

BAB II

LANDASAN TEORI

2.1. Sistem Pendukung Keputusan

Menurut (Wibowo, 2011) sistem pendukung keputusan ialah proses pengambilan keputusan dibantu menggunakan komputer untuk membantu pengambil keputusan dengan menggunakan beberapa data dan model tertentu untuk menyelesaikan beberapa masalah yang tidak terstruktur. Keberadaan SPK pada perusahaan atau organisasi bukan untuk menggantikan tugas-tugas pengambil keputusan, tetapi merupakan sarana yang membantu bagi mereka dalam pengambilan keputusan. Dengan menggunakan data-data yang diolah menjadi informasi untuk mengambil keputusan dari masalah-masalah semi-terstruktur. Dalam implementasi SPK, hasil dari keputusan-keputusan dari sistem bukanlah hal yang menjadi patokan, pengambilan keputusan tetap berada pada pengambil keputusan. Sistem hanya menghasilkan keluaran yang mengkalkulasi data-data sebagaimana pertimbangan seorang pengambil keputusan. Sehingga kerja pengambil keputusan dalam mempertimbangkan keputusan dapat dimudahkan.

Menurut (Fitriani, 2012) sistem pendukung keputusan dirancang untuk mendukung seluruh tahap pengambilan keputusan mulai dari mengidentifikasi masalah, memilih data yang relevan, dan menentukan pendekatan yang digunakan dalam proses

pengambilan keputusan sampai mengevaluasi pemilihan alternatif-alternatif yang ada.

Menurut (Khoiruddin, 2008) Dari beberapa definisi di atas dapat ditarik satu definisi tentang SPK yaitu sebuah sistem berbasis komputer yang adaptif, fleksibel, dan interaktif yang digunakan untuk memecahkan masalah-masalah tidak terstruktur sehingga meningkatkan nilai keputusan yang diambil.

2.1.1. Komponen Sistem Pendukung Keputusan

Dalam bukunya, Turban dan Aronson (2011: 85-88) menyatakan bahwa sebuah SPK dapat terdiri dari empat buah komponen, yaitu:

a. Subsistem Manajemen Data

Termasuk basis data yang berisi data-data relevant untuk situasi yang terjadi dan dikelola dalam sebuah piranti lunak yang disebut *database management system* (DBMS). Subsistem ini adalah bagian yang menangani semua penyimpanan maupun pengelolaan data dalam SPK.

b. Subsistem Manajemen Model

Subsistem Manajemen Model adalah sebuah paket piranti lunak yang meliputi model keuangan, statistik, ilmu manajemen, atau model kuantitatif lainnya yang menyediakan kemampuan analitis bagi sistem dan manajemen piranti lunak yang layak. Piranti lunaknya sering disebut *model database management system* (MBMS).

c. Subsistem Antarmuka

Subsistem antarmuka berfungsi sebagai penghubung pengguna dengan sistem. Pengguna dapat berkomunikasi dan memberi perintah pada sistem dengan menggunakan komponen-komponen yang disediakan pada antarmuka.

d. Subsistem Manajemen Berbasis Pengetahuan

Subsistem ini dapat berdiri sebagai komponen sendiri atau mendukung komponen lain. Fungsinya adalah untuk menyediakan intelijen untuk kepentingan sang pengambil keputusan.

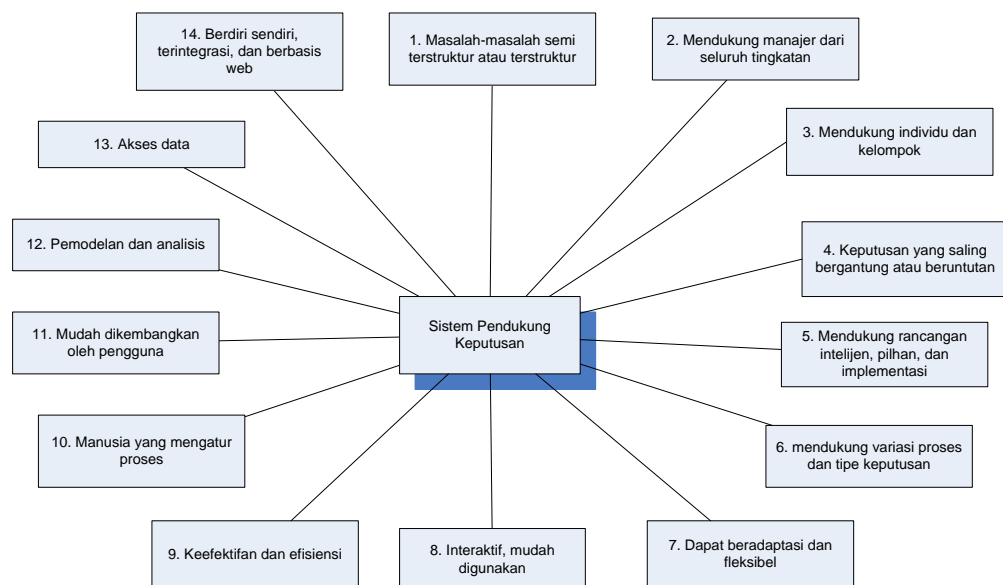
Sebuah SPK harus memiliki tiga komponen utama, yaitu DBMS, MBMS, dan antarmuka. Subsistem manajemen berbasis pengetahuan merupakan pilihan opsional.

2.1.2. Karakteristik Sistem Pendukung Keputusan

Karakteristik sistem pendukung keputusan menurut Wibowo (Wibowo, 2011) :

- a. Sistem Pendukung Keputusan dirancang untuk membantu pengambil keputusan dalam memecahkan masalah yang sifatnya semi terstruktur ataupun tidak terstruktur dengan menambahkan kebijaksanaan manusia dan informasi komputerisasi.
- b. Dalam proses pengolahannya, sistem pendukung keputusan mengkombinasikan penggunaan model-model analisis dengan teknik pemasukan data konvensional serta fungsi-fungsi pencari/interogasi informasi.
- c. Sistem Pendukung Keputusan, dirancang sedemikian rupa sehingga dapat digunakan/dioperasikan dengan mudah.
- d. Sistem Pendukung Keputusan dirancang dengan menekankan pada aspek fleksibilitas serta kemampuan adaptasi yang tinggi.

Menurut Turban dan Aronson (2011: 77), karakteristik yang menyatakan suatu sistem merupakan SPK ada 14. Karakteristik dan kemampuan inti SPK terangkas dalam gambar berikut ini:



Gambar 2.1: Karakteristik dan Kemampuan Inti SPK

Sumber: Turban dan Aronson (2011: 77)

2.1.3. Klasifikasi Sistem Pendukung Keputusan

Klasifikasi SPK bermacam-macam sesuai dengan tujuan dan strukturnya. Menurut Turban dan Aronson (2011: 79-81), Klasifikasi SPK termasuk dalam beberapa kategori di bawah ini.

a. *Communications-driven and group DSS*

SPK yang termasuk jenis ini adalah SPK yang menggunakan komputer, kolaborasi, dan teknologi komunikasi untuk mendukung tugas kelompok yang dapat melibatkan maupun tak melibatkan pengambilan keputusan.

b. *Data-driven DSS*

SPK jenis ini terutama berhubungan dengan data, memprosesnya menjadi informasi, dan menuajikannya untuk pengambil keputusan.

Dalam SPK jenis ini, organisasi database memiliki peranan besar dalam struktur SPK.

c. *Document-driven DSS*

SPK ini bergantung pada *knowledge coding* dan analisis. SPK jenis ini juga memiliki penekanan yang minimal terhadap pemanfaatan model matematis. Tujuan utama *document-driven DSS* ini adalah untuk menyediakan penunjang dalam mengambil keputusan dengan menggunakan dokumen dalam berbagai bentuk, yaitu: lisan, tertulis, dan multimedia.

d. *Knowledge-driven DSS, data mining, and management applications*

SPK jenis ini melibatkan aplikasi teknologi pengetahuan untuk membahas kebutuhan-kebutuhan dalam penunjang keputusan.

e. *Model-driven DSS*

Penekanan utamanya adalah menciptakan satu atau lebih optimisasi atau model simulasi yang biasanya menyertakan aktivitas penting dalam formulasi model, pemeliharaan model, manajemen model dalam lingkungan komputasi terdistribusi, dan *what-if analyses*. Fokus dari sistem ini adalah menggunakan model-model untuk mengoptimalkan satu atau lebih tujuan (misalnya keuntungan).

Selain kelima kategori tersebut, terdapat juga *compound DSS*. SPK ini terdiri dari dua atau lebih dari kategori-kategori yang telah disebutkan sebelumnya.

Tabel 2.1 Tabel Kategori *Decision Support System*

Orien- tasi	Kategori	Tipe Orientasi	Tipe Task	Pengguna	Pola yang Digunakan	Waktu
Data	Sistem penyim- panan data	Akses data	Operasi- onal	Personil non- manajer	Pencarian sederhana	Tidak teratur

	Sistem analisis data	Analisis <i>ad hoc</i> dari <i>data files</i>	Analisis Operasional	Staf analisis atau personil manajerial	Manipulasi dan tampilan data	Tidak teratur atau periodik
Data atau model	Sistem informasi analisis	Analisis <i>ad hoc</i> yang melibatkan lebih dari satu database dan model-model kecil	Analisis, perencanaan	Staf analisis	Pemrograman laporan khusus, mengembangkan model-model kecil	Tidak teratur, sesuai permintaan
Model	Model akuntansi	Perhitungan dasar yang memperkirakan hasil mendatang dengan dasar definisi akuntansi	Perencanaan, anggaran	Staf analisis atau manajer	Memasukkan perkiraan aktivitas; menerima hasil moneter yang diperkirakan sebagai keluaran (<i>output</i>)	Periodik
	Model representasional	Memperkirakan konsekuensi dari aksi-aksi tertentu	Perencanaan, anggaran	Staf analisis	Memasukkan keputusan yang memungkinkan; menerima hasil yang diperkirakan sebagai <i>output</i>	Periodik atau analisis tidak beraturan (<i>ad hoc</i>)
	Model optimisasi	Memperhitungkan solusi optimal dari	Perencanaan, alokasi	Staf analisis	Batasan <i>input</i> dan tujuan; menerima jawaban	Periodik atau analisis tidak

		kombinasi masalah	sumber daya			beraturan (<i>ad hoc</i>)
	Model perusulan	Melakukan perhitungan yang menghasilkan keputusan yang diusulkan	Operasional	Personil non-manajer	Memasukkan deskripsi terstruktur dari situasi keputusan; menerima keputusan yang diusulkan sebagai <i>output</i>	Harian atau periodik

Sumber: Turban dan Aronson (2011)

2.1.4. Konsep Keputusan

Pengambilan keputusan merupakan hal yang pokok bagi pemegang jabatan manajer. Karena keputusan merupakan rangkaian tindakan yang perlu diikuti dalam memecahkan masalah untuk menghindari atau mengurangi dampak negatif atau untuk memanfaatkan kesempatan di dalam perusahaan. Model sistem yang dipergunakan untuk mengambil keputusan dapat bersifat tertutup atau terbuka. Sistem pengambilan tertutup menganggap bahwa keputusan dipisahkan dari masukan-masukan yang tidak diketahui dari lingkungannya. Dalam sistem ini pengambil keputusan dianggap:

- a. Mengetahui semua alternatif dan akibat atau hasil dari masing-masing alternatif.

- b. Mempunyai suatu metode (aturan, hubungan dan sebagainya) yang memungkinkan ia membuat urutan alternatif yang lebih disukainya.
- c. Memilih alternatif yang memaksimalkan sesuatu seperti keuntungan, volume penjualan atau kegunaan.

Paham pengambilan keputusan yang tertutup jelas menganggap bahwa orang yang rasional secara logis menguji semua alternatif, membuat urutan berdasarkan hasilnya yang lebih disukai, dan memilih alternatif yang mendatangkan hasil terbaik.

Sistem pengambilan keputusan terbuka adalah keputusan yang dipengaruhi oleh lingkungan, dan proses pengambilan keputusan selanjutnya juga mempengaruhi lingkungan tersebut. Pengambil keputusan dianggap tidak harus logis dan sepenuhnya rasional, tetapi lebih banyak menunjukkan rasionalitas hanya dalam batas-batas yang ditentukan oleh latar belakang, penglihatan alternatif-alternatif, kemampuan untuk menangani model keputusan dan sebagainya. Mengingat tujuan model tertutup telah dirumuskan dengan baik, tujuan model terbuka sama dengan tingkat keinginan sebab model terbuka dapat berubah apabila pengambil keputusan menerima bukti keberhasilan atau kegagalan. Dibandingkan dengan ketiga anggapan model tertutup, model keputusan terbuka menganggap bahwa pengambil keputusan:

- a. Tidak mengetahui semua alternatif dan semua hasil.
- b. Melakukan penyelidikan secara terbatas untuk menemukan beberapa alternatif yang memuaskan.

- c. Mengambil keputusan yang memuaskan tingkat keinginannya.

Model terbuka adalah dinamis atas urutan pilihan-pilihan karena tingkatan keinginan berubah menangani perbedaan antara hasil dan tingkat keinginan.

2.1.5. Jenis-jenis Keputusan

Menurut Laudon dan Laudon (2010: 478), keputusan ada tiga jenis, yaitu:

- a. Keputusan tidak terstruktur

Untuk jenis keputusan ini, pembuat keputusan harus menyediakan penilaian, evaluasi, dan visi untuk menyelesaikan masalah. Keputusan-keputusan tersebut penting, tidak teratur, dan tak ada prosedur pasti dalam pembuatan keputusannya.

- b. Keputusan Semiterstruktur

Keputusan semi terstruktur memiliki karakteristik yang berada di tengah-tengah keputusan tidak terstruktur dan keputusan terstruktur. Hanya sebagian dari keputusan tersebut memiliki jawaban yang jelas dan terdapat prosedur penyelesaiannya.

- c. Keputusan Terstruktur

Keputusan terstruktur bersifat berulang dan rutin, serta terdapat prosedur yang jelas dalam menyelesaikannya.

2.2. Metode Analytical Hierarchy Process (AHP)

Menurut Kazibudzki dan Tadeusz (2013) Analytic Hierarchy Process (AHP) adalah pengambilan keputusan multikriteria dengan dukungan metodologi yang telah diakui dan diterima sebagai prioritas yang secara teori

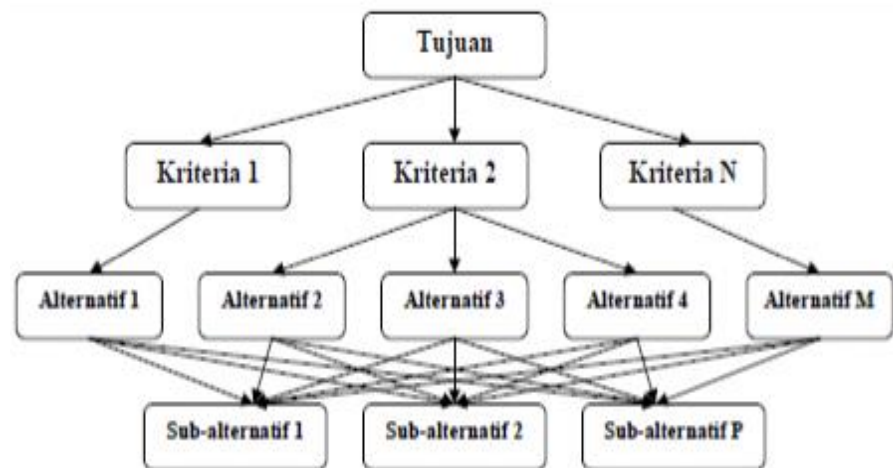
dapat memberikan jawaban yang berbeda dalam masalah pengambilan keputusan serta memberikan peringkat pada alternatif solusinya. Hirarki didefinisikan sebagai suatu representasi dari sebuah permasalahan yang kompleks dalam suatu struktur multi level dimana level pertama adalah tujuan, yang diikuti level faktor, kriteria, sub kriteria, dan seterusnya ke bawah hingga level terakhir dari alternatif.

Konsep metode AHP adalah merubah nilai-nilai kualitatif menjadi nilai kuantitatif. Sehingga keputusan-keputusan yang diambil menjadi lebih objektif. Metode AHP cukup mengandalkan intuisi sebagai input utamanya, namun intuisi tersebut harus datang dari pengambilan keputusan yang cukup informasi dan memahami masalah keputusan yang sedang dihadapi.

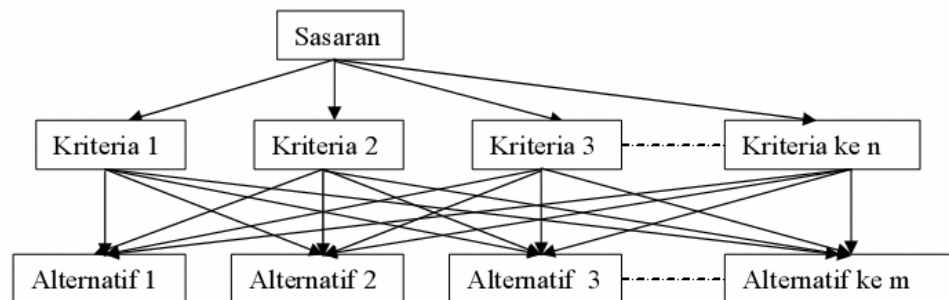
Untuk menyelesaikan masalah dengan menggunakan metode AHP, terdapat beberapa prinsip yang mendasari metode AHP, yaitu : *decomposition, comparative judgment, synthesis of priority dan logical consistency*.

- a. *Decomposition* Tahapan ini adalah pembuatan hierarki dari permasalahan yang dihadapi. Pembuatan hierarki perlu dilakukan untuk memecah persoalan yang utuh menjadi unsur-unsurnya. Struktur hirarki keputusan tersebut dapat dikategorikan sebagai complete dan *incomplete*. Suatu hirarki keputusan disebut *complete* jika semua elemen pada suatu tingkat memiliki hubungan terhadap semua elemen yang ada pada tingkat berikutnya, sementara hirarki keputusan *incomplete* kebalikan dari hirarki yang *complete* yakni tidak semua unsur pada masing-masing jenjang mempunyai hubungan (lihat gambar

2.2 dan 2.3). Pada umumnya problem nyata mempunyai karakteristik struktur yang *incomplete*. Bentuk struktur *decomposition* yakni :



Gambar 2.2 Struktur hirarki AHP



Gambar 2.3 Struktur hirarki AHP

Hirarki masalah disusun untuk membantu proses pengambilan keputusan dengan memperhatikan seluruh elemen keputusan yang terlibat dalam sistem. Sebagian besar masalah menjadi sulit untuk diselesaikan karena proses pemecahannya dilakukan tanpa memandang masalah sebagai suatu sistem dengan suatu struktur tertentu.

- b. Penilaian Kriteria dan alternatif (*Comparative judgment*) Prinsip ini berarti membuat penilaian tentang kepentingan relatif dua elemen pada

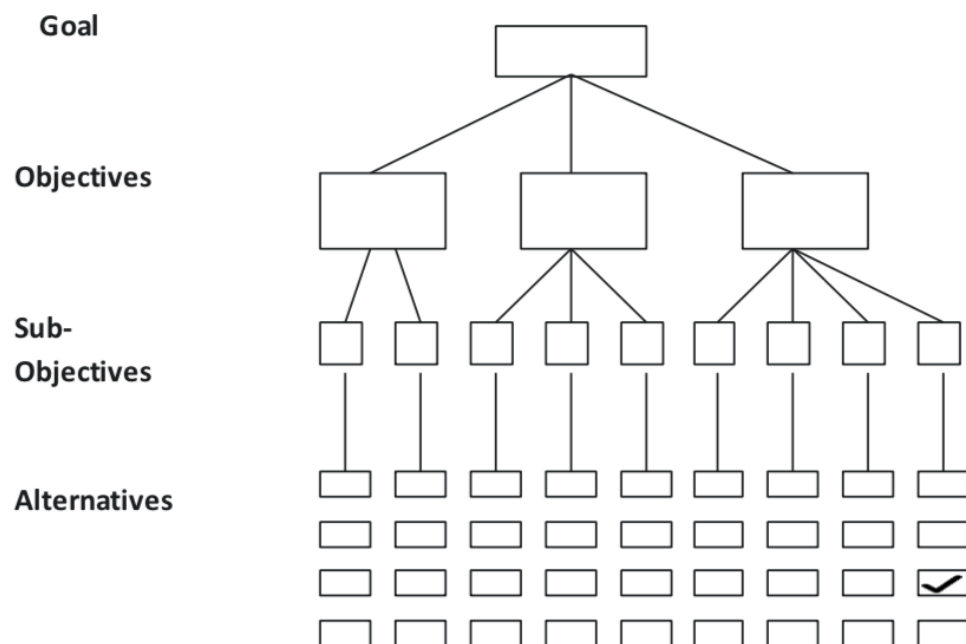
satu tingkat tertentu yang dalam kaitannya dengan satu tingkat di atasnya. Penilaian ini merupakan inti dari AHP, karena akan berpengaruh terhadap prioritas elemen-elemen.

- c. *Synthesis of priority* Dari setiap matriks *pairwise comparison* kemudian dicari *vector eigen* untuk mendapatkan *local priority*. Karena matriks *pairwise comparison* terdapat pada setiap tingkat, maka untuk mendapatkan *global priority* harus dilakukan sintesa di setiap *local priority*.
- d. *Logical consistency* Konsistensi memiliki dua makna. Pertama adalah bahwa objek-objek yang serupa dapat dikelompokkan sesuai dengan keseragaman dan relevansi. Makna yang kedua adalah menyangkut tingkat hubungan antar objek-objek yang didasarkan pada kriteria tertentu.

2.2.1. Langkah-langkah Perhitungan AHP

Untuk mendukung pengambilan keputusan yang akan dibuat ini, maka digunakan perhitungan bobot dengan metode AHP. Adapun tahapan dalam proses perhitungan bobot antara lain :

- a. Menyusun hirarki dari permasalahan yang dihadapi Persoalan yang akan diselesaikan diuraikan menjadi unsur-unsurnya, yaitu kriteria, subkriteria (bila ada) dan alternatif. Kemudian disusun menjadi struktur hirarki seperti pada gambar 2.4 di bawah ini :



Gambar 2.4 Struktur hierarki AHP

- b. Membuat matriks perbandingan berpasangan Pembuatan matriks perbandingan ini bertujuan untuk membandingkan tingkatan atau prioritas setiap elemen baik kriteria maupun subkriteria, kemudian menjumlahkan elemen setiap kolomnya untuk mendapatkan Σ kolom. Untuk lebih jelasnya dapat dilihat pada tabel 2.1.

Tabel 2.2 Matriks Perbandingan Berpasangan dan Penjumlahan Kolom

	K1	K2	...	Kn
K1	Nilai perbandingan K11
K2	+ ...	+...	...	+...
K3	+ ...	+...	...	+...
:	:	:	:	:
Kn	+ ... K1n	— +...	...	+...
Σ Kolom		—		

- c. Menghitung Total Priority Value (TPV) Untuk mendapatkan nilai bobot, kita harus menghitung TPV. Perhitungan TPV akan mengacu kepada Σ_{kolom} yang terdapat pada tabel 2.1. Untuk keterangan selengkapnya dapat dilihat pada tabel 2.2.

Tabel 2.3 Pembagian Nilai Perbandingan dengan Jumlah Kolom

	K1	K2	...	Kn
K1	Nilai perbandingan K11 / Σ_{kolom}
K2	Nilai perbandingan K12 / Σ_{kolom}
K3	Nilai perbandingan K13 / Σ_{kolom}
:	:	:	:	:
Kn	Nilai perbandingan K1n / Σ_{kolom}

Langkah selanjutnya untuk mendapatkan nilai TPV adalah menghitung menjumlahkan nilai elemen matriks setiap baris dari tabel 2.2 kemudian membagi jumlah baris tersebut dengan banyaknya kriteria (n) seperti pada tabel 2.3.

Tabel 2.4 Σ_{baris} dan Nilai TPV

	K1	K2	...	Kn	TPV
K1	Nilai hasil bagi K11	+...	...	+...	$\Sigma_{\text{baris1n/n}}$
K2	Nilai hasil bagi K12	+...	...	+...	$\Sigma_{\text{baris2n/n}}$
:	:	:	:	:	:
Kn	Nilai hasil bagi K13	$\Sigma_{\text{barisnn/n}}$

Keterangan :

K = Kriteria

n = Banyaknya Kriteria

TPV = Total Priority Value

Nilai TPV yang didapatkan dari hasil perhitungan ini merupakan nilai bobot dari setiap kriteria yang ada. Tahap-tahap diatas juga dilakukan apabila kita akan menghitung bobot subkriteria.

d. Memeriksa konsistensi matriks Dalam memeriksa konsistensi

matriks ada beberapa langkah, yaitu sebagai berikut :

- 1) Pertama bobot yang didapat dari nilai TPV dikalikan dengan nilai-nilai elemen matriks perbandingan yang telah diubah menjadi bentuk desimal, dan dilanjutkan dengan menjumlahkan nilai-nilai pada setiap baris, untuk lebih jelas, dapat dilihat pada tabel 2.4.

Tabel 2.5 Perkalian TPV dengan Nilai Elemen Matriks

K	TPV K1	TPV K2	TPV Kn
K1	Nilai perbandingan K11 * TPV K1	...	Nilai perbandingan K1n * TPV Kn
K2
K3
:	:	:	:
Kn	Nilai perbandingan Kn1 * TPV Kn	...	Nilai perbandingan Knn * TPV Knn

- 2) Kemudian pada tabel 2.5 ini merupakan hasil penjumlahan dari setiap barisnya.

Tabel 2.6 Penjumlahan Baris Setelah Perkalian

K	TPV K1	TPV K2	...	TPV Kn	Σ_{baris}
K1	Nilai perbandingan K11 * TPV K1	+...	...	+...	Σ_{barisk1}
K2	...	+...	...	+...	...
K3	...	+...	...	+...	...
:	:	:	:	:	:
Kn	Nilai perbandingan Kn1 * TPV Kn	+...	...	+...	Σ_{bariskn}

- 3) Setelah itu, kita perlu menghitung nilai λ_{maks} . Langkah pertama untuk menghitung nilai λ_{maks} adalah mencari nilai rata-rata setiap kriteria atau subkriteria. Dengan menghitung jumlah hasil pada langkah sebelumnya yaitu Σ_{baris} dibagi dengan TVP dari setiap kriteria.

$$\begin{pmatrix} \Sigma_{\text{baris}} K_1 \\ \dots \\ \Sigma_{\text{baris}} K_n \end{pmatrix} \div \begin{pmatrix} \text{TPV } K_1 \\ \dots \\ \text{TPV } K_n \end{pmatrix} = \begin{pmatrix} \lambda_{\text{maks}} K_1 \\ \dots \\ \lambda_{\text{maks}} K_n \end{pmatrix}$$

Kemudian akan diperoleh λ_{maks} dengan cara sebagai berikut :

$$\lambda_{\text{maks}} = \lambda_{\text{maks}} K_1 + \dots + \dots + \lambda_{\text{maks}} K_n \div n \text{ Keterangan : } \lambda_{\text{maks}}$$

= nilai rata – rata dari keseluruhan kriteria

n = jumlah matriks perbandingan suatu kriteria

- 4) Setelah mendapatkan nilai λ_{maks} kita perlu menghitung nilai Consistency Index (CI) dengan persamaan berikut ini : $CI = \lambda_{\text{maks}} - n$
- 5) Kemudian menghitung Consistency Ratio (CR). Nilai CR didapatkan dari hasil perhitungan CI dibagi dengan Random Index (RI). Nilai RI didapatkan dari tabel ketentuan sesuai dengan jumlah jumlah kriteria yang ada (n). Pada tabel 2.6 dapat dilihat nilai RI :

Tabel 2.7 Nilai Ratio Index (RI)

n	1	2	3	4	5	6	7	8	9
RI	0,000	0,000	0,580	0,900	1,120	1,240	1,320	1,410	1,450

n	10	11	12	13	14	15
RI	1,490	1,510	1,480	1,560	1,570	1,590

Sedangkan persamaan yang digunakan untuk perhitungan CR adalah sebagai berikut :

$$CR = \frac{CI}{RI}$$

Dari hasil perhitungan CR akan didapatkan nilai yang menjadi nilai pertimbangan rasio konsistensi. Nilai rasio akan diterima apabila $CR < 0,1$ dan perlu diperbaiki apabila $CR > 0,1$

2.2.2. Kelebihan AHP

AHP telah banyak penggunaannya dalam berbagai skala bidang kehidupan. Kelebihan metode ini dibandingkan dengan pengambilan keputusan kriteria majemuk lainnya adalah :

- a. Struktur yang berhirarki, sebagai konsekuensi dari kriteria yang dipilih pada sub-sub kriteria yang paling dalam
- b. Memperhitungkan validitas sampai dengan batas toleransi inkonsistensi berbagai kriteria dan alternatif yang dipilih oleh para pengambil keputusan
- c. Memperhitungkan daya tahan atau ketahanan output analisis sensitivitas pengambilan keputusan
- d. Metode AHP memiliki keunggulan dari segi proses pengambilan keputusan dan akomodasi untuk atribut-atribut baik kuantitatif maupun kualitatif
- e. Metode AHP juga mampu menghasilkan hasil yang lebih konsisten dibandingkan dengan metode-metode yang lainnya
- f. Metode AHP memiliki sistem yang mudah dipahami dan digunakan

2.2.3. Kekurangan AHP

Sedangkan kelemahan metode AHP diantaranya adalah sebagai berikut :

- a. Ketergantungan model AHP pada input utamanya. Input utama ini berupa persepsi seorang ahli sehingga dalam hal ini melibatkan subjektivitas sang ahli selain itu juga model menjadi tidak berarti jika ahli tersebut memberikan penilaian yang keliru.

- b. Metode AHP ini hanya metode matematis tanpa ada pengujian secara statistik sehingga tidak ada batas kepercayaan dari kebenaran model yang terbentuk.

2.3. Basis Data

Basis data terdiri atas dua kata yaitu basis dan data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang atau berkumpul. Sedangkan data adalah representasi dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya. Basis data sendiri dapat didefinisikan dalam sejumlah sudut pandang seperti :

- a. Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
- b. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan yang tidak perlu, untuk memenuhi berbagai kebutuhan.
- c. Kumpulan *file* atau tabel atau arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronik.

2.3.1. Perancangan Basis data

Perancangan basis data dimaksudkan untuk mendefinisikan isi atau struktur dari tiap-tiap file yang didefinisikan didesain secara umum. Elemen-elemen data di suatu file basis data garus dapat digunakan untuk pembuatan suatu *output*. Demikian juga dengan input yang direkam di basis data, file-file harus mampu mempunyai elemen-elemen untuk menampung input yang dimasukkan. Dengan demikian isi atau struktur suatu file basis data tergantung dari arus data masuk dan data keluar atau dari file.

2.3.2. Entity Relationship Diagram

ERD (*Entity Relationship Diagram*) merupakan model yang mendeskripsikan hubungan antar penyimpanan dalam DFD. ERD digunakan untuk memodelkan struktur data dan hubungan antar data. ERD menggunakan sejumlah notasi dan simbol untuk menggambarkan struktur dan hubungan antar data. Terdapat tiga simbol yang digunakan yaitu :

- a. Entitas, adalah suatu objek yang dapat diidentifikasi dalam lingkungan pemakai, sesuatu yang penting bagi pemakai dalam konteks sistem yang akan dibuat.
- b. Atribut, entitas mempunyai elemen yang disebut atribut dan berfungsi mendeskripsikan karakter entitas.
- c. Hubungan, entitas dapat berhubungan satu sama lain, hubungan ini dinamakan *relationship*. Sebagaimana halnya *entity* maka dalam hubungan juga harus dibedakan antara hubungan dan isi hubungan. Pada suatu hubungan antar entitas terdapat tiga jenis hubungan yaitu:

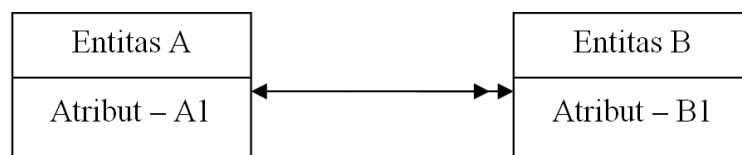
1) Hubungan satu ke satu (*One to one relationship*) Artinya

setiap entitas pada himpunan entitas pertama berhubungan dengan paling banyak satu entitas pada himpunan kedua, begitu juga sebaliknya.

Gambar 2.4 ERD dengan relasi satu ke satu

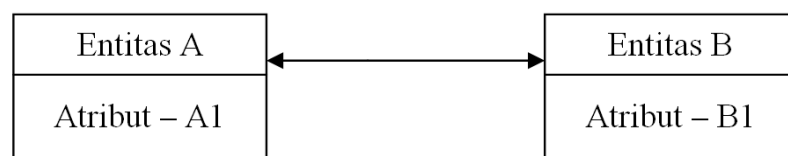
2) Hubungan satu ke banyak (*One to many relationship*) Artinya

setiap entitas pada himpunan entitas pertama berhubungan dengan banyak entitas pada himpunan entitas kedua, tetapi setiap entitas pada himpunan entitas kedua hanya dapat berhubungan dengan paling banyak satu entitas pada himpunan entitas pertama.

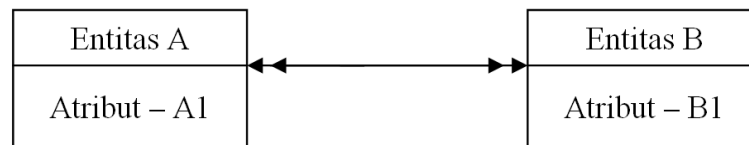


Gambar 2.5 ERD dengan relasi satu ke banyak

3) Hubungan banyak ke banyak (*Many to many relationship*) Artinya setiap entitas pada himpunan entitas



pertama berhubungan dengan banyak entitas pada himpunan entitas kedua, dan demikian juga sebaliknya.



Gambar 2. 6 ERD dengan relasi banyak ke banyak

2.3.2.1. Diagram Konteks

Diagram konteks adalah suatu alat atau metode penggambaran suatu sistem informasi secara global, baik sistem informasi yang berbasis komputer atau tidak berbasis komputer. Diagram konteks terdiri dari sebuah simbol proses yang mewakili keseluruhan proses dalam sistem dan minimal sebuah *external entity* (entitas luar) yang merupakan sumber atau tujuan data dari sistem tersebut dan aliran data yang menggambarkan aliran suatu masukan ataupun keluaran dari sistem tersebut. Berdasarkan notasi Yourdon proses digambarkan dengan lingkaran, entitas luar dengan persegi panjang, dan aliran data digambarkan dengan garis yang diberi mata panah.

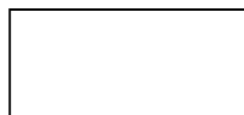
2.3.2.2. Diagram Arus Data

DAD (Diagram Arus Data) merupakan alat yang digunakan pada metodologi pengembangan sistem yang tersruktur (*Structured Analysis and Design*). DAD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut akan

disimpan. Untuk mewakili arus data dalam suatu sistem digunakan notasi atau simbol sehingga sangat membantu dalam komunikasi dengan pemakai sistem untuk memahami sistem secara logika. Beberapa simbol yang sering digunakan di DAD untuk maksud mewakili :

- a. *External entity* (kesatuan luar) atau *boundary* (batas sistem)

Setiap sistem pasti mempunyai batas sistem (*boundary*) yang memisahkan suatu sistem dengan lingkungan luarnya. Sistem akan menerima input dan menghasilkan *output* kepada lingkungan luarnya. Kesatuan luar (*external entity*) merupakan kesatuan di lingkungan luar sistem yang dapat berupa orang, organisasi atau sistem lainnya yang berada di lingkungan luarnya yang akan memberikan input atau menerima output dari sistem. Suatu kesatuan luar dapat disimbolkan dengan suatu notasi kotak sebagai berikut :

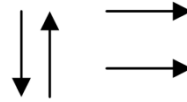


Gambar 2.7 Notasi Kesatuan Luar

- b. *Data flow* (arus data)

Arus data menunjukkan arus dari data yang dapat berupa masukan untuk sistem atau hasil dari proses sistem. Arus data ini mengalir diantara proses, simpanan data dan kesatuan luar. Arus data di DAD diberi simbol suatu panah. Arus data sebaiknya diberi nama yang

jelas dan mempunyai arti. Nama dari arus data dituliskan diatas garis panahnya sebagai berikut :



Gambar 2.8 Notasi Arus Data

c. *Process* (proses)

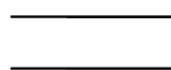
Suatu proses adalah kegiatan atau kerja yang dilakukan oleh orang, mesin atau komputer dari hasil suatu arus data yang masuk kedalam proses untuk dihasilkan arus data yang akan keluar dari proses. Suatu proses data ditunjukkan dengan simbol lingkaran. Setiap proses harus diberi penjelasan yang lengkap meliputi identifikasi proses, nama proses dan pemroses.



Gambar 2.9 Notasi Proses

d. *Data Store* (Simpanan data)

Simpanan data merupakan simpanan dari data yang dapat berupa suatu file atau database di sistem komputer. Simpanan data di DAD dapat disimbolkan dengan sepasang garis horizontal paralel.



Gambar 2.10 Notasi Simpanan Data

2.3.3. Kamus Data

Kamus data (*data dictionary*) adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Kamus data dibuat pada tahap analisis maupun pada tahap perencanaan sistem. Pada tahap analisis, kamus data dapat digunakan sebagai alat komunikasi antara analis sistem dengan pemakai sistem tentang data yang mengalir di sistem, yaitu tentang data yang masuk ke sistem dan tentang informasi yang dibutuhkan oleh pemakai sistem. Kamus data harus memuat hal-hal berikut ini:

- a. Nama arus data
- b. Panjang karakter
- c. Tipe data
- d. Deskripsi *field*

2.4. Unified Modeling Language (UML)

Unified Modeling Language (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat (*tools*) untuk mendukung pengembangan sistem tersebut. UML mulai diperkenalkan oleh *Object Management Group*, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh

para praktisi OOP. UML merupakan dasar bagi perangkat desain berorientasi objek dari IBM. UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri perangkat lunak dan pengembangan sistem.

UML menyediakan 10 (sepuluh) macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu:




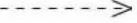






- a. *Use Case Diagram* untuk memodelkan proses bisnis.
- b. *Conceptual Diagram* untuk memodelkan konsep-konsep yang ada di dalam aplikasi.
- c. *Sequence Diagram* untuk memodelkan pengiriman pesan (*message*) antar objek.
- d. *Collaboration Diagram* untuk memodelkan interaksi antar objek.
- e. *State Diagram* untuk memodelkan perilaku objects di dalam sistem.
- f. *Activity Diagram* untuk memodelkan perilaku use case dan objects di dalam sistem.

- g. *Class Diagram* untuk memodelkan struktur kelas.
- h. *Object Diagram* untuk memodelkan struktur objek.
- i. *Component Diagram* untuk memodelkan komponen objek.
- j. *Deployment Diagram* untuk memodelkan distribusi aplikasi.

Pada penulisan ini, penulis menggunakan dua diagram UML, yaitu *use case diagram* dan *activity diagram*.

2.4.1. Use Case Diagram

Use case diagram yaitu salah satu jenis diagram pada UML yang menggambarkan interaksi antara sistem dan aktor, *use case diagram* juga


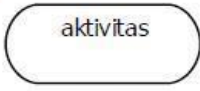



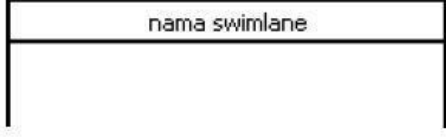
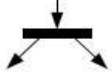

GAMBAR	NAMA	KETERANGAN
	<i>Actor</i>	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> .
	<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
	<i>Generalization</i>	Hubungan dimana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).
	<i>Include</i>	Menspesifikasikan bahwa <i>use case</i> sumber secara <i>eksplisit</i> .
	<i>Extend</i>	Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.
	<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.
	<i>System</i>	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
	<i>Use Case</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
	<i>Collaboration</i>	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (<i>sinergi</i>).
	<i>Note</i>	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

dapat mendeskripsikan tipe interaksi antara si pemakai sistem dengan sistemnya. Gambar 2.11 adalah simbol *usecase diagram*.

Gambar 2.11 Simbol *Use case diagram*

2.4.2. Activity Diagram

Activity diagram atau diagram aktivitas yaitu salah satu jenis diagram pada UML yang dapat memodelkan proses-proses apa saja yang terjadi pada sistem. *Activity diagram* juga berupa gambaran alur dari bagaimana suatu sistem mengawali, melakukan, dan mengakhiri proses tersebut bekerja. Gambar 2.12 adalah simbol *activity diagram*.

Simbol	Deskripsi
status awal 	status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
aktivitas 	aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
percabangan / <i>decision</i> 	asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
penggabungan / <i>join</i> 	asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
status akhir 	status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
swimlane 	memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi
<i>fork,</i> 	digunakan utk menunjukkan kegiatan yg dilakukan secara paralel
<i>join,</i> 	digunakan utk menunjukkan kegiatan yg digabungkan

Gambar 2.12 Simbol *activity diagram*

2.4.3. Class Diagram

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas sebagai berikut:

a. Kelas main

Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

b. Kelas yang menangani tampilan sistem

Kelas yang mendefinisikan dan mengatur tampilan kepemakai

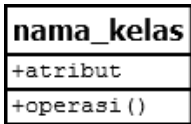
c. Kelas yang diambil dari pendefinisian use case







Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian use case

d. Kelas yang diambil dari pendefinisian data

Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Tabel 2.7 Simbol-simbol *class diagram*

Simbol	Deskripsi
<p>Kelas</p> 	Kelas pada struktur sistem
Antarmuka / interface	Sama dengan konsep interface dalam pemrograman berorientasi objek

	
Asosiasi / Association 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity
Asosiasi berarah / Directed association 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan / dependency 	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi / aggregation 	Relasi antar kelas dengan makna semua-bagian (whole-part)

2.5. Struktur Navigasi

Struktur navigasi adalah alur yang digunakan dalam aplikasi yang dibuat. Sebelum menyusun aplikasi multimedia ke dalam sebuah *software*, kita harus menentukan terlebih dahulu alur apa yang akan digunakan dalam aplikasi yang dibuat. Bentuk dasar dari struktur navigasi yang biasa digunakan dalam proses pembuatan aplikasi multimedia ada empat macam, yaitu struktur navigasi linier, hirarki, non linier, dan campuran.

2.5.1. Struktur Navigasi Linier

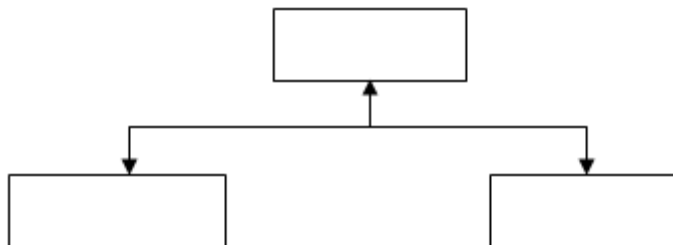
Struktur navigasi *linier* merupakan struktur yang mempunyai satu rangkaian cerita berurutan. Struktur ini menampilkan satu demi satu tampilan layer secara berurutan menurut aturannya. Gambar 2.13 adalah gambar struktur navigasi linier.



Gambar 2.13 Struktur Navigasi Linier

2.5.2. Struktur Navigasi Hirarki

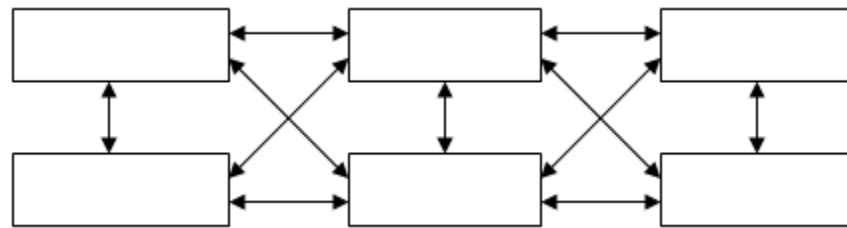
Struktur navigasi hirarki sering disebut struktur navigasi bercabang, yaitu merupakan suatu struktur yang mengandalkan percabangan untuk menampilkan data atau gambar pada layer dengan kriteria tertentu. Tampilan pada menu utama disebut *master page* (halaman utama satu), halaman tersebut mempunyai halaman percabangan yang disebut *slave page* (halaman pendukung) dan jika dipilih akan menjadi halaman kedua, begitu seterusnya. Gambar 2.14 adalah gambar struktur navigasi hirarki.



Gambar 2.14 Struktur Navigasi Hirarki

2.5.3. Struktur Navigasi Non Linier

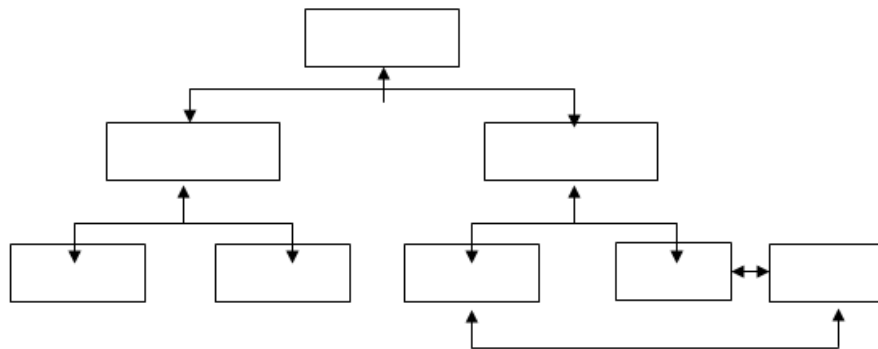
Struktur navigasi non linier (tidak teratur) merupakan pengembangan dari struktur navigasi *linier*, hanya saja pada struktur ini diperkenankan untuk membuat percabangan. Percabangan pada struktur *non linier* berbeda dengan percabangan pada struktur hirarki, pada struktur ini kedudukan semua *page* sama, sehingga tidak dikenal adanya *master* atau *slave page*. Gambar 2.15 adalah gambar struktur navigasi non linier.



Gambar 2.15 Struktur Navigasi Non Linier

2.5.4. Struktur Navigasi Campuran

Struktur navigasi campuran (*composite*) merupakan gabungan dari struktur sebelumnya dan disebut juga struktur navigasi bebas, maksudnya adalah jika suatu tampilan membutuhkan percabangan maka dibuat percabangan. Struktur ini paling banyak digunakan dalam pembuatan aplikasi multimedia. Gambar 2.16 adalah gambar struktur navigasi *composite*.



Gambar 2.16 Struktur Navigasi Campuran (*Composite*)

2.6. *Entity Relationship Diagram (ERD)*

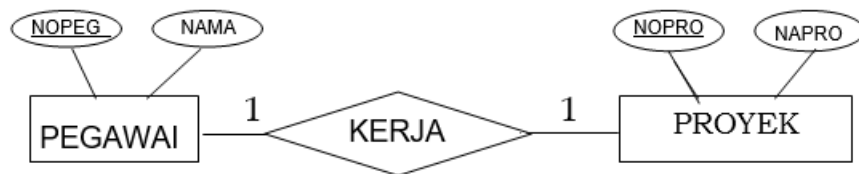
ERD adalah suatu pemodelan dari basis data relasional yang didasarkan atas persepsi di dalam dunia nyata, dunia ini senantiasa terdiri dari sekumpulan objek yang saling berhubungan antara satu dengan yang lainnya. Suatu objek disebut *entity* (entitas) dan hubungan yang dimilikinya disebut *relationship*. Suatu *entity* bersifat unik dan memiliki atribut sebagai pembeda dengan *entity* lainnya. Diagram ER terdiri dari:

- Kotak persegi panjang, menggambarkan himpunan entitas.
- Elip, menggambarkan atribut-atribut entitas.
- Diamond*, menggambarkan hubungan antara himpunan entitas.
- Garis, yang menghubungkan antar objek dalam diagram ER.

ER Diagram merupakan suatu bahasa pemodelan yang dimana posisinya dapat dianalogikan dengan *storyboard* dalam industri film, *blueprint* arsitektur suatu bangunan, miniatur, dan lain-lain. Dalam praktiknya, membangun suatu sistem terlebih dahulu dilakukannya suatu perencanaan. Pemodelan merupakan suatu sub bagian dari perencanaan secara keseluruhan

sebagai salah satu upaya *feedback* evaluasi perampungan suatu perencanaan. ER Diagram sebagai suatu pemodelan setidaknya memiliki beberapa karakteristik dan manfaat sebagai berikut:

- a. Memudahkan untuk dilakukannya analisis dan perubahan sistem sejak dini, bersifat murah dan cepat.
- b. Memberikan gambaran umum akan sistem yang akan di buat sehingga memudahkan developer.
- c. Menghasilkan dokumentasi yang baik untuk *client* sebagai bahan diskusi dengan bentuk ER Diagram itu sendiri.
- d. Kamus data bagi bagi para pengembang basis data.



Gambar 2.17 Contoh ER Diagram

2.6.1. Komponen ERD

Komponen utama ERD adalah *entity*, *relationship*, dan *attribute*.

a. *Entity*

Entity adalah objek yang dapat dibedakan dalam dunia nyata. Entity set adalah kumpulan dari entitas yang sejenis. Entity set dapat berupa: 1.

Objek secara fisik: Rumah, Kendaraan, Peralatan. 2. Objek secara

konsep: Pekerjaan , Perusahaan, Rencana

b. Relationship

Relationship adalah hubungan yang terjadi antara satu atau lebih entitas. *Relationship set* adalah kumpulan *relationship* yang sejenis.

c. Attribute

Attribute adalah karakteristik dari *entity* atau *relationship*, yang menyediakan penjelasan detail tentang *entity* atau *relationship* tersebut.

Nilai atribut merupakan suatu data aktual atau informasi yang disimpan pada suatu atribut di dalam suatu entitas atau *relationship*. Jenis-jenis atribut:

- 1) *Key* adalah atribut yang digunakan untuk menentukan suatu *entity* secara unik.
- 2) *Attribute Simple* adalah atribut yang bernilai tunggal.
- 3) *Attribute Multivalue* adalah atribut yang memiliki sekelompok nilai untuk setiap instant *entity*.
- 4) *Attribute Composite* adalah suatu atribut yang terdiri dari beberapa atribut yang lebih kecil yang mempunyai arti tertentu.
- 5) *Attribute Derivatif* adalah suatu atribut yang dihasilkan dari atribut yang lain.

2.6.2. Participation Constraint

Menjelaskan apakah keberadaan suatu *entity* tergantung pada hubungannya dengan *entity* lain. Terdapat 2 macam *Participation Constraint*:

a. Total Participation

Keberadaan suatu *entity* tergantung pada hubungannya dengan *entity* lain. Gambar 2.18 adalah contoh *Total Participation*.



Gambar 2.18 Contoh *Total Participation*

b. Partial Participation

Keberadaan suatu *entity* tidak tergantung pada hubungannya dengan *entity* lain. Gambar 2.19 adalah contoh *Partial Participation*.



Gambar 2.19 Contoh *Partial Participation*

2.6.3. Weak Entity

Weak Entity adalah suatu *entity* dimana keberadaan dari *entity* tersebut tergantung dari keberadaan *entity* lain. *Entity* yang merupakan induknya disebut *Identifying Owner* dan *relationshipnya* disebut *Identifying Relationship*. *Weak Entity* selalu mempunyai *Total Participation Constraint* dengan *Identifying Owner*.

2.6.4. Derajat *Relationship*

Menjelaskan jumlah entitas yang berpartisipasi dalam suatu *relationship*. Ada tiga derajat *relationship* yang digunakan dalam ERD, yaitu *unary* (derajat satu), *binary* (derajat dua) dan *ternary* (derajat tiga).

2.6.4.1. *Unary Degree*

Relationship Unary yang sering disebut juga *relationship* rekursif merupakan *relationship* antara beberapa *instance* dari satu entitas saja. Contoh *Unary degree* dapat dilihat pada Gambar 2.20.



Gambar 2.20 Contoh *Relationship Unary*

2.6.4.2. *Binary Degree*

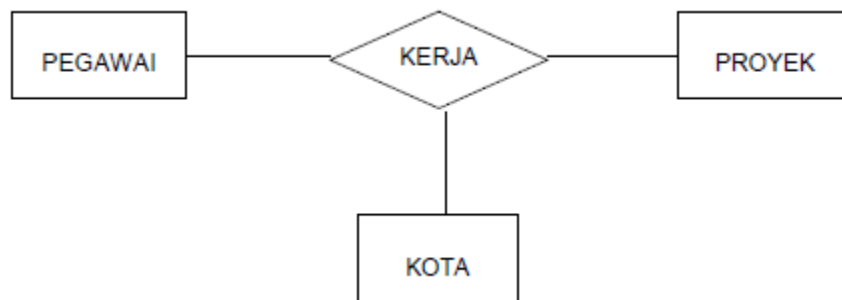
Relationship Binary merupakan *relationship* antara beberapa *instance* dari dua tipe entitas. *Relationship* ini paling umum digunakan dalam pembuatan model data. Contoh *binary degree* dapat dilihat pada Gambar 2.21.



Gambar 2.21 Contoh *Relationship Binary*

2.6.4.3. Ternary Degree

Relationship Ternary merupakan *relationship* antara beberapa *instance* dari tiga tipe entitas secara serentak. Contoh *ternary degree* dapat dilihat pada Gambar 2.22.



Gambar 2.22 Contoh *Relationship Ternary*

2.6.5. Mapping Entity Relationship Diagram

Entity Relationship Diagram bisa dituliskan menjadi suatu basis data secara fisik. Himpunan entitas dan relasi ditransformasikan menjadi tabel dan masing-masing atribut yang melekat akan dinyatakan sebagai *field* dari tabel yang sesuai. Berikut aturan-aturan dalam mentransformasi ERD menjadi basis data relasional:

- a. Setiap tipe *entity* dibuat suatu relasi yang memuat semua atribut *simple*, sedangkan untuk atribut *composite* hanya dimuat komponen-komponennya saja.
- b. Setiap relasi yang mempunyai atribut *multivalued*, buatlah relasi baru dimana *Primary Key* nya merupakan gabungan dari *Primary Key* dari relasi tersebut dengan atribut *multivalued*.

- c. Setiap *Unary Relationship* 1:N, pada relasi perlu ditambahkan suatu *Foreign Key* yang menunjukan ke nilai *Primary Key* nya.
- d. Setiap *Unary Relationship* M:N, buatlah relasi baru dimana *Primary Key*-nya merupakan gabungan dari dua atribut dimana keduanya menunjuk ke *Primary Key* relasi awal dengan penamaan yang berbeda.
- e. Setiap *Binary Relationship* 1:1, dimana *Participation Constraint* keduanya total, buatlah suatu relasi gabungan dimana *Primary Key* nya dapat dipilih salah satu.
- f. Setiap *Binary relationship* 1:1 dan salah satu *Participation Constraint* nya total, maka *Primary Key* nya pada relasi yang *Participation Constraint Partial* menjadi *Foreign Key* pada relasi yang lainnya.
- g. Setiap *Binary Relationship* 1:1, dimana kedua *Participation Constraint* nya *partial*, maka selain kedua relasi perlu dibuat relasi baru yang berisi *Primary Key* gabungan dari *Primary Key* kedua tipe *entity* yang berelasi.
- h. Setiap *Binary Relationship* 1:N dimana tipe *entity* yang bersisi N mempunyai *Participation Constraint* Total, maka *Primary Key* pada relasi yang bersisi 1 dijadikan *Foreign Key* pada relasi yang bersisi N.
- i. Setiap *Binary Relationship* 1:N, dimana tipe *entity* yang bersisi N mempunyai *Participation Constraint Partial*, buatlah relasi baru dimana *Primary Key* nya merupakan gabungan dari *Primary Key* kedua tipe *entity* yang berelasi.

- j. Setiap *Binary Relationship* N:N, buatlah relasi baru dimana *Primary Key* nya merupakan gabungan dari *Primary Key* kedua tipe entity yang berelasi.
- k. Setiap *Ternary relationship*, buatlah relasi baru dimana *Primary Key* nya merupakan gabungan dari *Primary Key* ketiga tipe entity yang berelasi.
- l. Setiap tipe *Weak Entity*, dibuat suatu relasi yang memuat semua atributnya dimana *Primary Key* nya adalah gabungan dari *partial key* dan *Primary Key* dari relasi induknya (*identifying owner*).

2.7.MySQL

MySQL dapat disebut sebagai sebuah implementasi dari sistem manajemen basis data relasional (RDBMS - *Relational Database Management System*) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public Lisen*ce). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial.

MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basis data yang telah ada sebelumnya, yaitu SQL. SQL (*Structured Query Language*) adalah bahasa yang digunakan untuk mengelola data pada RDBMS (*Relational Database Management System*). SQL awalnya dikembangkan berdasarkan teori aljabar relasional dan kalkulus.

2.8.HTML

HTML atau *Hyper Text Markup Language* adalah suatu format data yang digunakan untuk membuat dokumen *hypertext* (teks pada komputer yang memungkinkan *user* saling mengirimkan informasi (*request-respon*)). Dokumen HTML harus disimpan dengan ekstensi .htm atau .html. HTML memiliki *tag-tag* yang telah didefinisikan untuk membuat halaman *website*. Penulisan *tag-tag* html dapat menggunakan huruf besar atau huruf kecil, karena HTML tidak *case sensitive* (membedakan huruf besar dan huruf kecil memiliki maksud yang berbeda). Contoh sejumlah *tag* HTML dapat dilihat pada Tabel 2.7.

Tabel 2.7 Contoh Sejumlah Tag pada Sebuah Dokumen HTML

Tag	Keterangan
<html></html>	Tag dasar yang menandakan dokumen yang merupakan dokumen HTML.
<head>....</head>	Tag untuk mengisikan informasi tentang dokumen HTML.
<title>....</title>	Tag yang berada di dalam tag head untuk menuliskan judul website pada caption website.
<body>....</body>	Tag untuk mengisikan isi dokumen website yang ingin ditampilkan sebagai halaman website.
<p>....</p>	Menyatakan paragraf
<div>....</div>	Menyatakan divisi
<h1>....</h1> <h2>....</h2> <h3>....</h3>	Untuk mengatur judul, semakin besar angkanya maka ukuran font semakin kecil
<a>....	Untuk membuat tautan (link)
	Untuk menyajikan gambar
....	Untuk membuat bullet
....	Untuk membuat nomor urut
....	Daftar yang diatur oleh atau
<form>....</form>	Untuk menangani formulir yang berguna untuk memasukkan data oleh pemakai

<input type="tipe">	Untuk menentukan kontrol di formulir yang digunakan untuk memasukkan data
---------------------	---

2.8.1. HTML5

HTML5 adalah standar baru untuk HTML yang hadir setelah kemunculan HTML 4. Tujuan utama pengembangan HTML5 adalah untuk memperbaiki teknologi HTML agar mendukung teknologi multimedia terbaru, mudah dibaca oleh manusia, dan juga mudah dimengerti oleh mesin. Beberapa hal baru yang didukung oleh HTML5 tetapi tidak tersedia di pendahulunya antara lain, *kanvas*, *website SQL database*, audio (memungkinkan penyajian player untuk memutar suara), video (memungkinkan player untuk memainkan film), *drag and drop*, dan dokumen HTML5 diawali dengan <!doctype html>.

Adapun beberapa *tag* HTML yang tidak lagi didukung di HTML5 antara lain, <acronym>, <applet>, <basefont>, <big>, <dir>, , <center>, <frame>, <frameset>, <isindex>, <noframes>, <s>, <strike>, <tt>, <u>.

Dalam penulisan HTML5, ada struktur dasar yang biasa dipakai untuk membuat halaman *website*. Dengan menggunakan elemen HTML yang dimulai dengan *tag* awal yang diikuti dengan isi elemen, dan ditutup oleh *tag* akhir seperti berikut:

```
<!doctype html>

<html>

  <head>
    <title>Ini Judul HTML</title>
  </head>
  <body>
    Ini isi HTML
  </body>

</html>
```

2.9.Cascading Style Sheet (CSS)

CSS atau *Cascading Style Sheet* adalah suatu fasilitas untuk mempermudah pemeliharaan sebuah halaman *website*, dengan menggunakan CSS sebuah halaman *website* dapat diubah tampilannya tanpa harus mengubah dokumen HTML nya.

Ketika *tag* seperti `` dan atribut warna diperkenalkan sejak HTML 3.2, banyak *website developer* yang menjadi repot karena membuat banyak sekali informasi masuk ke dalam dokumen HTML, terlebih untuk *website* besar dimana banyak sekali *font*, warna, dan informasi khusus yang dimasukkan di tiap halaman. Sehingga ini menjadikan proses pembuatan dan manajemen *website* menjadi sangat lama dan melelahkan.

Untuk menanggulangi masalah ini, *World Wide Website Consortium* (W3C) menciptakan CSS. Dengan CSS, semua pemformatan bisa dihilangkan dari dokumen HTML dan diletakkan di file CSS terpisah.

sehingga ini memunculkan istilah baru, dimana ada pemisahan antara dokumen HTML dan CSS.

Style pada umumnya disimpan di file .css eksternal, ini memudahkan perubahan *style* dibandingkan jika file diletakkan satu file di dokumen HTML. Saat ini, hampir semua *browser* mendukung CSS, dan membuat pekerjaan para *website programmer* menjadi ringan. CSS membantu menentukan bagaimana elemen HTML disiapkan.

Format penulisan file CSS seperti berikut ini:

```
nama_style_atau_nama_tag_HTML {  
  
    nama_properti : nilai_properti;  
    .....  
    nama_properti : nilai_properti;  
  
}
```

2.9.1. Twitter Bootstrap

Bootstrap adalah kumpulan alat gratis untuk membuat *website* dan aplikasi *website*. Bootstrap ini berisi HTML dan CSS berbasis desain template untuk tipografi, bentuk, tombol, navigasi, dan komponen antarmuka lainnya, serta opsional ekstensi JavaScript. Bootstrap dikembangkan oleh Mark Otto dan Jacob Thornton di Twitter sebagai kerangka untuk mendorong konsistensi di alat internal. Sebelum Bootstrap, berbagai perpustakaan yang digunakan untuk pengembangan antarmuka, yang menyebabkan inkonsistensi dan beban pemeliharaan yang tinggi.

Bootstrap adalah modular dan pada dasarnya terdiri dari serangkaian *stylesheet LESS* yang menerapkan berbagai komponen *toolkit*. Sebuah *stylesheet* disebut *bootstrap.less* mencakup komponen-komponen *stylesheet*. Pengembang dapat menyesuaikan *file* Bootstrap sendiri, memilih komponen yang ingin mereka gunakan dalam proyek mereka.

Bootstrap dilengkapi dengan beberapa komponen JavaScript dalam bentuk *plugin* jQuery. Mereka menyediakan elemen antarmuka pengguna tambahan seperti kotak dialog, *tooltips*, dan *carousels*. Mereka juga memperluas fungsionalitas dari beberapa elemen antarmuka yang ada, termasuk misalnya fungsi *autocomplete* untuk bidang masukan. Dalam versi 2.0, plugin JavaScript berikut ini didukung: *Modal*, *Dropdown*, *Scrollspy*, *Tab*, *Tooltip*, *Popover*, *Alert*, *Button*, *Tutup*, *Carousel*, dan *Typehead*.

Untuk menggunakan Bootstrap dalam sebuah halaman HTML, pengembang cukup mengunduh Bootstrap CSS *stylesheet* dan termasuk *link* dalam *file* HTML yang tersedia di *website* resmi <http://getbootstrap.com/>.

2.10. Bahasa Pemrograman PHP

PHP diperkenalkan pada tahun 1994 sebagai sebuah kumpulan *script freeware* yang berbasis Perl dan dikenal sebagai "*Personal Home Page Tools*". Pembuatnya bernama Rasmus Lerdorf. Ternyata paket tersebut banyak mengundang minat para developer dan profesional. Pada tahun 1995, sebuah milis dibuat untuk menyediakan tempat diskusi termasuk memberikan *feedback*, perbaikan *bug* dan ide-ide kode *script* tersebut.

Terdorong untuk mengembangkan paket aslinya dengan fitur-fitur tambahan, Rerdorf mengeluarkan PHP-F1 (atau PHP2 pada tahun 1995). Versi ini sudah memiliki kemampuan untuk mengambil informasi yang dikirim dari *form website* dan mengubahnya menjadi variabel yang dapat digunakan. Hal yang penting dari fungsi ini adalah bahwa kita bisa menangkap dan mengolah variabel tadi sehingga memungkinkan pengembangan aplikasi *website* yang interaktif dan lebih kompleks.

Kira-kira pada waktu yang sama, PHP berubah dari pekerjaan satu orang menjadi pekerjaan kelompok yang terdiri dari 7 orang developer utama. Mereka memperbaiki sintaks dan menambahkan fungsi dan metode tambahan, serta kemampuan bagi programmer PHP lain untuk meningkatkan kemampuan bahasa PHP tersebut dengan modul-modul *plugin*.

Kemiripan PHP dengan bahasa pemrograman lain seperti C dan Perl, mendorong para *programmer* berpengalaman untuk pindah ke PHP dan secara cepat menumbuhkan pengguna-pengguna baru.

Contoh file PHP:

```
<html>
<?php
Print("Contoh text yang menggunakan kode PHP");
?>
</html>
```

2.10.1. Konsep MVC

Dengan menggunakan prinsip MVC suatu aplikasi atau *website* dapat di buat dengan lebih mudah. Dengan adanya prinsip MVC, pekerjaan pembuatan *website* pun dapat dikerjakan oleh dua bagian yang berbeda, untuk *front end*

website pada bagian *view* dapat dikerjakan oleh bagian *designer*, dan untuk bagian *back end website* dapat dikerjakan oleh *programmer*, ataupun jika pengerjaan *front end* dan *back end* dilakukan oleh satu orang *programmer* pun akan bisa menjadi lebih mudah dan rapih karena adanya prinsip MVC. Dengan adanya pembagian tugas yang sesuai dengan arsitektur MVC dapat meningkatkan *maintanability* dan koordinasi pembuatan program. Berikut adalah penjelasan singkat tentang MVC:

- a. *Model*, pada umumnya digunakan untuk akses dengan *database* berupa manipulasi data (*insert, update, delete, select*), validasi pada bagian *controller*.
- b. *View*, merupakan bagian utama dari *website*, digunakan untuk menampilkan halaman *website* atau antarmuka untuk pengguna aplikasi. Pada bagian *view* biasanya berupa *file HTML* yang diatur pemanggilannya oleh *controller*.
- c. *Controller*, merupakan bagian yang mengatur alur informasi data antara *model* dengan bagian *view*. Merupakan komponen yang digunakan untuk menangani interaksi pengguna, bekerja dengan *model*, dan memilih *view* mana yang digunakan untuk *render* data.

2.10.2. Codeigniter

CodeIgniter (CI) adalah *framework opensource* pengembangan aplikasi (*Application Development Framework*) dengan menggunakan PHP dengan

model MVC (*Model, View, Controller*), suatu kerangka untuk bekerja atau membuat program dengan menggunakan PHP yang lebih sistematis dan membuat *website* dinamis. CI dikembangkan oleh Rick Ellis, selain CI masih terdapat beberapa *framework* php seperti cake, symphony, yii, zend dan prado.

Tujuan dari pembuatan *framework* CI ini menurut petunjuk penggunaannya adalah pemrogram tidak perlu membuat program dari awal (*from scratch*), karena CI menyediakan sekumpulan *library* yang banyak diperlukan untuk menyelesaikan pekerjaan yang umum, dengan menggunakan antarmuka dan struktur logika yang sederhana untuk mengakses librarinya. Pemrogram dapat memfokuskan diri pada kode yang harus dibuat untuk menyelesaikan suatu pekerjaan dan dapat lebih cepat dalam membuat suatu *website*.

2.10.2.1. Keuntungan Framework CodeIgniter

Framework CI merupakan *framework* yang memiliki dokumentasi yang jelas dan lengkap, yang memudahkan pengembang untuk mempelajari dengan lebih mudah. CI adalah *framework* yang dapat digunakan secara gratis, dan dapat berjalan di PHP versi 4 dan 5, ringan, dan cepat.

Beberapa kelebihan yang dimiliki oleh CI adalah:

a. Performa yang cepat

CodeIgniter hanya *me-load* fungsi atau *library* yang digunakan saja, berbeda dengan *framework* lainnya yang menggunakan seluruh *library*

walaupun *library* tersebut tidak digunakan. Alasan inilah yang menjadikan CodeIgniter dengan akses tercepat dan ringan.

b. Dokumentasi

Framework yang baik pastinya dilengkapi dengan dokumentasi yang lengkap dan mendukung, agar bisa mudah dipahami oleh pengguna nya. Didukung oleh *user guide* yang mudah dimengerti, mulai dari langkah instalasi sampai dokumentasi fungsi-fungsinya tersedia.

c. Memakai konsep MVC

CodeIgniter memakai konsep MVC (*Model View Controller*), konsep modern yang banyak dipakai oleh *framework* PHP lainnya. Dengan adanya MVC, Pengerjaan antara logika dengan *layout* telah dipisahkan, sehingga antara *programmer* dan *designer* dapat fokus untuk melakukan tugasnya.

d. Komunitas

Framework CodeIgniter memiliki sebuah komunitas, dapat bergabung pada situs codeigniter.com/forums yang akan memudahkan untuk berinteraksi dengan sesama pengguna CI.