BAB II

KAJIAN PUSTAKA

2.1 Teori Dasar

Deskripsi teori paling tidak berisi tentang penjelasan terhadap variabel-variabel yang diteliti melalui pendefinisian, dan uraian yang lengkap dan mendalam dari berbagai referensi, sehingga ruang lingkup, kedudukan dan prediksi terhadap hubungan antara variabel yang akan diteliti menjadi lebih jelas dan terarah (Sugiyono, 2014).

Pada bab ini akan dijelaskan tentang beberapa teori dasar antara lain kecerdasan buatan atau *Artificial Intelligence* (*AI*) dan beberapa subdisiplin ilmunya seperti logika *fuzzy* (*fuzzy logic*), jaringan saraf tiruan (*artificial neural network*), dan sistem pakar (*expert system*); *web,* basis data, dan validitas sistem.

2.1.1 Kecerdasan buatan (artificial intelligence)

Menurut (Sri Hartati & Sari Iswanti, 2008) kecerdasan buatan adalah salah satu bidang ilmu komputer yang mendayagunakan komputer sehingga dapat berperilaku cerdas seperti manusia. Ilmu komputer tersebut mengembangkan perangkat lunak dan perangkat keras untuk menirukan tindakan manusia seperti penalaran, pembelajaran, pemecahan masalah, dan sebagainya.

Cerdas berarti memiliki pengetahuan, pengalaman, dan penalaran untuk membuat keputusan dan mengambil tindakan. Untuk membuat sebuah mesin menjadi cerdas (dapat bertindak seperti manusia) maka harus diberi bekal pengetahuan dan diberi kemampuan untuk menalar. Kecerdasan buatan memungkinkan komputer untuk berpikir atau menalar dan menirukan proses belajar manusia sehingga informasi baru dapat diserap sebagai pengetahuan, pengalaman, dan proses pembelajaran serta dapat digunakan sebagai acuan di masa-masa yang akan dating (Sutojo, Mulyanto, & Suhartono, 2011)

Menurut (Jones, 2008) tahun 1950 merupakan saat-saat awal dari *AI* yaitu saat awal sistem komputer dibangun dan ide-ide pembangunan mesin cerdas mulai terbentuk. Pada tahun 1950, Alan Turing menyimpan pertanyaan dalam pikirannya "apakah sebuah mesin mampu untuk berpikir". Alan Turing melakukan percobaan yang cukup sederhana untuk menentukan apakah suatu mesin bisa dikatakan cerdas. Hasil percobaannya ini disebut dengan *Turing Test*. Dalam *Turing Test*, jika sebuah mesin mampu mengelabui seseorang yang menganggap mesin itu adalah manusia, maka mesin itu dianggap telah lulus dari tes kecerdasan (*intelligence test*).

Seseorang yang menjadi subjek percobaan diminta untuk menentukan terminal mana yang terkoneksi dengan komputer. Subjek boleh mengajukan pertanyaan, membuat pernyataan, menanyakan perasaan dan motivasi selama diperlukan. Jika subjek ternyata gagal menentukan terminal mana yang terkoneksi dengan komputer, maka komputer dinyatakan lulus tes dan dikatakan memiliki *consciousness/* kesadaran (Fatta, 2009).

Menurut (Jones, 2008) pada tahun 1956, *Darmouth AI Conference* membawa para peneliti yang terlibat dalam penelitian *AI* seperti John McCarthy bersama peneliti-peneliti lainnya untuk sesi diskusi dan penelitian *AI*di *Dartmouth College*. Sejak saat itu, banyak konferensi *AI* telah diselenggarakan di seluruh dunia, dan berbagai disiplin ilmu belajar di bawah nama *AI*. Pada awal 80-an penelitian tentang *AI* sukses di bidang komersial dari *software* jenis *Expert System*. Pada era 90-an dan awal abad 21, *AI* mencapai kesuksesan terbesarnya setelah diadopsi secara luas oleh industri teknologi, memberikan sumbangan besar pada logistik, data mining, diagnosis medis dan berbagai bidang lainnya (Fatta, 2009).

Kombinasi antara *AI* dengan bidang ilmu yang lainnya melahirkan subdisiplin ilmu dalam *AI*. Beberapa diantaranya adalah logika *fuzzy* (*fuzzy logic*), jaringan syaraf tiruan (*artificial neural network*), dan sistem pakar (*expert system*) (Sutojo et al., 2011).

2.1.2 Logika fuzzy *(fuzzy logic)*

Logika *fuzzy* adalah metodologi sistem kontrol pemecahan masalah, yang sesuai untuk diimplementasikan pada sistem, mulai dari sistem yang sederhana, sistem kecil, *embedded system*, jaringan komputer, *multi-channel* atau *workstation* berbasis akuisisi data, dan sistem kontrol. Dalam logika *fuzzy* memungkinkan nilai keanggotaan berada di antara 0 dan 1, artinya suatu keadaan memungkinkan mempunyai dua nilai “Ya” dan “Tidak” secara bersamaan, namun besar nilainya tergantung pada bobot keanggotaan yang dimilikinya. Logika *fuzzy* dapat

digunakan di berbagai bidang seperti pada sistem diagnosis penyakit (dalam bidang kedokteran); pemodelan sistem pemasaran, sistem operasi (dalam bidang ekonomi); kendali kualitas air, prediksi adanya gempa bumi, klasifikasi dan pencocokan pola (dalam bidang teknik) (Sutojo et al., 2011).

Ada beberapa keuntungan yang dapat diambil ketika menggunakan logika *fuzzy* untuk memecahkan suatu masalah, yaitu (Sutojo et al., 2011):

1. Perancangannya tidak memerlukan persamaan matematik yang rumit
2. Mudah dimengerti
3. Memiliki toleransi terhadap data-data yang tidak tepat
4. Mampu memodelkan fungsi-fungsi nonlinear yang sangat kompleks
5. Dapat membangun dan mengaplikasikan pengalaman-pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan
6. Dapat bekerja sama dengan teknik-teknik kendali secara konvensional
7. Logika *fuzzy* didasarkan pada bahasa alami

Sistem inferensi *fuzzy* adalah cara memetakan ruang *input* menuju ruang *output* menggunakan logika *fuzzy*. Empat elemen dasar sistem inferensi *fuzzy* antara lain (Sutojo, dkk., 2011: 232):

1. Basis pengetahuan *fuzzy*, yaitu kumpulan aturan (*rule*) *fuzzy* dalam bentuk pernyataan *IF…THEN*.
2. Fuzzifikasi, yaitu proses untuk mengubah *input* sistem yang mempunyai nilai tegas menjadi variabel linguistik menggunakan fungsi keanggotaan yang disimpan dalam basis pengetahuan *fuzzy*.

3. Mesin inferensi, yaitu proses untuk mengubah *inputfuzzy* menjadi *outputfuzzy* dengan cara mengikuti aturan-aturan yang telah ditetapkan pada basis pengetahuan *fuzzy*.
4. Defuzzifikasi, yaitu mengubah *outputfuzzy* yang diperoleh dari mesin inferensi menjadi nilai tegas menggunakan fungsi keanggotaan yang sesuai dengan saat dilakukan fuzzifikasi.

Beberapa metode yang digunakan dalam sistem inferensi *fuzzy* adalah (Sutojo et al., 2011):

1. Metode Tsukamoto

Dalam inferensinya, metode Tsukamoto menggunakan tahapan sebagai berikut:

a. Fuzzifikasi
b. Pembentukan basis pengetahuan *fuzzy* (*rule* dalam bentuk *IF…THEN*)
c. Mesin inferensi menggunakan fungsi implikasi *MIN* (*Minimum*)
d. Defuzzifikasi menggunakan metode Rata-rata (*Average*)

2. Metode Mamdani

Metode ini sering digunakan karena strukturnya yang sederhana. Pada metode ini, untuk mendapatkan *output* diperlukan 4 tahapan sebagai berikut:

a. Fuzzifikasi
b. Pembentukan basis pengetahuan *fuzzy* (*rule* dalam bentuk *IF…THEN*)
c. Aplikasi fungsi implikasi menggunakan fungsi *MIN* (*Minimum*) dan komposisi antar-*rule* menggunakan fungsi *MAX*(*Maximum*) dengan menghasilkan himpunan *fuzzy* baru

d. Defuzzifikasi menggunakan metode *Centroid* (Titik Tengah)

3. Metode Sugeno

Metode ini diperkenalkan oleh Takagi-Sugeno Kang pada tahun 1985. Dalam metode ini, *output* sistem berupa konstanta atau persamaan liniear. Dalam inferensinya, metode Sugeno menggunakan tahapan sebagai berikut:

a. Fuzzifikasi
b. Pembentukan basis pengetahuan *fuzzy* (*rule* dalam bentuk *IF…THEN*)
c. Mesin inferensi menggunakan fungsi implikasi *MIN* (*Minimum*)
d. Defuzzifikasi menggunakan metode Rata-rata (*Average*)

2.1.3 Jaringan syaraf tiruan (*artificial neural network*)

Jaringan saraf tiruan adalah paradigma pengolahan informasi yang terinspirasi oleh sistem saraf secara biologis, seperti proses informasi pada otak manusia. Elemen utamanya adalah struktur dari sistem pengolahan informasi yang terdiri dari sejumlah besar elemen pemrosesan yang saling berhubungan (*neuron*), bekerja serentak untuk menyelesaikan masalah tertentu. Cara kerja jaringan saraf tiruan sama seperti cara kerja manusia, yaitu belajar melalui contoh. Beberapa contoh aplikasi jaringan saraf tiruan adalah implementasi di bidang kedokteran, yaitu pemodelan dan diagnosis sistem kardiovaskular, hidung elektronik, dan dokter instan; dan implemetasi di bidang bisnis, yaitu jaringan saraf tiruan yang diintegrasikan dengan merek dagang *The Airline Marketing Tactician* (*AMT*) menggunakan *back-propagation* untuk membantu kontrol pemasaran dari alokasi kursi penerbangan (Sutojo et al., 2011).

Beberapa kelebihan yang dimiliki jaringan saraf tiruan antara lain (Sutojo et al., 2011):

1. Belajar adaptif, yaitu kemampuan untuk mempelajari bagaimana melakukan pekerjaan berdasarkan data yang diberikan untuk pelatihan atau pengalaman awal.
2. *Self-Organization*, yaitu kemampuan membuat organisasi sendiri atau representasi dari informasi yang diterimanya selama waktu belajar.
3. *Real Time Operation*, yaitu perhitungan jaringan saraf tiruan yang dapat dilakukan secara paralel sehingga perangkat keras yang dirancang dan diproduksi secara khusus dapat mengambil keuntungan dari kemampuan ini.

Selain mempunyai beberapa kelebihan, jaringan saraf tiruan juga mempunyai kelemahan-kelemahan, yaitu (Sutojo et al., 2011):

1. Tidak efektif jika digunakan untuk melakukan operasi-operasi numerik dengan presisi tinggi.
2. Tidak efisien jika digunakan untuk melakukan operasi algoritma aritmatika, operasi logika, dan simbolis.
3. Membutuhkan pelatihan untuk dapat beroperasi sehingga bila jumlah datanya besar, waktu yang digunakan untuk proses pelatihan sangat lama.

Salah satu elemen yang menentukan baik tidaknya suatu mode jaringan saraf tiruan adalah hubungan antar-*neuron* atau arsitektur jaringan. *Neuron-neuron* tersebut terkumpul dalam lapisan-lapisan yang disebut *neuron layers*. Terdapat 3 bagian lapisan penyusun jaringan saraf tiruan, yaitu (Sutojo et al., 2011):

1. Lapisan *Input* (*Input Layer*)

Unit-unit dalam lapisan ini disebut unit-unit *input* yang bertugas menerima pola *input*-an dari luar yang menggambarkan suatu permasalahan.

2. Lapisan Tersembunyi (*Hidden Layer*)

Unit-unit dalam lapisan ini disebut unit-unit tersembunyi, yang mana nilai *output*-nya tidak dapat diamati secara langsung.

3. Lapisan *Output* (*Output Layer*)

Unit-unit dalam lapisan ini disebut unit-unit *output*, yang merupakan solusi jaringan saraf tiruan terhadap suatu permasalahan.

Beberapa arsitektur jaringan yang sering digunakan dalam jaringan saraf tiruan antara lain (Sutojo et al., 2011):

1. Jaringan Lapisan Tunggal

Jaringan ini terdiri dari 1 lapisan *input* dan 1 lapisan *output*, yang mana setiap unit dalam lapisan *input* selalu terhubung dengan setiap unit yang terdapat pada lapisan *output*. Jaringan ini menerima *input* kemudian mengolahnya menjadi *output* tanpa melewati lapisan tersembunyi. Contoh jaringan saraf tiruan yang menggunakan jaringan ini adalah *ADALINE, Hopfield*, dan *Perceptron*.

2. Jaringan Lapisan Banyak

Jaringan ini mempunyai 3 jenis lapisan, yaitu lapisan *input*, lapisan tersembunyi, dan lapisan *output*. Jaringan ini dapat menyelesaikan permasalahan yang lebih kompleks dibandingkan dengan jaringan lapisan tunggal. Contoh jaringan saraf tiruan yang menggunakan jaringan ini adalah *MADALINE, backpropagation*, dan *Neocognitron*.

3. Jaringan dengan Lapisan Kompetitif

Jaringan ini memiliki bobot yang telah ditentukan dan tidak memiliki proses pelatihan. Jaringan ini digunakan untuk mengetahui *neuron* pemenang dari sejumlah *neuron* yang ada sehingga sekumpulan *neuron* bersaing untuk mendapatkan hak menjadi aktif. Contoh jaringan saraf tiruan yang menggunakan jaringan ini adalah *Learning Vector Quantization* (*LVQ*).

Berdasarkan cara memodifikasi bobotnya, pelatihan jaringan saraf tiruan dibagi menjadi dua, yaitu (Sutojo et al., 2011):

1. Pelatihan dengan Supervisi (pembimbing)

Dalam pelatihan ini, jaringan dipandu oleh sejumlah pasangan data (masukan dan target) yang berfungsi sebagai pembimbing untuk melatih jaringan hingga diperoleh bobot yang terbaik. Algoritma yang termasuk dalam pelatihan dengan supervisi antara lain:

a. *Hebb-Rule*

Model ini diperkenalkan oleh D.O. Hebb yang menggunakan cara menghitung bobot dan bias secara iteratif dengan memanfaatkan model pembelajaran dengan supervisi sehingga bobot dan bias dapat dihitung secara otomatis tanpa harus melakukan cara coba-coba. Arsitektur jaringan ini terdiri dari beberapa unit *input* dihubungkan langsung dengan sebuah unit *output*, ditambah dengan sebuah bias.

b. *Perceptron*

Model ini ditemukan oleh Rosenblatt (1962) dan Minsky – Papert (1969). Model jaringan ini merupakan model yang terbaik pada saat itu. Algoritma

pelatihan *perceptron* digunakan baik untuk *input* biner maupun bipolar, dengan θ tertentu.

c. *Delta-Rule*

Selama pelatihan pola, *Delta-Rule* akan mengubah bobot dengan cara meminimalkan *error* antara *output* jaringan dengan target.

d. *Backpropagation*

Backpropagation adalah metode penurunan gradien untuk meminimalkan kuadrat *error* keluaran. Pelatihan jaringan ini terdiri dari 3 tahap, yaitu tahap perambatan maju (*forward propagation*), tahap perambatan balik, dan tahap perubahan bobot dan bias. Arsitektur jaringan ini terdiri dari *input layer*, *hidden layer*, dan *output layer*.

e. *Heteroassociative Memory*

Jaringan saraf *heteroassociative memory* adalah jaringan yang dapat menyimpan kumpulan pengelompokan pola dengan cara menentukan bobot-bobotnya sedemikian rupa. Algoritma pelatihan yang biasa digunakan adalah *Hebb-Rule*.

f. *Bidirectional Associative Memory* (*BAM*)

Bidirectional Associative Memory (*BAM*) adalah model jaringan saraf yang memiliki 2 lapisan, yaitu lapisan *input* dan lapisan *output* yang mempunyai hubungan timbal balik antara keduanya (bersifat *bidirectional*). Arsitektur jaringan ini terdiri dari 3 *neuron* pada lapisan *input* dan 2 *neuron* pada lapisan *output*. Model jaringan ini terbagi menjadi 2 jenis yaitu *BAM* Diskrit dan *BAM*Kontinu.

g. *Learning Vector Quantization* (*LVQ*)

Learning Vector Quantization (*LVQ*) adalah suatu model pelatihan pada lapisan kompetitif terawasi yang akan belajar secara otomatis untuk mengklasifikasikan vektor-vektor *input* ke dalam kelas-kelas tertentu.

2. Pelatihan tanpa Supervisi

Dalam pelatihan ini, tidak ada pembimbing yang digunakan untuk memandu proses pelatihan. Jaringan hanya diberi *input* tetapi tidak mendapatkan target yang diinginkan sehingga modifikasi bobot pada jaringan dilakukan menurut parameter tertentu. Model jaringan yang termasuk dalam pelatihan tanpa supervisi adalah jaringan kohonen yang diperkenalkan oleh Prof. Teuvo Kohonen pada tahun 1982.

Pada jaringan kohonen, *neuron-neuron* pada suatu lapisan data akan menyusun dirinya sendiri berdasarkan *input* nilai tertentu dalam suatu *cluster*. *Cluster* yang dipilih sebagai pemenang adalah *cluster* yang mempunyai vektor bobot paling cocok dengan pola *input*, yaitu *cluster* yang memiliki jarak yang paling dekat.

2.1.4 Sistem pakar (expert sytem)

System pakar pertama kali di kembangkan oleh komunitas AI pada pertengahan tahun 1960. Sistem pakar yang muncul pertama kali adalah general-purpose problem solver (GPS) yang di kembangkan oleh newel dan simon. GPS dan program-program yang serupa ini mengalami kegagalan di karenakan

cakupannya terlalu luas sehingga terkadang meninggalkan pengetahuan-pengetahuan penting yang seharusnya di sediakan.

Sampai saat ini sudah banyak system pakar yang di buat, menurut (Sutojo et al., 2011) beberapa contoh MYCIN digunakan untuk Diagnosa penyakit, DENDRAL digunakan untuk Mengidentifikasi struktur molecular campuran yang tak di kenal, XCON dan XSEL digunakan untuk Membantu konfigurasi system computer besar, SOPHIE digunakan untuk Analisis sirkit elektronik, Prospector di gunakan di dalam geologi untuk membantu mencari dan menemukan deposit, POLIO digunakan untuk Membantu memberikan keputusan bagi seorang manager dalam hal stok broker dan investasi, DELTA digunakan untuk Pemeliharaan lokomotif listrik disel.

Menurut (Sri Hartati & Sari Iswanti, 2008) sistem pakar merupakan salah satu teknik kecerdasan buatan yang menirukan proses penalaran manusia. Sistem pakar adalah suatu sistem yang dirancang untuk dapat menirukan keahlian seorang pakar dalam menjawab pertanyaan dan memecahkan suatu masalah. Dengan bantuan sistem pakar, seseorang yang bukan pakar dapat menjawab pertanyaan, menyelesaikan masalah serta mengambil keputusan yang biasanya dilakukan oleh seorang pakar (Sutojo et al., 2011).

Pakar adalah seseorang yang memiliki pengetahuan khusus, pemahaman, pengalaman, dan metode-metode yang digunakan untuk memecahkan masalah dalam bidang tertentu. Seorang pakar memiliki kemampuan kepakaran seperti mengenali dan merumuskan suatu masalah, menyelesaikan masalah dengan cepat dan tepat, menjelaskan solusi dari suatu masalah, restrukturisasi pengetahuan,

belajar dari pengalaman, memahami batas kemampuan, kemampuan untuk mengaplikasikan pengetahuannya dan memberi saran serta pemecahan masalah pada bidang tertentu (Sri Hartati & Sari Iswanti, 2008). Suatu sistem dikatakan sebagai sistem pakar jika memiliki ciri-ciri sebagai berikut (Sutojo et al., 2011):

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 tertentu
5. Mudah dimodifikasi
6. Basis pengetahuan dan mekanisme inferensi diletakkan terpisah
7. Keluarannya (*output*) bersifat anjuran
8. Sistem dapat mengaktifkan kaidah secara terpisah secara searah, sesuai dengan dialog dengan pengguna

Menurut (Sri Hartati & Sari Iswanti, 2008) representasi pengetahuan dimaksudkan untuk mengorganisasikan pengetahuan dalam bentuk dan format tertentu agar dapat dimengerti oleh komputer. Pemilihan representasi pengetahuan yang tepat akan menghasilkan sebuah sistem pakar yang efektif. Salah satu model representasi pengetahuan yang penting yaitu kaidah produksi (*production rule*).

Sistem pakar pada penelitian ini menggunakan model representasi pengetahuan berbasis kaidah produksi. Menurut Firebaugh (1988) dalam (Sri Hartati & Sari Iswanti, 2008) struktur sistem pakar yang berbasis kaidah produksi terdiri dari 4 komponen, yaitu:

1. Antarmuka pemakai

antarmuka merupakan penghubung antara pemakai dengan sistem pakar. Komponen ini berfungsi sebagai alat komunikasi antara sistem dan pengguna (*user*) yang penting sekali bagi pengguna. Komponen ini harus didesain sedemikian rupa sehingga efektif dan mudah digunakan terutama bagi pengguna yang tidak ahli dalam bidang yang diterapkan pada sistem pakar (Sri Hartati & Sari Iswanti, 2008).

2. Basis pengetahuan

Basis pengetahuan adalah komponen yang berisi sekumpulan kaidah yang berasal dari pengetahuan dalam domain tertentu dan secara umum disajikan dalam bentuk kaidah produksi (*IF…THEN…*). Pengetahuan pakar yang disajikan dalam format tertentu didapat dari sekumpulan pengetahuan pakar dan sumber-sumber pengetahuan lainnya seperti buku-buku, jurnal ilmiah, majalah, maupun dokumentasi tercetak lainnya. Basis pengetahuan diletakkan terpisah dari mesin inferensi agar pengembangan pengetahuan sistem pakar dapat dilakukan secara leluasa tanpa mengganggu mesin inferensi (Sri Hartati & Sari Iswanti, 2008).

3. Struktur kontrol (Mesin Inferensi)

Struktur kontrol merupakan *interpreter* kaidah atau mesin inferensi yang menggunakan pengetahuan-pengetahuan yang tersimpan dalam basis pengetahuan untuk memecahkan atau menyelesaikan permasalahan yang ada.

Dalam melakukan proses inferensi, sistem pakar memerlukan adanya proses pengujian kaidah-kaidah dalam urutan tertentu untuk mencari suatu kondisi yang sesuai dengan kondisi awal atau untuk memastikan kondisi yang sedang berjalan

sudah dimasukkan ke dalam *database*. Proses pengujian itu disebut dengan perunutan atau penalaran, yaitu proses pencocokan fakta atau kondisi tertentu yang tersimpan dalam basis pengetahuan maupun pada memori kerja dengan kondisi yang dinyatakan dalam premis atau bagian kondisi pada suatu kaidah atau aturan (Sri Hartati & Sari Iswanti, 2008).

Ada beberapa konsep penalaran yang dapat digunakan oleh mesin inferensi yaitu:

a. Penalaran maju (*forward chaining*)

Konsep ini dapat juga disebut sebagai pencarian yang dimotori data (*data driven search*). Runut maju melakukan proses perunutan (penalaran) dimulai dari premis-premis atau informasi masukan (*IF*) terlebih dahulu kemudian menuju konklusi atau *derived information* (*THEN*). Konsep ini dapat dimodelkan sebagai berikut:

IF (informasi masukan)

THEN (konklusi)

Informasi masukan dapat berupa suatu pengamatan sedangkan konklusi dapat berupa diagnosa sehingga dapat dikatakan jalannya penalaran runut maju dimulai dari pengamatan menuju diagnosa. Pada metode ini, sistem tidak melakukan praduga apapun terhadap konklusi, namun sistem akan menerima semua gejala yang diberikan pengguna lalu sistem akan memeriksa gejala-gejala tersebut dan selanjutnya mencocokkan dengan konklusi yang sesuai (Sri Hartati & Sari Iswanti, 2008).

b. Penalaran mundur (*backward chaining*)

Secara umum, konsep ini diaplikasikan ketika tujuan ditentukan sebagai kondisi atau keadaan awal. Konsep ini disebut juga *goal-driven search*. Arah penalaran atau perunutan dalam konsep ini berlawanan dengan *forward chaining*. Konsep ini dapat dimodelkan sebagai berikut:

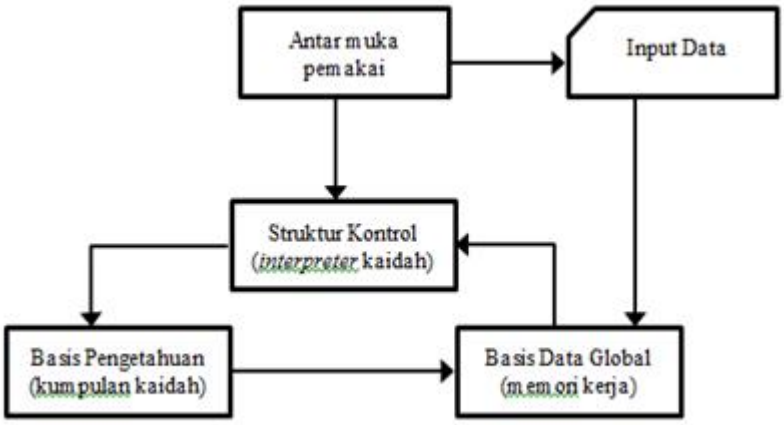
Tujuan,

IF (kondisi)

Proses penalaran pada *backward chaining* dimulai dari tujuan kemudian merunut balik ke jalur yang mengarah ke tujuan tersebut, untuk membuktikan bahwa bagian kondisi pada kaidah atau aturan benar-benar terpenuhi. Proses *internal* selalu memeriksa konklusi (tujuan) terlebih dahulu sebagai praduga awal, kemudian memeriksa dan memastikan gejala-gejala (kondisi) telah terpenuhi dan selanjutnya mengeluarkan konklusi sebagai *output*. Jika sistem menemukan ada bagian kondisi yang tidak terpenuhi maka sistem akan memeriksa konklusi (tujuan) pada aturan atau kaidah berikutnya (Sri Hartati & Sari Iswanti, 2008).

4. *Working memory* (memori kerja) atau basis data global

Berfungsi untuk mencatat status masalah yang terjadi dan *history* solusi. Memori kerja merupakan bagian yang berisi fakta-fakta masalah yang ditemukan dalam suatu sesi saat proses konsultasi terjadi.



Gambar 2. 1 Struktur Sistem Pakar Kaidah Produksi

(sumber : firebought, 1988 dalam (Sri Hartati & Sari Iswanti, 2008))

Kaidah *IF-THEN* menghubungkan antesenden (*antecedent*) dengan konsekuensi yang diakibatkannya. Berikut ini adalah contoh struktur kaidah *IF-THEN* yang menghubungkan obyek (Adedeji, 1992 *dalam* (Sri Hartati & Sari Iswanti, 2008)):

1. *IF* premis *THEN* konklusi
2. *IF* masukan *THEN* keluaran
3. *IF* kondisi *THEN* tindakan
4. *IF* antesenden *THEN* konsekuen
5. *IF* data *THEN* hasil
6. *IF* tindakan *THEN* tujuan
7. *IF* aksi *THEN* reaksi
8. *IF* gejala *THEN* diagnosa

Premis mengacu pada fakta yang harus benar sebelum konklusi tertentu dapat diperoleh. Masukan mengacu pada data yang harus tersedia sebelum

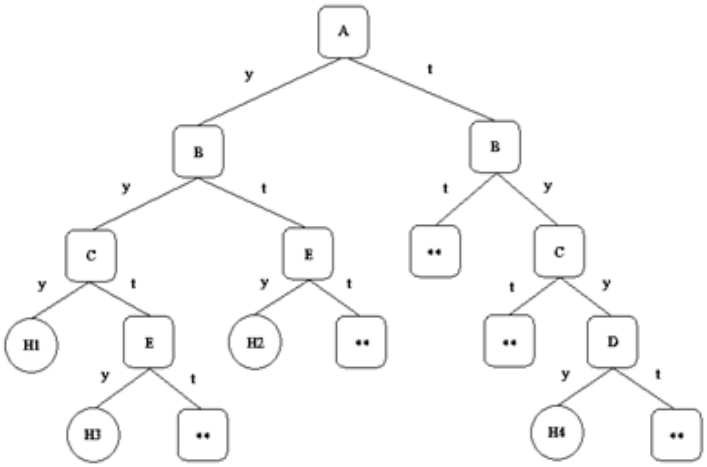
keluaran dapat diperoleh. Kondisi mengacu pada keadaan yang harus berlaku sebelum tindakan dapat diambil. Antesenden mengacu situasi yang terjadi sebelum konsekuensi dapat diamati. Data mengacu pada informasi yang harus tersedia sehingg sebuah hasil dapat diperoleh. Tindakan mengacu pada kegiatan yang harus dilakukan sebelum hasil dapat diharapkan. Aksi mengacu pada kegiatan yang menyebabkan munculnya efek dari tindakan tersebut. Gejala mengacu pada keadaan yang menyebabkan adanya kerusakan atau keadaan tertentu yang mendorong adanya pemeriksaan (diagnosa) (Sri Hartati & Sari Iswanti, 2008).

Sebelum sampai pada bentuk kaidah produksi, pengetahuan yang berhasil didapatkan dari domain tertentu disajikan dalam bentuk tabel keputusan kemudian dibuat pohon keputusannya. Berikut ini adalah contoh penyajian dalam bentuk tabel keputusan dan pohon keputusan (Sri Hartati & Sari Iswanti, 2008).

Tabel 2. 1 Tabel Keputusan

Hipotesa	**Hipotesa**	**Hipotesa**	**Hipotesa**	**Hipotesa**
Evidence A	Ya	Ya	Ya	Tidak
Evidence B	Ya	Tidak	Ya	Ya
Evidence C	Ya	Tidak	Tidak	Ya
Evidence D	tidak	Tidak	Tidak	Ya
Evidence E	tidak	Ya	Ya	Tidak

Sumber : (Sri Hartati & Sari Iswanti, 2008)



Gambar 2. 2 Pohon Keputusan

(Sri Hartati & Sari Iswanti, 2008)

Keterangan:

A = *evidence* A, H1 = hipotesa 1, y = ya

B = *evidence* B, H2 = hipotesa 2, t = tidak

C = *evidence* C, H3 = hipotesa 3, ** = tidak menghasilkan hipotesa tertentu

D = *evidence* D, H4 = hipotesa 4

Dari gambar 2.2 dapat diketahui bahwa hipotesa H1 terpenuhi jika memenuhi *evidence* A, B, dan C. Hipotesa H2 terpenuhi jika memiliki *evidence* A dan *evidence* E. Hipotesa H3 akan terpenuhi jika memiliki *evidence* A, B, dan E. Hipotesa H4 akan dihasilkan jika memenuhi *evidence* B, C, dan D. Notasi "y" mengandung arti memenuhi *node* (*evidence*) di atasnya, notasi "t" artinya tidak memenuhi.

Dalam sesi konsultasi pada sistem pakar, *node-node* yang mewakili *evidence* biasanya akan menjadi pertanyaan yang diajukan oleh sistem. Dengan

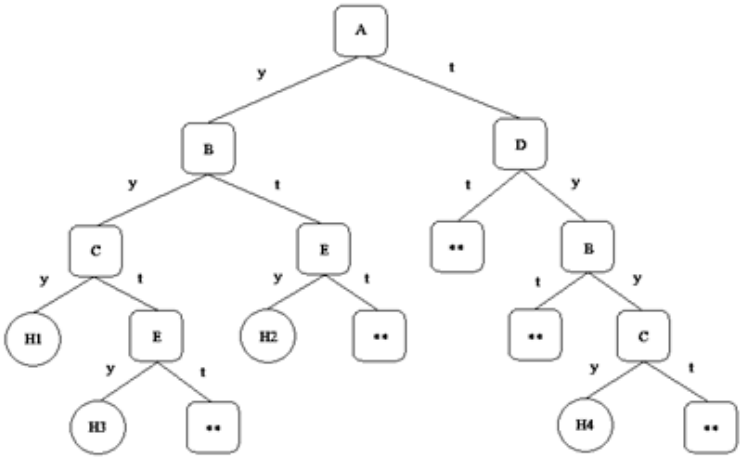
melihat pohon keputusan pada gambar 2.2 permasalahan dapat saja terjadi pada awal sesi konsultasi yaitu pada saat sistem pakar menanyakan “apakah memiliki *evidence* A?”. Permasalahannya adalah apapun jawaban pengguna baik “ya” atau “tidak” maka sistem akan menanyakan *evidence* B. Ini berarti jawaban pengguna tidak akan mempengaruhi sistem. Salah satu cara untuk mengatasi hal ini adalah dengan mengubah urutan pada tabel keputusan seperti terlihat pada tabel 2.2

Tabel 2. 2 Alternatif Tabel Keputusan

Hipotesa	**Hipotesa**	**Hipotesa**	**Hipotesa**	**Hipotesa**
Evidence A	Ya	Ya	Ya	Tidak
Evidence D	Tidak	Tidak	Tidak	ya
Evidence B	Ya	Tidak	Ya	Ya
*Evidence*C	Ya	Tidak	Tidak	Ya
Evidence E	Tidak	Ya	Ya	Tidak

Sumber : (Sri Hartati & Sari Iswanti, 2008)

Berdasarkan tabel 2.2 dapat dihasilkan pohon keputusan sebagai berikut:



Gambar 2. 3 Alternatif Pohon Keputusan

sumber : (Sri Hartati & Sari Iswanti, 2008)

Keterangan:

A = *evidence* A, H1 = hipotesa 1, y = ya

B = *evidence* B, H2 = hipotesa 2, t = tidak

C = *evidence* C, H3 = hipotesa 3, ** = tidak menghasilkan hipotesa tertentu

D = *evidence* D, H4 = hipotesa 4

Dilihat dari gambar 2.3, masing-masing *node* yang mewakili *evidence* tertentu untuk kondisi "y" dan "t" sudah tidak mengarah pada *evidence* yang sama. Hal ini berarti jawaban pengguna yang berbeda akan mengarah pada pertanyaan yang berbeda pula.

Kaidah yang dapat dihasilkan berdasarkan pohon keputusan pada gambar 2.3 adalah sebagai berikut:

1. Kaidah 1: *IF* A *AND* B *AND* C *THEN* H1
2. Kaidah 2: *IF* A *AND* B *AND* E *THEN* H3
3. Kaidah 3: *IF* A *AND* E *THEN* H2
4. Kaidah 4: *IF* D *AND* B *AND* C *THEN* H4

Model representasi pengetahuan kaidah produksi banyak digunakan pada aplikasi sistem pakar karena model representasi ini mudah dipahami dan bersifat deklaratif sesuai dengan jalan pikiran manusia dalam menyelesaikan suatu masalah, dan mudah diinterpretasikan.

Adapun kelebihan yang dimiliki sistem pakar antara lain:

1. Membuat seorang yang awam dapat bekerja seperti layaknya seorang pakar.
2. Dapat bekerja dengan informasi yang tidak lengkap atau tidak pasti.

3. Meningkatkan *output* dan produktifitas, bekerja lebih cepat dari manusia sehingga mengurangi jumlah pekerja yang dibutuhkan dan akan mereduksi biaya.
4. Meningkatkan kualitas.
5. Sistem pakar menyediakan nasihat yang konsisten dan dapat mengurangi tingkat kesalahan.
6. Membuat peralatan yang kompleks lebih mudah dioperasikan karena sistem pakar dapat melatih pekerja yang tidak berpengalaman.
7. Handal (*realibility*).
8. Sistem pakar tidak dapat lelah atau bosan serta konsisten dalam memberikan jawaban dan selalu memberikan perhatian penuh.
9. Memiliki kemampuan untuk memecahkan masalah yang kompleks.
10. Memungkinkan pemindahan pengetahuan ke lokasi yang jauh serta memperluas jangkauan seorang pakar, dapat diperoleh dan dipakai dimana saja. Sistem pakar merupakan arsip yang terpercaya dari sebuah keahlian sehingga *user* seolah-olah berkonsultasi langsung dengan sang pakar meskipun sang pakar sudah pensiun.

Selain memiliki beberapa kelebihan yang dapat dimanfaatkan, sistem pakar juga memiliki beberapa kekurangan, yaitu (Sutojo et al., 2011):

1. Biaya yang sangat mahal untuk membuat dan memeliharanya.
2. Sulit dikembangkan karena keterbatasan keahlian dan ketersediaan pakar.
3. Sistem pakar tidak 100% bernilai benar.

2.2 Variabel penelitian

Variabel penelitian merupakan segala sesuatu yang berbentuk apa saja yang dimiliki orang, obyek atau kegiatan yang mempunyai variasi tertentu yang ditetapkan oleh peneliti untuk dipelajari sehingga diperoleh informasi tentang hal tersebut, kemudian ditarik kesimpulannya (Sugiyono, 2014). Obyek yang digunakan dalam penelitian ini adalah narkoba dan variabel penelitian yang ditetapkan yaitu mengenai kecanduan narkoba.

2.2.1 Narkoba

NAPZA adalah singkatan dari narkotika alkohol, psikotropika dan zat adiktif lainnya (Fr & W, 2013). Napza ini juga di sebut dengan istilah "NARKOBA" singkatan dari narkotika dan obat berbahaya. Napza maupun Narkoba dua istilah yang marak di pergunjingkan orang dan menyerang masyarakat kita terutama genari mudanya. Narkotika secara etimologi berasal dari bahasa yunani narkoum, yang berari membuat lumpuh atau membuat mati rasa. Pada dasarnya narkotika memiliki khasiat dan manfaat digunakan dalam bidang kedokteran, kesehatan dan pengobatan serta berguna abagi penelitian perkembangan, ilmu pengetahuan farmasi atau farmakologi itu sendiri. Sedangkan dalam bahasa inggris *narcotic* lebih mengarah keobat yang membuat penggunanya kecanduan. Narkotika adalah zat yang dapat menimbulkan pengaruh tertentu bagi mereka yang menggunakan dengan cara memasukkan obat tersebut kedalam tubuhnya, pengaruh tersebut berupa pembiusan, hilangnya rasa sakit rangsangan, semangat dan halusinasi. Dengan timbunya efek halusinasi inilah

yang menyebabkan kelompok masyarakat terutama di kalangan remaja ingin menggunakan narkotika meskipun tidak menderita apa-apa. Hal ini yang mengakibatkan terjadinya penyalahgunaan narkotika (obat). Bahaya menggunakan narkotika bila tidak sesuai dengan peraturan dapat menyebabkan adanya adiksi/ketergantungan obat (ketagihan). Adiksi adalah suatu kelainan obat kronik/periodic sehinggga penderita kehilangan control terhadap dirinya dan menimbulkan kerugian terhadap dirinya dan masyarakat. Orang orang yang sudah terlibat pada penyalahgunaan narkotika pada mulanya masih pada ukuran (dosis) yang normal. Lama-lama pengguna obat menjadi kebiasaan, setelah biasa menggunakan narkotika, kemudian untuk menimbulkan efek yang sama di perlukan dosis yang tinggi (toleransi). Setelah fase toleransi ini berakhir menjadi ketergantungan, merasa tidak dapat hidup tanpa narotika.

Narkotika adalah zat atau obat yang berasal dari tanaman atau bukan tanaman, baik sintesis maupun semi sintesis (Fr & W, 2013). Zat tersebut menyebabkan penurunan atau perubahan kesadaran, mengilanhkan rasa, mengurangi hingga menghilangkan rasa nyeri, dan dapat menimbulkan ketergantungan (adiktif). (UU No.22 tahun 1997), WHO sendiri memberikan definisi tentang narkotika sebagai berikut “Narkotika merupakan suatu zat yang apabila di masukkan kedalam tubuh akan memengaruhi fungi fisik dan psikologi kecuali (makanan, air, dan oksigen)”. Narkotika secara farmakologiik adalah apioida, seiring berjalannya waktu keberadaan narkoba bukan sebagai penyembuh namun justru menghancurkan. Awalnya narkoba masih di gunakan sesekali dalam dosis kecil dan tentu sejak dampaknya tidak terlalu berarti. Namun perubahan

zaman dan mobilititas kehidupan membuat narkoba menjadi bagian dari gaya hidup, dari yang tadinya hanya sekedar perangkat medis, kini narkoba mulai tenar digabungkan sebagai dewa dunia, penghilang rasa sakit.

1. Pengertian kecanduan narkoba

Ketergantungan/kecanduan narkoba adalah suatu keadaan atau kondisi yang diakibatkan penyalahgunaan narkoba yang disertai dengan adanya toleransi zat (dosis semakin meningkat) dan gejala putus zat (withdrawal syindrome). Sedangkan menurut pakar dari hasil penelitian ketergantungan/ kecanduan adalah dorongan untuk menggunakan narkoba terus-menerus, dan apabila pemakaiannya di hentikan maka terjadi putus zat. Berat ringannya gejala kecanduan tergantung pada jenis narkoba, dosis yang digunakan, serta semakin lam pemakaian akan semakin meningkat. Menurut jenis tingkatan/range kecanduan narkoba di bagi menjadi 3 yaitu:

a. Kecanduan ringan
b. Kecanduan sedang
c. Kecanduan berat

Kecanduan ringan adalah apabila pemakaian narkoba masih dalam tahap 1-4 kali dalam satu bulan.

Kecanduan sedang adalah apabila pemakaian narkoba masih dalam tahap 4-8 kali dalam satu bulan.

Kecanduan berat adalah apabila pemakaian narkoba di konsumsi setiap hari hari dalam satu bulan dan terus-menerus.

2.2.2 Gejala kecanduan narkoba

Narkoba dapat dideteksi dari jenis gejala atau ciri-ciri yang ditimbulkan akibat jenis narkoba yang dikonsumsi, hal tersebut biasanya selalu berhubungan dengan perubahan baik fisik maupun perilaku yang ditimbulkan oleh pengguna narkoba. Jenis narkoba yang dikonsumsi oleh pengguna narkoba seperti ganja, cocain, heroin, ekstasi, shabu, inhalen, dan alkohol. Untuk mengetahui seseorang pengguna narkoba, maka dibutuhkan suatu program sistem pakar, dimana sistem tersebut dapat mengenali jenis narkoba yang digunakan berdasarkan gejala-gejala yang ditimbulkan oleh pengguna narkoba.

Efek narkoba/narkotika tergantung pada dosis pemakaian, cara pemakaian, pemakaian sebelumnya dan harapan pengguna. Selain kegunaan medis untuk mengobati neyeri, batuk dan diare akut, narkotika menghasilkan perasaan “lebih membaik” yang dikenal dengan eforia dengan mengurangi tekanan psikis. Efek ini dapat mengakibatkan ketergantungan (Fr & W, 2013) .Adapun tanda-tanda umum seseorang yang telah kecanduan narkoba/narkotika dapat di lihat dari beberapa hal antara lain:

1. Anak jadi pemurung dan penyendiri
2. Wajah anak pucat dan kuyu
3. Terdapat bau aneh yang tidak biasa di kamar anak
4. Matanya berair dan tangannya gemetar
5. Nafasnya tersengal dan susah tidur
6. Badanya lesu dan selalu gelisah

7. Anak menjadi mudah tersinggung, dan marah.
8. Merokok pada usia remaja
9. Cenderung menarik diri dari acara keluarga dan lebih senang mengurung di kamar
10. Bergaul dengan teman hingga larut malam bahkan jarang pulang kerumah
11. Sering bersenang-senang di pesta, diskotik maupun kumpul di mall
12. Menghindar dari tanggung jawab yang sesuai, malas menyelesaikan tugas rutin di rumah
13. Prestasi belajar menurun, dan sering bolos sekolah
14. Prilaku mulai menyimpang seperti mencuri, pergaulan seks bebas, dan suka mabuk-mabukan

2.3 *Software* Pendukung

Software pendukung merupakan beberapa perangkat lunak yang digunakan untuk mendukung pembuatan sistem pakar dalam penelitian ini. Perangkat lunak tersebut antara lain: *XAMPP, phpMyAdmin, PHP, HTML, CSS, jQuery, MySQL, Notepad++,* dan *StarUML*.

2.3.1Web

Menurut (Sidik & Pohan, 2009) *WWW* (*World Wide Web*) atau yang lebih dikenal dengan *web* merupakan salah satu layanan yang didapat oleh pengguna komputer yang terhubung dengan internet. *Web* pada awalnya adalah ruang informasi dalam internet, dengan menggunakan teknologi *hypertext*. Pengguna

dituntun untuk menemukan informasi dengan mengikuti *link* yang disediakan dalam dokumen *web* yang ditampilkan dalam *browserweb*. Sekarang *web* menjadi standar *interface* pada layanan-layanan yang ada di internet seperti komunikasi melalui *e-mail, chatting,* transaksi bisnis, pencarian informasi, dan sebagainya.

Web memudahkan pengguna komputer untuk berinteraksi dengan pelaku internet lainnya dan menelusuri informasi di Internet. Banyak perusahaan yang mengadopsi *web* sebagai bagian dari strategi teknologi informasinya karena beberapa alasan yaitu: akses informasi yang mudah, *setup server* lebih mudah, informasi mudah didistribusikan, dan bebas *platform*, artinya informasi dapat disajikan oleh *browser web* pada sistem operasi apapun karena adanya standar dokumen berbagai tipe data yang disajikan (Sidik & Pohan, 2009).

2.3.2 Database (*basis data*)

Menurut (A.S & Shalahuddin, 2013) sistem *database* adalah sistem terkomputerisasi yang tujuan utamanya adalan memelihara data atau informasi yang sudah diolah dan membuat informasi tersedia saat dibutuhkan. *Database* adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat. Kebutuhan basis data meliputi memasukkan, menyimpan, dan mengambil data serta membuat laporan berdasarkan data yang telah disimpan. Salah satu bentuk basis data yang dibutuhkan dalam sebuah sistem yaitu *Database Management System* (*DBMS*). *DBMS* adalah suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola, dan menampilkan data. Syarat minimal dari *DBMS* antara lain (A.S & Shalahuddin, 2013):

1. Menyediakan fasilitas untuk mengelola akses data
2. Mampu menangani integritas data
3. Mampu menangani akses data yang dilakukan secara bersamaan
4. Mampu menangani *backup* data

Ada beberapa *DBMS* yang paling banyak digunakan saat ini antara lain:

1. *DBMS* versi komersial, yaitu *Oracle, Microsoft SQL Server*, *IBM DB2*, dan *Microsoft Access*
2. *DBMS* versi *open source*, yaitu *MySQL, PostgreSQL, Firebird*, dan *SQLite*

Dalam alur hidup basis data (*Database Life Cycle*), terdapat tahapan yang dinamakan *physical database design*. Biasanya pada tahap ini dibuat rancangan fisik *database* yaitu *Physical Data Model* (*PDM*). *PDM* adalah model yang menggunakan sejumlah tabel untuk mengambarkan data serta hubungan antar data. Setiap tabel mempunyai sejumlah kolom yang mempunyai nama unik beserta tipe data yang digunakan. *PDM* merupakan konsep yang digunakan untuk menerangkan secara detail bagaimana data disimpan dalam *database*. *PDM* sudah dalam bentuk fisik perancangan *database* yang siap diimplementasikan ke dalam *DBMS* sehingga nama tabel pada *PDM* merupakan nama asli tabel yang akan diimplementasikan ke dalam *DBMS* (A.S & Shalahuddin, 2013).

2.3.3 *XAMPP* (*XApache MySQL PHP Perl*)

Gambar 2. 4 Logo XAMPP

(Sumber: https://wiki.bitnami.com/@api/deki/files/527/=xampp-logo.jpg)

Menurut (Sidik & Pohan, 2009) *server web* adalah komputer yang digunakan untuk menyimpan dokumen-dokumen *web* yang akan melayani permintaan dokumen *web* dari kliennya. *XAMPP* adalah sebuah perangkat lunak *web server apache* yang didalamnya sudah tersedia *databaseserver MySQL* dan dapat mendukung pemrograman *PHP*. *XAMPP* merupakan *software* yang mudah digunakan , gratis, dan mendukung instalasi di *Linux* dan *Windows*. Keuntungan lainnya adalah cukup dengan menginstal *XAMPP* sudah tersedia *Apache Web Server*, *MySQL Database Server*, *PHPsupport* (*PHP* 4 dan *PHP* 5) dan beberapa modul lainnya.

2.3.4 *PhpMyAdmin*

Gambar 2. 5 Logo phpMyAdmin

(Sumber: https://www.phpmyadmin.net/static/images/logo-og.png)

phpMyAdmin adalah perangkat lunak gratis yang ditulis dalam bahasa pemrograman *PHP* bertujuan untuk menangani administrasi *MySQL* melalui *web*. *phpMyAdmin* mendukung berbagai operasi pada *MySQL* dan *MariaDB*. Operasi-operasi yang sering digunakan seperti mengelola *database*, tabel, kolom, relasi, indeks, *users*, *permissions*, dan lain-lain, dapat dilakukan melalui antarmuka pengguna dengan tetap dapat mengeksekusi pernyataan *SQL* secara langsung. (www.phpmyadmin.net/).

2.3.5 *PHP: Hypertext Preprocessor*(*PHP*)

Menurut (Welling & Thomson, 2009) *PHP Hypertext Prepocessor* (*PHP*) adalah bahasa *scripting server-side* yang didesain khusus untuk sebuah *web*. Kode *PHP* yang akan dijalankan dapat ditanamkan atau disisipkan dalam sebuah halaman *HTML* (*Hyper Text Markup Language*). Kode *PHP* akan

diinterpretasikan pada *web server* dan menghasilkan *HTML* atau *output* lainnya yang akan dilihat oleh pengunjung.

PHP adalah proyek *open source*, yang berarti memiliki akses ke kode sumber dan tidak membutuhkan biaya untuk menggunakan, mengubah, dan mendistribusikannya. Awalnya kepanjangan dari *PHP* adalah *Personal Home Page*, tetapi mengalami perubahan sejalan dengan konvensi penamaan *GNU* rekursif (*GNU = Gnu's Not Unix*) dan sekarang kepanjangan dari *PHP*adalah*PHP*: *Hypertext Preprocessor* (Welling & Thomson, 2009).

Gambar 2. 6 Logo php

(Sumber: https://www.php.net/download-logos.php)

Beberapa pesaing utama *PHP* adalah *Perl*, *Microsoft ASP.NET*, *Ruby on Rails*, *JavaServer Pages* (*JSP*), dan *ColdFusion*. Dibandingkan dengan produk-produk tersebut, *PHP* memiliki banyak kelebihan, yaitu (Welling & Thomson, 2009):

1. Kinerja

Kinerja *PHP* sangat cepat. Cukup dengan menggunakan server tunggal yang tidak mahal sudah dapat melayani jutaan hits per hari.

2. Skalabilitas

PHP memiliki arsitektur "*shared-nothing*" yang berarti bahwa skala horizontal dengan sejumlah besar komoditas server dapat diterapkan secara efektif dengan biaya yang murah.

3. Integrasi *database*

PHP memiliki koneksi bawaan (asli) yang tersedia untuk beberapa sistem *database* seperti: *MySQL, PostgreSQL*, *Oracle*, *dbm*, *FilePro*, *DB2*, *Hyperwave*, *Informix, Interbase*, dan *Sybase database*.

4. *Built-in libraries*

PHP dirancang untuk digunakan pada *web* sehingga memiliki banyak fungsi *built-in* untuk dapat melakukan banyak tugas *web* yang berguna seperti menghasilkan gambar dengan cepat, koneksi ke layanan *web* dan layanan jaringan lainny

poa, *parse XML* (*eXtended Markup Language*), mengirim *e-mail*, bekerja dengan *cookie*, dan menghasilkan dokumen *PDF*. Semua itu dilakukan cukup dengan beberapa baris kode.

5. Biaya yang murah

Menggunakan *PHP* tidak membutuhkan biaya alias gratis. *PHP* versi terbaru dapat di-*download* setiap saat dari situs resminya tanpa biaya.

6. Mudah dipelajari dan digunakan

Sintaks *PHP* didasarkan pada bahasa pemrograman lainnya, terutama bahasa *C* dan *Perl*. Jika pengguna sudah mengetahui bahasa *C* atau *Perl*, atau bahasa *C* lainnya seperti *C++* atau *Java*, maka akan segera menjadi produktif dalam menggunakan *PHP*.

7. Dukungan *object-oriented* yang kuat

PHP memiliki rancangan fitur *object-oriented* yang baik seperti pewarisan (*inheritance*),atribut dan metode *private* dan *protected*, kelas dan metode *abstract*, antarmuka (*interfaces*), *constructors*, *destructors*, dan *iterators*.

8. Portabilitas

PHP tersedia untuk banyak sistem operasi yang berbeda. Kode *PHP* dapat ditulis pada sistem operasi *Unix* seperti *Linux*, *FreeBSD*, *Solaris* dan *IRIX*, *Operating System X* (*OS X*) maupun sistem operasi *Microsoft Windows* yang memiliki versi yang berbeda-beda. Kode yang ditulis biasanya akan bekerja tanpa dimodifikasi pada sistem lain yang menjalankan *PHP*.

9. Fleksibilitas dalam pembangunan

PHP memungkinkan penerapan tugas-tugas sederhana secara ringkas dan sama-sama mudah beradaptasi dalam penerapan aplikasi yang besar menggunakan kerangka kerja berdasarkan pola desain seperti *Model View Control* (*MVC*).

10. Tersedianya kode sumber (*source code*)

PHP menyediakan akses ke kode sumber *PHP* (bersifat *open-source*), tidak seperti produk komersial lainnya yang bersifat *closed-source*. Pada *PHP*, dapat dilakukan modifikasi atau penambahan di dalamnya secara bebas tanpa harus menunggu produsen untuk mengeluarkan *patches* (program kecil untuk perbaikan sistem).

11. Tersedianya *support* (dukungan) dan dokumentasi

Zend Technologies, perusahaan di balik mesin yang menggerakkan *PHP*, mendanai pengembangan *PHP* dengan menawarkan *support* (dukungan) dan

perangkat lunak terkait pada basis komersial. Dokumentasi dan komunitas *PHP* adalah sumber daya yang matang dan kaya akan informasi yang dapat dibagi.

2.3.6 *HTML* (*Hyper Text Markup Language*)

Gambar 2. 7 Logo HTML

(Sumber: https://www.w3.org/html/logo/downloads/HTML5_Logo_512.png)

HTML adalah bahasa pemrograman yang digunakan untuk membuat dokumen *HTML* atau dikenal sebagai *web page*. Dokumen *HTML* merupakan file teks murni yang dapat dibuat menggunakan *editor* teks apapun. Dokumen *HTML* disajikan dalam *browser web surfer* seperti *Mozzila*, *Google Chrome*, *Internet Explorer*, dan sebagainya. Dokumen ini biasanya berisi informasi atau *interface* aplikasi di dalam internet (Sidik & Pohan, 2009).

Elemen merupakan istilah bagi komponen-komponen dasar pembentuk dokumen *HTML*. *TagHTML* digunakan untuk menandai berbagai elemen dalam suatu dokumen *HTML*. Elemen dasar *HTML* yang dibutuhkan untuk membuat suatu dokumen *HTML* ditandai dengan *tag*<html>, <head>, dan <body> berikut

tag-tag pasangannya yaitu </html>, </head>, dan </body>. Setiap dokumen *HTML* harus mempunyai pola dasar sebagai berikut (Sidik & Pohan, 2009):

<html>

<head>

… informasi tentang dokumen *HTML*

</head>

<body>

… informasi yang ditampilkan dalam *browser web*

</body>

</html>

1. Setiap dokumen *HTML* harus diawali dengan menuliskan *tag*<html> dan *tag*</html> di akhir dokumen. *Tag* ini menandai elemen *html* yang berarti dokumen ini adalah dokumen *HTML*.
2. Elemen *head* ditandai dengan *tag*<head> dan *tag*</head>. Elemen ini berisi informasi tentang dokumen *HTML*. Minimal informasi yang dituliskan dalam elemen ini adalah judul dokumen yang ditandai dengan menggunakan *tag*<title> dan diakhiri dengan *tag*</title>.
3. Elemen *body* ditandai dengan *tag*<body> dan diakhiri dengan *tag*</body>. Elemen ini merupakan elemen terbesar di dalam dokumen *HTML*. Elemen ini

mengandung isi dokumen yang akan ditampilkan pada *browser* yang meliputi paragraf, grafik, *link*, tabel, dan sebagainya.

2.3.7 *CSS* (*Cascading Style Sheet*)

Menurut (Sidik & Pohan, 2009) *CSS* merupakan fitur baru dari *HTML 4.0*. Hal ini diperlukan setelah melihat perkembangan *HTML* menjadi kurang praktis karena *web pages* terlalu banyak dibebani hal-hal yang berkaitan dengan faktor tampilan seperti *font*, dan lain-lain. Bentuk penggunaan *CSS* dapat dimodelkan sebagai berikut :

Selector {*property*: *value*}

Selector merupakan elemen yang akan didefinisikan, *property* adalah *atribute* yang akan diubah, dan *value* adalah nilai yang akan diberikan. Jika nilai yang akan diberikan berupa kata-kata, gunakan tanda petik ganda sebelum dan sesudah *value*(”*value*”).

Gambar 2. 8 Logo CSS

(Sumber: http://w3widgets.com/responsive-slider/img/css3.png)

Terdapat tiga cara pendefinisian dalam menggunakan *CSS*, yaitu (Sidik & Pohan, 2009):

1. *Style sheet external*

Pada teknik ini, *style sheet* didefinisikan di luar dokumen *HTML* dan disimpan dalam *file* berekstensi *css* (*.css). Dalam pendefinisian *external* tidak perlu lagi menggunakan *taghtml* diawal dan akhir dokumen.

2. *Style sheet internal*

Style sheet didefinisikan secara *internal* biasanya karena *web page* tertentu bersifat sangat unik sehingga membutuhkan definisi terpisah dibandingkan dengan *web page* lainnya.

3. *Inline style sheet*

Style sheet inline hanya bisa digunakan pada lokasi yang sangat spesifik dimana *style sheet* ditempatkan. Kekurangan dari teknik ini adalah dokumen menjadi lebih besar karena *style* didefinisikan satu per satu.

2.3.8 *JavaScript dan jQuery*

Gambar 2. 9 Logo JavaScript

(Sumber: http://www.w3devcampus.com/wp-content/uploads/logoAndOther/logo_JavaScript.png)

Menurut (Sidik & Pohan, 2009) *JavaScript* merupakan modifikasi dari bahasa *C++* dengan pola penulisan yang lebih sederhana. Beberapa hal penting dalam *JavaScript* adalah:

1. Menggunakan blok awal "{" dan blok akhir "}"
2. *Automatic conversion* dalam pengoperasian tipe data yang berbeda
3. *Sensitive case*, yaitu membedakan antara huruf kecil dan huruf capital sehingga harus berhati-hati dalam menggunakan nama variabel , fungsi, dan lain-lain.
4. *Extension* umumnya menggunakan "*.js"
5. Setiap *statement* dapat diakhiri tanda baca *semi colon* (;) dapat juga tidak
6. Jika tidak didukung oleh *browser* versi lama, *script* dapat disembunyikan diantara *tag* "<!--" dan "-->"
7. Jika program dalam satu baris terlalu panjang dapat disambung ke baris berikutnya menggunakan karakter *backslash* (\)

Gambar 2. 10 Logo jQuery

(Sumber: http://precision-software.com/wp-content/uploads/2014/04/jQurery.gif)

jQuery adalah sebuah *framework* berbasiskan *JavaScript*, yang berisi kumpulan kode atau fungsi *JavaScript* yang siap digunakan sehingga

mempermudah dalam membuat kode *JavaScript*. Beberapa kemampuan yang dimiliki *jQuery* antara lain (Warman & Zahni, 2013):

1. memanipulasi elemen *HTML*
2. memanipulasi *CSS*
3. penanganan *event* pada *HTML*
4. efek-efek *JavaScript* dan animasi

2.3.9 *MySQL*dan *SQL*

MySQL adalah *RDBMS* (*Relational Database Management System*) yang sangat cepat dan kuat. Sebuah *database* memungkinkan untuk menyimpan, mencari, mengurutkan, dan mengambil data secara efisien. *MySQL server* bertugas mengendalikan akses ke data yaitu untuk memastikan bahwa beberapa pengguna dapat bekerja dengan data-data itu secara konkuren, untuk memberikan akses yang cepat terhadap data, dan memastikan bahwa hak akses diberikan hanya kepada orang yang berwenang. *MySQL* adalah *multiuser* dan *multithread server*yang menggunakan*SQL (Sturctured Query Language*) sebagai standar bahasa pengelolaan *database* (Welling & Thomson, 2009).

Gambar 2. 11 Logo MySQL

(Sumber: https://www.mysql.com/about/legal/logos.html)

Ada beberapa pesaing utama *MySQL*, yaitu *PostgreSQL*, *Microsoft SQL Server*, dan *Oracle*. Dibanding pesaing utamanya, *MySQL* memiliki banyak kelebihan, antara lain (Welling & Thomson, 2009):

1. Kinerja yang tinggi

MySQL tidak dapat disangkal mengenai kecepatannya. Pada tahun 2002, *MySQL* menjadi yang terbaik dibandingkan *database* penggerak aplikasi *web* lainnya.

2. Biaya yang murah

MySQL tersedia tanpa biaya di bawah lisensi *open source* atau dengan biaya yang murah di bawah lisensi komersial. Lisensi diperlukan jika pengguna ingin mendistribusikan *MySQL* sebagai bagian dari aplikasi dan tidak ingin lisensi aplikasi menjadi lisensi *open source*. Lisensi tidak perlu dibeli jika pengguna tidak berniat untuk mendistribusikan aplikasi (seperti pada kebanyakan aplikasi *web*), atau ingin menggunakan perangkat lunak secara gratis atau *open source*.

3. Mudah digunakan

Kebanyakan *database* modern menggunakan *SQL* karena *MySQL* lebih mudah dikonfigurasi dan digunakan dibanding produk lain sejenisnya. Jika pengguna sudah pernah menggunakan *RDBMS* lain, akan menjadi lebih mudah untuk beradaptasi menggunakan *MySQL*.

4. Portabilitas

MyQL dapat digunakan pada banyak sistem operasi baik sistem *Unix*maupun *Microsoft Windows*.

5. Tersedianya kode sumber

Sebagaimana *PHP*, MySQL menyediakan akses untuk memperoleh dan memodifikasi kode sumber (*source code*). Hal ini memang tidak begitu penting bagi kebanyakan pengguna selama ini, tetapi dengan adanya akses ke kode sumber akan memberikan ketenangan pikiran, memastikan kelangsungan data di masa depan dan memberikan pilihan untuk pengguna dalam keadaan darurat (*emergency*) tertentu.

6. Tersedianya dukungan (*support*)

Tidak semua produk *open source* memiliki induk perusahaan yang menawarkan dukungan (*support*), pelatihan, konsultasi, dan sertifikasi. Pengguna dapat mendapatkan semua manfaat itu dari *MySQL AB*.

Menurut (A.S & Shalahuddin, 2013) hampir semua *DBMS* baik yang komersial maupun *open source* saat ini berbasis *Relational DBMS* (*RDBMS*). *SQL* (*Structured Query Language*) adalah bahasa yang digunakan untuk mengelola data pada *RDBMS*. *SQL* mulai digunakan sebagai standar yang resmi pada tahun 1986 oleh *ANSI* (*American National Standards Institute*) dan pada tahun 1987 oleh *ISO* (*International Organization for Standardization*). Secara umum, terdapat 4 teknik pengaksesan data pada *DBMS* menggunakan *SQL*, antara lain (A.S & Shalahuddin, 2013):

1. Memasukkan data (*insert*)

Bentuk *query* (perintah) untuk memasukkan data dapat dimodelkan sebagai berikut:

```
INSERT INTO nama_tabel(kolom1, kolom2, ...,kolomN)
```

```
VALUES (‘isikolom1’, ‘isikolom1’, …, ‘isikolomN’);
```

2. Mengubah data (*update*)

Bentuk *query* untuk mengubah data dapat dimodelkan sebagai berikut:

```
UPDATE nama_tabel

SET

    kolom1 = ‘isikolom1’

WHERE

    kolom2 = ‘isikolom2’;
```

3. Menghapus data (*delete*)

Bentuk *query* untuk menghapus data dapat dimodelkan sebagai berikut:

```
DELETE FROM nama_tabel

WHERE

    kolom1 = ‘isikolom1’;
```

4. Menampilkan data (*select*)

Bentuk *query* untuk menampilkan data dapat dimodelkan sebagai berikut:

```
SELECT kolom1, kolom2

FROM nama_tabel

WHERE

    kolom1 = ‘isikolom1’;
```

2.3.10 *Notepad++*

Gambar 2. 12 Logo notepad++

(Sumber: https://www.gavick.com/blog/wp-content/uploads/2010/07/notepadd.jpg)

Menurut (Gilmore, 2010) *Notepad++* merupakan editor teks *open source* yang matang dan diakui sebagai pengganti *Notepad.Notepad++* tersedia untuk platform *Windows* yang dapat digunakan untuk menulis kode dengan beberapa pilihan bahasa (pemrograman). *Notepad++* menawarkan beragam kenyamanan fitur yang diharapkan dari setiap kemampuan *IDE*(*Integrated Development Environment*), termasuk kemampuan untuk menunjukkan baris tertentu dari suatu dokumen sebagai referensi yang mudah; sintaks, tanda kurung, *indentation highlighting*, fasilitas pencarian yang tangguh, *macro recording* untuk tugas-tugas seperti memasukkan *template* komentar, dan sebagainya. Salah satu kelebihan *Notepad++* adalah dukungan dasar untuk *auto-completion* dari nama fungsi yang ditawarkan sehingga akan mengurangi beberapa proses pengetikan kode.

2.3.11 *StarUML*

Salah satu pemodelan yang saat ini paling banyak digunakan adalah *UML*(*Unified Modeling Language*). *UML* adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek (A.S & Shalahuddin, 2013).

Menurut (A.S & Shalahuddin, 2013) *UML* muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak. *UML* tidak terbatas pada metodologi pemrograman tertentu, meskipun pada kenyataanya *UML* paling banyak digunakan pada metodologi berorientasi objek.

Gambar 2. 13 Logo StarUML

(Sumber: http://staruml.sourceforge.net/image/staruml-logo.jpg)

StarUML merupakan salah satu *CASE* (*Computer-Aided Software Engineering*) *tools* atau perangkat pembantu berbasis komputer untuk rekayasa perangkat lunak yang mendukung alur hidup perangkat lunak (*life cycle support*). *StarUML* termasuk ke dalam kelompok *upperCASE tools* yang mendukung perencanaan strategis dan pembangunan perangkat lunak (A.S & Shalahuddin, 2013).

Terdapat 13 macam diagram dalam *UML 2.3* yang dibagi menjadi 3 kategori yaitu (A.S & Shalahuddin, 2013):

1. *Structure diagrams*

Kategori ini terdiri dari kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan. Diagram *UML* yang termasuk dalam kategori ini antara lain *class diagram, object diagram, component diagram, composite structure diagram, package diagram,* dan *deployment diagram*.

2. *Behaviour diagrams*

Kategori ini terdiri dari kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem. Diagram *UML* yang termasuk dalam kategori ini antara lain *use case diagram, acitivity diagram,* dan *state machine diagram*.

3. *Interaction diagrams*

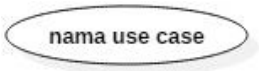
Kategori ini terdiri dari kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem. Diagram *UML* yang termasuk dalam kategori ini antara lain *sequence diagram, communication diagram, timing diagram,* dan *interaction overview diagram*.

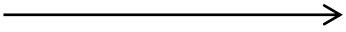
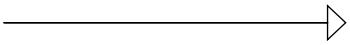
Menurut (A.S & Shalahuddin, 2013) *use case* dan *sequence diagram* merupakan bagian dari desain sistem. Dalam penelitian ini, diagram yang akan digunakan untuk desain sistem yaitu:

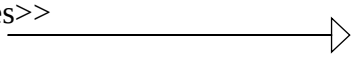
1. *Use case diagram*

Use case diagram merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu sistem atau lebih aktor dengan sistem informasi yang akan dibuat. *Use case diagram*digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Ada 2 hal utama yang terdapat pada *use case* yaitu aktor dan *use case*. Berikut ini adalah simbol-simbol yang digunakan dalam *use case diagram* (A.S & Shalahuddin, 2013).

Tabel 2. 3 Simbol use case diagram

Simbol	**Deskripsi**
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*
Aktor/*actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri. Aktor belum tentu merupakan orang, biasanya

	dinyatakan menggunakan kata benda di awal frase nama actor
asosiasi/*association* 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan actor
Ekstensi/*extend* <<extend>> 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walaupun tanpa *use case* tambahan itu. Arah panah mengarah pada *use case* yang ditambahkan.
generalisasi/*generalization* 	Hubungan generalisasi dan spesialisasi (umum – khusus) antara 2 buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari fungsi lainnya. Arah panah mengarah pada *use case* yang menjadi generalisasinya (umum)

Menggunakan/*include*/*uses* <<include>>  <<uses>> 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankannya *use case* ini. Arah panah mengarah pada *use case* yang ditambahkan

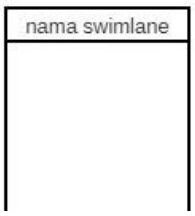
Sumber : (A.S & Shalahuddin, 2013)

2. *Activity diagram*

Activity diagram merupakan diagram yang menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Jadi dapat dikatakan bahwa *activity diagram* menggambarkan aktifitas sistem, bukan apa yang dilakukan oleh aktor. Simbol-simbol yang digunakan dalam *activity diagram* ditampilkan dalam tabel berikut (A.S & Shalahuddin, 2013).

Tabel 2. 4 Simbol activity diagram

Simbol	**Deskripsi**
Status awal 	Status awal aktivitas sistem, sebuah diagram aktifitas memiliki sebuah status awal
Aktifitas	Aktifitas yang dilakukan sistem,

	aktifitas biasanya diawali dengan kata kerja
Percabangan/*decision* 	Asosiasi percabangan dimana jika ada pilihan aktifitas lebih dari satu
Penggabungan/*join* 	Asosiasi penggabungan dimana lebih dari satu aktifitas digabungkan menjadi satu
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktifitas memiliki sebuah status akhir
Swimlane  atau 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktifitas yang terjadi

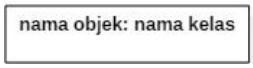
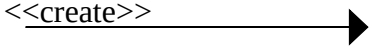
Sumber : (A.S & Shalahuddin, 2013)

3. *Sequence diagram*

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup (*life cycle*) objek dan *message* (pesan) yang dikirimkan dan diterima antar objek. Jumlah *sequence diagram* yang harus digambar minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri. Semakin banyak *use case* yang didefinisikan semakin banyak pula *sequence diagram* yang harus dibuat. Simbol-simbol yang digunakan pada *sequence diagram* ditampilkan dalam tabel berikut (A.S & Shalahuddin, 2013).

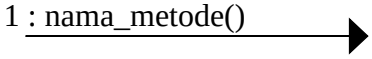
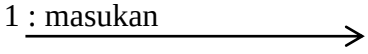
Tabel 2. 5 Simbol sequence diagram

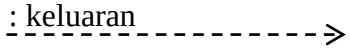
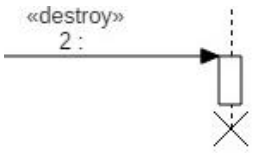
Simbol	**Deskripsi**
Aktor/*actor*  	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri. Aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase nama actor
Garis hidup/*lifeline* 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan actor

Objek 	Menyatakan objek yang berinteraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya. Aktor tidak memiliki waktu aktif
Pesan tipe *create* 	Menyatakan suatu objek membuat objek yang lain. Arah panah mengarah pada objek yang dibuat

Sumber : (A.S & Shalahuddin, 2013)

Tabel 2. 6 Lanjutan

Simbol	**Deskripsi**
pesan tipe *call* 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri. Arah panah mengarah pada objek yang memiliki operasi/metode.
Pesan tipe *send* 	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya. Arah panah mengarah pada objek yang dituju

pesan tipe *return* 1 : keluaran 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu. Arah panah mengarah pada objek penerima
Pesan tipe *destroy* 	Menyatakan suatu objek mengakhiri hidup objek lain. Arah panah mengarah pada objek yang diakhiri

Sumber : (A.S & Shalahuddin, 2013)

2.4 Penelitian Terdahulu

Untuk mendukung teori yang berkaitan dengan penelitian, peneliti mencantumkan beberapa penelitian terdahulu di bidang sistem pakar dalam kategori diagnosis.

1. Judul Jurnal : Sistem Pakar Untuk Mendiagnosa Pengguna Narkoba Dengan Menggunakan Metode Bayes.

   Nama Jurnal : Pelita Informatika Budi Darma

   Penulis Jurnal : Ismail Syaputra

   ISSN/Vol/No/Thn: 2301-9425/V/3/2013

   Pembahasan : Sistem pakar ini menggunakan metode bayes yang proses pencarian solusinya berdasarkan nilai probabilitas hipotesa dan probabilitas *evidence*. Pada aplikasi ini, setiap gejala akan merujuk pada satu kesimpulan

yakni berupa hasil diagnosa. Sistem pakar ini menghasilkan berupa hasil diagnosa yang disertai dengan nilai-nilai pada perhitungan masing-masing narkoba tersebut serta penelusuran diagnosa dari gejala- gejala narkoba yang diderita. Penulis dapat merancang aplikasi sistem pakar untuk mendiagnosa gejala narkoba yang ditimbulkan dengan menggunakan metode bayes.

2. Judul Jurnal : Sistem Pakar Diagnosa Gejala Kecanduan Game Online Dengan Menggunakan Metode Certainty Factor.

   Nama Jurnal : Pelita Informatika Budi Darma

   Penulis Jurnal : Ericksan Sianturi

   ISSN/Vol/No/Thn: 2301-9425/VII/3/2014

   Pembahasan : Proses diagnosa gejala kecanduan *game online* dapat dilakukan dengan mengenali semua gejala-gejala kecanduan *game online* dan hasil kriteria yang didapat akan dianalisa dengan menggunakan metode tertentu. Penerapan Metode *Certainty Factor* sangat cocok digunakan untuk melakukan diagnose gejala kecanduan *game online,* karna dengan menggunakan metode ini sistem yang dirancang akan lebih mudah untuk dipahami oleh para penggunanya. Perancangan aplikasi Sistem Pakar diagnose gejala kecanduan *game online* dilakukan dengan menggunakan pemograman Visual Studio 2008, dan semua data yang dimasukkan merupakan hasil perhitungan menggunakan metode *certainty factor* (Sianturi, 2014)

3. Judul Jurnal : Sistem Pakar Diagnosa Kecanduan Menggunakan Internet (Internet Addiction) Menggunakan Metode Certainty Factor.

Nama Jurnal : Pelita Informatika Budi Darma

Penulis Jurnal : Adlin Hasibuan

ISSN/Vol/No/Thn: 2301-9425/VI/3/2014

Pembahasan : *Internet addiction* diukur melalui alat ukur berbentuk skala *Internet Addiction.* Semakin tinggi skor yang diperoleh seseorang dalam skala *internet addiction* yang diberikan, semakin tinggi *internet addiction* yang dirasakannya. *Certainty Factor* diterapkan dalam mengukur *Internet Addiction* dengan cara, skala *Internet Addiction* diberi nilai bobot untuk setiap gejala. Kemudian nilai bobot tersebut dikalikan dengan nilai yang diberikan user, selanjutnya nilai setiap gejala dikombinasikan dan terakhir dihitung persentasi keyakinannya. Perancangan aplikasi ini menggunakan *Visual Basic 6.0* dan database *Microsoft Acces* 2007.

4. Judul Jurnal : Sistem Pakar Diagnosa Penyakit Puyu Dengan Metode Forward Chaining, *Expert System of Quail Disease Diagnosis Using Forward Chaining Method.*

Nama Jurnal : *Indonesian Journal Of Electrical Engineering And Computer Science*

Penulis Jurnal : B.Herawan Hayadi, Kasman Rukun, Rizky Ema Wulansari, Tutut Herawan, Dahliusmanto, Dafid Setaiwan, Safril

ISSN/Vol/No/Thn: 2502-4752/5/1/2017

Pembahasan : Sistem pakar adalah ilmu yang dikembangkan sejak tahun 1950, yang aplikasinya sangat luas, terutama bagi organisasi yang mengharapkan peningkatan nilai, produktivitas, dan kemampuan manajerial

dalam mengambil keputusan. Sistem pakar unggas di dunia merupakan elemen penting untuk keunggulan dalam membuat diagnosis perawatan antisipasi orthat dapat dilakukan dengan lebih tepat dan akurat. Tiga bagian utama yaitu karakteristik sistem, ada expertis Knowledge Base, Inference Engine, dan User Interface. Penyakit yang menyerang burung puyuh membantu petani dalam mengantisipasi gejala yang disebabkan pengobatan dengan cepat, akurat, dan efisien. Hal ini bisa mengurangi kerugian yang bisa diakibatkan oleh penyebaran penyakit yang kini cenderung berbahaya. Peternak dapat meningkatkan produktivitas dengan mendeteksi penyakit dini. Keuntungan dari penerapan sistem pakar untuk diagnosis penyakit ini tergantung pada hasil perhitungan tingkat kepercayaan dalam mendukung proses inferensi terhadap penyimpanan data dan fakta pada Knowledge Base. Metode forward chaining dapat memberikan hasil yang akurat dari kesimpulan diagnosis yang dihasilkan. Penggunaan metode Forward Chaining sangat mudah dengan penentuan Rule, dan perhitungan berdasarkan fakta yang muncul sebagai sebuah gejala. Hal yang perlu diperhatikan dalam metode Forward Chaining ini terhadap gejala yang ditimbulkan akan mempengaruhi besarnya kesimpulan yang didapat. Jangan mengesampingkan kemungkinan untuk pengembangan lebih lanjut dengan kombinasi aturan yang lebih kompleks sehingga kompleksitas diagnosisnya bisa memberikan hasil yang lebih memuaskan.

The expert system is the science being developed since 1950, whose application has been very extensive, especially for organizations who expect increased value, productivity, and managerial ability in decision decision. Poultry expert system in

the world an as important element to excellence in making a diagnosis orthat anticipation treatment can be done more precisely and accurately. The three main parts that is characteristic of systems, there are expertis the Knowledge Base, Inference Engine, and User Interface. The diseases that attack quail is helping farmers in anticipation the symptoms caused to treatment quickly, accurately, and efficiently. This can reduce the losses which can be caused by spreading the disease which is now likely to be dangerous. The rancher can increase productivity by detection of early disease. The advantage of the application of expert system for diagnosis of the disease is dependent on the results of the calculation of the level of confidence in supporting the process of inference to data and facts store on the Knowledge Base. Forward chaining method can provide accurate results from the conclusion of the resulting diagnosis. Use of Forward Chaining method is very easy with the determination of the Rule, and calculation based on facts that appear as a symptom. Things to consider in this Forward Chaining method to the symptoms caused will affect the magnitude of the conclusions obtained. Do not rule out the possibility for further development by a combination of rule more complex so that the complexity of the diagnosis can provide more satisfactory results.

5. Judul Jurnal : Sistem Pakar untuk Menentukan Skala Prioritas Kasus di Indonesia Laboratorium Forensik Menggunakan Forward Chaining dan Metode Chaining Backward Rule Based. *Expert System to Determine the Priority Scale of Case in Laboratory of Forensic Using Forward Chaining and Backward Chaining Methods Rule Based.*

   Nama Jurnal : International Journal Of Innovative Research In Advanced Engineering (IJIRAE)

   Penulis Jurnal : Setiawan widiyanto, bayu surarso, oky dwi nurhayati

ISSN/Vol/No/Thn: 2349-2163/IV/3/2017

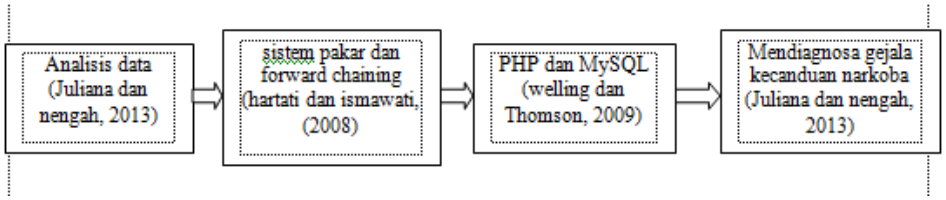
Pembahasan : Sistem pakar dengan metode forward chaining dan backward chaining menerapkan 16 peraturan dan 6 fakta dengan 8 kondisi serta skala prioritas 6 level dengan menggunakan usia tersangka, pendudukan korban, pelaku kerja, kerangka waktu. Adegan dan perhatian sebagai basis kasus prioritas ke objek laboratorium Forensik cabang Semarang memiliki kecocokan 100% dan mampu memberikan informasi kasus yang masuk dalam enam tahap pemeriksaan prioritas berdasarkan peraturan yang mana telah dipasang di sistem yang bisa dijadikan referensi ke tingkat tertentu dan juga output informasi yang dihasilkan mampu memberikan prioritas informasi status secara berkala dan periodik untuk meningkatkan tingkat penyelesaian kasus ditargetkan sehingga bisa digunakan untuk memantau penyelesaian kasus.

Expert system with methods forward chaining and backward chaining are implementing 16 rules and 6 facts with 8 conditions as well as 6-level scale of priorities using a suspect's age, occupation of victims, work actors, time frames the scene and attention as the basis for priority cases to the object Forensic laboratory Semarang branch have compatibility 100% and able to provide information cases that fall into six levels of priority examination is based on the rules which have been installed in the system that can be used as a reference to a certain level and also outputs information produced is capable of providing status information priorities periodically and periodic

case to raise the completion level of case were targeted so that it can be used for monitoring the completion of case.

2.5 Kerangka Pemikiran

Kerangka pemikiran memuat pemikiran terhadap alur yang dipahami sebagai acuan dalam pemecahan masalah yang diteliti secara logis dan sistematis. Kerangka berfikir yang baik akan menjelaskan secara teoritis pertautan antar variabel yang diteliti (Sugiyono, 2014). Berikut ini adalah kerangka pemikiran yang mendasari penelitian ini.



Gambar 2. 14 Kerangka Pemikiran

(Sumber : Data Penelititian 2017)

Data-data yang dibutuhkan berkaitan dengan narkoba dianalisis terlebih dahulu agar lebih sederhana dan mudah dilakukan proses pengolahan datanya. Data-data tersebut kemudian diolah menggunakan metode sistem pakar *forward chaining* untuk membuat aturan (*rule*) yang akan digunakan. Sistem pakar dengan metode *forward chaining* dibuat menggunakankan bahasa pemrograman *PHP* dan *database MySQL* sehingga menghasilkan sebuah sistem pakar untuk diagnosa gejala kecanduan narkoba menggunakan metode *forward chaining* berbasis *web*.