BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Secara sederhana suatu sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari suatu unsur, komponen, atau variabel yang terorganisir, saling tergantung satu sama lain dan terpadu. Teori sistem secara umum yang pertama kali diuraikan oleh Kennet Boulding, terutama menekan pentingnya perhatian terhadap setiap bagian yang membentuk sebuah sistem. Kecenderungan manusia yang mendapat tugas memimpin suatu organisasi adalah terlalu memusatkan perhatian pada salah satu komponen saja dari sistem organisasi.

Teori sistem melahirkan konsep-konsep futuristik, antara lain yang terkenal adalah konsep sibernetika (*cybernetics*). Konsep atau dibidang kajian ilmiah ini berkaitan dengan upaya menerapkan berbagai ilmu yaitu ilmu perilaku, fisika, biologi, dan teknik. Oleh karena itu sibernetika biasanya berkaitan dengan usaha-usaha otomasi tugas-tugas yang dilakukan manusia, sehingga melahirkan studi-studi tentang robotika, kecerdasan buatan (*artificial intelegence*). Unsur-unsur yang mewakili suatu sistem secara umum adalah masukan (*input*), pengolahan (*processing*), dan keluaran (*output*).

selain itu, suatu sistem tidak bisa lepas dari lingkungan maka umpan balik (*feed back*) dapat berasal dari lingkungan sistem yang dimaksud. Organisasi dipandang sebagai suatu sistem yang tentunya akan memiliki semua unsur ini (Tata Sutabri; 2012 : 10)

II.1.1. Karakteristik Sistem

Model umum sebuah sistem adalah input, proses, dan output. Hal ini merupakan konsep sebuah sistem yang sangat sederhana sebab sebuah sistem dapat mempunyai beberapa masukan dan keluaran. Selain itu sebuah sistem memiliki karakteristik atau sifat-sifat tertentu, yang mencirikan bahwa hal tersebut bisa dikatakan sebagai suatu sistem. Adapun karakteristik yang dimaksud adalah sebagai berikut:

1. Komponen Sistem (*Component*)

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, artinya saling berkerja sama membentuk satu kesatuan. Komponen-komponen sistem tersebut dapat berupa suatu bentuk subsistem. Setiap subsistem memiliki sifat dari sistem yang menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai sistem yang lebih besar, yang disebut "supra sistem

2. Batas Sistem *(Boundary)*.

Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem yang lain atau sistem dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat dipisah-pisahkan

3. Lingkungan Luar Sistem *(Environment)*.

Bentuk apapun yang ada di luar lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut operasi lingkungan luar sistem. Lingkungan luar sistem ini dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut. Lingkungan yang menguntungkan merupakan bagi sistem tersebut. Dengan demikian, lingkungan luar tersebut harus dijaga dan dipelihara. Lingkungan luar yang merugikan harus dikendalikan. Kalau tidak maka akan mengganggu kelangsungan hidup sistem tersebut

4. Penghubung Sistem *(Interface)*

Media yang menghubungkan sistem dengan subsistem lainnya disebut penghubung sistem atau *interface*. Penghubung ini memungkinkan sumber-sumber daya mengalir dari subsistem lain. Bentuk keluaran dari satu subsistem akan menjadi masukan untuk subsistem lain melalui penghubung tersebut. Dengan demikian dapat terjadi suatu integrasi sistem untuk membentuk satu kesatuan.

5. Masukan Sistem *(Input)*

Energi yang dimasukan ke dalam sistem disebut masukan sistem, yang dapat berupa pemeliharaan (*maintenance input*) dan sinyal (*signal input*). Contoh di dalam suatu sistem unit komputer. "program" adalah *maintenance*

input yang digunakan untuk mengoperasikan komputernya dan “data” adalah *signal input* untuk diolah menjadi informasi.

6. Keluaran Sistem (*Output*)

   yaitu hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain. Contoh, sistem informasi. Keluaran yang dihasilkan adalah informasi. Informasi ini dapat digunakan sebagai masukan untuk pengambil keputusan atau hal-hal lain yang menjadi *input* bagi subsistem lain

7. Pengolah Sistem (*Proses*).

   Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran. Contoh, sistem akuntansi. Sistem ini akan mengolah data transaksi menjadi laporan-laporan yang dibutuhkan oleh pihak manajemen.

8. Sasaran Sistem (*Objective*)

   Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat *deterministik*. Kalau suatu sistem tidak memiliki sasaran, maka operasi sistem tidak ada gunanya. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuan yang telah direncanakan (Tata Sutabri; 2012 :20-21).

II.1.2. Klasifikasi Sistem

Sistem merupakan suatu bentuk integrasi antara satu komponen dengan komponen lain karena sistem memiliki sasaran yang berbeda setiap kasus yang terjadi yang ada di dalam sistem tersebut. Oleh karena itu sistem dapat diklasifikasikan dar beberapa sudut pandangannya antara lain.

1. Sistem Abstrak Dan Sistem Fisik

   Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik, misalnya sistem teologia, yaitu sistem yang berupa pemikiran hubungan antara manusia dengan tuhan, sedangkan sistem fisik merupakan sistem yang ada secara fisik, misalnya sistem komputer, sistem

produksi, sistem penjualan, sistem administrasi personalia, dan lain sebagainya.

2. Sistem Alamiah Dan Sistem Buatan Manusia.

Sistem alamiah adalah sistem yang terjadi melalui proses alam, tidak dibuat oleh manusia, misalnya sistem perputaran bumi, terjadinya siang malam, pergantian musim. Sedangkan sistem buatan manusia dengan mesin, merupakan melibatkan interaksi manusia dengan mesin, yang disebut "*human machine system*". Sistem informasi berbasis komputer merupakan contoh *human machine system* karena menyangkut penggunaan komputer yang berinteraksi dengan manusia.

3. Sistem Deterministik Dan Sistem Probabilistik.

Sistem yang beroparasi dengan tingkah laku yang dapat diprediksi disebut sistem deterministic. Sistem komputer adalah contoh dari sistem yang tingkah lakunya dapat dipastikan berdasarkan program-program komputer yang dijalankan. Sedangkan sistem bersifat probabilistik adalah sistem yang mana kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilistic.

4. Sistem Terbuka Dan Sistem Tertutup.

Sistem tertutup merupakan sistem yang tidak berhubungan dan tidak terpengaruh oleh lingkungan luarnya. Sistem ini bekerja secara otomatis tanpa campur tangan pihak luar. Sedangkan sistem terbuka adalah sistem yang berhubungan dan dipengaruhi oleh lingkungan luarnya. Sistem ini menerima masukan dan menghasilkan keluaran untuk subsistem lainya (Tata Sutabri; 2012 : 22-26).

II.1.3. Pakar

Pakar adalah seseorang yang mempunyai pengetahuan, pengalaman, dan metode khusus serta mampu menerapkannya untuk memecahkan masalah atau memberi nasehat (T. Sutojo, dkk ; 2010:163).

II.2. Sistem Pakar

II.2.1. Pengertian Sistem Pakar

Sistem pakar adalah suatu sistem yang dirancang untuk dapat menirukan keahlian seorang pakar dalam menjawab pertanyaan dan memecahkan suatu masalah. Sistem pakar akan memberikan pemecahan suatu masalah yang didapat dari dialog dengan pengguna (T.Sutojo, dkk ; 2010:163).

II.2.2. Manfaat Sistem Pakar

Sistem pakar sangat populer karena sangat banyak kemampuan dan manfaat yang diberikannya, diantaranya :

1. Meningkatkan produktivitas, karena sistem pakar dapat bekerja lebih cepat daripada manusia.
2. Membuat seorang yang awam bekerja seperti layaknya seorang pakar.
3. Meningkatkan kualitas, dengan memberi nasehat yang konsisten dan mengurangi kesalahan.
4. Mampu menangkap pengetahuan dan kepakaran seseorang.
5. Dapat beroperasi dilingkungan yang berbahaya.
6. Memudahkan akses pengetahuan seorang pakar.
7. Andal. Sistem pakar tidak pernah menjadi bosan dan kelelahan atau sakit.
8. Meningkatkan kapabilitas sistem komputer, integrasi sistem pakar dengan komputer lain membuat sistem lebih efektif dan mencakup lebih banyak aplikasi.
9. Meningkatkan kemampuan untuk menyelesaikan masalah karena sistem pakar mengambil sumber pengetahuan dari banyak pakar. (T.Sutojo, dkk ; 2010:160).

II.2.3. Kekurangan Sistem Pakar

Selain manfaat, ada juga beberapa kekurangan yang ada pada sistem pakar, diantaranya :

1. Biaya yang sangat mahal untuk membuat dan memeliharanya.
2. Sulit dikembangkan karena keterbatasan dan ketersediaan pakar.
3. Sistem pakar tidak 100% bernilai benar. (T.Sutojo, dkk ; 2010:161).

II.2.4. Ciri – Ciri Sistem Pakar

Ciri – ciri dari sistem pakar adalah sebagai berikut :

1. Terbatas pada domain keahlian tertentu.
2. Dapat memberikan penalaran untuk data – data yang tidak lengkap atau tidak pasti.
3. Dapat menjelaskan alasan – alasan dengan cara yang dapat dipahami
4. Bekerja berdasarkan kaidah/rule tertentu
5. Mudah dimodifikasi.
6. Basis pengetahuan dan mekanisme inferensi terpisah.
7. Keluarannya bersifat anjuaran.
8. Sistem dapat mengaktifkan kaidah secara searah yang sesuai, dituntun oleh dialog dengan pengguna. (T.Sutojo, dkk ; 2010:162).

II.3. Konsep Dasar sistem Pakar

II.3.1. Kepakaran *(Expertise)*

Kepakaran merupakan suatu pengetahuan yang diperoleh dari pelatihan, membaca, dan pengalaman. Kepakaran inilah yang memungkinkan para ahli dapat mengambil keputusan lebih cepat dan lebih baik daripada seorang yang bukan pakar. Kepakaran itu sendiri meliputi pengetahuan tentang :

1. Fakta – fakta tentang bidang permasalahan tertentu.
2. Teori – teori tentang bidang permasalahan tertentu.
3. Aturan – aturan dan prosedur – prosedur menurut bidang permasalahan umumnya.
4. Aturan *heuristic* yang harus dikerjakan dalam suatu situasi tertentu.
5. Strategi global untuk memecahkan permasalahan.
6. Pengetahuan tentang pengetahuan (*meta knowledge*).
   (T.Sutojo, dkk ; 2010:161).

II.3.2. Pemindahan Kepakaran (*Transferring Expertise*)

Tujuan dari sistem pakar adalah memindahkan kepakaran dari seorang pakar kedalam komputer, kemudian ditransfer kepada orang lain yang bukan pakar. Prose ini melibatkan empat kegiatan yaitu:

1. Akuisisi pengetahuan (dari pakar atau sumner lain).

2. Repsentasi pengetahuan (pada komputer).
3. Inferensi pengetahuan.
4. Pemindahan pengetahuan ke pengguna. (T.Sutojo, dkk ; 2010:164).

II.3.3. Inferensi (*inferencing*)

Inferensi adalah sebuah prosedur (program) yang mempunyai kemampuan dalam melakukan penalaran. Inferensi ditampilkan pada suatu komponen yang disebut mesin inferensi yang mencakup prosedur – prosedur mengenai pemecahan masalah. Semua pengetahuan yang dimiliki seoarang pakar disimpan pada basis pengetahuan oleh sistem pakar. Tugas mesin inferensi adalah mengambil kesimpulan berdasarkan pengetahuan yang dimilikinya.

(T.Sutojo, dkk; 2010:164).

II.3..4. Aturan – aturan (*Rule*)

Kebanyakan software sistem pakar komersial adalah sistem yang berbasis rule (*rule-based systems*), yaitu pengetahuan disimpan terutam dalam bentuk rule sebagi prosedur – prosedur pemecahan masalah. (T.Sutojo, dkk ; 2010:165).

II.3.5. Kemampuan Menjelaskan (*Explanation Copability*)

Fasilitas lain dari sistem pakar adalah kemampuannya untuk menjelaskan saran atau rekomendasi yang diberikannya. Penjelasan dilakukan dalam subsistem yang disebut subsistem penjelasan (*explanation*). Bagian dari sistem ni memungkinkan sistem untuk memeriksa penalaran yang dibuatnya sendiri dan menjelaskan operasi – operasinya.

Karakteristik dan kemampuan yang dimiliki oleh sistem pakar berbeda dengan sistem pakar konvensional. Perbedaan ini ditunjukkan pada tabel II.1.

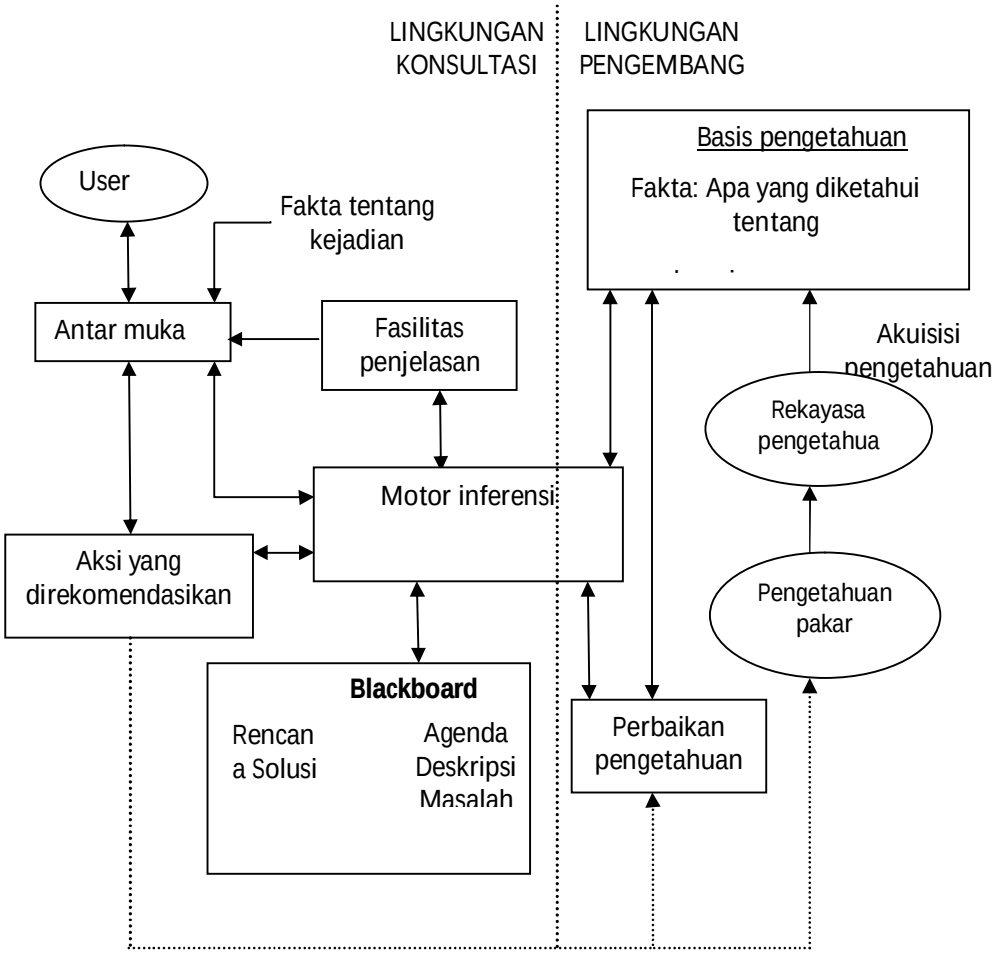
Tabel II.1. Perbandingan antara sistem konvensional dengan sistem pakar

Sistem Konvensional	Sistem Pakar
Informasi dan pemrosesannya biasanya digabungkan dalam suatu program.	Basis pengetahuan dipisahkan secara jelas dengan mekanisme inferensi.
Program tidak membuat kesalahan (yang membuat kesalahan: pemrogram atau pengguna.	Program dapat berbuat kesalahan.
Biasanya tidak menjelaskan mengapa dimasukkan diperlukan atau bagaimana output dihasilkan.	Penjelasan merupakan bagian terpenting dari semua sistem pakar.
Perubahan program sangat menyulitkan.	Perubahan dalam aturan – aturan mudah dilakukan.
Sistem hanya bisa beroperasi setelah lengkap atau selesai.	Sistem dapat beroperasi hanya dengan aturan – aturan yang sedikit (sebagai prototipe awal).
Manipulasi efektif dari basis data yang besar.	Manipulasi efektif dari basis pengetahuan yang besar

(Sumber : T.Sutojo, dkk ; 2010:165)

II.3.6. Struktur Sistem Pakar

Ada dua bagian penting dari sistem pakar, yaitu lingkungan pengembangan (development environment) dan lingkunagan konsultasi (consultation environment). Lingkungan pengembangan digunakan oleh pembuat sistem pakar untuk membangun komponen – komponennya dan memperkenalkan pengetahuan kedalam *knowledge base*. Lingkungan konsultasi digunakan oleh pengguna untuk berkonsultasi sehingga pengguna mendapatkan pengetahuan dan nasihat dari sistem pakar layaknya berkonsultasi dengan seorang pakar. Gambar II.1. menunjukkan komponen – komponen yang penting dalam sebuah sistem pakar. (T.Sutojo, dkk ; 2010:166).

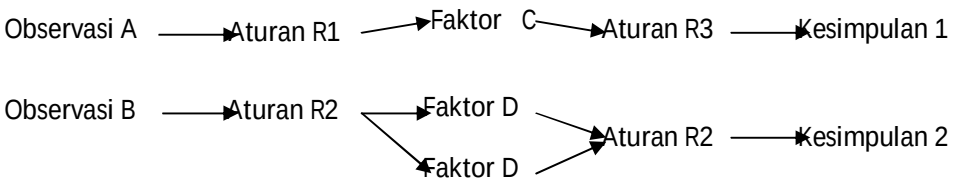


Gambar II.1. Komponen Sistem Pakar

(Sumber : T.Sutojo;2010:167)

II.4. Forward Chaining

Forward chaining adalah teknik pencarian yang dimulai dengan fakta yang diketahui, kemudian mencocokkan fakta – fakta tersebut dengan bagian IF dari *rules* IF-THEN. Bila ada fakta yang cocok dengan bagia IF, maka rule tersebut dieksekusi. (T.Sutojo, dkk ; 2010:171).



Gambar II.2. Proses Forward Chaining

(Sumber : Hersatoto Listiyono. 2010)

Forward chaining merupakan metode inferensi yang melakukan penalaran dari suatu masalah kepada solusinya. Jika klausa premis sesuai dengan situasi (bernilai TRUE), maka proses akan menyatakan konklusi. Forward chaining adalah *data driven* karena inferensi dimulai dengan informasi yang tersedia dan baru konklusi diperoleh. Jika suatu aplikasi menghasilkan tree yang lebar dan tidak dalam, maka gunakan forward chaining.

Tipe sistem yang dapat dicari dengan Forward Chaining :

1. Sistem yang dipersentasikan dengan satu atau beberapa kondisi.
2. Untuk setiap kondisi, sistem mecari rule-rule dalam knowledge base untuk rule-rule yang berkorespondensi dengan kondisi dalam bagian IF.
3. Setiap rule dapat menghasilkan kondisi baru dari konklusi yang diminta pada bagian THEN. Kondisi baru ini ditambahkan ke kondisi lain yang sudah ada.
4. Setiap kondisi yang ditambahkan ke sistem akan diproses. Jika ditemui suatu kondisi baru dari konklusi yang diminta, sistem akan kembali ke langkah 2 dan mencari rule-rule dalam knowledge base kembali. Jika tidak ada konklusi baru, sesi ini berakhir.(sumber : jurnal Arga Dian Setyo Wicaksono,2010)

Forward chaining juga merupakan grup dari *multiple inferensi* yang melakukan pencarian dari suatu masalah kepada solusinya. Metode *Forward chaining* dimulai dari sejumlah fakta-fakta yang telah diketahui, untuk

mendapatkan suatu fakta baru dengan memakai rule-rule yang memiliki ide dasar yang cocok dengan fakta dan terus dilanjutkan sampai mendapatkan tujuan atau sampai tidak ada rule yang punya ide dasar yang cocok atau sampai mendapatkan fakta.

Forward chaining bisa dikatakan sebagai *strategi inference* yang bermula dari sejumlah fakta yang diketahui. Pencarian dilakukan dengan menggunakan rulesyang premisnya cocok dengan fakta yang diketahui tersebut untuk memperoleh fakta baru dan melanjutkan proses hingga goal dicapai atau hingga sudah tidak ada rules lagi yang premisnya cocok dengan fakta yang diketahui maupun fakta yang diperoleh.*Forward chaining* bisa disebut juga runut maju atau pencarian yang dimotori data (data driven search). Jadi pencarian dimulai dari premis-premis atau informasi masukan (if) dahulu kemudian menuju konklusi atau *derived information (then). Forward Chaining* berartimenggunakan himpunan aturan kondisi-aksi. Dalam metode ini, data digunakan untuk menentukan aturan mana yang akan dijalankan atau dengan menambahkan data ke memori kerja untuk diproses agar ditemukan suatu hasil.

Forward Chaining digunakan jika :

1. Banyak aturan berbeda yang dapat memberikan kesimpulan yang sama.
2. Banyak cara untuk mendapatkan sedikit konklusi
3. Benar-benar sudah mendapatkan pelbagai fakta, dan ingin mendapatkan konklusi dari fakta-fakta tersebut.

Forward chaining juga digunakan jika suatu aplikasi menghasilkan tree yang lebar dan tidak dalam. Pada metode forward chaining, ada 2 cara yang dapat dilakukan untuk melakukan pencarian, yaitu :

1. Dengan memasukkan semua data yang tersedia ke dalam sistem pakar pada satu kesempatan dalam sesi konsultasi. Cara ini banyak berguna pada sistem pakar yang termasuk dalam proses terautomatisasi dan menerima data langsung dari komputer yang menyimpan database, atau dari satu set
2. Dengan hanya memberikan elemen spesifik dari data yang diperoleh selama sesi konsultasi kepada sistem pakar. Cara ini mengurangi jumlah data yang

diminta, sehingga data yang diminta hanyalah data-data yang benar-benar dibutuhkan oleh sistem pakar dalam mengambil kesimpulan.

Metode *inferensi* runut maju cocok digunakan untuk menangani masalah pengendalian (controlling) dan peramalan (prognosis) (Giarattano dan Riley, 1994). Teknik *Forward Chaining* merupakan teknik yang sering digunakan untuk proses inferensia yang memulai penalarannya dan sekumpulan data menuju kesimpulan yang dapat ditarik. Teknik Forward Chaining yaitu metode penalaran yang bergerak dan IF part menuju THEN part.

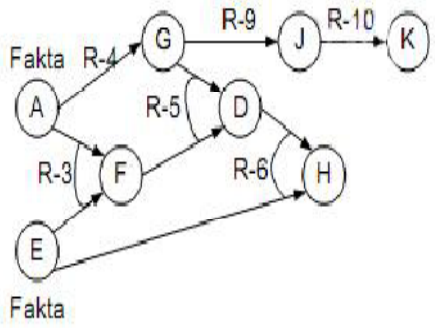
Dicontohkan pada tabel di bawah ini terlihat 10 aturan yang tersimpan dalam basis pengetahuan. Fakta awal yang diberikan hanya A & F (artinya A dan F bernilai benar). Ingin dibuktikan apakah K bernilai benar (hipotesis : K) ?

Tabel II.2. Perantaian Maju

No	Aturan
R-1	IF A & B THEN C
R-2	IF C THEN D
R-3	IF A & E THEN F
R-4	IF A THEN G
R-5	IF F & G THEN D
R-6	IF G & E THEN H
R-7	IF C & H THEN I
R-8	IF I & A THEN J
R-9	IF G THEN J
R10	IF J THEN K

(Sumber : Arga Dian Setyo Wicaksono;2010)

Penyelesaian dengan *Forward Chaining* pada Gambar berikut.



Gambar II.3 : Penyelesaian Forward Chaining

(Sumber : Arga Dian Setyo Wicaksono;2010)

Langkah-langkah yang harus dilakukan dalam membuat sistem *forward chaining* berbasis aturan, yaitu:

1. Pendefinisian Masalah

   Tahap ini meliputi pemilihan domain masalah dan akusisi pengetahuan.

2. Pendefinisian Data Input

   Sistem *forward chaining* memerlukan data awal untuk memulai inferensi.

3. Pendefinisian Struktur Pengendalian Data.

Aplikasi yang kompleks memerlukan premis tambahan untuk membantu mengendalikan pengaktifan suatu aturan.

4. Penulisan Kode Awal

   Tahap ini berguna untuk menentukan apakah sistem telah menangkap domain pengetahuan secara efektif dalam struktur aturan yang baik.

5. Pengujian Sistem

   Pengujian sistem dilakukan dengan beberapa aturan untuk menguji sejauh mana sistem berjalan dengan benar.

6. Perancangan Antarmuka

   Antarmuka adalah salah satu komponen penting dari suatu sistem. Perancangan antarmuka dibuat bersama-sama dengan pembuatan basis pengetahuan.

7. Pengembangan Sistem

Pengembangan sistem meliputi penambahan antarmuka dan pengetahuan sesuai dengan prototipe sistem.

8. Evaluasi Sistem

   Pada tahap ini dilakukan pengujian sistem dengan masalah yang sebenarnya. Jika

sistem belum berjalan dengan baik maka akan dilakukan pengembangan kembali.

Pada penalaran maju, aturan-aturan diuji satu demi satu dalam urutan tertentu, urutan tersebut mungkin berupa aturan ke dalam perangkat aturan atau dapat juga urutan lain yang ditentukan oleh pemakai. Dalam pengujian tersebut sistem pakar berusaha mengevaluasi apakah kondisinya benar atau salah. Jika kondisi benar, aturan tersebut ditembakkan dan aturan berikutnya diuji. Jika kondisinya salah, aturan tersebut tidak ditembakkan dan aturan berikutnya diuji. Jika kondisinya salah, aturan tersebut tidak ditembakkan dan aturan berikutnya diuji.Suatu aturan mungkin tidak dievaluasi sebagi benar atau salah. Mungkin kondisinya mencakup satu atau beberapa variabel dengan nilai yang tidak diketahui Dalam hal itu, kondisi aturannya tidak diketahui. Jika kondisi aturan tidak diketahui, aturan tidak ditembakkan dan aturan berikutnya diuji Proses pengujian aturan satu demi satu berlanjut sampai putaran lengkap melalui seluruh perangkat aturan. Biasanya diperlukan lebih dari satu putaran untuk memberikan suatu nilai pada variabel sasaran. Mungkin informasi yang diperlukan untuk mengevaluasi satu aturan dihasilkan oleh aturan lain yang diuji kemudian. Ketika tidak ada lagi aturan yang dapat ditembakkan, maka proses penalaran berhenti. (sumber : jurnal Arga Dian Setyo Wicaksono,2010)

II.5. PHP

Menurut dokumen resmi PHP, php merupakan singkatan dari PHP Hypertext Preprocessor. Ia merupakan bahasa berbentuk skrip yang ditempatkan dalam server dan di proses di server. Hasinyalah yang dikirimkan ke klien, tempat pemakai menggunakan browser.

Secara khusus, PHP dirancang untuk membentuk aplikasi web dinamis, artinya ia dapat membentuk suatu tampilan berdasarkan permintaan

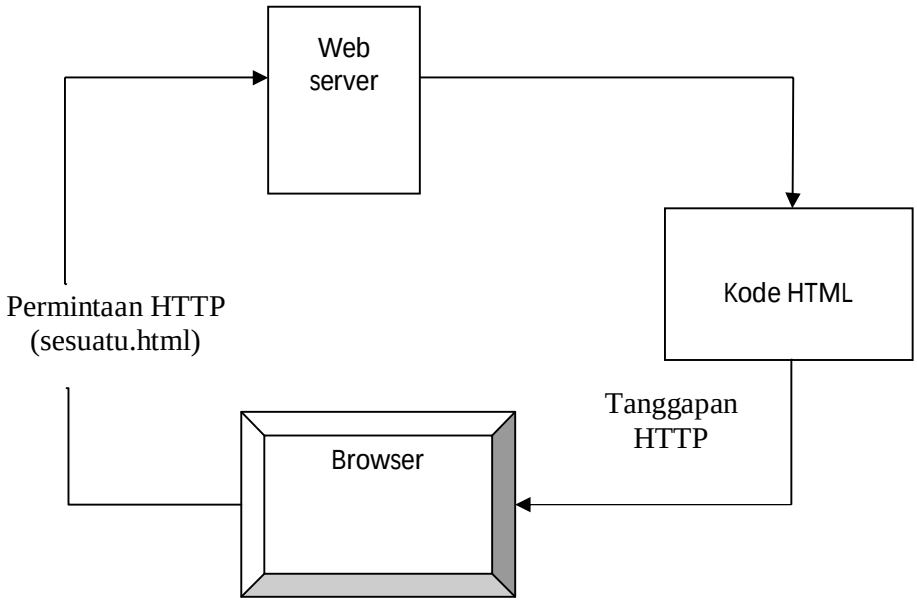
terkini.Misalnya anda bisa menamplkan isi database ke halaman web. Pada prinsipnya PHP mempunyai fungsi yang sama dengan skrip – skrip seperti ASP (Active Server Page), Cold Fusion, ataupun Perl. Namun perlu diketahui bahwa PHP sebenarnya bisa dipakai secara *command line* artinya skrip PHP dapat dijalankan tanpa melibatkan web server maupun browser.

Kelahiran PHP bermula saat Rasmus Lerdorf membuat sejumlah skrip perl yang dapat mengamati siapa saja yang melihat – lihat daftar riwayat hidupya, yakni pada tahun 1994. Skrip - skrip ini selanjutnya dikemas menjadi tool yang disebut ”Personal Home Page”, paket inilah yang menjadi cikal bakal PHP. Pada tahun 1995, Rasmus menciptakan menciptakan PHP/F1 versi 2. Pada versi inilah pemrograman dapat menempelkan kode terstruktur di dalam tag HTMl. Yang menarik kode PHP juga bisa berkomunikasi dengan database dan melakukan perhitungan – perhitungan yang kompleks.

Pada saat ini PHP cukup populer sebagai peranti pemrograman web, terutama dilingkungan linux. Walaupun demikian PHP sebenarnya dapat berfungsi pada server – server yang berbasis UNIX, Windows, Macintosh. Pada awalnya PHP dirancang untuk diintegrasikan dengan web server Apache, namun belakangan PHP juga dapat bekerja dengan web server seperti PWS (Personal Web Server),IIS (Internet Informasion Server), dan Xitami. (Abdul Kadir ; 2008 : 2).

II.5.1. Konsep Kerja PHP

Model kerja HTML diawali dengan permintaan suatu halaman web oleh browser. Berdasarkan URL (Uniform Resource Locator) atau dikenal dengan sebutan alamat internet, browser mendapat alamat dari web server, mengidentifikasi halaman yang dikehendaki dan menyampaikan segala informasi yang dibutuhkan oleh web server. Selanjutnya web server yang mencarikan file yang diminta dan memberikan isinya ke web browser, browser yang mendapatkan isinya segera melakukan proses penerjemahan kode HTML dan menampilkannya ke layar pemakai. (Abdul Kadir ; 2008 : 4-5).



Gambar II.4. Skema HTML

(Sumber : Abdul Kadir;2008 :5)

II.5.2. PHP dan Database

Salah satu kelebihan dari PHP adalah mampu berkomunikasi dengan berbagai database. Dengan demikian, menampilkan data yang bersifat dinamis, yang diambil dari database, merupakan hal yang mudah untuk di implementasikan. Itulah sebabnya sering dikatakan bahwa PHP sangat cocok untuk membangun halaman – halaman web dinamis.

Pada saat ini PHP sudah dapat berkomunikasi dengan berbagai database meskipun dengan kelengkapan yang berbeda – beda. Beberapa diantaranya DBM, FilePro, Informix, Ingres, Interbase, Microsoft Accsess, MSQL, MySQL, Oracle, Sybase, PostgreSQL. (Abdul Kadir ; 2008 : 6-7).

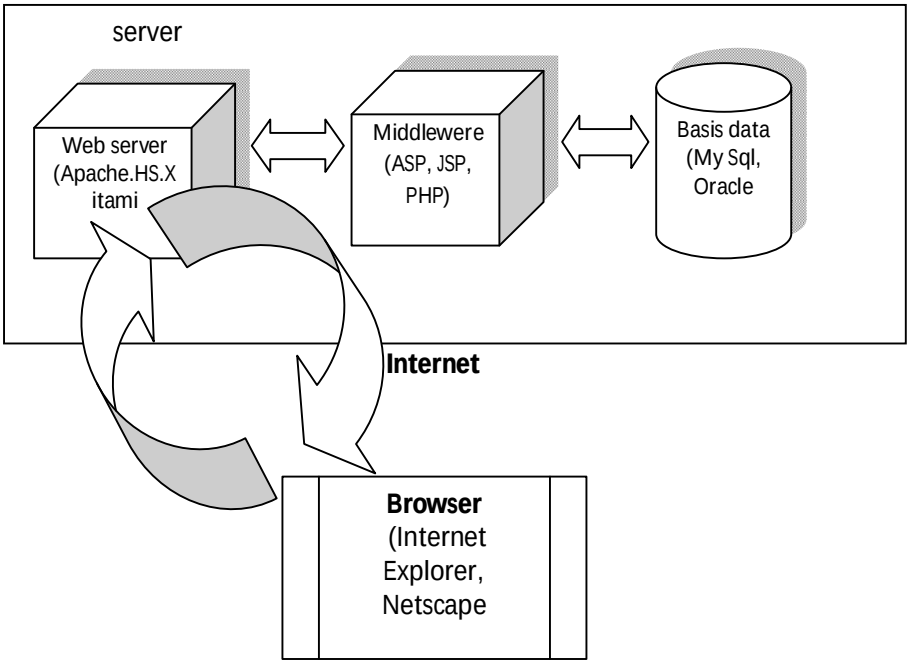
II.6. Web

World Wide Web (WWW) atau biasa disebut web, merupakan salah satu sumber daya internet yang berkembang pesat. Informasi web didistribusikan melalui pendekatan hypertext, yang memungkinkan suatu text pendek menjadi acuan yang membuka dokumen yang lain. Dengan pendekatan hypertext ini seseorang dapat memperoleh informasi dengan meloncat dari suatu dokumen ke dokumen yang lain. Dokumen – dokumen yang diakses pun dapat tersebar

diberbagai mesin dan bahkan diberbagai negara. Web telah membentang mempublikasikan hasil risetnya, web juga banyak digunakan oleh perusahaan bisnis yang ingin mengiklankan produk atau untuk melakukan tansaksi lainnya. (Abdul Kadir ; 2008 : 4).

II.6.1 Aplikasi Web

Pada Awalnya aplikasi web dibangun hanya dengan menggunakan bahasa yang disebut HTML dan protokol yang digunakan yang dinamakan HTTP. Pada perkembangan berikutnya sejumlah skrip dan objek dikembangkan untuk memperluas kemampuan HTML. Alikasi web sendiri dapat dibagi menjadi, web statis dibentuk dengan menggunakan HTML saja, kekurangan aplikasi ini terletak pada keharusan untuk memelihara program secara terus – menerus untuk mengikuti setiap perubahan yang terjadi, kelemahan ini diatasi dengan model aplikasi web dinamis. Sebagai implementasinya aplikasi web dapat dikoneksikan ke basis data, dengan demikian perubahan informasi dapat dilakukan oleh operator atau orang yang bertanggung jawab terhadap kemutakhiran data. (Abdul kadir ; 2008 : 5 - 6).



Gambar II.5. Arsitektur Aplikasi Web

(Sumber : Abdul Kadir;2008 :7)

II.7. Adobe Dreamweaver CS6

Dreamweaver CS6 adalah versi terbaru dari *adobe dreamweaver* yang merupakan bagian dari adobe Creative Suite 6. *Adobe dreamweaver* sendiri merupakan aplikasi yang digunakan sebagai *HTML* editor profesional untuk mendesain web secara visual. Aplikasi ini juga bisa dikenal dengan istilah WYSIWYG (*what you see is what you get*), yang intinya adalah anda tidak harus berurusan dengan tag – tag *HTML* untuk membuat sebuah site dan dapat melihat hasil desainnya secara langsung. Kemampuan adobe dreamweaver untuk berinteraksi dengan beberapa bahasa pemrograman seperti PHP, ASP, JavaScript, dan lainnya juga memberikan fasilitas maksimal kepada desainer web dengan menyertakan bahasa pemrograman didalamnya. (Madcoms ; 2013 :2)

II.8. *Entity Relationship Diargam* (ERD)

Entity Relationship (ER) data model didasarkan pada persepsi terhadap dunia nyata yang tersusun atas kumpulan objek-objek dasar yang disebut entitas dan hubungan antar objek. Entitas adalah sesuatu objek dalam dunia nyata yang

dapat dibedakan dari objek lain. Sebagai contoh, masing-masing mahasiswa adalah entitas dan mata kuliah dapat dianggap sebagai entitas.

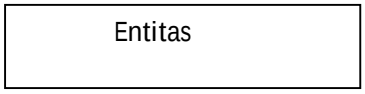
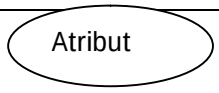
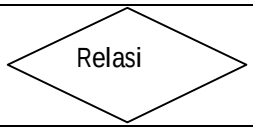
Entitas digambarkan dalam basis data dengan kumpulan atribut. Misalnya atribut nim, nama, alamat, dan kota bisa menggambarkan data mahasiswa tertentu dalam suatu universitas. Atribut-atribut membentuk entitas mahasiswa. Demikian pula, atribut kodeMK, namaMK, dan SKS mendeskripsikan mata kuliah.

Atribut NIM digunakan sebagai untuk mengidentifikasikan mahasiswa secara unik karena dimungkinkan terdapat dua mahasiswa dengan nama, alamat, dan kota yang sama. Pengenal unik harus diberikan pada masing-masing mahasiswa.

Relasi adalah hubungan antara beberapa entitas. Sebagai contoh relasi menghubungkan mahasiswa dengan mata kuliah yang diambilnya. Kumpulan semua entitas bertipe sama disebut kumpulan entitas (*entitas sel*), sedangkan kumpulan semua relasi bertipe sama disebut dengan kumpulan relasi (*relationship sel*).

Struktur logis (skema database) dapat ditunjukkan secara grafis dengan diagram ER yang dibentuk dari komponen-komponen berikut pada dilihat pada tabel II.3.

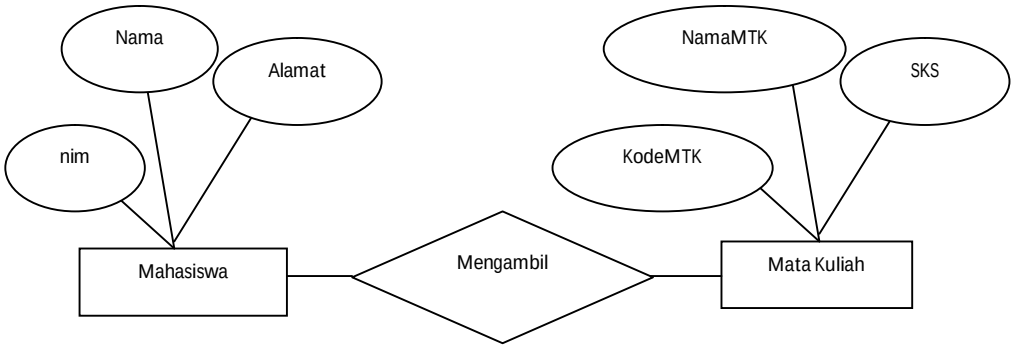
Tabel II.3. Komponen-Komponen Diagram ER

	Persegi Panjang mewakili kumpulan entitas
	Elips Mewakili Atribut
	Belah Ketupat Mewakili Relasi
	Garis Menghubungkan atribut dengan kumpulan entitas dan kumpulan entitas dengan relasi

(Sumber : Janner Simarmata, dkk ; 2009 :60)

Masing-masing komponen diberi nama entitas atau relasi yang diwakilinya.

Sebagai ilustrasinya bayangkan anda mengambil bagian sistem basis data universitas yang terdiri dari mahasiswa dan mata kuliah. gambar II.4. menunjukkan diagram ER dari contoh. Diagram menunjukkan bahwa ada dua kumpulan entitas yaitu mahasiswa dan mata kuliah dan bahwa relasi mengambil contoh mahasiswa dan mata kuliah (Janner Simarmata; 2009 : 59-60)



Gambar II.6. Diagram ER

(Sumber : Janner Simarmata, dkk ; 2009 : 60)

II.8.1. Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basis data relasional. Pada dasarnya normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel relasional (*www. utexas. edu*). Bentuk normalisasi yang sering dikenal yaitu sebagai berikut :

1. Bentuk Nornal Pertama (1 NF)

   Contoh yang kita gunakan di sini adalah sebuah perancangan yang mendapatkan barang dari sejumlah pemasok. Masing-masing pemasok berada pada satu kota. Sebuah kota dapat mempunyai lebih dari satu pemasok dan masing-masing kota mempunyai kode status tersendiri. Masing-masing pemasok bisa menyediakan banyak barang. Tabel relasionalnya dapat dituliskan sebagai berikut :

PEMASOK (P#, Status, Kota, b#, qty) di mana

p# : kode pemasok (kunci utama) status : kode status kota

Kota : nama kota

b# : barang yang dipasok

qty : jumlah barang yang dipasok.

Sebuah tabel relasional secara defenisi selalu berada dalam bentuk normal pertama. Semua nilai pada kolom-kolomnya adalah atomi. Ini berarti kolom-kolom tidak mempunyai nilai berulang.

Tabel II.4. Normalisasi Pertama Pemasok

P#	Status	Kota	B#	Qty
P1	20	Yogyakarta	B1	300
P1	20	Yogyakarta	B2	200
P1	20	Yogyakarta	B3	400
P1	20	Yogyakarta	B4	200
P1	20	Yogyakarta	B5	100
P1	20	Yogyakarta	B6	100
P2	10	Medan	B1	300
P2	10	Medan	B2	400
P3	10	Medan	B2	200
P4	20	Yogyakarta	B2	200
P4	20	Yogyakarta	B4	300
P4	20	Yogyakarta	B5	400

(Sumber : Janner Simarmata, dkk ; 2009 :80)

2 Bentuk Normal Kedua (2 NF).

Defenisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada 1 NF, tetapi tidak pada 2 NF, sebuah tabel relasional berada pada bentuk normal kedua jika dia berada pada 1 NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada kunci utama. Ini berarti bahwa setiap kolom bukan kunci harus tergantung pada seluruh kolom yang membentuk kunci utama. Tabel pemasok berada pada 1 NF, tetapi tidak pada 2 NF karena status dan kota tergantung secara fungsional hanya pada kolom p# dari kunci gabungan (p#, b#). Ini dapat digambarkan dengan membuat daftar ketergantungan fungsional.

P# ⟶ Kota, Status

Kota ⟶ Status

(P#, B#) ⟶ qty

Proses mengubah tabel 1 NF ke 2 NF adalah :

a. tentukan sembarang kolom penentu selain kunci gabungan dan kolom-kolom yang ditentukannya.
b. Buat dan beri nama tabel baru untuk masing-masing penentu dan kolom-kolom yang ditentukan.
c. Pindahkan kolom-kolom yang ditentukan dari tabel asal ke tabel baru penentu akan menjadi kunci utama pada tabel baru.
d. Hapus kolom yang baru dipindahkan dari tabel asal, kecuali penentu yang akan berfungsi sebagai kunci tamu.
e. Tabel asal bisa diberi nama baru.

Pada contoh, kita memindahkan kolom p#, status, dan kota ke tabel baru yang disebut pemasok2. Kolom p# menjadi kunci utama tabel ini. Tabel II.5. menunjukkan hasilnya.

Tabel II.5. Tabel Bentuk Normal Kedua (2NF).

Barang

P#	Status	Kota
P1	20	Yogyakarta
P2	10	Medan
P3	10	Medan
P4	20	Yogyakarta
P5	30	Bandung

Pemasok2

P#	B#	Qty
P1	B1	300
P1	B2	200
P1	B3	400
P1	B4	200
P1	B5	100
P1	B6	100
P2	B1	300
P2	B2	400
P3	B2	200
P4	B2	200
P4	B4	300
P4	B5	400

(Sumber : Janner Simarmata, dkk ; 2009 :82)

3. Bentuk Normal Ketiga (3 NF).

Bentuk normal ketiga mengharuskan semua kolom pada tabel relasional hanya pada kunci utama. Secara defenisi, sebuah tabel berada pada bentuk normal ketiga (3 NF) jika tabel sudah berada pada 2 NF dan setiap kolom yang bukan kunci tidak tergantung secara transistif pada kunci utamanya. Dengan kata lain, semua atribut bukan kunci tergantung secara fungsional hanya pada kunci utama. Tabel barang sudah dalam bentuk normal ketiga. Kolom bukan kunci, qty, tergantung sepenuhnya pada kunci utama (p#, b#). Pemasok masih berada pada 2 NF, tetapi belum berada pada 3 NF karena dia mengandung ketergantungan transitif. Ketergantungan transitif terjadi ketika sebuah kolom bukan kunci, yang ditentukan oleh kunci utama, menentukan kolom lainnya. Konsep ketergantungan transistif daopat digambarkan dengan menunjukkan ketergantungan fungsional pada pemasok2, yaitu :

Pemasok2. p# ⟶ Pemasok2, status

Pemasok2. p# ⟶ Pemasok2, kota

Pemasok2. kota ⟶ Pemasok2, status

Perlu dicatat bahwa pemasok2, status ditentukan, baik oleh kunci utama p#, maupun kolom bukan kunci, kota

Proses mengubah tabel menjadi 3 NF adalah :

a. Tentukan semua penentu selain kunci utama dan kolom yang ditentukannya.
b. Buat dan beri nama tabel baru untuk masing-masing penentu dan kolom yang ditentukannya.
c. Pindahkan kolom yang ditentukan dari tabel asal ke tabel baru. Penentu menjadi kunci utama tabel baru.
d. Hapus kolom yang baru saja dipindahkan dari tabel asal, kecuali penentu yang akan berfungsi sebagai kunci tamu.
e. Tabel asal bisa diberi nama baru.

Untuk mengubah PEMASOK2 menjadi 3 NF, kita membuat tabel baru yang disebut KOTA_STATUS dan memindahkan kolom kota dan status ke tabel baru. Status dihapus dari tabel diberi nama baru

PEMASOK_KOTA. Tabel II.6 menunjukkan hasilnya

Tabel II.6. Tabel Bentuk Normal Ketiga (3 NF)

PEMASOK_KOTA

P#	Kota
P1	Yogyakarta
P2	Medan
P3	Medan
P4	Yogyakarta
P5	Bandung

KOTA_STATUS

Kota	Status
Yogyakarta	20
Medan	10
Bandung	30
Semarang	40

(Sumber : Janner Simarmata, dkk ; 2009 :83)

4. Bentuk Normal Keempat (4 NF)

Sebuah tabel relasional berada pada bentuk normal keempat (4 NF) jika dia dalam BCNF dan semua ketergantungan multivalue merupakan ketergantungan fungsional.

Bentuk normal keempat (4 NF) didasarkan pada konsep ketergantungan multivalue (MVD). Sebuah ketergantungan multivalue terjadi ketika dalam sebuah tabel relasional yang mengandung setidaknya tiga kolom, satu kolom mempunyai banyak baris bernilai sama, tetapi kolom lain bernilai berebeda.

Defenisi secara formal diberikan oleh CJ. Date, yaitu :

Misalnya, ada sebuah tabel relasional R dengan kolom A, B dan C, Maka R.A $\rightarrow$ R.B (kolom A menentukan kolom B).

Adalah benar jika dan hanya jika himpunan nilai B yang cocok dengan pasangan nilai A dan nilai C pada R hanya tergantung pada nilai A dan tidak tergantung pada nilai C.

MVD selalu terjadi dalam pasangan, yaitu R.A $\rightarrow$ R.B dipenuhi jika dan hanya jika R.A $\rightarrow$ R.C dipenuhi pula.

5. Bentuk Normal Kelima (5 NF).
6. Sebuah tabel berada pada bentuk normal kelima jika dia tidak dapat mempunyai dekomposisi lossless menjadi sejumlah tabel lebih kecil.
7. Empat bentuk normal pertama berdasarkan pada konsep ketergantungan fungsional, sedangkan bentuk normal kelima berdasarkan pada konsep ketergantungan gabungan (*join dependence*). Ketergantungan gabungan berarti sebuah tabel, setelah deskomposisi menjadi tiga atau lebih tabel yang lebih kecil, harus dapat digabungkan kembali untuk membentuk tabel asal. Dengan kata lain 5 NF menunjukkan ketika sebuah tabel tidak dapat dideskomposisi lagi (Janner Simarmata; 2009: 77 - 86)

II.8.2. Kamus data

Menurut Roger. S. Pressman [3], kamus data adalah sebuah daftar yang terorganisasi dari elemen data yang berhubungan dengan system, dengan definisi yang tegaran teliti, sehingga pemakai dana nalisis system akan memiliki pemahaman yang umum mengenai input, output, dan komponen penyimpan dan bahkan kalkulasi intermediate. (jurnal : Syahrani Dhimas Prabowo dkk,2013)

II.9. *Unified Modeling Language* (UML)

UML singkatan dari *Unified Modelling Langguage* adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembangan sistem untuk membuat cetak biru atau visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi dan mengkomunikasikan rancangan mereka dengan yang lain.

(jurnal: Haviluddin; 2011 : 1)

II.9.1. Diagram-*Diagram UML*

Beberapa literatur menyebutkan bahwa *UML* menyediakan Sembilan jenis diagram, yang lain menyebutkan delapan karena ada beberapa yang digabung, misalnya diagram komunikasi, diagram urutan, dan diagram pewaktuan digabung menjadi diagram interaksi. Namun demikian model-model itu dapat

dikelompokan berdasarkan sifatnya yaitu statis atau dinamis. Jenis diagram itu antara lain :

1. Diagram kelas bersifat statis. Diagram ini memperlihatkan himpunan kelas kelas,antarmuka – antarmuka, kolaborasi-kolaborasi relasi.
2. Use case Diagram. Diagram ini memperlihatkan objek-objek serta relasi antar objek. Diagram objek memperlihatkan instansiasi statis dari segala sesuatu yang dijumpai pada diagram kelas
3. Diagram ini bersifat statis. Diagram ini memperlihatkan himpunan use case dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku dari suatu sistem yang dibutuhkan sertadiharapkan pengguna.
4. Sequence Diagram. Diagram ini bersifat dinamis. Diagram sequence merupakan diagram interaksi yangmenekankan pada pengiriman pesan (message) dalam suatu waktu tertentu.

5. Collaboration Diagram. Diagram ini bersifat dinamis. Diagram kolaborasi adalah diagram interaksi yang menekankan organisasi struktural dari objek – objek yang menerima serta mengirim pesan (message)

6. Statechart Diagram. Diagram ini bersifat dinamis. Diagram ini memperlihatkan state–state pada sistem,memuat state, transisi, event, serta aktifitas. Diagram ini terutama penting untukmemperlihatkan sifat dinamis dari antarmuka, kelas, kolaborasi dan terutamapenting pada pemodelan sistem – sistem yang reaktif.
7. Activity Diagram. Diagram ini bersifat dinamis. Diagram ini adalah tipe khusus dari diagram state yangmemperlihatkan aliran dari suatu aktifitas keaktifitas lainnya dari suatu sistem. Diagramnini terutama penting dalam pemodelannfungsi – fungsi dalam suatu sistem danmemberi tekanan pada aliran kendali antar objek

8. Component Diagram. Diagram ini bersifat statis. Diagram ini memperlihatkan organisasi serta kebergantungan pada komponen – komponen yang telah ada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan ke dalam satu

atau lebih kelas - kelas, antarmuka – antarmuka serta kolaborasi – kolaborasi

9. Deployment Diagram. Diagram ini bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (saat run time). Dengan ini memuat simpul – simpul (node) beserta Komponen – komponen yang ada didalamnya. Deployment diagram berhubungan erat dengan diagram kompoen dimana deployment diagram memuat satu atau lebih komponen – komponen. Diagram ini sangat berguna saat aplikasi berlakusebagai aplikasi yang dijalankan pada banyak mesin *(distributed computing)*. (jurnal : Prastuti Sulistyorini,2009).

Tabel II.7. Tipe Diagram UML

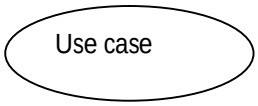
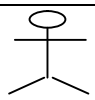
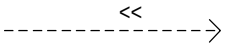
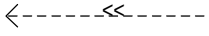
Diagram	Tujuan	Keterangan
Activity	Perilaku prosedural & pararel	Sudah ada di UML 1
Class	Class, fitur dan relasinya	Sudah ada di UML 1
Comunication	Interaksi diantara objek, lebih menekankan ke link	Di UML 1 disebut collaboration
Component	Struktur dan koneksi dari komponen	Sudah ada di UML 1
Composite structure	Dekomposisi sebuah class saat runtime	Baru untuk di UML 2
Deployment	Penyebaran/instalasi ke klien	Sudah ada di UML 1
Interactive Overview	Gabungan antara activity & sequence diagram	Baru untuk di UML 2
Object	Contoh konfigurasi instance	Tidak resmi ada di UML 1
Package	Struktur hierarki saat kompilasi	Tidak resmi ada di UML 1
Sequence	Interaksi antar objek, lebih menekankan pada urutan	Sudah ada di UML 1
State Machine	Bagaimana event mengubah objek	Sudah ada di UML 1
Timing	Interaksi antar objek, lebih menekankan pada waktu	Baru untuk di UML 2
Use case	Bagaimana user berinteraksi dengan sebuah sistem	Sudah ada di UML 1

(Sumber : Munawar; 2005 : 23)

II.9.2. Use Case

Use Case adalah rangakaian atau uraian sekelompok yang saling terkait dan berbentuk system secara teratur yang dilakukan atau diawasi oleh sebuah actor. *Use case* digunakan untuk memebetuk tingkah laku bendah dalam sebuah model serta direalisasikan yang diharapkan sebuah sistem. Use Case menggambarkan hubungan antara entitas yang biasa disebut aktor dengan suatu proses yang dapat dilakukannya. Simbol - simbol yang digunakan dalam Use Case beserta deskripsinya dapat dilihat pada table II.8.

Tabel II.8. Simbol Use Case

Simbol	Nama	Deskripsi
	Case	*Use case* mengindentifikasi fitur kuci dari sitem. Tanpa fitur ini, sitem tidak akan memenuhi permintaan *user/actor*
	Aktor	Actor dapat berupa manusia, sitem, atau device yang memiliki peran dalam keberhasilan operasi dari system
	Include	Relasi use case tambahan ke sebuah usecase dimana usecase yang ditambahkan memerlukan usecase ini untuk menjalankan fungsinya.
	Extend	Relasi use case tambahan ke sebuah usecase dimana usecase yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan.

	*Sistem*	Sistem menyatakan batasan sitem dalam relasi dengan *actor-actor* yang menggunakan (diluar sistem) dan fitur-fitur yang disediakan.

(Sumber : Hamim Tohari; 2014 : 52-54)

II.9.3. Class Diagram

Class Diagram adalah hubungan antar kelas yang menghasilkan sebuah objek dan merupakan unit dari pengembangan dan desain berorientasi objek.

Class Diagram menunjukkan atribut-atribut dan oprasi-oprasi dari sebuah kelas dan *constarin*t yang berhunbugan dangan objek yang dikoneksikan.

Class Diagram menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti *Contaiment, Pewarisan asosiasi*, dll, *Class* secara meliputi:

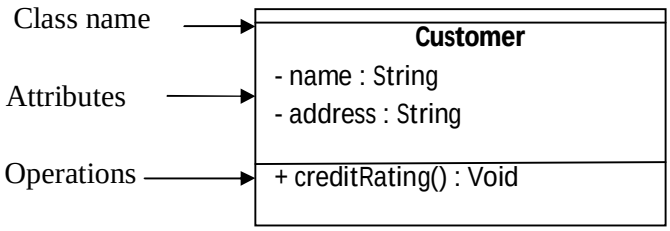
1. Kelas *(Class),*
2. Atribut *(Attributes),*
3. Operasi *(Operations/Method),*

   Atribut dan metode dapat memiliki salah satu sifat berikut :

1. Private, tidak di panggil dari luar class yang bersangkutan.
2. Protected, hanya dapat dipanggil oleh class yang bersangkutan.
3. Public, dapat dipanggil siapa saja.

   Class dapat merupakan implementasi dari sebuah interface, yaitu class abstrak yang hanya memiliki metode. Interface tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah class. Contoh diagram class dapat dilihat pada gambar II.7. dibawah ini :

   \



Gambar II.7. Class Diagram

(Sumber : Hamim Tohari ; 2010 : 84)

Hubungan antara mempunyai keterangan yang disebut dengan *multiplicity* atau *kardinality*, Tabel II.9. Berikut keteranganya :

Tabel II.10.*Multiplicity Class Diagram*

Indikator	Arti
0..1	Nola tau satu
1	Hanya satu
0..*	Nol atau lebih
1..*	Satu atau lebih
N	Hanya n (dengan n> 1)
0..n	Nol sampai n (dengan n> 1)
1..n	Satu sampai n (dengan n > 1)

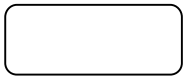
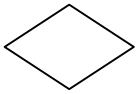
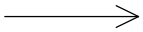
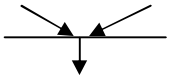
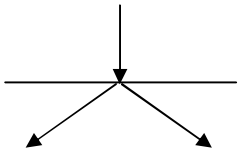
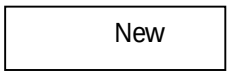
(Sumber : Hamim Tohari; 2014 : 86)

II.9.4. Aktivity Diagram

Activity Diagram adalah teknik untuk mendiskripsikan logika procedural, proses bisnis dan aliran kerja dalam banyak kasus . *Activity Diagram* mempunyai peran seperti halnya *flowchart* , akan tetapi perbedaanya dengan *flowchar* adalah

activity diagram bisa mendukung prilaku parallel sedangkan *flowchart* tidak bisa. Adapun elemen-elemen *activity diagram* dapat dilihat pada stabel II.10.

Tabel II.10. Simbol Aktivity diagram

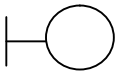
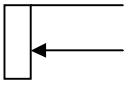
Simbol	Nama	Deskripsi
	*Activites*	Menggambarkan suatu proses/kegiatan bisnis
	*Decision Points*	Menggambarkan pilihan untuk pengambilan keputusan, *true, false,*
	Flow Control	Mengggambarkan aliran aktifitas dari suatu elemen ke elemen lain
	Initial State	Menggambarkan titik awal siklus hidup suatu elemen.
	Final State	Menggambarkan titik akhir yang menjadi kondisi akhir suatu elemen.
	Join (Penggabungan)	Digunakan untuk menunjukkan adanya dekomposisi
	Fork (Pencabangan)	Digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan parallel menjadi satu
	Swimlane	Pembagian Activity diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Hamim Tohari ; 2014 : 114-115)

II.9.5.Sequence Diagram

Sequence diagram menggunakkan interaksi antara sejumlah objek dalam urutan waktu. Kegunaannya untuk menujukkan rangkaian pesan yang dikiri antara objek juga interaksi atar objek yang terjadi pada titik tertentu dalam eksekusi system. Dalam UML , pada diagram sequence diagambarkan dengan segi epat, yang berisi nama dari objek yang digaris bawahi.

Tabel II.11. Simbol Sequence Diagram

Symbol	Nama	Keterangan
	Entity Class	Merupakan bagian dari system yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambarkan awal system dan menjadi landasan untuk menyusun basis data
	Boundary Class	Berisi kumpulan kelas yang menjadi *interface* atau interaksi antar satu atau lebih actor dengan system, seperti tampilan *formentry* dan *form* cetak
		Suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi aturan bisnis yang melibatkan berbagi abjek .
	Massage	Simbol mengirim pesan antar class
	Recursive	Menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri

	Activation	Activation mekili sebuah ekotal sekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah opraasi
	Lifeline	Garis titik-titik yang terhubung dengan objek sepanjang lifeline terdapat

(Sumber : Hamim Tohari: 2014 : 101)

II.10 *MYSQL*

MySQL (*My Structured Query Language*)atau yang biasa dibaca mai-se-kuel adalah sebuah program pembuat dan pengelola database atau yang sering disebut dengan DBMS (*DataBase Management System*), sifatdari DBMS ini adalah Open Source.MySQL sebenarnya produk yang berjalanpada platform Linux, dengan adanya perkembangan dan banyaknya pengguna, sertalisensi dari database ini adalah Open Source,maka para pengembang kemudian merilis versiWindows. Selain itu MySQL juga merupakan program pengakses database yang bersifat jaringan, sehingga dapat digunakan untukaplikasi Multi User (*Banyak Pengguna*).Kelebihan lain dari MySQL adalah menggunakan bahasa query (*permintaan*) standard SQL (*Structured Query Language*). Sebagai sebuah program penghasil database, MySQL tidak mungkin berjalan sendiri tanpa adanya sebuah aplikasi pengguna(*interface*) yang berguna sebagai program aplikasi pengakses database yang dihasilkan. MySQL dapat didukung oleh hampir semua program aplikasi baik yang Open Sourceseperti PHP maupun yang tidak Open Sourceyang ada pada platform windows seperti Visual Basic, Delphi dan lainnya.(jurnal : uswatun hasanah,2013).