkan *output* salah pada perangkat lunak yang sedang diuji.
c. Menentukan *output* untuk satu jenis input.
d. Pengujian dilakukan dengan input-input yang telah benar-benar diseleksi.
e. Melakukan pengujian.
f. Perbandingan *output* yang dihasilkan dengan *output* yang diharapkan.
g. Menentukan fungsionalitas yang seharusnya ada pada perangkat lunak yang sedang diuji.

2.2.5 Pemrograman Berorientasi Objek

Metodologi berorientasi objek adalah strategi pembangunan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis[15].

Metodologi berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan tersruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu**.**

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut:

1. Meningkatkan produktifitas
   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai diulang kembali untuk masalah lainnya yang melibatkan objek tersebut (*reusable*).
2. Kecepatan pengembangan
   Karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengkodean.
3. Kemudahan pemeliharaan
   Karena dengan model objek, pola-pola yang cenderung tetapdan stabil dapat dipisahkan dan pola-pola mungkin sering berubahubah.
4. Adanya konsistensi
   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.
5. Meningkatkan kualitas perangkat lunak
   Karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek:

1. Kelas (*class*)
   Class adalah *prototype*, atau *blueprint*, atau rancangan yang mendefinisikan variabel dan metode-metode pada seluruh objek tertentu. *Class* berfungsi untuk menampung isi dari program yang akan dijalankan, didalamnya berisi atribut / *type* data dan *method* untuk menjalankan suatu program.
2. Objek (*object*)
   Objek adalah *instance* dari *class*. Jika *class* secara umum mempresentasikan (*templates*) sebuah objek, sebuah *instance* adalah representasi nyata dari *class* itu sendiri.
3. Metode (*method*)
   Metode adalah merupakan suatu operasi berupa fungsi-fungsi yang dapat dikerjakan oleh suatu objek. *Method* didefinisikan pada *class* akan tetapi dipanggil melalui objek.

4. Atribut (*attribute*)
   Atribut merupakan nilai data yang terdapat pada suatu objek yang berasal dari class. *Attribute* mempresentasikan karakteristik dari suatu objek.
5. Abstraksi (*abstract*)
   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.
6. Enkapsulasi (*encapsulation*)
   Pembungkusan *attribute* data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.
7. Pewarisan (*inheritance*)
   Mekanisme yang memungkinkan suatu objek mewarisi sebagian atas seluruh definisi objek lain sebagai bagian dari dirinya.
8. Antarmuka (*interface*)
   Antarmuka atau *interface* sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.

Pada pemrograman berorientasi objek, UML digunakan untuk pemodelan sistem. UML adalah bahasa untuk menspesifikasi, memvisualisasi, membangun dan mendokumentasikan *artifacts* (bagian dari informasi yang digunakan atau dihasilkan oleh proses pembuatan perangkat lunak, *artifact* tersebut dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya.

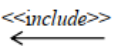
A. *Use Case Diagram*

Use Case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use Case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut *scenario* sedangkan pengguna disebut *actor*. *Actor* adalah sebuah peran yang biasa dimainkan oleh pengguna dalam interaksinya dengan sistem. Model *use case* adalah bagian dari model *requitment*. Definisi lain *use case* adalah

abstraksi dari interaksi antara sistem dan *actor*. *Use case* dibuat berdasarkan keperluan *actor*.

Berdasarkan definisi diatas maka dapat disimpulkan bahwa *Use Case* adalah kontruks untuk mendeskripsikan bagaimana sistem akan terlihat dimata pengguna potensial yang terdiri dari sekumpulan *scenario* dan *actor*. Sedangkan *use case* diagram memfasilitasi komunikasi diantara analis dan pengguna serta analis dan klien. Simbol *Use Case* dapat dilihat pada Table 2.3.

Tabel 2. 1 Simbol *Use Case Diagram*

No.	Simbol	Nama	Keterangan
1.		*Actor*	Menggambarkan *actor* yang berinteraksi dengan sistem dan dapat menerima dan memberi informasi pada sistem.
2.		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*not independent*)
3.		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada diatasnya objek induk (*ancestor*).
4.		*Include*	Menunjukkan bahwa *use case* satu merupakan bagian dari *use case* lainnya.
5.		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6.		*Association*/ Asosiasi	Menghubungkan antara objek satu dengan objek yang lainnya.
7.		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8.		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang

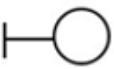
			menghasilkan suatu hasil yang terukur bagi suatu actor.
9.		*Collaborattion*	Interaksi aturan-aturan dan elemen lain yang bekerjasama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10.		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

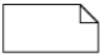
B. *Sequence Diagram*

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan sejumlah objek dan *message* yang diletakan antara objek-objek didalam *use case*.

Komponen utama *sequence diagram* terdiri atas objek yang dituliskan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan progress *vertical*. *Sequence diagram* menambahkan dimensi waktu pada interaksi diantara objek. Simbol-simbol yang dipakai dalam pembuatan *sequence diagram* dapat dilihat pada Tabel 2.4.

Tabel 2. 2 Simbol *Sequence Diagram*

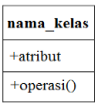
No	Simbol	Nama	Keterangan
1.		*Actor*	Menggambarkan *user* atau pengguna yang sedang berinteraksi dengan sistem.
2.		*Entity Class*	Menggambarkan suatu tempat atau mekanisme yang menangkap pengetahuan atau informasi dalam suatu sistem.
3.		*Boundary Class*	Menggambarkan hubungan suatu elemen yang berbeda, secara khas merupakan penghubung actor dengan tampilan layar.
4.		*Control Class*	Menggambarkan suatu pengendalian yang mengorganisir dan menjadwalkan aktivitas elemen – elemen

5.		*Lifeline*	Menggambarkan lamanya suatu pesan diproses.
6.		*Line Message*	Perilaku sistem yang menandai adanya suatu alur informasi atau tarnsisi kendali antar elemen.
7.		*Note*	Menunjukkan catatan untuk komentar dari suatu pesan antar elemen.

C. *Class Diagram*

Class diagram atau diagram kelas merupakan diagram yang memodelkan sekumpulan kelas, *interface*, kolaborasi dan relasinya[16]. Diagram kelas digambarkan dengan bentuk kotak. Simbol dan keterangan *class diagram* seperti pada Tabel 2.5.

Tabel 2. 3 Simbol *Class Diagram*

No	Simbol	Nama	Keterangan
1.		Kelas	Kelas pada struktur sistem.
2.		Asosiasi/ *Association*	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
3.		Asosiasi berarah / *Direct Association*	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain.
4.		Generalisasi	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum-khusus).
5.		Kebergantungan / *Dependency*	Relasi antarkelas dengan makna kebergantungan antarkelas.
6.		Agregasi/ *Aggregation*	Relasi antarkelas dengan makna semua-bagian (*whole-part*).

2.2.6 Basis Data

Basis data adalah kumpulan file-file yang mempunyai kaitan antara satu file dengan file lain sehingga membentuk satu bangun data

untuk penginformasikan suatu perusahaan instansi, dalam bahasan tertentu[17].

A. *Database Management System* (DBMS)

DBMS atau *Database Management System* merupakan sebuah perangkat lunak yang memungkinkan pengguna untuk mendefinisikan, membuat, mengambil data dan mengontrol akses kepada *database*. DBMS merupakan sebuah perangkat lunak yang mengintegrasikan *database* dengan aplikasi program pada pengguna. Biasanya, DBMS menyediakan fasilitas sebagai berikut :

1. *Data Definition Language* (DDL)

DDL memperbolehkan pengguna untuk mendeskripsikan *database*, misalnya merinci tipe dan batasan data yang akan disimpan dalam *database*. Terdapat tiga perintah penting dalam DDL, yaitu :

a) *CREATE* adalah perintah yang digunakan untuk membuat, termasuk didalamnya membuat *database* baru, tabel baru, *view* baru, dan kolom baru. Contoh : *CREATE DATABASE* nama *database*.
b) *ALTER* berfungsi untuk mengubah struktur tabel yang telah dibuat. Mencakup didalamnya mengubah nama tabel, menambah kolom, mengubah kolom, menghapus kolom, dan memberikan atribut pada kolom. Contoh : *Alter Table* nama tabel ADD nama kolom datatype.
c) *DROP* perintah *DROP* berfungsi untuk menghapus *database* atau tabel. Contoh : *DROP DATABASE* nama *database*.

2. *Data Manipulation Language* (DML)

DML memperbolehkan pengguna untuk memanipulasi data dalam *database* yang telah dibuat. Terdapat 4 (empat) perintah penting dalam DML, yaitu :

a) *INSERT* perintah ini digunakan untuk memasukkan data baru ke dalam sebuah tabel. Perintah ini tentu saja bisa dijalankan ketika database dan tabel sudah dibuat. Contoh : *INSERT INTO* nama tabel *VALUES* (data1, data2, dst..)
b) *SELECT* perintah ini digunakan untuk mengambil dan menampilkan data dari tabel atau bahkan dari beberapa tabel dengan penggunaan relasi. Contoh : *SELECT* nama_kolom1, nama_kolom2 *FROM* nama tabel.

c) *UPDATE* perintah ini digunakan untuk memperbaharui data pada sebuah tabel. Contoh : *UPDATE* nama_tabel *SET* kolom1=data1, kolom2=data2,.. *WHERE* kolom=data.
d) *DELETE* perintah ini digunakan untuk menghapus data dari sebuah tabel. Contoh : *DELETE FROM* nama tabel *WHERE* kolom=data.

B. MySQL

MySQL adalah sebuah program *database server* yang mampu menerima dan mengirimkan datanya dengan sangat cepat, multi *user* serta menggunakan perintah standar *Structured Query Language* (SQL). MySQL banyak digunakan untuk membangun aplikasi web yang menggunakan *database* sebagai sumber dan pengelolaan datanya[18]. MySQL memiliki dua bentuk lisensi, yaitu *Free Software* dan *Shareware*. MySQL yang biasa kita gunakan adalah MySQL *Free Software* yang berada dibawah Lisensi *General Public License* (GPL).

MySQL juga dapat didefinisikan sebagai sebuah *database server*, dapat juga berperan sebagai *client* sehingga sering disebut *database client /server* yang *open source* dengan kemampuan dapat berjalan baik di Operasi Sistem maupun dengan *Platform Windows* maupun *Linux*.

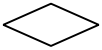
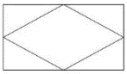
MySQL dikembangkan oleh sebuah perusahaan Swedia bernama MySQL AB, yang pada saat itu bernama TcX Data Konsult AB sekitar tahun 1994-1995. MySQL sudah ada sejak 1979. MySQL termasuk jenis *Relation Database Management System* (RDBMS) digunakan oleh banyak portal-portal internet sebagai basis data dari informasi yang ditampilkan pada situs *web*. Kepopuleran MySQL dimungkinkan karena kemudahannya untuk digunakan, cepat secara kinerja *query*, dan mencukupi untuk kebutuhan basis data perusahaan-perusahaan skala menengah dan kecil. Istilah seperti tabel, baris, dan kolom tetap digunakan dalam MySQL. Sebuah basis data yang terdapat pada MySQL mengandung satu atau beberapa tabel yang terdiri dari sejumlah baris dan kolom.

C. *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram (ERD) adalah suatu rancangan atau bentuk hubungan suatu kegiatan di dalam sistem yang berkaitan langsung dan mempunyai fungsi di dalam proses tersebut. ERD merupakan suatu pemodelan dari basis data relasional yang didasarkan atas persepsi di

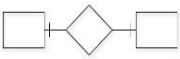
dalam dunia nyata, dunia ini senantiasa terdiri dari sekumpulan objek yang saling berhubungan antara satu dengan yang lainnya. Suatu objek disebut *entity* dan hubungan yang dimilikinya disebut *relationship*. Suatu *entity* bersifat unik dan memiliki atribut sebagai pembeda dengan *entity* lainnya. Berikut adalah simbol – simbol dari ERD yang dapat dilihat pada Tabel 2.6 :

Tabel 2. 4 Simbol *Entity Relationship Diagram* (ERD)

No.	Nama	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.	Associative Entity		Entitas yang digunakan pada relasi many-to-many.

ERD memiliki derajat relasi atau biasa disebut kardinalitas. Kardinalitas menjelaskan batasan jumlah keterhubungan satu *entity* dengan *entity* lainnya. Berikut adalah macam-macam kardinalitas yang dapat dilihat pada Tabel 2.7.

Tabel 2. 5 Macam-Macam Kardinalitas

No.	Simbol	Nama	Keterangan
1.		Relasi Satu ke Satu (*One to One*)	Relasi yang menunjukkan bahwa setiap himpunan entitas

			berhubungan dengan tepat satu himpunan entitas lainnya
2.		Relasi Satu ke Banyak (*One to Many*)	Relasi yang menunjukkan bahwa hubungan antara entitas pertama dengan entitas kedua adalah satu banding banyak, begitu pula sebaliknya
3.		Relasi Banyak ke Banyak (*Many to Many*)	Relasi yang menunjukkan bahwa setiap himpunan entitas boleh berhubungan dengan banyak himpunan entitas lainnya dan sebaiknya

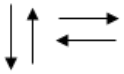
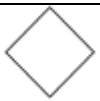
2.2.7 Flowchart

Flowchart merupakan gambar atau bagan yang memperlihatkan urutan dan hubungan antar proses beserta instruksinya yang dinyatakan dengan simbol[19]. Setiap simbol menggambarkan proses tertentu. Sedangkan hubungan antar proses digambarkan dengan garis penghubung. *Flowchart* ini merupakan langkah awal pembuatan program. Dengan adanya *flowchart* urutan poses kegiatan menjadi lebih jelas. Jika ada penambahan proses maka dapat dilakukan lebih mudah. Setelah *flowchart* selesai disusun, selanjutnya programmer menerjemahkannya ke bentuk program dengan bahasa pemrograman. Simbol yang dipakai dalam *flowchart* dibagi menjadi 3 kelompok, yaitu :

1. *Flow direction symbols*
   Digunakan untuk menghubungkan simbol satu dengan yang lain (*connecting line*).
2. *Processing symbols*
   Menunjukan jenis operasi pengolahan dalam suatu proses atau prosedur.
3. *Input / Output symbols*
   Menunjukkan jenis peralatan yang digunakan sebagai media input atau output.

Flowchart disusun dengan simbol-simbol. Simbol ini dipakai sebagai alat bantu menggambarkan proses di dalam program. Simbol-simbol yang dipakai antara lain :

Tabel 2. 6 Simbol *Flowchart* dan Keterangan

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

~Halaman ini sengaja dikosongkan~