BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

1.1 Tinjauan Pustaka

Penelitian yang dilakukan oleh[3] "Sistem Pendukung Keputusan Seleksi Penerimaan Staff Administrasi Menggunakan Metode *Profile Matching*". Dalam penelitian ini membahas tentang penerimaan karyawan baru untuk mendapatkan karyawan yang tepat bagi suatu jabatan tertentu, sehingga karyawan tersebut mampu bekerja secara optimal dan mengerti apa yang seharusnya karyawan kerjakan sehingga orang tersebut dapat bertahan di perusahaan. Dalam penelitian ini menggunakan metode *Profile Matching* diperlukannya kriteria, kategori dan bobot untuk melakukan perhitungan. Dalam kasus ini kriteria dibagi menjadi tiga, yaitu kecerdasan, sikap kerja dan perliku. Dalam setiap kriteria memiliki sub kriteria dan nilai target. Untuk kategori ada dua sub, *core factor* dan *secondary factor*. Nilai bobot *core factor* yaitu 60%, sedangkan *secondary factor* nilai bobotnya 40%. Dalam penelitian ini kriteria yang paling diprioristaskan. Kelauaran yang nantinya dihasilkan adalah urutan alternatif dari yang tertinggi sampai terendah.

Penelitian yang dilakukan oleh[4] "Implementasi Sistem Pendukung Keputusan Seleksi Penerimaan Karayawan Baru Metode *Profile Matching ".* Dalam penelitian ini yang dengan tujuan untuk merancang dan membuat sebuah aplikasi *Decision Support System* (DSS) agar memberian alternatif solusi dalam sebuah pengambilan keputusan pemilihan karyawan baru yang sesusai dengan kebutuhan perusahaan. Dalam penelitian ini menggunakan metode *Profile Matching* dengan kriteria yang telah ditentukan , hasil yang di peroleh pun di nilai lebih adil dikarenakan tidak ada unsur subjektif dalam penentuan karyawan yang diterima.

Penelitian yang dilakukan oleh[5] "Sistem Pendukung Keputusan Penerimaan Karyawan Menggunakan Metode Analytical Hierarchy Proses (AHP)". Pada penelitian ini menggunakan beberapa kriteria untuk menentukan calon pelamar yang mana akan diterima. Spk ini membantu pimpinan perusahaan dalam memutuskan pelamar mana yang akan dipilih. Penelitian ini menitih beratkan kepada bagaimana merancang dan mengimplementasikan program serta dimasukan agar memudahkan dalam

hal perhitungan, penelitian ini menggunakan metode AHP dalam perhitungan seleksi karyawan.

Penelitian yang dilakukan oleh[6] “Proses Seleksi Karyawan Baru Bagian *Sales* Pada PY Mitra Sukse KARYA Bersama Bekasi”. Pada penelitian ini proses seleksi penerimaan karyawan baru sales melalui bebarapa tahap diantaranya seleksi berkas lamaran kerja, pemanggil pelamar kerja, wawancara atasan langsung, pemeriksa referensi oleh manajer HRD dan terkahir keputusan penerimaan apakah pelamar diterima berkerja atau ditolak. Disarankan dalam proses penerimaan karyawan baru dilakukan tes penerimaan, tes kesehatan dan tes pengetahuan produk. Tes penerima dilakukan untuk mengetahui kepribadian calon pelamar, sedang tes kesehatan dilakuka untuk menngetahui kesehatan calon pelamar, tes pengetahuan produk bertujuan untuk mengatahui tingkat pengetahuan pelamar tentang produk yang akan dijual. Dalam penelitian ini tidak dijelaskan dengan jelas menggunakan metode apa dalam sistem pendukung keputusan. Tujuan dari penelitian ini adalah untuk mendapatkan karyawan yang jujur dan dapat melaksanakan tugas yang diberikan oleh perusahaan.

Pada penelitian in, penulis bermaksud membangun sebuah sistem pendukung keputusan penerimaan karyawan baru jabatan tenaga ahli *welder* berbasis *website* di PT. Ratu Mula Jadi. Sistem dibangun menggunakan metode *Profile Matching*. Pada sistem ini, peneliti dalam merekomendasikan pelamar yang diterima berdasarkan kriteria yang sudah ditentukan. Penelitian yang dilakukan memiliki perbedaan dengan penelitian sebelumnya. Perbedaan tersebut diantaranya pada setiap proyek yang dipegang oleh PT. Ratu Mula Jadi memiliki kriteria dan bobot yang berbeda dan diinputkan oleh HRD. Metode pengembangan perangkat lunak yang dipakai adala metode *Prototype.* Sistem ini dibuat dengan menggunakan bahasa *CodeIgniter* 4 dan *database mysql.*

2.2 Landasan Teori

2.2.1 Sistem Pendukung Keputusan

Dalam kehidupan sehari-hari akan ditemukan pemilihan keputusan yang akan dihadapi oleh manusia. Sistem pendukung keputusan secara sederhana adalah sebuah sistem komputer yang membantu dalam mengampil keputusan dengan kriteria dan alternatif. *Decision Supoort System* (DSS) atau sistem pendukung keputusan adalah sebuah sistem yang mampu memberikan kemampuan pemecahan masalah maupun kemampuan pengkomunikasian untuk masalah dengan kondisi semi terstruktur dan tak terstruktur[7].

Dalam sistem pendukung keputusan bagian dari *management system* yang terdiri dari *Decision Support System, Group Support System, Excutive Information System, Expert System, Artifical Neural Network, Hybird Support System.* Dalam pembuatan keputusan akan dihadapkan dengan kerumitan dan begitu banyak data. Oleh karena itu pembuatan keputusan akan mempertimbangkan resiko, manfaat, biaya dan dihadapkan pada suatu keharusan mengandalkan seperangkat sistem yang mampu memecahkan masalah secara objektif berdasarkan kriteria atau pertimbangan yang telah diberikan sebelumnya, kemudian sistem ini disebut sistem pendukung keputusan[8]. Tujuan sistem pendukung keputusan memberikan dukungan berdasarkan analisis data, informasi dan kriteria yang telah ditentukan, sehingga dapat menghasilkan keputusan yang lebih baik dan lebih terinformasi.

2.2.2 Profile Matching

Profile Matcing merupakan suatu proses yang sangat penting dalam manajemen sumber daya manusia yang dibutuhkan kemampuannya dalam sebuah jabatan[9]. *Profile Matching* secara garis besar merupakan proses membandingakan antara kompetensi individu ke dalam kompetensi kinerja sehingga dapat diketahui perbedaan komptensinya[10]. Kelibihan dari metode ini adalah kemudahan dalam pemahaman dan implementasi, terutama jika preferensi pengambilan keputusan sudah dinyatakan dengan jelas. Langkah-langkah dalam penyelesaian perhitungan dengan menggunakan metode *Profile Matcing* yaitu[11]:

1. Aspek penilaian.
   Langkah pertama yang harus dilakukan yaitu menentukan aspek-aspek penilaian pada *core factor* (faktor utama) dan *secondary factor* (faktor kedua).
2. Pemetaan GAP Kompetensi
   GAP Kompetensi adalah perbedaan antara kriteria-kriteria yang dimiliki seseorang dengan kriteria yang diinginkan, rumus GAP Kompetensi yaitu,
   GAP = Nilai Kriteria – Nilai Minimal ……...……(1)
3. Pembobotan
   Apabila pemetaan GAP sudah selesai dilakukan, maka hasil dari pemetaan tersebut diberi bobot nilai sesuai dengan yang telah ditentukan.
4. Perhitungan dan pengelompokan *Core Factor* dan *Secondary Factor*
   Setelah pembobotan nilai GAP ditentukan, maka dibagi menjadi 2 kelompok yaitu *core factor* dan *secondary factor*. Dengan rumus *core factor* sebagi berikut:
   $\boldsymbol{NCF = \sum NC / \sum IC}$ ……..……(2)
   Keterangan:
   NFC = nilai rata-rata *core factor*
   NC (aspek) = jumlah nilai *core factor*
   IC = jumlah *item core factor*
   Sedangkan rumus *secondary factor* sebagai berikut:
   $\boldsymbol{NSF = \sum NS}$ **(aspek) /** $\boldsymbol{\sum IC}$ …….(3)
   Keterangan:
   NSF = nilai rata-rata *secondary factor*
   NS (aspek) = jumlah nilai secondary factor
   IS = jumlah *item secondary factor*
5. Perhitungan Nilai Total
   Untuk menghitung nilai total, rumus yang digunakan yaitu:
   Nilai Total = (x)%NCF + (x)%NSP ……...(4)
   Keterangan:
   NCF (aspek)= nilai rata-rata core factor.
   NSF (aspek)= nilai rata-rata secondary factor.
   N(aspek) = nilai total dari aspek
   (x)% = nilai persen yang diinputkan

6. Perhitungan nilai rangkingan
   Menentukan nilai perangkingan mengacu pada hasil perhitungan menggunakan rumus sebagai berikut:
   Rangking
   (x)%NS……..……...…….(5)
   Keterangan:
   Ns = Nilai aspek
   (x)% = Nilai persen yang diinputkan

2.2.3 ColdeIgniter (CI)

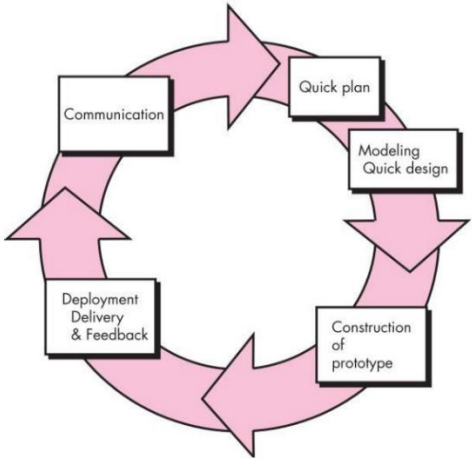
CodeIgniter merupakan sebuah kerangka kerja atau *freamwork* untuk pengembangan web berbasis bahasa pemrograman PHP. *Codeigniter* merupakan sebuah *freamwork* yang bersifat *open source* dan menggunakan metode *Mode View Controller* (MVC). *Freamwork* ini dikembangkan untuk membantu membangun aplikasi web dengan lebih cepat dan efisien tanpa harus membuat dari awal[12].

CodeIgniter memiliki fungsi untuk mengembangkan proyek jauh lebih cepat dari pada jika harus menulis *code* dari awal, dengan menyediakan banyak kumpulan pustaka kerangka program yang umumnya butuhkan, serta antarmuka sederhana dan logis untuk mengakses pustaka.

2.2.4 Metode Prototype

Metode *prototype* merupakan metode pengembangan perangkat lunak yang memungkinkan adanya interkasi antara pengembang sistem dengan pengguna sistem, sehingga dapat mengatasi ketidakserasian antara pengembang dan pengguna[13].

Metode *prototype* merupakan salah satu metode *software development life cycle* (SDLC) yang didasarkan pada konsep model bekerja. Metode *prototype* dinilai mampu menghasilkan sistem yang sesuai dengan kebutuhan pengguna, berikut langkah-langkah metode *prototype*[13]:



Gambar 2. 1 Gambar Metode *Prototype*

1. *Comunication*
   Pengumpulan kebutuhan bermaksud untuk mengetahui komponen yang akan digunakan selama pembuatan ataupun pengembangan aplikasi, seperti perangkat dan perangkat lunak. Dalam proses *comunication* untuk analisis kebutuhan dengan cara observasi dan wawancara.
2. *Quick Plan*
   Pada tahapan ini akan melakukan perencanaan cepat sesuai dengan spesifikasi kebutuhan *user* berdasarkan data yang telah dikumpulkan pada tahapan *comunniacation* dengan merancang *flowchart* untuk mengetahui komponen-komponen yang dibutuhkan dalam pembuatan sistem, serta *use case diagram* untuk mengetahui bagaimana jalannya sistem.
3. *Quick Design*
   Pada tahapan ini membuat model *design* sistem yang dibutuhkan dengan waktu perancangan yang efektif untuk mendeskripsikan kebutuhan dari pengguna berdasarkan analisis yang telah dilakukan sebelumnya.

4. Pembentukan *Prototype*
   Pada tahapan ini membuat sistem berdasarkan data yang telah dikumpulkan sebelumnya, proses pembangunan ini lebih berfokus terhadap aspek utama perangkat lunak dengan maksud pada saat proses selanjutnya perancangan bisa dengan cepat mendapatkan *feddback* dari pengguna tentang sistem yang dibuat.
5. *Development, Delevery and Feedback*
   a. *Development*
      Pada tahapan ini dilakukan jika *prototype* yang dibuat sudah diterima dan disepakati yang dilanjutkan dengan pengkodean ahasa program yang sesuai.
   b. *Delivery and Feedback*
      Setelah tahapan *development* dan pengujian selesai, *prototype* akan diserahkan ke pengguna untuk mendapatkan *feedback* dari hasil *prototype* yang telah diserahkan tersebut, *feedback* tersebut akan digunakan sebagai landasan untuk memperbaiki *prototype* agar sesuai dengan spesifikasi kebutuhan pengguna.

2.2.5 *Modeling View Controller*

Modeling view controller merupakan konsep pemisahan antara *logic* dengan tampilan dan *database* yang mana konsep ini diterapkan oleh *freamwork codeigniter*. MVC merupakan sebuah pola arsitekutr perangkat lunak yang digunakan dalam pengembangan aplikasi berbasis *web*. Pola ini membantu dalam mengorganisir kode aplikasi menjadi lebih terstruktur. Manfaat dari konsep ini adalah membuat *coding logic* menjadi lebih simple, karena terpisah dengan *code* tampilan dan membuat pengembang sistem dapat bekerja dengan mudah[14].

2.2.6 *Black Box Testing*

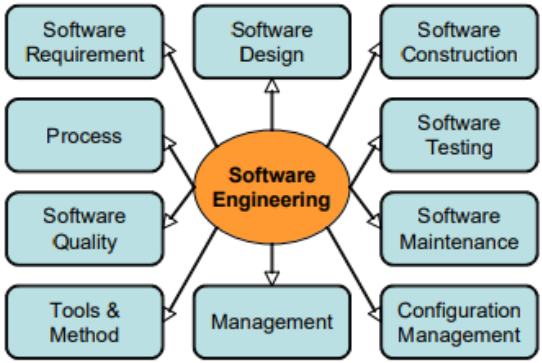
Black box testing merupakan teknik pengujian perangkat lunak yang dimana pengetahuan detail tentang struktur apabila tidak diperlukan. Fokusnya hanya aspek-aspke mendasar dari sistem yang berjalan. *Black box testing* untuk mengetahui fungsional pada perangkat lunak. Pengujian menggunakan *black box testing* umumnya lebih cepat dalam pelaksanaan. *Black box testing* cenderung dapat menemukan beberapa hal seperti fungsional yang tidak benar atau tidak ada, kesalahan basis data, kesalahan

struktur data, kesalahan akses data, kesalahan antarmuka, kesalahaan pengguna, kesalahaan *performance*, serta kesalahan inisialisasi dan terminasi[15].

2.2.7 Rekayasa Perangkat Objek (PBO)

Perangkat lunak *software* merupakan sekumpulan perintah untuk memproses sebuah informasi. Perangkat lunak berupa program atau prosedur. Program merupakan kumpulan perintah yang dimengerti oleh komputer sedangkan prosedur adalah perintah yang dibutuhkan oleh pengguna dalam memproses informasi.

Rekayasa perangkat lunak (*software engineering)* merupakan proses pengembangan perangkat lunak yang sistematis, terstruktur dan mematuhi standar. Rekayasa perangkat lunak bertujuan agar bidang rekayasa akan selalu berusaha menghasilkan *output* dengan kinerja yang tinggi, biaya yang rendah dan penyelesaian yang tepat. Berikut adalah ruang lingkup rekayasa perang lunak menurut Abran, 2004[15]. Seperti pada Gambar 2.2 ruang lingkup rekayasa perangkat lunak.



Gambar 2. 2 Gambar Ruang Lingkup Rekayasa Perangkat Lunak

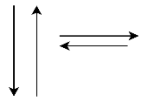
1. *Software requirement* berhubungan dengan spesifikasi kebutuhan dan persyaratan perangkat lunak.
2. *Software design* mencakup proses penetuan arsitektut, komponen, antarmuka dan karakteristik lain dari perangkat lunak.

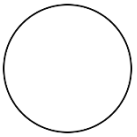
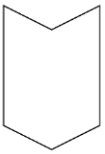
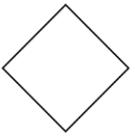
3. *Software construction* berhubungan dengan detik pengembangan perangkat lunak, termasuk algoritma, pengkodean, pengujian dan pencarian kesalahan.
4. *Software testing* meliputi pengujian pada keseluruhan perilaku perangkat lunak.
5. *Software maintenance* mencakup upaya-upaya perawat ketika perangkat lunak telah dioperasikan.
6. *Software configuration management* berhubungan dengan usaha perubahan konfigurasi perangkat lunak untuk memenuhi kebutuhan tertentu.
7. *Software engineering management* berkaitan dengan pengelolaan dan pengukuran RPL, termasuk perencanaan proyek perangkat lunak.
8. *Software engineering tools and methods* mencakup kajian teoritis tentang alat bantu dan metode RPL
9. *Software engineering process* berhubungan dengan definisi, implementasi, pengukuran, pengelolaan, perubahan dan perbaikan proses RPL.

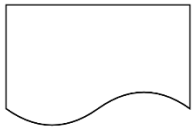
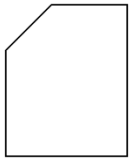
2.2.8 *Software quality* menitikberatkan pada kualitas dan daur hidup perangkat lunak *Flowchart*

Flowchart adalah jenis diagram yang mempresentasikan alogoritma atau langkah-langkah instruksi yang berurutan dalam sebuah sistem. *Flowchart* sebagai bukti dokumentasi untuk menjelaskan gambaran *login* sebuah sistem yang akan dikembangkan. Berikut adalah simbol-simbol yang dimiliki oleh *flowchart*. Berikut adalah tabel 2.1 simbol-simbol *flowchart*[16].

Tabel 2. 1 Tabel Simbol-Simbol *Flowchart*

No	Simbol	Nama	Keterangan
1.		*Connection line*	Digunakan untuk menghubungkan antara simbol yang yang satu dengan simbol yang lain.

2.		*Terminator*	Sebagai mulai (*star)* atau akhir (*stop)* dari sebuah program.
3.		*Connector*	Keluar masuk atau penyambungan proses dalam halam yang sama .
4.		*Connector*	Keluar masuknya atau penyambungan sebuah proses pada halaman yang berbeda.
5.		*Process*	Menunjukan pengolahan yang akan dilakukan pada sebuah sistem.
6.		*Input / output*	Merepresentasikan input atau output data proses atau informasi.
7.		*Manual operation*	Menunjukkan pengolahan yang tidak dilakukan oleh komputer.
8.		*Decision*	Pemilihan proses berdasarkan yang tidak dilakukan oleh komputer.

9.		*Predefine* Proses	Pelaksanaan suatu bagian (sub-program) atau prosedur.
10.		Dokumen	Menyatakan input berasal dari dokumen dalam bentuk kertas atau output dicetak ke kertas.
11.		*Punch Card*	Menyatakan bahwa input berasal dari kartu atau output tulis ke kartu.

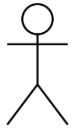
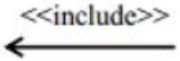
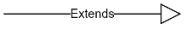
2.2.9 *Unified Modeling Langage* (UML)

Unified modeling language merupakan sebuah "bahasa" yang telah menjadi standa dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem perangkat lunak. UML ada karena adanya kebutuhan pemodelan visual untuk menspesifikan, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak[17]

1. Use Case Diagram

Use case diagram menggambarkan fungsional dari sebuah sistem, *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. Aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan program. *Use case* digunakan dalam pemodelan perangkat lunak untuk menggambarkan interaksi antara aktor-aktor *eksternal* dengan sistem yang sedang dikembangkan, berikut tabel 2.2 adalah simbol-simbol *use case_diagram*[17].

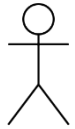
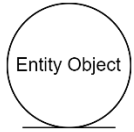
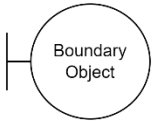
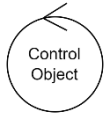
Tabel 2. 2 Tabel Simbol-Simbol *Use Case Diagram*

No	Simbol	Nama	Keterangan
1.		*Actor*	Menggambarkan semua objek di luar sistem yang berinteraksi dengan sistem yang dikembangkan.
2.		*Generaliization*	Relasi antara pengklafikasi yang memiliki deskripsi yang bersifat lebih umum dengan berbagai pengklafikasi yang lebih spesifik, digunakan dalam struktur pewarisan.
3.		*Include*	Penambahan perilaku ke suatu *use case* dasar yang secara eksplisit mendeskripsikan penambahan.
4.		*Extend*	Penambahan perilaku ke suatu *use case.*

2. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan disekitar sistem yang berupa message yang dikirimkan dan diterima oleh objek. Sequence diagram yang terdiri antara dimenasi vertikal atau waktu dan dimensi horizontal berupa objek-objek yang terkait. Berikut adalah simbol-simbol yang terdapat pada sequence diagram, berikut adalah tabel 2.3 simbol-simbol *sequence diagram*[15].

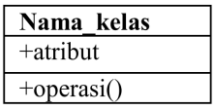
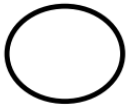
Tabel 2. 3 Tabel Simbol-Simbol *Sequence Diagram*

No	Simbol	Nama	Keterangan
1.		*Actor*	Menggambarkan user atau pengguna berinteraksi dengan sistem.
2.		*Entity Class*	Menggambarkan hubungan yang akan dilakukan.
3.		*Boundary Class*	Menggambarkan sebuah gambaran dari form.
4.		*Control Class*	Sebuah penghubung antara *boundary* dengan tabel.
5.		*A Focus Of Control & A Life Line*	Menggambarkan tempat mulai dan berakhirnya massage.
6.		*A Massage*	Mengirimkan pesan.

3. Class Diagram

Class adalah sebuah spesifikasi yang menggambarkan keadaan suatu sistem, sekalagus menawarkan kriteria untuk memanipulasi metode atau fungsi. Class diagram menggambarkan struktur dan deskripsiin class, package dan objek berserta hubungan satu sama lainnya seperti containment, pewaris dan asosias. Berikut adalah tabel 2.4 simbol-simbol class diagram [15].

Tabel 2. 4 Tabel Simbol-Simbol *Class Diagram*

No	Simbol	Nama	Keterangan
1.		*class*	Kelas pada struktur sistem.
2.		*interface*	Sama seperti konsep *interface* dalam pemprograman berorientasi obje.
3.		*association*	Menggambarkan sebuah gambaran dari form.
4.		*Directed Association*	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.

5.		*Generalisasi*	Relasi antarkelas dengan makna generalisasi-spesialisasi.
6.		*Dependency*	Relasi antar kelas dengan makna kebergantungan anterkelas
7.		*Aggregation*	Relasi antarkelas dnegan makna semua bagian.

2.2.10 Basis Data

Basis data terdiri dari dua kata yaitu basis dan data, basis dapat dikatakan gudang, markas, atau tempat berkumpul. Sedangkangkan data dapat diartikan sepresentasi dari fakta dunia yang mewakili sebuah objek yang direkam dalam bentung angka, huruf, simbol, teks, gambar, bungi atau kombinasi. Basis data merupakan kumpulan data dari berbagai sumber yang secara logika mempunyai arti *implicit* atau terkandung. Apabila data terkumpul secara acak dan tanpa mempunyai arti, tidak dapat disebut basis data. Basis data dapat digunakan oleh beberapa pemakai dan beberapa apliksi yang sesudai dengan kebutuhan pemaikai. Basis data perlu dirancang, dibangun dan data dikumpulkan untuk suatu tujuan tertentu[18].

1. (*Database Management System)* DBMS

Database Management System DBMS adalah perangkat lunak yang memungkinkan pemakai untuk mendefinisikan, mengelola dan mengontrol akses ke basis data. DBMS yang mengelola basis data relational disebut dengan Relational DBMS (RDBMS). Berikut adalah perangkat lunak yang mendukung DBMS yaitu, MS-SQL, Server, Oracle, Informix, Sybase, MySQL. Komponen-komponen yang dimiliki oleh DBMS sebagai berikut[19]:

a. *Query Prosesor*, komponen yang mengubah bentuk *query* kedalam instruksi kedalam *database manager*.
b. *Database Manager*, menerima *query* & menguji eksternal dan konseptual untuk menentukan apakah *record-record* tersebut dibutuhkan untuk memenuhi permintaan kemudian *database* manager memanggil file *manager* untuk menyelesaikan perminataan.
c. *File Manager*, memanipulasi penyimpanan file dan mengatur alokasi ruang penyimanan *disk.*
d. DML *Precompiler*, modul yang mengubah perintah DML yang ditempelkan kedalam program aplikasi dalam bentuk fungsi-fungsi.
e. DLL *Compiler*, merubah statement DDL menjadi kumpulan tabel atau file yang berisi data *dictionary* atau meta data.
f. *Dictionary Manager*, mengatur akses dan memelihata data *dictionary.*

2. *Structured Query Language* (SQL)

SQL merupakan bahasa *query* terapan yang banyak digunakan oleh berbagai DBMS, diterapkan dalam berbagai *development tools* dan program aplikasi untuk berinteraksi dengan basis data. SQL adalah bahasa standar dari basis data yang digunakan aplikasi atau pemakai untuk berinteraksi dengan basis data melalui DBMS, SQL memiliki 2 bagian yatu Data *Definision Language* (DLL) dan Data Manipulation Language DML[19].

3. *Data Definition Language* (DDL)

Data *Definition Language* dalam bahasa ini dapat membuat tabel baru, membuat indeks, menentukan struktur penyimpanan tabel. Hasil kompilkasi perintah DDL, disimpan dalam *file* khusus yang disebut Kamus Data (Data *Dictionary*). Kamus data adalah suatu metadata (Super-data) yaitu data yang mendeskripsikan data sesungguhnya[19].

4. Data *Manipulation Language* (DML)

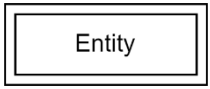
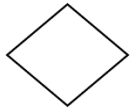
DML bahasa yang berguna untuk melakukan manipulasi data pada suatu basis data.Manipulasi dapat berupa: penambahan, penghapusan, pengubahan data pada suatu basis data. DML memiliki 2 tipe yaitu[19]:

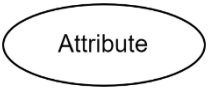
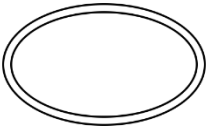
a. Prosedural, bahasa yang mensyaratkan pemakai untuk menentukan data apa yang diinginkan serta bagaimana cara mendapatkannya.
b. Non Prosedural, bahasa yang membuat pemakai dapat menentukan data apa yang diinginkan tanpa menyebutkan bagaimana cara untuk mendapatknya.

5. *Entity Relationship Diagram* (ERD)

ERD merupakan model untuk menjelaskan hubungan antar data dalam basis data berdasarkan suatu persepsi bahwa *real word* terdiir dari objek-objek dasar yang mempunyai hubungan atau relasi antara objek-objek tersebut. ERD dibuat berdasarkan anggapan bahwa dunia nyata terdiri dari koleksi objek-objek dasar yang dinamakan entitas (*entity*) serta hubungan (*relationship*) antara entitas-entitas. Entitas adalah "sesuatu" atau "objek" pada dunia nyata yang dapat dibedakan antara satu dengan yang lainnya, yang bermnafaat bagi aplikasi yang akan dikembangkan. ERD merupakan suatu model jaringan yang menggunakan susunan data yang disimpan dalam sistem secara abstrak. ERD biasa digunakan oleh pengembang sistem dengan pengguna guna mengembangkan sistem yang baik dan efektif. Berikah adalah tabel 2.5 simbol-simbol ERD[19].

Tabel 2. 5 Tabel Simbol-Simbol *Entity Relationship Diagram*

No	Simbol	Nama	Keterangan
1.		*Entity*	Objek atau konsep yang akan disimpan informasinya.
2.		*Weak Entity*	Harus dihubungkan kunci asing dengan entitas lain, karena tidak dapat mendefinisikan dengan atribut sendiri.
3.		*Relationship*	Hubungan antara satu entitas atau lebih.

4.		*Attribute*	Sekumpulan elemen data yang membentuk suatu entitas.
5.		*Attribute Mulinilai*	Sekelompok nilai untuk setiap instan entity/