BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Secara leksikal, sistem berarti susunan yang teratur dari pandangan, teori, asas dan sebagainya. Dengan kata lain, sistem adalah suatu kesatuan usaha yang terdiri dari bagian-bagian yang berkaitan satu sama lain yang berusaha mencapai suatu tujuan dalam suatu lingkungan kompleks. Pengertian tersebut mencerminkan adanya beberapa bagian dan hubungan antara bagian, ini menunjukkan kompleksitas dari sitem yang meliputi kerja sama antara bagian yang interpenden satu sama lain. Selain itu dapat dilihat bahwa sistem berusaha mencapai tujuan. Pencapaian tujuan ini menyebabkan timbulnya dinamika, perubahan-perubahan yang terus menerus perlu dikembangkan dan dikendalikan. Definisi tersebut menunjukkan bahwa sistem sebagai gugus dan elemen-elemen yang saling berinteraksi secara teratur dalam rangka mencapai tujuan atau subtujuan (Prof. Dr. Ir. Marimin, M.Sc ; 2008 : 1).

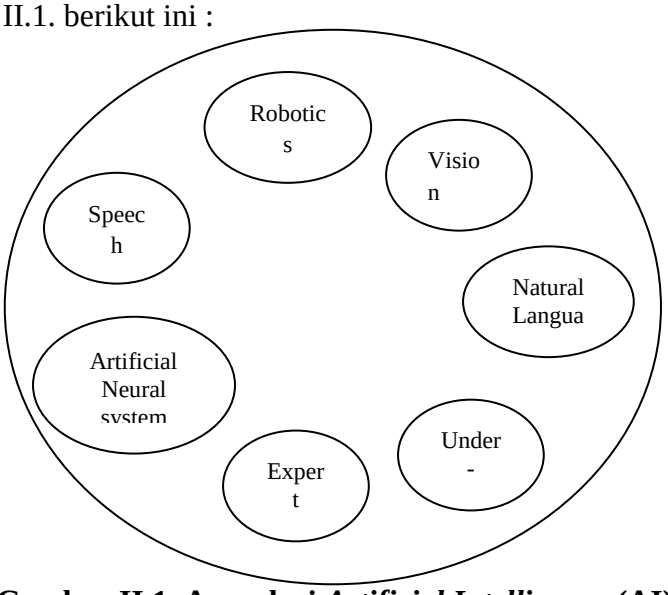
II.2. *Artificial Intelegence* (Kecerdasan Buatan)

Artificial Intelegence atau kecerdasan buatan adalah menjadikan komputer berakting dan bergaya seperti halnya para artis berakting di bioskop. Untuk saat ini banyak permasalahan dunia nyata yang diselesaikan menggunakan AI dan banyak juga aplikasinya yang komersialkan (Rika Rosnelly; 2012: 1).

Walaupun penyelesain umum untuk masalah AI klasik seperti tranlasi bahasa alami, pemahaman ucapan, dan visi belum ditemukan, tetapi pembatasan domain permasalahannya telah dapat menghasilkan suatu penyelesaian yang bermanfaat. Sebagai contoh, tidaklah terlalu sukar untuk membangun suatu sistem bahasa alami yang sederhana jika masukan dibatasi untuk kalimat dengan bentuk kata benda, kata kerja dan objek (Rika Rosnelly; 2012: 2).

Sistem pakar *(Expert System)* merupakan solusi *Artificial Intelligence* (AI) bagi masalah pemrograman pintar (*intelligent). Profesor Edward Feigenbaum* dari *Stanford University* yang merupakan pionir dalam teknologi sistem pakar mendefinisikan sistem pakar sebagai sebuah program komputer pintar (*intelligent computer program*) yang memanfaatkan pengetahuan *(knowledge)* dan prosedur inferensi *(inference procedure)* untuk memecahkan masalah yang cukup sulit hingga membutuhkan keahlian khusus dari manusia (Rika Rosnelly; 2012: 2).

Artificial Intelligence (AI) memiliki berapa domain masalah atau area seperti gambar II.1. berikut ini :



Gambar II.1. Area dari *Artificial Intelligence* (AI)
(Sumber : Rika Rosnelly; 2012: 3)

II.3. Sistem Pakar

II.3.1. Pengertian Sistem Pakar

Sistem Pakar adalah sistem komputer yang ditujukan untuk meniru semua aspek (*emulates*) kemampuan pengambilan keputusan (*decision making*) seorang pakar. Sistem pakar memanfaatkan secara maksimal pengetahuan khusus selayaknya seorang pakar untuk memecahkan masalah (Rika Rosnelly; 2012: 2).

Sistem Pakar adalah aplikasi berbasis komputer yang digunakan untuk menyelesaikan masalah sebagaimana yamg dipikirkan oleh pakar. Pakar yang dimaksud disini adalah orang yang mempunyai keahlian khusus yang dapat menyelesaikan masalah yang tidak dapat diselesaikan oleh orang awam. Sebagai contoh mekanik adalah seorang pakar yang mampu mendiagnosa kerusakan yang dialami mobil serta dapat memberikan penatalaksanaan terhadap kerusakan tersebut. Tidak semua orang dapat mengambil keputusan mengenai identifikasi dan memberikan penatalaksanaan suatu kerusakan.

II.3.2. Kelebihan Sistem Pakar

Sistem Pakar memiliki beberapa fitur menarik yang merupakan kelebihannya, seperti :

1. Meningkatkan ketersedian (*increased availability*). Kepakaran atau keahlian menjadi tersedia dalam sistem komputer. Dapat dikatakan bahwa sistem pakar merupakan produksi kepakaran secara masal (*massproduction*).
2. Mengurangi biaya *(reduced cost).* Biaya yang diperlukan untuk menyediakan keahlian per satu orang *user* menjadi berkurang.

3. Mengurangi Bahaya *(reduced danger)*. Sistem pakar dapat digunakan di lingkungan yang mungkin berbahaya bagi manusia.
4. Permanen *(permanence)*. Sistem pakar dan pengetahuan yang terdapat di dalamnya bersifat lebih permanen dibandingkan manusia yang dapat merasa lelah, bosan, dan pengetahuannya hilang saat sang pakar meninggal dunia.
5. Keahlian Multipel *(multiple expertise)*. Pengetahuan dari beberapa pakar dapat dimuat ke dalam sistem dan bekerja secara simultan dan kontinyu menyelesaikan suatu masalah setiap saat. Tingkat keahlian atau pengetahuan yang digabungkan dari beberapa pakar dapat melebihi pengetahuan satu orang pakar .

II.3.3. Konsep Umum Sistem Pakar

Konsep dasar sistem pakar tediri dari beberapa unsur atau elemen antara lain (Rika Rosnelly; 2012: 10) :

1. Pakar

Pakar adalah seorang individu yang memiliki pengetahuan khusus, pemahaman, pengalaman, dan metode-metode yang digunakan untuk memecahkan persoalan dalam bidang tertentu. Pakar juga memiliki kemampuan untuk mengaplikasikan pengetahuannya dan memberikan saran serta pemecahan masalah pada domain tertentu.

Seorang pakar mengetahui fakta – fakta mana yang penting, sebab akibat, fenomena – fenomena yang terkait dengan fakta, memahami arti hubungan antar fakta, juga hubungan sebab akibat, dan hubungan dengan fenomena – fenomena

yang terkait serta mampu menginterprestasikan akibat – akibat yang terjadi karena sesuatu sebab terjadi (Rika Rosnelly; 2012: 11).

2. Pembangun / Pembuat Pengetahuan

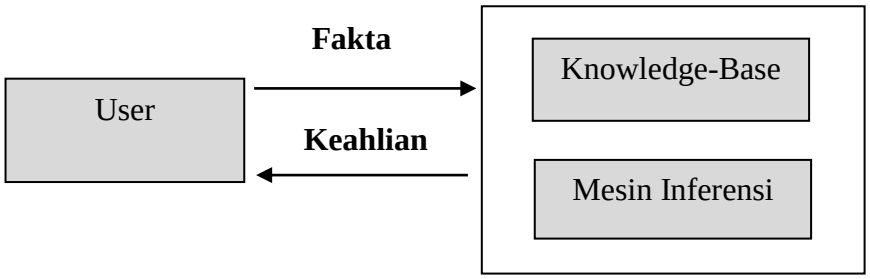
Pembangun pengetahuan memiliki tugas utama menerjemahkan dan merepresentasikan pengetahuan yang diperoleh dari pakar, baik berupa pengalaman pakar dalam menyelesaikan masalah maupun sumber terdukumentasi lainnya kedalam bentuk yang bisa diterima oleh sistem pakar. Dalam hal ini pembangun pengetahuan *(knowledge engineer)* menginterpretasikan dan merepresentasikan pengetahuan yang diperoleh dalam bentuk jawaban-jawaban atas pertanyaan-pertanyaan yang diajukan pada pakar atau pemahaman, penggambaran, analogis, sistematis, konseptual yang diperoleh dari membaca beberapa dokumen cetak seperti *text book*, jurnal, makalah, dan sebagainya, Kurangnya pengalaman *Knowledge engineer* merupakan kesulitan utama dalam mengkonstruksi sistem pakar. Untuk mengatasi hal tersebut, perancang sistem pakar menggunakan *tools* komersial. (Seperti pada editor-editor khusus maupun *logic debuggers*) dan usahanya akan dipusatkan pada pembangunan mesin inferensi (Rika Rosnelly; 2012: 12).

3. Pembangun / Pembuat Sistem

Pembangun sistem adalah orang yang bertugas untuk merancang antarmuka pemakai sistem pakar, merancang pengetahuan yang sudah diterjemahkan oleh pembangun pengetahuan kedalam bentuk yang sesuai dan dapat diterima oleh sistem pakar dan mengimplementasikannya kedalam mesin infrensi (Rika Rosnelly; 2012: 12).

4. Pengguna *(User)*

Banyak sistem berbasis komputer mempunyai susunan pengguna tunggal. Hal ini berbeda jauh dengan sistem pakar yang memungkinkan mempunyai beberapa kelas pengguna. Berikut gambar konsep dan fungsi sistem pakar ditunjukkan pada Gambar II.2. (Rika Rosnelly; 2012: 12).



Gambar II.2. Konsep dan Fungsi Sistem Pakar Berbasis Pengetahuan
(Sumber : Rika Rosnelly; 2012: 4)

II.3.4. Stuktur Sistem Pakar

Komponen yang terdapat dalam stuktur sistem pakar ini adalah sebagai berikut (Rika Rosnelly; 2012: 14) :

1. *Knowledge Base* (Basis Pengetahuan)

Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah. Komponen sistem pakar disusun atas dua elemen dasar, yaitu fakta dan aturan.

2. *Inference Engine* (Mesin Inferensi)

Mesin Inferensi merupakan otak dari sebuah sistem pakar dan dikenal juga dengan sebutan *control structure* (stuktur kontrol) atau *rule interpreter* (dalam sistem pakar berbasis kaidah). Komponen ini mengandung mekanisme pola pikir dan penalaran yang digunakan oleh pakar dalam menyelesaikan suatu masalah.

3. Working Memory

Berguna untuk menyimpan fakta yang dihasilkan oleh *inference engine* dengan penambahan parameter berupa derajat kepercayaan atau dapat juga dikatakan sebagai global database dari fakta yang digunakan rule – rule yang ada.

4. Explanation Facility

Menyediakan kebenaran dari solusi yang dihasilkan kepada user *(reasoning chain).*

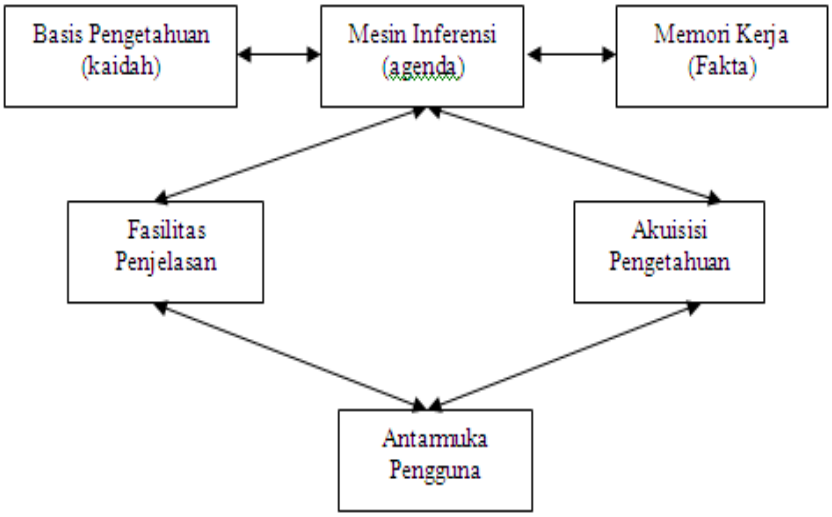
5. Knowledge Acquisition Facility

Meliputi proses pengumpulan, pemindahan dan perubahan dari kemampuan pemecahan masalah seorang pakar atau sumber pengetahuan terdokumentasikan ke program computer, yang bertujuan untuk memperbaiki atau mengembangkan basis pengetahuan.

6. User Interface

Mekanisme untuk memberi kesempatan kepada *user* dan sistem pakar untuk berkomunikasi. Antarmuka menerima informasi dari pemakai dan mengubahnya kedalam bentuk yang dapat diterima oleh sistem. Selain itu antarmuka menerima informasi dari sistem dan menyajikannya ke dalam bentuk yang dapat dimengerti oleh pemakai.

Adapun Gambar Stuktur Sistem Pakar adalah sebagai berikut:



Gambar II.3. Stuktur Sistem Pakar
(Sumber : Rika Rosnelly; 2012: 13)

II.3.5. Karakteristik Sistem Pakar

Sistem Pakar umumnya dirancang untuk memenuhi beberapa karaktristik umum berikut (Rika Rosnelly; 2012: 20) :

1. Kinerja yang sangat baik *(high performance).* Sistem harus mampu memberikan respon berupa saran (advice) dengan tingkat kualitas yang sama dengan seorang pakar atau melebihinya.
2. Waktu respon yang baik *(adequate respon time).* Sistem juga harus mampu bekerja dalam waktu yang sama baiknya *(reasonable)* atau lebih cepat dibandingkan dengan seorang pakar dalam menghasilkan keputusan. Hal ini sangat penting terutama pada sistem waktu nyata (real-time).
3. Dapat diandalkan *(good reliability).* Sistem harus dapat diandalkan dan tidak mudah rusak/crash.
4. Dapat Dipahami *(understandable).* Sistem ini harus mampu menjelaskan langkah-langkah penalaran yang dilakukannya seperti seorang pakar.

5. Fleksibel *(flexibility).* Sistem harus menyediakan mekanisme untuk menambah, mengubah, dan menghapus pengetahuan.

II.3.6. Metode Inferensi

Komponen ini mengandung mekanisme pola pikir dan penalaran yang digunakan oleh pakar dalam menyelesaikan suatu masalah. Metode inferensi adalah program komputer yang memberikan metedologi untuk penalaran tentang informasi yang ada dalam basis pengetahuan dan dalam *workplace*, dan untuk memformulasikan kesimpulan.

Kebanyakan sistem pakar berbasis aturan menggunakan strategi inferensi yang dinamakan modus ponen. Berdasarkan strategi ini, jika terdapat aturan "IF A THEN B", dan jika diketahui bahwa A benar, maka dapat disimpulkan bahwa B juga benar. Strategi inferensi modus ponen dinyatakan dalam bentuk :

[A And (A→B)] →B (1)

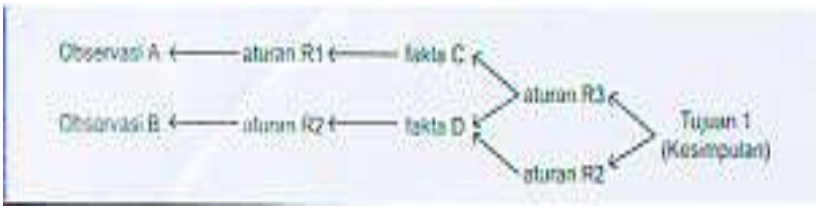
dengan A dan A→B adalah proposisi-proposisi dalam basis pengetahuan.

Terdapat dua pendekatan untuk mengontrol inferensi dalam sistem pakar berbasis aturan, yaitu pelacakan ke belakang (*Backward chaining*) dan pelacakan ke depan (*forward chaining*).

1. Pelacakan ke belakang (*Backward Chaining*)

Pelacakan ke belakang adalah pendekatan yang dimotori oleh tujuan (*goaldriven*). Dalam pendekatan ini pelacakan dimulai dari tujuan, selanjutnya dicari aturan yang memiliki tujuan tersebut untuk kesimpulannya. Selanjutnya proses pelacakan menggunakan premis untuk aturan sebagai tujuan baru dan mencari aturan lain dengan tujuan baru sebagai kesimpulannya. Proses berlanjut

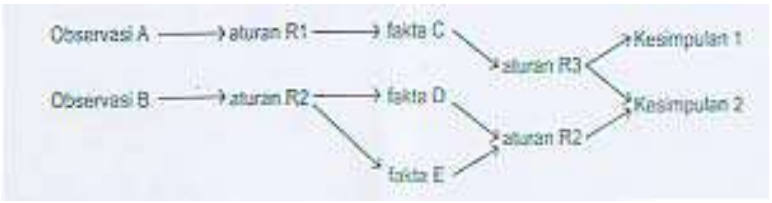
sampai semua kemungkinan ditemukan. Gambar II.4 menunjukan proses *backward chaining*.



Gambar II.4. Proses *backward chaining*

2. Pelacakan ke depan (*forward chaining*)

Pelacakan kedepan adalah pendekatan yang dimotori data (*data-driven*). Dalam pendekatan ini pelacakan dimulai dari informasi masukan, dan selanjutnya mencoba menggambarkan kesimpulan. Pelacakan ke depan, mencari fakta yang sesuai dengan bagian IF dari aturan IF-THEN. Gambar II.5 menunjukkan proses *forward chaining* (Feri Fahrur Rohman ; 2008 : 6).



Gambar II.5.. Proses *forward chaining*

II.3.7. Representasi Pengetahuan

Setelah menerima bidang kepakaran yang telah diaplikasikan pada sistem pakar, kemudian mengumpulkan pengetahuan yang sesuai dengan *domain* keahlian tersebut. Pengetahuan yang dikumpulkan tersebut tidak bisa diaplikasikan begitu saja dalam sistem. Pengetahuan harus direpresentasikan dalam *format* tertentu dan dihimpun dalam suatu basis pengetahuan.

Pengetahuan yang dilakukan pada sistem pakar merupakan serangkaian informasi pada *domain* tertentu. Kedua hal tersebut menurut *ekspresi* klasik oleh

Wirth ditulis sebagai berikut:

Algoritma + Struktur Data = Program

Pengetahuan + *Inferensi* = Sistem Pakar

Noise merupakan suatu item yang tidak mempunyai maksud (*interest*). *Noise* merupakan data yang masih kabur atau tidak jelas. *Data* adalah item yang mempunyai makna potensial. Data diolah menjadi pengetahuan. *Meta knowledge* adalah pengetahuan tentang pengetahuan dan keahlian.

Karakteristik pengetahuan yang diperoleh tergantung pada sifat masalah yang akan diselesaikan, tipe dan tingkat pengetahuan seorang pakar. Pengetahuan harus *diekstraksikan* dan dikodekan dalam suatu bentuk tertentu untuk memecahkan masalah. Ketika pengetahuan dalam suatu bidang kepakaran tersedia, maka dipilih representasi pengetahuan yang tepat. Pengetahuan dapat digolongkan menjadi dua kategori, yaitu: pengetahuan *deklaratif* dan pengetahuan *prosedural*.

Pengetahuan *deklaratif* mengacu pada fakta, sedangkan pengetahuan *prosedural* mengacu pada serangkaian tindakan dan konsekuensinya. Pengetahuan *deklaratif* juga terlibat dalam pemecahan masalah, sedangkan pengetahuan *prosedural* diasosiasikan dengan bagaimana menerapkan strategi atau *prosedur* penggunaan pengetahuan yang tepat untuk memecahkan masalah.

Pengetahuan *deklaratif* menggunakan basis logika dan pendekatan relasi. Representasi logika menggunakan *logika proporsional* dan *logika predikat*. Model relasi menggunakan jaringan *semantik*, *graph* dan pohon keputusan

(decision tree). Pengetahuan *prosedural* menggunakan algoritma sebagai *prosedural* pemecahan masalah (Feri Fahrur Rohman ; 2008 : 7).

II.4. *Rheumatic*

Arthritis atau biasa disebut *Rheumatic* adalah penyakit yang menyerang persendian dan struktur disekitarnya. Masyarakat pada umumnya menganggap *Rheumatic* adalah penyakit sepele karena tidak menimbulkan kematian. Pada hal, jika tidak segera ditangani rernatik bisa membuat anggota tubuh berfungsi tidak normal, mulai dari benjol-benjol, sendi kaku, sulit berjalan, bahkan kecacatan seumur hidup. Rasa sakit yang timbul bisa sangat mengganggu dan membatasi aktivitas kegiatan sehari-hari. jumlah penderita arthritis atau gangguan sendi kronis lain di Amerika Serikat terus meningkat. Pada tahun 1990 terdapat 38 juta penderita dari sebelumnya 35 juta pada tahun 1985. Data tahun 1998 memperlihatkan hampir 43 juta atau 1 dari 6 orang di Amerika menderita gangguan sendi, dan pada tahun 2005 jumlah penderita arthritis sudah mencapai 66 juta atau hampir 1 dari 3 orang menderita gangguan sendi. Sebanyak 42,7 juta di antaranya telah terdiagnosis sebagai arthritis dan 23,2 juta sisanya adalah penderita dengan keluhan nyeri sendi kronis. Sedangkan prevalensi *Rheumatic* di Indonesia menurut hasil penelitian yang dilakukan oleh Zeng QY *et al* mencapai 23,6% sampai 31,3%.

Penyakit *Rheumatic* itu sebenarnya terdiri lebih dari 100 (seratus) jenis, tetapi bagi orang awam, setiap gejala nyeri, kaku, bengkak, pegal-pegal, atau kesemutan itu semua sering disebut *Rheumatic* dan dianggap sama saja. Penyakit *Rheumatic* yang paling banyak ditemukan pada golongan usia lanjut di Indonesia

adalah osteoarkitis (OA) (50-60)%. Yang kedua adalah kelompok *Rheumatic* luar sendi (gangguan pada komponen penunjang sendi, peradangan penggunaan berlebihan, dan sebagainya). Yang ketiga adalah asam urat (gout) sekitar 6-7%. Sementara penyakit *rematoid arthritis* (RA) di Indonesia hanya 0,l% (1 di antara 1000-5000 orang), sedangkan di negara-negara Barat sekitar 3%.

Rheumatic merupakan salah satu penyebab nyeri sendi, khususnya sendi-sendi kecil di daerah pergelangan tangan dan jari-jari. Keluhan kaku, nyeri dan bengkak akibat penyakit *Rheumatic* dapat berlangsung terus-menerus dan semakin lama semakin berat tetapi ada kalanya hanya berlangsung selama beberapa hari dan kemudian sembuh dengan pengobatan. Namun demikian, kebanyakan penyakit *Rheumatic* berlangsung kronis, yaitu sembuh dan kambuh kembali secara berulang sehingga menyebabkan kerusakan sendi secara menetap. Keluhan kaku dan nyeri sendi pada penyakit *Rheumatic* adakalanya disertai oleh perasaan mudah lelah (Olwin Nainggolan; 2009: 589).

II.5. Metode *Dempster-Shafer*

Ada berbagai macam penalaran dengan model yang lengkap dan sangat konsisten, tetapi pada kenyataannya banyak permasalahan yang tidak dapat terselesaikan secara lengkap dan konsisten. Ketidak konsistenan yang tersebut adalah akibat adanya penambahan fakta baru. Penalaran yang seperti itu disebut dengan penalaran *non monotonis*. Untuk mengatasi ketidak konsistenan tersebut maka dapat menggunakan penalaran dengan teori *Dempster-Shafer*.

Dempster-Shafer adalah suatu teori matematika untuk pembuktian berdasarkan *belief functions and plausible reasoning* (fungsi kepercayaan dan

pemikiran yang masuk akal), yang digunakan untuk mengkombinasikan potongan informasi yang terpisah (bukti) untuk mengkalkulasi kemungkinan dari suatu peristiwa. Teori ini dikembangkan oleh Arthur P. Dempster dan Glenn Shafer (Muhammad Dahria ; 2013 : 5).

Metode *Dempster-Shafer* pertama kali diperkenalkan oleh *Dempster*, yang melakukan percobaan model ketidakpastian dengan *range probabilities* dari pada sebagai probabilitas tunggal. Kemudian pada tahun 1976 Shafer mempublikasikan teori Dempster itu pada sebuah buku yang berjudul *Mathematical Theory Of Evident*. *Dempster-Shafer Theory Of Evidence*, menunjukkan suatu cara untuk memberikan bobot kenyakinan sesuai fakta yang dikumpulkan.

Pada teori ini dapat membedakan ketidakpastian dan ketidaktahuan. Teori *Dempster-Shafer* adalah representasi, kombinasi dan propogasi ketidakpastian, dimana teori ini memiliki beberapa karakteristik yang secara instutitif sesuai dengan cara berfikir seorang pakar, namun dasar matematika yang kuat (Elyza Gustri Wahyuni; 2013: 135-136).

Secara umum teori *Dempster-Shafer* ditulis dalam suatu interval :

[Belief, Plausibility]

1. Belief (Bel) adalah ukuran kekuatan evidence dalam mendukung suatu himpunan proposisi. Jika bernilai 0 maka mengindikasikan bahwa tidak ada *evidence*, dan jika bernilai 1 menunjukkan adanya kepastian. Dimana nilai bel yaitu (0-0.9).

2. *Plausibility* (Pls) akan mengurangi tingkat kepastian dari *evidence. Plausibility* juga bernilai 0 sampai 1. Jika yakin akan X', maka dapat dikatakan bahwa Bel(X') = 1, sehingga rumus di atas nilai dari Pls(X)=0.

Menurut Giarratano dan Riley fungsi *Belief* dapat diformulasikan dan ditunjukkan pada persamaan (1) :

$$Bel\ (X) = \sum_{Y \subseteq X} m(Y) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ (1)$$

Dan *Plausibility* dinotasikan pada persamaan (2) :

$$Pls\ (X) = 1 - Bel\ (X) = 1 - \sum_{Y \subseteq X} m(X) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ (2)$$

Dimana :

Bel (X) = *Belief* (X)

Pls (X) = *Plausibility* (X)

m (X) = *mass function* dari (X)

m (Y) = *mass function* dari (Y)

Pada teori *Dempster-Shafer* dikenal adanya *frame of discrement* yang dinotasikan dengan simbol θ. *Frame of discrement* merupakan semesta pembicaraan dari sekumpulan hipotesis sehingga sering disebut dengan *environment* yang ditunjukkan pada persamaan (3) :

$$\Theta = \{ \theta 1, \theta 2, \ldots \theta N\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ (3)$$

Dimana :

Θ = *frame of discrement atau environment*

θ1,…,θN = element/ unsur bagian dalam *environment*

Environment mengandung elemen-elemen yang menggambarkan kemungkinan sebagai jawaban, dan hanya ada satu yang akan sesuai dengan jawaban yang dibutuhkan. Kemungkinan ini dalam teori *Dempster-Shafer* disebut

dengan *power set* dan dinotasikan dengan P (Θ), setiap elemen dalam *power set* ini memiliki nilai interval antara 0 sampai 1.

m : P (Θ) [0,1]

Sehingga dapat dirumuskan pada persamaan (4) :

$$\sum_{X \in P(\Theta)} m(X) = 1 \quad \text{..... (4)}$$

Dengan :

P (Θ) = *power set*

m (X) = *mass function* (X)

Mass function (m) dalam teori *Dempster-shafer* adalah tingkat kepercayaan dari suatu *evidence* (gejala), sering disebut dengan *evidence measure* sehingga dinotasikan dengan (m). Tujuannya adalah mengaitkan ukuran kepercayaan elemen-elemen θ. Tidak semua evidence secara langsung mendukung tiap-tiap elemen. Untuk itu perlu adanya probabilitas fungsi densitas (m). Nilai m tidak hanya mendefinisikan elemen-elemen θ saja, namun juga semua subsetnya. Sehingga jika θ berisi n elemen, maka subset θ adalah 2*n* . Jumlah semua m dalam subset θ sama dengan 1. Apabila tidak ada informasi apapun untuk memilih hipotesis, maka nilai : m{θ} = 1,0.

Apabila diketahui X adalah subset dari θ, dengan m1 sebagai fungsi densitasnya, dan Y juga merupakan subset dari θ dengan m2 sebagai fungsi densitasnya, maka dapat dibentuk fungsi kombinasi m1 dan m2 sebagai m3, yaitu ditunjukkan pada persamaan (5) :

$$M_3(Z) = \frac{\sum x \cap y = z^{m} 1(x).^{m} 2(y)}{1 - \sum x \cap y = \emptyset^{m} 1(x).^{m} 2(y)} \quad \text{..... (5)}$$

Dimana :

$m3(Z)$ = *mass function* dari *evidence* (Z)

$m_1(X)$ = *mass function* dari *evidence* (X), yang diperoleh dari nilai keyakinan suatu *evidence* dikalikan dengan nilai *disbelief* dari *evidence* tersebut.

$m_2(Y)$ = *mass function* dari *evidence* (Y), yang diperoleh dari nilai keyakinan suatu *evidence* dikalikan dengan nilai *disbelief* dari *evidence* tersebut.

$\sum_{X \cap Y = Z} m_1(X).m_2(Y)$ = merupakan nilai kekuatan dari *evidence* Z yang diperoleh dari kombinasi nilai keyakinan sekumpulan *evidence* (Elyza Gustri Wahyuni ; 2013 : 135-137).

II.6. Macromedia Dreamweaver

Macromedia Dreamweaver adalah sebuah *software web design* yang menawarkan cara mendesain *website* dengan dua langkah sekaligus dalam satu waktu, yaitu mendesain dan memprogram. Dreamweaver memiliki satu jendela mini yang disebut HTML *Source*, tempat kode-kode HTML tertulis. Setiap kali kita mendesain *web*, seperti menulis kata-kata, meletakkan gambar, membuat tabel dan proses lainnya, tag-tag HTML akan tertulis secara langsung mengiringi proses pengaturan *website*. Artinya kita memiliki kesempatan untuk mendesain. *Website* sekaligus mengenal tag-tag HTML yang membangun *website* itu. Di lain kesempatan kita juga dapat mendesain *website* hanya dengan menulis tag-tag dan teks lain di jendela HTML *Source* dan hasilnya dapat di lihat langsung di layar (M. Suyanto; 2009: 244).

II.7. PHP

PHP merupakan bahasa *scripting* yang berjalan di sisi *server* (*server-side*). Semua perintah yang ditulis akan dieksekusi oleh *server* dan hasil jadinya dapat dilihat melalui *browser*. Saat ini PHP versi 4 sudah di-*release* di pasaran, mengikuti jejak kesuksesan versi sebelumnya, PHP 3. Selain dapat digunakan untuk berbagai sistem operasi, koneksi *database* yang sangat mudah menyebabkan bahasa *scripting* ini digemari para *programmer web*. Beberapa perintah PHP yang kita pelajari sebatas pada perintah untuk menampilkan *tag-tag* wml, akses *database* MySQL, dan pengiriman *email* (Ridwan Sanjaya; 2009: 73).

II.8. *Database*

Banyak sekali definisi tentang database yang diberikan oleh para pakar dibidang ini. Database terdiri dari dua penggalan kata yaitu data dan base, yang artinya berbasiskan pada data. Tetapi secara konseptual, database diartikan sebuah koleksi atau kumpulan data yang saling berhubungan (*relation),* disusun menurut aturan tertentu secara logis, sehingga menghasilkan informasi. Sebuah informasi yang berdiri sendiri tidaklah dikatakan database.

Contoh : Nomor telpon seorang pelanggan, disimpan dalam banyak tempat apakah itu di file pelanggan, di file alamat dan dilokasi yang lain. Antara file yang satu denga file yang lainnya tidak saling berhubungan, sehingga apabila salah seorang pelanggan berganti nomor telpon dan anda hanya mengganti di file pelanggan saja, akibatnya akan terjadi ketidakcocokan data, karena dilokasi yang lain masih tersimpan data nomor telpon yang lama.

Dalam sistem database hal ini tidak boleh dan tidak bisa terjadi, karena antara file yang satu dengan file yang lain saling berhubungan. Jika suatu data yang sama anda ubah, data tersebut di file yang lain akan otomatis berubah juga. Sehingga tingkat keakuratan/kebenaran data sangat tinggi.

Secara prinsip, dalam suatu database tercakup dua komponen penting, yaitu Data dan Informasi. Jadi tujuan akhir anda adalah bagaimana mengelola data sehingga mampu menjadi informasi yang diinginkan dan dapat dilakukan proses pengambilan,penghapusan, pengeditan terhadap data secara mudah dan cepat (Efektif, Efisien dan Akurat)

Data adalah fakta, baik berupa sebuah objek, orang dan lain-lain yang dapat dinyatakan dengan suatu nilai tertentu (angka, simbol, karakter tertentu, dll). Sedangkan Informasi adalah data yang telah diolah sehingga bernilai guna dan dapat dijadikan bahan dalam pengambilan keputusan. Hubunga data dan informasi dapat digambarkan sebagai berikut :

Data ⟶ Proses ⟶ Informasi

Gambar II.6. Data dan Informasi

Banyak sekali contoh database yang ada disekeliling anda. Tanpa disadari ternyata anda telah menggunakan manfaat dari database itu sendiri. Misalnya ATM tempat anda mengambil dan transfer uang yang dapat dilakukan dimana saja, membayar rekening telpon atau PDAM yang dapat dilakukan di berbagai tempat, registrasi akademik di kampus dan lain sebagainya. Semua itu telah dibuat secara database (Yuhefizard, S.Kom; 2010: 2).

II.9. MySQL

Mysql pertama kali dirintis oleh seorang programmer database bernama Michael Widenius, yang dapat anda hubungi di emailnya monty@analytikerna. Mysql *database server* adalah RDBMS (*Relasional Database Management System*) yang dapat menangani data yang bervolume besar. meskipun begitu, tidak menuntut *resource* yang besar. Mysql adalah *database* yang paling populer di antara *database* yang lain.

MySQL adalah program *database* yang mampu mengirim dan menerima data dengan sangat cepat dan *multi user*. MySQL memiliki dua bentuk lisensi, yaitu *free software* dan *shareware*. Penulis sendiri dalam menjelaskan buku ini menggunakan *database* ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensi, yang berada di bawah lisensi GNU/GPL (*general public license*) (Wahana Komputer; 2010: 5).

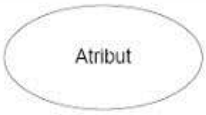
Gambar II.7. Tampilan MySQL

(Sumber : Wahana Komputer; 2010: 5)

II.10. *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram atau ERD adalah alat pemodelan data utama dan akan membantu mengorganisasi data dalam suatu proyek ke dalam entitas-entitas dan menentukan hubungan antarentitas. Proses memungkinkan analis menghasilkan struktur basisdata yang baik sehingga data dapat disimpan dan diambil secara efisien (Janner Simarmata; 2010: 67). Simbol-simbol yang digunakan dalam ERD, yaitu :

Tabel II.1. Simbol ERD

Notasi	Keterangan
	**Entitas**, adalah suatu objek yang dapat diidentifikasi dalam lingkungan pemakai.
	**Relasi**, menunjukkan adanya hubungan di antara sejumlah entitas yang berbeda.
	**Atribut**, berfungsi mendeskripsikan karakter entitas (atribut yg berfungsi sebagai key diberi garis bawah)
	Garis, sebagai penghubung antara relasi dengan entitas, relasi dan entitas dengan atribut.

(Sumber : Janner Simarmata; 2010: 67)

II.11. Kamus Data

Kamus data (data *dictionary*) mencakup definisi-definisi dari data yang disimpan di dalam basis data dan dikendalikan oleh sistem manajemen basis data. Figur 6.5 menunjukkan hanya satu tabel dalam basis data jadwal. Struktur basis data yang dimuat dalam kamus data adalah kumpulan dari seluruh definisi *field*,

definisi tabel, relasi tabel, dan hal-hal lainnya. Nama *field* data, jenis data (seperti teks atau angka atau tanggal), nilai-nilai yang valid untuk data, dan karakteristik-karakteristik lainnya akan disimpan dalam kamus data. Perubahan-perubahan pada struktur data hanya dilakukan satu kali di dalam kamus data, program-program palikasi yang mempergunakan data tidak akan ikut terpengaruh (Raymond McLeod; 2008: 171).

II.12. Teknik 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 rasional.

Teori normalisasi didasarkan pada konsep bentuk normal. Sebuah tabel relasional dikatakan berada pada bentuk normal tertentu jika tabel memenuhi himpunan batasan tertentu. Ada lima bentuk normal yang telah ditemukan.

II.12.1. Bentuk-Bentuk Normalisasi

1. Bentuk Normal Tahap Pertama (1” Normal Form)

Contoh yang kita gunakan di sini adalah sebuah perusahaan 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.

2. Bentuk Normal Tahap Kedua (2nd Normal Form)

Definisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada 1NF, tetapi tidak pada 2NF. Sebuah tabel relasional berada pada bentuk normal kedua jika dia berada pada bentuk normal kedua jika dia berada pada 1NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada seluruh kolom yang membentuk kunci utama.

3. Bentuk Normal Tahap Ketiga (3rd Normal Form)

Bentuk normal ketiga mengharuskan semua kolom pada tabel relasional tergantung hanya pada kunci utama. Secara definisi, sebuah tabel berada pada bentuk normal ketiga (3NF) jika tabel sudah berada pada 2NF dan setiap kolom yang bukan kunci tidak tergantung secara transitif pada kunci utamanya.

4. Boyce Code Normal Form (BCNF)

Setelah 3NF, semua masalah normalisasi hanya melibatkan tabel yang mempunyai tiga kolom atau lebih dan semua kolom adalah kunci. Banyak praktisi berpendapat bahwa menempatkan entitas pada 3NF sudah cukup karena sangat jarang entitas yang berada pada 3NF bukan merupakan 4NF dan 5NF.

5. Bentuk Normal Tahap Keempat

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

6. Bentuk Normal Tahap Kelima

Sebuah tabel berada pada bentuk normal kelima (5NF) jika ia tidak dapat mempunyai dekomposisi lossless menjadi sejumlah tabel lebih kecil.

Empat bentuk normal pertama berdasarkan pada konsep ketergantungan fungsional, sedangkan bentuk normal kelima berdasarkan pada konsep ketergantungan gabungan (*join dependence*) (Janner Simarmata; 2010: 76).

II.13. UML (*Unified Modeling Language*)

Menurut (Windu Gata; 2013: 4) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

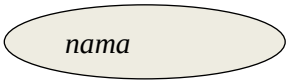
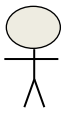
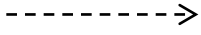
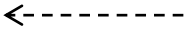
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

1. *Use Case* Diagram

Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem

informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram, yaitu sebagai berikut :

Tabel II.2. Simbol-Simbol pada diagram *Use Case*

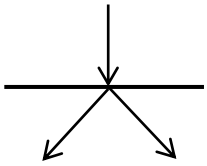
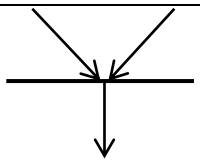
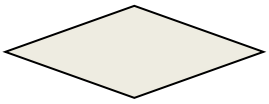
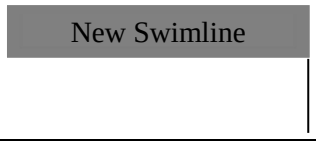
Simbol	Nama	Keterangan
	*Use Case*	Menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
 nama aktor	Aktor	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
		Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
<<include>> 	*Include*	*Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
<<extend>> 	*Extend*	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Windu Gata; 2013: 4)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram*, yaitu :

Tabel II.3. Simbol *Activity Diagram*

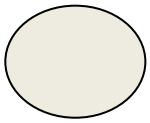
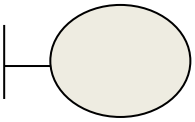
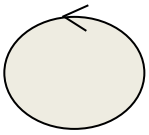
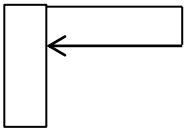
Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktivitas.
	*End point*, akhir aktivitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.
	*Join* (penggabungan) atau Rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false*.
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Windu Gata; 2013: 6)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

Tabel II.4. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek. Control object mengkoordinir pesan antara boundary dengan entitas.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation*, *activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Windu Gata; 2013: 7)

4. *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem.

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan.

Class diagram secara khas meliputi : Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti.

Tabel II.5. *Multiplicity Class Diagram*

Multiplicity	**Penjelasan**
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Windu Gata; 2013: 9)