BAB II
KAJIAN PUSTAKA

2.1. Teori Dasar

Teori adalah seperangkat konstruk (konsep), definisi dan proposisi yang berfungsi untuk melihat fenomena secara sistematik melalui spesifikasi hubungan antar variabel (Sugiyono, 2013:52).

2.1.1. Kecerdasan Buatan (*Artificial Intelligence*)

Kecerdasan buatan berasal dari Bahasa inggris "*Artificial Intelligence*" atau disingkat AI, yaitu *intelligence* adalah kata sifat yang berarti cerdas, sedangkan *Artificial* artinya buatan. Kecerdasan buatan yang dimaksud di sini merujuk pada mesin yang mampu berpikir, menimbang tindakan yang akan diambil dan mampu mengambil keputusan seperti yang dilakukan manusia (Sutojo, dkk., 2011:1).

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. Aktivitas menusia yang ditirukan seperti penalaran, penglihatan, pembelajaran, pemecahan masalah, pemahaman bahasa alami dan sebagainya (Hartati dan Iswanti, 2008: 1).

Beberapa macam cabang ilmu kecerdasan buatan antara lain Jaringan Saraf Tiruan, Logika *Fuzzy* dan Sistem Pakar.

1. Jaringan Saraf Tiruan

Jaringan saraf tiruan adalah paradigma pengolahan informasi yang terinspirasi oleh sistem saraf secara biologis, seperti proses informasi pada otak manusia. Elemen kunci dari paradigma ini adalah struktur dari sistem pengolahan informasi yang terdiri dari senjumlah besar elemen pemrosesan yang saling berhubungan (neuron), bekerja serentak untuk menyelesaikan masalah tertentu (Sutojo, dkk., 2011: 283).

Beberapa metode yang digunakan dalam inferensi jaringan saraf tiruan adalah:

1. *Hebb Rule*

   Hebb rule adalah metode pembelajaran yang dilakukan dengan cara memperbaiki nilai bobot sedemikian rupa sehingga jika ada 2 neuron yang terhubung, dan keduanya pada kondisi hidup pada saat yang sama, maka bobot keduanya dinaikkan.

2. *Perceptron*

   Perceptron biasanya digunakan untuk mengklasifikasikan suatu tipe pola tertentu yang sering dikenal dengan pemisahan secara linear.

3. *Delta Rule*

   Pada *delta rule* akan mengubah bobot yang menghubungkan antara jaringan input ke unit output dengan nilai target.

4. *Backpropagation*

Backpropagation merupakan algoritma pembelajaran yang terawasi dan biasanya digunakan oleh perceptron dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan neuron-neuran yang ada pada lapisan tersembunyi.

2. Logika *Fuzzy (Fuzzy Logic)*

Dalam logika klasik dinyatakan bahwa segala sesuatu bersifat biner, yang artinya adalah hanya mempunyai dua kemungkinan, "Ya atau Tidak", "Benar atau Salah", "Baik atau Buruk" dan lain-lain. Oleh karena itu, semua ini dapat mempunyai nilai keanggotaan 0 atau 1. Akan tetapi, dalam logika *fuzzy* memungkinkan nilai keanggotaan berada di antara 0 dan 1. Artinya, bisa saja suatu keadaan mempunyai dua nilai "Ya atau Tidak", "Benar atau Salah", "Baik atau Buruk" secara bersamaan, namun besar nilainya tergantung pada bobot keanggotaan yang dimilikinya (Sutojo, dkk., 2011: 211).

Beberapa metode yang digunakan dalam Sistem Inferensi *Fuzzy* adalah:

1. Metode Tsukamoto

Pada metode Tsukamoto, setiap konsekuen pada aturan yang berbentuk *IF-THEN* harus direpresentasikan dengan suatu himpunan *fuzzy* dengan fungsi keanggotaan yang monoton. Sebagai hasilnya, *output* hasil inferensi dari tiap-tiap aturan yang diberikan secara tegas. Hasil akhirnya diperoleh dengan menggunakan rata-rata terbobot (Kusumadewi, 2003: 180).

2. Metode Mamdani

Metode Mamdani sering juga dikenal dengan nama metode *Max-Min*. Pada metode mamdani, fungsi implikasi yang digunakan adalah Min. Apabila digunakan fungsi implikasi *Min*, maka metode komposisi ini sering disebut dengan nama *Max-Min* atau *Min-Max* atau Mamdani (Kusumadewi, 2003:186).

3. Metode Sugeno

Penalaran dengan metode Sugeno hampir sama dengan penalaran Mamdani, hanya saja output sistem tidak pernah berupa himpunan *fuzzy*, melainkan berupa konstanta atau persamaan linear (Kusumadewi, 2003:194).

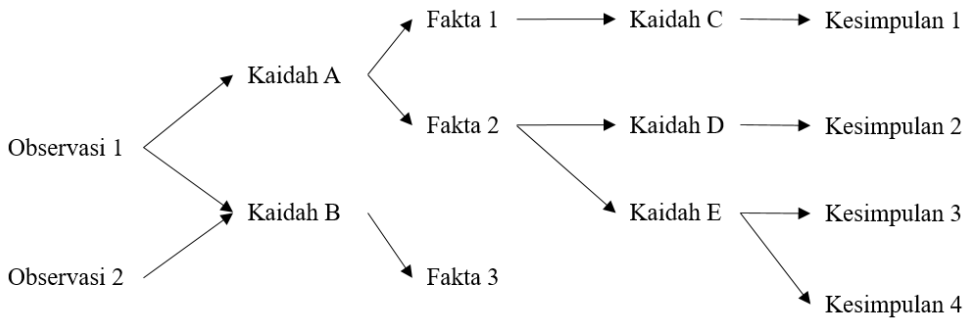
3. Sistem Pakar (*Expert System*)

Sistem pakar suatu cabang dari *Artificial Intelligent* (AI) yang cukup tua karena sistem ini mulai dikembangkan pada tahun 1960. Sistem pakar adalah program AI dengan basis pengetahuan (*Knowledge Base*) yang diperoleh dari pengalaman atau pengetahuan pakar atau ahli dalam memecahkan persoalan pada bidang tertentu dan didukung mesin Interensi yang melakukan penalaran atau pelacakan terhadap sesuatu atau fakta-fakta dan aturan kaidah yang ada di basis pengetahuan setelah dilakukan pencarian, sehingga dicapai kesimpulan (Hayadi, 2016: 1).

Beberapa metode yang digunakan dalam Sistem Pakar adalah (Sutojo, dkk., 2011: 171-178):

1. *Forward Chaining*

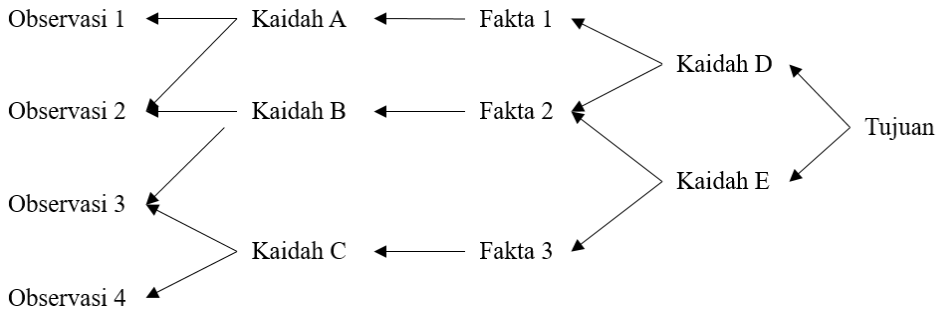
   Forward Chaining adalah teknik pencarian yang dimulai dengan fakta yang diketahui, kemudian mencocokkan fakta-fakta tersebut dengan bagian *IF* dari *rules IF-THEN*.



Gambar 2.1 Diagram *Forward Chaining*

2. *Backward Chaining*

   Backward Chaining adalah metode inferensi yang bekerja mundur ke arah kondisi awal. Proses diawali dari *Goal* (yang berada dibagian *THEN* dari *rule IF-THEN*), kemudian pencarian mulai dijalankan untuk mencocokkan apakah fakta-fakta yang ada cocok dengan premis-premis di bagian *IF*.



Gambar 2.2 Diagram *Backward Chaining*

2.1.2. Sistem Pakar

Sistem pakar atau *Expert System* biasa disebut juga dengan *Knowledge Based System* yaitu suatu aplikasi komputer yang ditujukan untuk membantu pengambilan keputusan dalam bidang yang spesifik. Sistem ini bekerja dengan menggunakan pengetahuan dan metode analisis yang telah didefinisikan terlebih dahulu oleh pakar yang sesuai dengan bidang keahliannya. Sistem ini disebut sistem pakar karena fungsi dan perannya sama seperti seorang ahli yang harus memiliki pengetahuan, pengalaman dalam memecahkan suatu persoalan. Sistem biasanya berfungsi sebagai kunci penting yang akan membantu suatu sistem pendukung keputusan atau sistem pendukung eksekutif (Hayadi, 2016: 1).

Menurut Sutojo, dkk. (2011: 160) Sistem pakar menjadi sangat popular karena sangat banyak kemampuan dan manfaat yang diberikan, di antaranya:

1. Meningkatkan produktivitas, karena sistem pakar dapat bekerja lebih cepat daripada manusia.
2. Membuat seorang yang awam bekerja seperti layaknya seorang pakar.
3. Meningkatkan kualitas, dengan memberi nasehat yang konsisten dan mengurangi kesalahan.
4. Mampu menangkap pengetahuan dan kepakaran seseorang.
5. Dapat beroperasi di lingkungan yang berbahaya.
6. Memudahkan akses pengetahuan seorang pakar.
7. Andal. Sistem pakar tidak pernah menjadi bosan dan kelelahan atau sakit.

8. Meningkatkan kapabilitas sistem komputer. Integrasi sistem pakar dengan sistem komputer lain membuat sistem lebih efektif dan mencakup lebih banyak aplikasi.
9. Mampu bekerja dengan informasi yang tidak lengkap atau tidak pasti. Berbeda dengan sistem komputer konvensional. Sistem pakar dapat bekerja dengan informasi yang tidak lengkap. Pengguna dapat merespons dengan: “tidak tahu” atau “tidak yakin” pada satu atau lebih pertanyaan selama konsultasi dan sistem pakar tetap akan memberikan jawabannya.
10. Bisa digunakan sebagai media pelengkap dan pelatihan. Pengguna pemula yang bekerja dengan sistem pakar akan menjadi lebih berpengalaman karena adanya fasilitas penjelas yang berfungsi sebagai guru.
11. Meningkatkan kemampuan untuk menyelesaikan masalah karena sistem pakar mengambil sumber pengetahuan dari banyak pakar.

Menurut Sutojo, dkk. (2011: 161) Selain Manfaat, ada juga beberapa kekurangan yang ada pada sistem pakar, di antaranya:

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.

2.1.2.1. Struktur Sistem Pakar

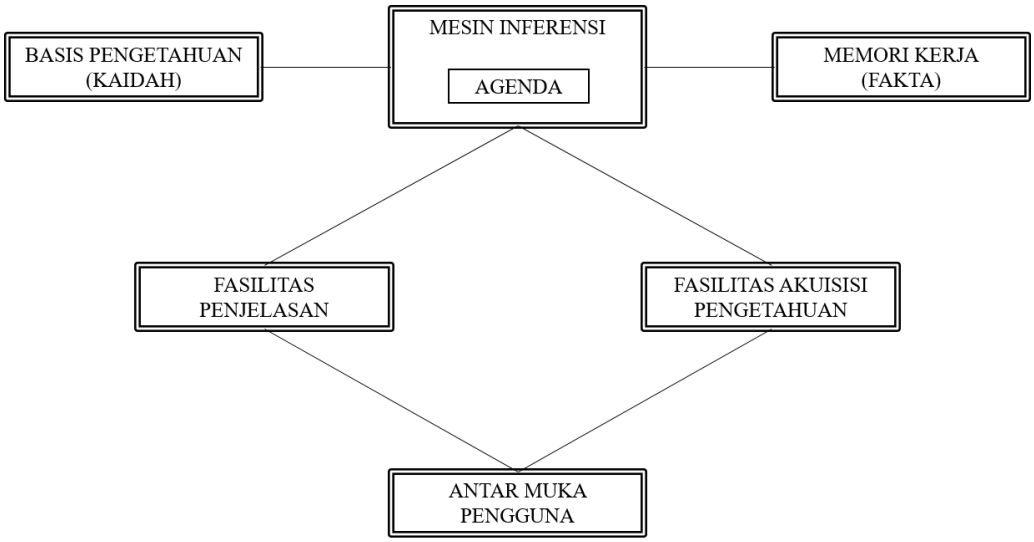
Sistem pakar sebagai sebuah program uang difungsikan untuk menirukan pakar manusia harus bisa melakukan hal-hal yang dapat dikerjakan oleh seorang pakar. Untuk membangun sistem yang seperti itu maka komponen-komponen yang harus dimiliki adalah sebagai berikut (Hartati dan Iswanti, 2008: 3):

1. Antar Muka Pengguna (*User Interface*)
2. Basis Pengetahuan (*Knowledge Base*)
3. Mesin Inferensi (*Inference Machine*)
4. Memori Kerja (*Working Memory*)

Sedangkan untuk menjadikan sistem pakar menjadi lebih menyerupai seorang pakar yang berinteraksi dengan pemakai, maka dilengkapi dengan fasilitas berikut:

1. Fasilitas Penjelasan (*Explanation Facility*)
2. Fasilitas Akuisisi Pengetahuan (*Knowledge Acquisition Facility*)

Hal ini terlihat dalam struktur sistem pakar pada gambar 2.3.



Gambar 2.3 Struktur Sistem Pakar

1. Antar Muka Pengguna (*User Interface*)

   Sistem pakar menggantikan seorang pakar dalam suatu situasi tertentu, maka sistem harus menyediakan pendukung yang diperlukan oleh pemakai yang tidak memahami masalah teknis. Sistem pakar juga menyediakan komunikasi antara sistem dan pemakainya, yang disebut sebagai antar muka. Antar muka yang efektif dan ramah pengguna (*user friendly*) penting sekali terutama bagi pemakai yang tidak ahli dalam bidang yang diterapkan pada sistem pakar.

2. Basis Pengetahuan (*Knowledge Base*)

   Basis pengetahuan merupakan kumpulan pengetahuan bidang tertentu pada tingkatan pakar dalam format tertentu. Pengetahuan ini diperoleh dari akumulasi pengetahuan pakar dan sumber-sumber pengetahuan lainnya seperti yang telah disebutkan sebelumnya. Basis pengetahuan bersifat dinamis, bisa berkembang dari waktu ke waktu. Perkembangan ini disebabkan karena pengetahuan selalu bertambah. Pada sistem pakar basis pengetahuan terpisah dari mesin inferensi. Pemisahan ini bermanfaat untuk pengembangan sistem pakar secara leluasa disesuaikan dengan perkembangan pengetahuan pada suatu domain. Penambahan dan pengurangan dapat dilakukan pada basis pengetahuan ini tanpa mengganggu mesin inferensi.

3. Mesin Inferensi (*Inference Machine*)

   Mesin inferensi merupakan otak dari sistem pakar, berupa perangkat lunak yang melakukan tugas inferensi penalaran sistem pakar, biasa dikatakan

sebagai mesin pemikir (*Thinking machine*). Pada prinsipnya mesin inferensi inilah yang akan mencari solusi dari suatu permasalahan. Konsep yang biasanya digunakan untuk mesin inferensi adalah runut balik (*top-down*), yaitu proses penalaran yang berawal dari tujuan yang kita inginkan, menelusuri fakta-fakta yang mendukung untuk mencapai tujuan. Selain itu dapat juga menggunakan runut maju (*bottom-up*), yaitu proses penalaran tang bermula dari kondisi yang diketahui menuju tujuan yang diinginkan.

4. Memori Kerja (*Working Memory*)

Memori kerja merupakan bagian dari sistem pakar yang menyimpan fakta-fakta yang diperoleh saat dilakukan proses konsultasi. Fakta-fakta inilah yang nantinya akan diolah oleh mesin inferensi berdasarkan pengetahuan yang disimpan dalam basis pengetahuan untuk menentukan suatu keputusan pemecahan masalah.

5. Fasilitas penjelasan (*Explanation Facility*)

Proses menentukan keputusan yang dilakukan oleh mesin inferensi selama sesi konsultasi mencerminkan proses penalaran seorang pakar. Karena pemakai kadangkala bukanlah ahli dalam bidang tersebut, maka dibuatlah fasilitas penjelasan. Tujuan adanya fasilitas penjelasan dalam sistem pakar antara lain membuat sistem menjadi lebih cerdas, menunjukkan adanya proses analisa dan yang tidak kalah pentingnya adalah memuaskan psikologis pemakai.

6. Fasilitas Akuisisi Pengetahuan (*Knowledge acquisition facility*)

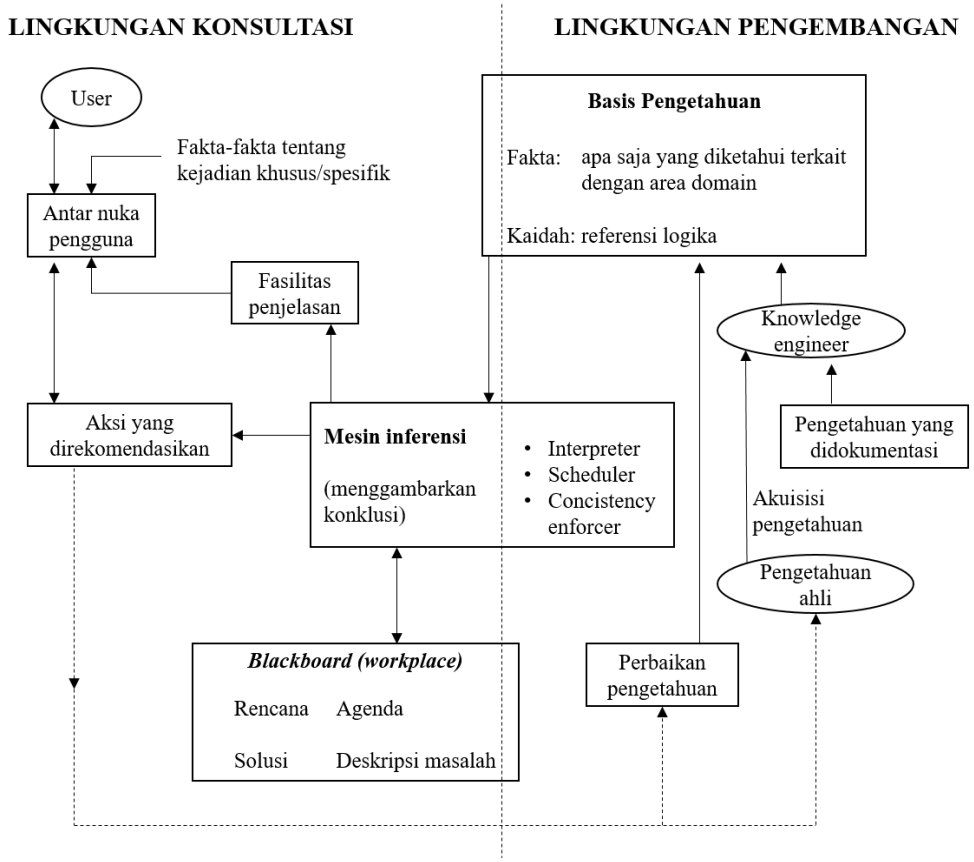
Pengetahuan pada sistem pakar dapat ditambahkan kapan saja pengetahuan baru diperoleh atau saat pengetahuan yang sudah ada dan sudah tidak berlaku lagi. Dengan adanya fasilitas ini pada sistem, maka seorang pakar akan mudah menambahkan pengetahuan ataupun kaidah baru pada sistem pakar. Untuk menjamin bahwa pengetahuan pada sistem pakar ini *up to date* dan valid, maka fasilitas akuisisi pengetahuan hanya bisa diakses oleh pakar.

Sistem pakar juga dapat dilihat dari sudut pandang lingkungan (*enviroment*) dalam sistem. Terdapat dua lingkungan yaitu lingkungan konsultasi dan lingkungan pengembangan. Lingkungan konsultasi diperuntukkan bagi pengguna non pakar untuk melakukan konsultasi dengan sistem yang tujuannya adalah mendapatkan nasehat pakar. Sedangkan, lingkungan pengembang ditujukan bagi pembangun sistem pakar untuk membangun komponen dan memasukkan pengetahuan hasil akuisisi pengetahuan ke dalam basis pengetahuan (Hartati dan Iswanti, 2008: 8).

Menurut Hartati dan Iswanti (2008: 10) hasil pemrosesan yang dilakukan oleh mesin inferensi dari sudut pandang pengguna non pakar berupa aksi/konklusi yang direkomendasikan oleh sistem pakar atau dapat juga berupa penjelasan jika memang dibutuhkan pengguna. Dari susut pandang pembangun sistem dalam lingkungan pengembangan, mesin inferensi terdiri dari 3 elemen penting yaitu:

1. Interpreter (interpreter kaidah terdapat pada sebagian besar sistem), elemen ini mengeksekusi item-item agenda yang terpilih dengan menggunakan kaidah basis pengetahuan yang bersesuain.
2. Penjadwalan, elemen ini mengelola pengontrolan terhadap agenda. Penjadwalan memperkirakan pengaruh-pengaruh dari penggunaan kaidah inferensi pada prioritas-prioritas item atau kriteria lain pada agenda.
3. Pelaksana konsistensi, elemen ini berusaha untuk mengelola penyajian solusi secara konsisten.

Blackboard adalah memori kerja yang digunakan untuk menyimpan kondisi/keadaan yang dialami oleh pengguna dan juga hipotesa serta keputusan sementara.



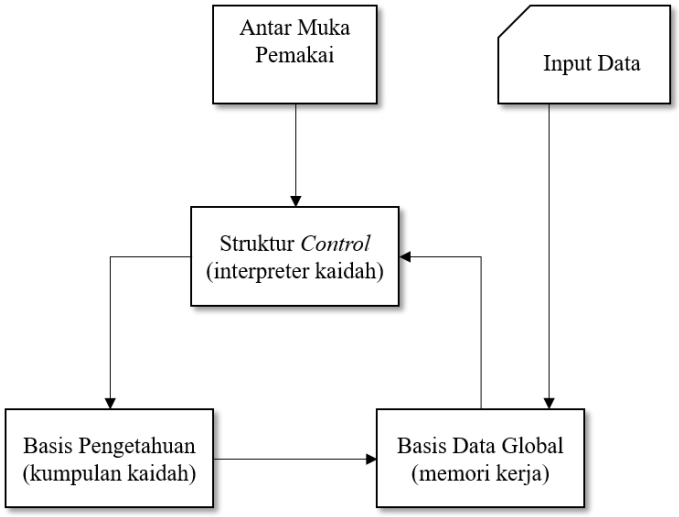
Gambar 2.4 Struktur Sistem Pakar

Pengetahuan yang dimasukkan ke dalam sistem pakar disajikan dalam bentuk yang dapat dimengerti dan diterima oleh sistem pakar, salah satunya adalah kaidah produksi (Hartati dan Iswanti, 2008: 10).

Struktur sistem pakar berbasis kaidah produksi terdiri dari empat komponen, yaitu:

1. Antar muka pemakai, antar muka penghubung antara pemakai dengan sistem pakar

2. Basis pengetahuan, berisi sekumpulan kaidah yang berasal dari pengetahuan dalam domain tertentu. Kaidah ini secara umum disajikan dalam bentuk kaidah produksi (*IF....THEN....*)
3. Struktur kontrol, merupakan interpreter kaidah atau mesin inferensi yang menggunakan pengetahuan-pengetahuan yang tersimpan dalam basis pengetahuan untuk menyelesaikan permasalahan yang ada.
4. *Working memory* atau basis data global, mencatat status problem saat ini dan histori solusi.



Gambar 2.5 Struktur Sistem Pakar Berbasis Kaidah Produksi

Kebanyakan *software* sistem pakar komersial adalah sistem yang berbasis *rule* (*rule-based systems*), yaitu pengetahuan disimpan terutama dalam bentuk *rule*, sebagai prosedur pemecahan masalah (Sutojo, dkk., 2011: 165).

Syntax Rule adalah:

IF E THEN H

E : *evidence* (fakta-fakta) yang ada

H : Hipotesis atau kesimpulan yang dihasilkan

Gambar 2.6 *Syntax Rule*

Secara umum, *rule* mempunyai *evidence* lebih dari satu yang dihubungkan oleh kata penghubung AND atau OR, atau kombinasi keduanya. Tetapi sebaiknya biasakan menghindari penggunaan AND dan OR secara sekaligus dalam satu *rule* (Sutojo, dkk., 2011).

IF (E_1 AND E_2 AND E_3 ……AND E_n) THEN H

IF (E_1 OR E_2 OR E_3 …… OR E_n) THEN H

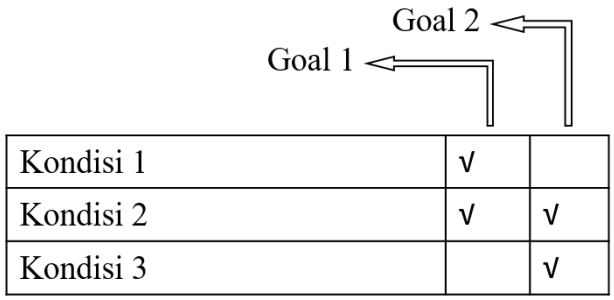
Satu *evidence* bisa juga mempunyai hipotesis lebih dari satu.

IF E THEN (H_1 AND H_2 AND H_3 ……AND H_n)

Gambar 2.7 Contoh Penggunaan *AND* Dan *OR*

2.1.2.2. Tabel Keputusan dan Pohon Keputusan

Menurut Hartati dan Iswanti (2008:26) tabel keputusan merupakan suatu cara untuk mendokumentasikan pengetahuan. Tabel keputusan merupakan matrik kondisi yang dipertimbangkan dalam pendeskripsian kaidah. Gambar 2.8 Merupakan suatu bentuk tabel keputusan.



Gambar 2.8 Tabel Keputusan

Menurut Hartati dan Iswanti (2008:26-27) kaidah yang disajikan dalam bentuk kaidah produksi disusun dari tabel keputusan (dibentuk dari pengubahan tabel keputusan). Pembuatan suatu kaidah dilakukan dengan beberapa tahapan. Sebagai contoh perhatikan pembuatan kaidah 1. Pertama, kita lihat **Goal 1** merupakan konklusi dari kaidah 1. Konklusi ini akan dapat dicapai bila kondisi-kondisi yang mendukungnya terpenuhi. Kedua, tanda centang pada kolom di bawah **Goal 1** menunjukkan kondisi mana yang harus dipenuhi untuk mencapai konklusi tersebut. Pada **Goal 1**, terlihat tanda centang berada pada **Kondisi 1** dan **Kondisi 2.** Ketiga, Pembuatan kaidah 1 menggunakan goal dan kondisi yang telah diperoleh dari langkah 1 dan 2, seperti berikut ini:

```
Kaidah 1:    Goal 1    IF
             Kondisi   1   AND
             Kondisi   2
```

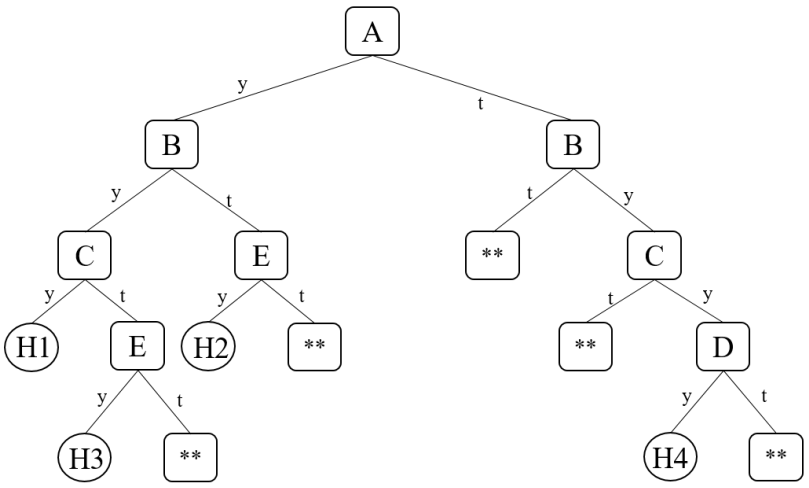
Gambar 2.9 Kaidah 1

Kaidah 2 dapat diperoleh dengan cara yang sama:

```
Kaidah 2:    Goal 2    IF
             Kondisi   2   AND
             Kondisi   3
```

Gambar 2.10 Kaidah 2

Meskipun kaidah secara langsung dapat langsung dapat dihasilkan sari tabel keputusan tetapi untuk menghasilkan kaidah yang efisien terdapat suatu langkah yang harus ditempuh yaitu membuat pohon keputusan terlebih dahulu. Dari pohon keputusan dapat diketahui atribut (kondisi) yang dapat direduksi sehingga menghasilkan kaidah yang efisien dan optimal (Hartati dan Iswanti, 2008:27).



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
D = evidence D,	H4 = hipotesa 4	tertentu

Gambar 2.11 Pohon Keputusan

Dengan melihat pohon keputusan yang dihasilkan, dapat diketahui hipotesa H1 terpenuhi jika memenuhi *evidence* A, B, dan C. Hipotesa H2 terpenuhi jika memiliki *evidence* A dan *evidence* E. Hipotesa H3 akan terpehuni jika memiliki

evidence A, B, dan E. Demikian juga untuk 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. Dapat dicontohkan untuk menghasilkan hipotesa H2 kalau diurutkan jalannya dari node A adalah memiliki *evidence* A, tidak memiliki *evidence* B, dan memiliki *evidence* E. Dalam hal itu jika tidak memenuhi *evidence* tertentu, maka diabaikan; sehingga hasil yang didapatkan adalah Hipotesa H2 terpenuhi jika memiliki *evidence* A dan E saja (Hartati dan Iswanti, 2008:33-34).

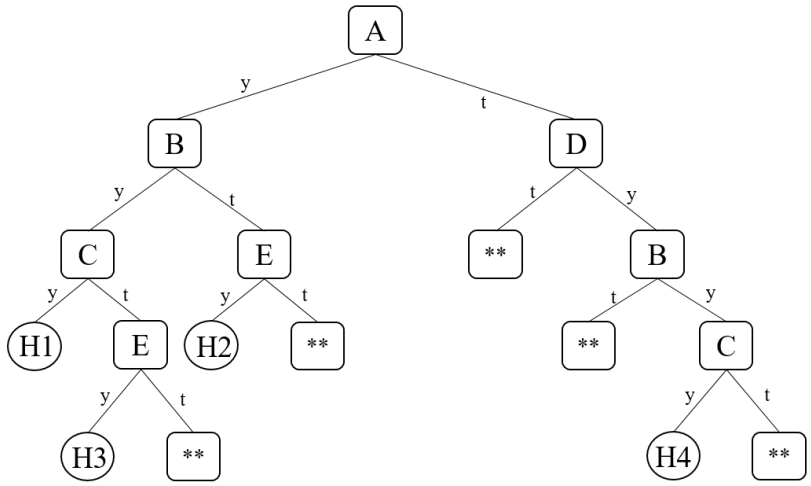
Dalam implementasi sistem pakar terutama dalam sesi konsultasi, node-node yang mewakili *evidence* biasanya akan menjadi pertanyaan yang diajukan oleh sistem. Mengacu pohon keputusan pada gambar 2.5 Permasalahan bisa muncul pada awal sesi konsultasi yaitu pada saat sistem pakar menanyakan "apakah memiliki *evidence* A?" Terlihat dari pohon keputusan apakah jawaban pengguna baik "ya" atau "tidak" maka sistem akan menanyakan *evidence* B dengan kata lain apapun jawaban pengguna maka tidak akan mempengaruhi sistem. Hal ini dapat diatasi dengan mengubah urutan pada tabel keputusan. Altenatif tabel keputusan yang lain seperti terlihat pada tabel 2.1 (Hartati dan Iswanti, 2008:34).

Tabel 2.1 Tabel Keputusan

Hipotesa	**Hipotesa 1**	**Hipotesa 2**	**Hipotesa 3**	**Hipotesa 4**
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: Hartati dan Iswanti (2008:32)

Mengacu tabel keputusan pada tabel 2.1 Dapat dihasilkan pohon keputusan sebagai berikut:



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
D = evidence D,	H4 = hipotesa 4	tertentu

Gambar 2.12 Pohon Keputusan

Terlihat dari pohon keputusan gambar 2.12, masing-masing node yang mewakili *evidence* tertentu untuk kondisi "y" dan "t" sudah tidak mengarah pada *evidence* yang sama. Dalam sesi konsultasi hal ini mengandung arti jawaban pengguna yang berbeda, akan mengarah pada pertanyaan yang berbeda pula (Hartati dan Iswanti, 2008:35).

Menurut Hartati dan Iswanti (2008:35-36) proses akuisisi pengetahuan dalam sistem pakar bukanlah hal yang mudah dan sederhana. Pengetahuan yang berhasil didapatkan dan disajikan dalam salah satu bentuk representasi pengetahuan misalnya disajikan dalam salah satu bentuk tabel keputusan dan diubah susunannya seperti contoh kasus di atas, sacara teori sudah benar tetapi pada domain

pengetahuan tertentu; pengubahan yang seperti itu terkadang tidak diperkenankan. Dengan kata lain, pengetahuan yang akan disajikan dalam bentuk representasi tertentu harus mengikuti kaida-kaidah kepakaran dalam domainnya dan tidak boleh bertentangan.

2.1.2.3. Basis Pengetahuan (Knowledge base)

Menurut Kusumadewi (2003: 115-116) basis pengetahuan berisi pengetahuan-pengetahuan dalam penyelesaian masalah, tentu saja di dalam domain tertentu. Ada 2 bentuk pendekatan basis pengetahuan yang sangat umum digunakan, yaitu:

1. Penalaran berbasis aturan (*Rule-Based Reasoning*)

   Pada penalaran berbasis aturan, pengetahuan direpresentasikan dengan menggunakan aturan berbentuk: *IF-THEN*. Bentuk ini digunakan apabila kita memiliki sejumlah pengetahuan pakar pada suatu permasalahan tertentu, dan si pakar dapat menyelesaikan masalah tersebut secara berurutan. Disamping itu, bentuk ini juga digunakan apabila dibutuhkan penjelasan tentang jejak (langkah-langkah) pencapaian solusi.

2. Penalaran berbasis kasus (*Case-Based Reasoning*)

   Pada penalaran berbasis kasus, basis pengetahuan akan berisi solusi-solusi yang telah dicapai sebelumnya, kemudian akan diturunkan suatu solusi untuk keadaan yang terjadi sekarang. Bentuk ini digunakan apabila pengguna menginginkan untuk mengetahui lebih banyak lagi pada kasus-kasus yang

hampir sama. Selain itu, bentuk ini juga digunakan apabila kita telah memiliki sejumlah situasi atau kasus tertentu dalam basis pengetahuan.

2.1.2.4. Motor Inferensi (*Inference Engine*)

Menurut Kusumadewi (2003: 116) ada 2 cara yang dapat dikerjakan dalam melakukan inferensi, yaitu:

1. *Forward Chaning.* Pencocokan fakta atau pernyataan dimulai dari bagian kiri (*IF* dulu). Dengan kata lain, penalaran dimulai dari fakta terlebih dahulu untuk menguji kebenaran hipotesis
2. *Backward Chaining.* Pencocokan fakta atau pernyataan dimulai dari bagian sebelah kanan (*THEN* dulu). Dengan kata lain, penalaran dimulai dari hipotesis terlebih dahulu, dan untuk menguji kebenaran hipotesis tersebut dicari harus dicari fakta-fakta yang ada dalam basis pengetahuan.

2.1.2.5. Metode *Forward Chaining*

Metode *Forward Chaining* adalah metode pencarian atau teknik pelacakan ke depan yang dimulai dengan informasi yang ada dan penggabungan *rule* untuk menghasilkan suatu kesimpulan atau tujuan (Hayadi, 2016: 9).

Forward Chaining adalah teknik pencarian yang dimulai dengan fakta yang diketahui, kemudian mencocokkan fakta-fakta tersebut dengan bagian *IF* dari *rules IF-THEN*. Bila ada fakta yang cocok dengan bagian *IF*, maka *rule* tersebut dieksekusi. Bila sebuah *rule* dieksekusi, maka sebuah fakta baru (bagian *THEN*)

ditambahkan ke dalam *database*. Setiap kali pencocokan dimulai dari *rule* teratas. Setiap *rule* hanya boleh dieksekusi sekali saja. Pencocokan berhenti bila tidak ada lagi *rule* yang bisa dieksekusi. Metode pencarian yang digunakan adalah *Depth-First Search* (DFS), *Breadth-First Search* (BFS) atau *Best First Search*. (Sutojo, dkk., 2011: 171)

Tabel 2.2 Contoh Aturan-Aturan

No.	Aturan
R-1	IF A & B THEN C
R-2	IF C THEN D
R-3	IF A & E THEN F
R-4	IF A THEN G
R-5	IF F & G THEN D
R-6	IF G & E THEN H
R-7	IF C & H THEN I
R-8	IF I & A THEN J
R-9	IF G THEN J
R-10	IF J THEN K

Sumber: Kusumadewi (2003, 116)

Pada tabel 2.2 terlihat ada 10 aturan yang tersimpan dalam basis pengetahuan. Fakta awal yang diberikan hanya: A & F (artinya: A dan F bernilai benar). Ingin buktikan apakah K bernilai benar (hipotesis: K)?

Langkah-langkah inferensi adalah sebagai berikut:

1. Dimulai dari R-1. A merupakan fakta sehingga bernilai benar, sehingga B belum bisa diketahui kebenarannya, sehingga C pun juga belum bisa diketahui kebenarannya. Oleh karena itu kita tidak mendapatkan informasi apapun pada R-1 ini. Sehingga kita menuju R-2.

2. Pada R-2, kita tidak mengetahui informasi apapun tentang C, sehingga kita juga tidak bisa memastikan kebenaran D. oleh karena itu kita tidak mendapatkan informasi apapun pada R-2 ini. Sehingga kita menuju R-3.
3. Pada R-3, baik A maupun E adalah fakta sehingga jelas benar. Dengan demikian F sebagai konsekuen juga ikut benar. Sehingga sekarang kita mempunyai fakta baru yaitu F. karena F bukan hipotesis yang hendak kita buktikan, maka penelusuran kita lanjutkan ke R-4.
4. Pada R-4, A adalah fakta sehingga benar. Dengan demikian G sebagai konsekuen juga ikut benar. Sehingga sekarang kita mempunyai fakta baru yaitu G. karena G bukan hipotesis yang hendak dibuktikan, maka penelusuran kita lanjutkan ke R-5.
5. Pada R-5, baik F maupun G bernilai benar berdasarkan aturan R-3 dan R-4. Dengan demikian D sebagai konsekuen juga ikut benar. Sehingga sekarang kita mempunyai fakta baru yaitu D. Karena D bukan hipotesis yang hendak kita buktikan, maka penelusuran kita lanjutkan ke R-6.
6. Pada R-6, baik A maupun G adalah benar berdasarkan fakta dari R-4. Dengan demikian H sebagai konsekuen juga ikut benar. Sehingga sekarang kita mempunyai fakta baru yaitu H. karena H bukan hipotesis yang hendak kita buktikan, maka penelusuran kita lanjutkan ke R-7.
7. Pada R-7, meskipun H benar berdasarkan R-6, namun kita tidak tahu kebenaran C, sehingga I pun juga belum bisa diketahui kebenarannya. Oleh karena itu kita tidak mendapatkan informasi apapun pada R-7 ini. Sehingga kita menuju ke R-8.

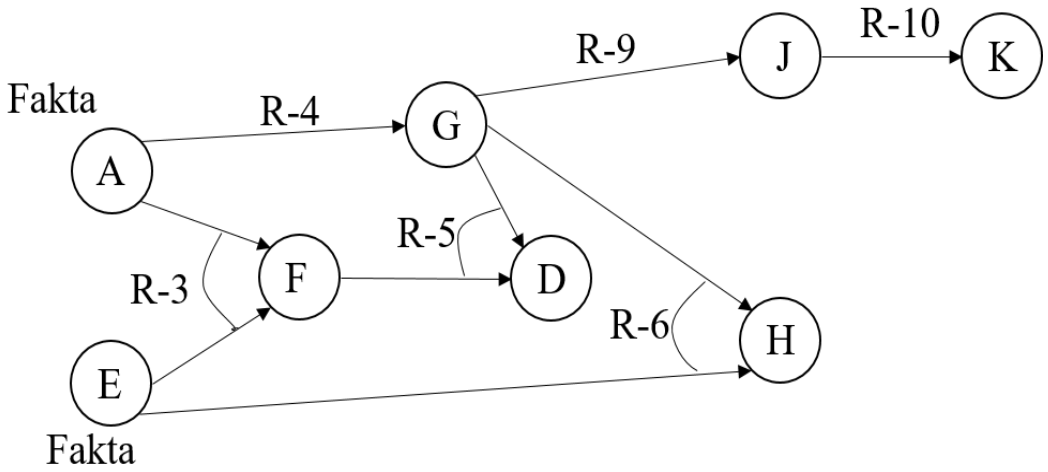
8. Pada R-8, meskipun A benar karena fakta, namun kita tidak tahu kebenaran I, sehingga J pun juga belum bisa diketahui kebenarannya. Oleh karena itu kita tidak mendapatkan informasi apapun pada R-8 ini. Sehingga kita menuju ke R-9.
9. Pada R-9, J bernilai benar karena G benar berdasarkan R-4. Karena J bukan hipotesis yang hendak kita buktikan, maka penelusuran kita lanjutkan ke R-10.
10. Pada R-10, K bernilai benar karena J benar berdasarkan R-9. Karena H sudah merupakan hipotesis yang hendak kita buktikan, maka terbukti bahwa K adalah benar.

Tabel munculnya fakta baru pada saat inferensi terlihat pada tabel 2.3. Sedangkan alur inferensi terlihat pada gambar 2.13.

Tabel 2.3 Fakta Baru

Aturan	Fakta Baru
R-3	F
R-4	G
R-5	D
R-6	H
R-9	J
R-10	K

Sumber: Kusumadewi (2003: 118)



Gambar 2.13 Alur Inferensi *Forward Chaining*

2.2. Variabel

Variabel penelitian pada dasarnya adalah segala sesuatu yang berbentuk apa saja yang ditetapkan oleh peneliti untuk dipelajari sehingga diperoleh informasi tentang hal tersebut, kemudian ditarik kesimpulannya (Sudaryono, 2015).

2.2.1. *Emotional Quotient* (EQ)

Daniel Goleman menyatakan bahwa kontribusi IQ (*Intelligence Quotient*) bagi keberhasilan seseorang hanya sekitar 20% dan sisanya yang 80% ditentukan oleh serumpun faktor-faktor yang disebut kecerdasan emosional. Karena itu, ada yang berpendapat bahwa IQ mengangkat fungsi pikiran dan EQ mengangkat fungsi perasaan. Orang yang ber-EQ tinggi akan berupaya menciptakan keseimbangan

dalam dirinya sendiri dan bisa mengubah sesuatu yang buruk menjadi sesuatu yang positif dan bermanfaat (Sunar, 2010:50-51).

Kecerdasan emosional atau yang biasa dikenal dengan EQ (*Emotional Quotient)* adalah kemampuan untuk memahami, mengendalikan dan mengevaluasi emosi. Beberapa peneliti menyarankan bahwa kecerdasan emosional dapat dipelajari dan diperkuat melalui pembelajaran dan lingkungan, sedangkan klaim lain adalah karakteristik bawaan (*genetics*) (Goleman, 2015).

Menurut Howard Gardner, terdapat lima pokok utama dari kecerdasan emosional seseorang yakni mampu menyadari dan mengelola emosi diri sendiri, memiliki kepekaan terhadap emosi orang lain, mampu merespon dan bernegosiasi dengan orang lain secara emosional, serta dapat menggunakan emosi sebagai alat untuk memotivasi diri (Sunar, 2010:129).

EQ mengukur tingkat keterampilan emosional dalam memahami emosi untuk mengendalikan reaksi emosional, untuk memotivasi diri sendiri, untuk memahami keadaan sosial dan untuk berkomunikasi secara baik dengan orang lain (Sunar, 2010:134).

2.2.2. Komponen Pada Kecerdasan Emosional

Menurut Baskara, dkk. (2008:103-104) Goleman mengadaptasi model kecerdasan emosi dari Salovey dan Meyer ke dalam sebuah versi yang menurutnya paling bermanfaat untuk memahami cara kerja kecerdasan emosi dalam kehidupan

sehari-hari ataupun kehidupan kerja. Goleman mengadaptasi lima komponen dasar kecakapan emosi dan kecakapan sosial sebagai berikut:

1. Kesadaran Diri

Kesadaran diri adalah kemampuan untuk mengenali apa yang individu rasakan pada suatu saat, dan menggunakannya untuk memandu pengambilan keputusan diri sendiri; memiliki tolak ukur yang realistis atas kemampuan diri dan kepercayaan diri yang kuat. Kesadaran diri dapat diuraikan menjadi tiga kemampuan, yaitu kesadaran emosi, penilaian diri secara teliti, dan percaya diri. Sadar emosi berarti individu dapat mengenali emosi diri sendiri dan efeknya. Kemampuan menilai diri secara teliti menunjukkan seberapa luas pengetahuan individu tentang kekuatan dan batas-batas diri sendiri. Kepercayaan diri menunjukkan seberapa besar keyakinan individu tentang harga diri dan kemampuan diri sendiri.

2. Pengaturan Diri

Pengaturan diri adalah kemampuan menangani emosi sedemikian rupa sehingga berdampak positif kepada pelaksanaan tugas; peka terhadap kata hati dan sanggup menunda kenikmatan sebelum tercapainya suatu sasaran; mampu pulih kembali dari tekanan emosi. Kemampuan pengaturan diri dapat diuraikan menjadi: (1) kendali diri, yaitu kemampuan mengelola emosi-emosi dan desakan-desakan hati yang bersifat merusak, (2) sifat dapat dipercaya, yaitu kemampuan memelihara norma kejujuran dan integritas, (3) kewaspadaan, yaitu sikap bertanggung jawab atas kinerja pribadi, (4) adaptibilitas, yaitu keluwesan dalam menghadapi

perubahan, dan (5) inovasi, yaitu kemampuan mudah menerima dan terbuka terhadap gagasan, pendekatan, dan informasi-informasi baru.

3. Motivasi

Motivasi adalah kemampuan menggunakan hasrat yang paling dalam untuk menggerakkan dan menuntun menuju sasaran, membantu mengambil inisiatif dan bertindak sangat efektif, serta untuk bertahan menghadapi kegagalan dan frustasi. Motivasi dapat diuraikan menjadi: (1) dorongan prestasi, yaitu dorongan untuk menjadi lebih baik atau memenuhi standar keberhasilan, (2) komitmen, yaitu kemampuan menyesuaikan diri dengan tujuan kelompok, (3) inisiatif, yaitu kesiapan untuk memanfaatkan kesempatan, (4) optimisme, yaitu kegigihan dalam memperjuangkan sasaran kendati ada halangan dan kegagalan.

4. Empati

Empati adalah kemampuan merasakan apa yang dirasakan oleh orang lain, mampu memahami perspektif mereka, menumbuhkan hubungan saling percaya dan menyelaraskan diri dengan bermacam-macam orang. Empati dapat diuraikan menjadi: (1) memahami orang lain, yaitu kemampuan mengindra perasaan dan perspektif orang lain, serta menunjukkan minat aktif terhadap kepentingan mereka, (2) orientasi pelayanan, yaitu mengantisipasi, mengenali, dan berusaha memenuhi kebutuhan pelanggan, (3) mengembangkan orang lain, yaitu merasakan kebutuhan perkembangan orang lain dan berusaha menumbuhkan kemampuan mereka, (4) menerima keragaman, yaitu menumbuhkan peluang melalui pergaulan dengan berbagai macam orang, (5) kesadaran politik, yaitu mampu membaca arus-arus emosi sebuah kelompok dan hubungannya dengan kekuasaan.

5. Keterampilan Sosial

Keterampilan sosial adalah kemampuan menangani emosi dengan baik ketika berhubungan dengan orang lain dan dengan cermat membaca situasi serta jaringan sosial; berinteraksi dengan lancar; menggunakan berbagai keterampilan ini untuk mempengaruhi dan memimpin, bermusyawarah dan menyelesaikan perselisihan, serta untuk bekerja sama dalam tim. Keterampilan sosial dapat diuraikan menjadi : (1) pengaruh, yaitu memiliki berbagai taktik dan strategi untuk melakukan persuasi, (2) komunikasi, yaitu mengirimkan pesan yang jelas dan meyakinkan, (3) kepemimpinan, yaitu kemampuan membangkitkan inspirasi dan memandu kelompok serta orang lain, (4) katalisator perubahan, yaitu kemampuan memulai dan mengelola perubahan, (5) manajemen konflik, yaitu negosiasi dan pemecahan silang pendapat, (6) pengikat jaringan, yaitu kemampuan menumbuhkan hubungan sebagai alat, (7) kolaborasi dan kooperasi, yaitu kerja sama dengan orang lain demi tujuan bersama, (8) kemampuan tim, yaitu menciptakan sinergi kelompok dalam memperjuangkan tujuan bersama.

2.3. *Software* Pendukung

Software pendukung merupakan perangkat lunak komputer yang digunakan untuk merancang suatu sistem. Adapun *software* atau perangkat lunak yang di gunakan dalam peneletian ini antara lain: *StarUML*, *PHP*, *HTML*, *CSS*, JavaScript, *Notepad++*, *Microsoft SQL server 2008* dan *XAMPP*.

2.3.1. *Unified Modeling Language* (UML)

UML (*Unified Modeling Language*) adlah salah standar Bahasa yang banyak digunakan di dunia industry untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek (Rosa dan Shalahuddin, 2016:133).

Pada perkembangan teknologi perangkat lunak, diperlukan adanya Bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. Seperti yang kita ketahui bahwa menyatukan banyak kepala untuk menceritakan sebuah ide dengan tujuan untuk memahami hal yang sama tidaklah mudah, oleh Karena itu diperlukan sebuah Bahasa pemodelan perangkat lunak yang dapat dimengerti oleh banyak orang (Rosa dan Shalahuddin, 2016:137).

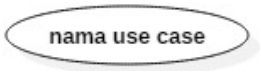
2.3.1.1. *Use Case Diagram*

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behaviour*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu (Rosa dan Shalahuddin, 2016:155).

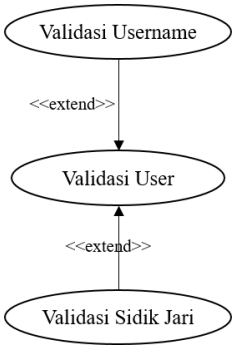
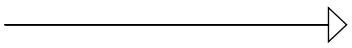
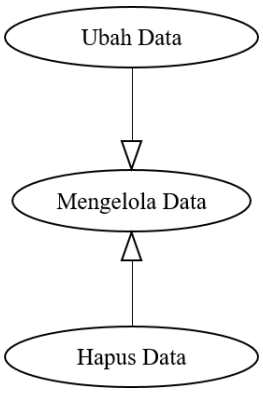
Menurut Rosa dan Shalahuddin (2016:155) Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utam pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat sendiri, jadi walaupun simbol dari actor adalah gambar orang, tapi actor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor. Berikut adalah simbol-simbol yang ada pada diagram *use case*:

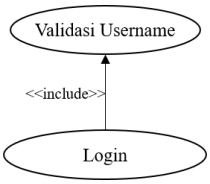
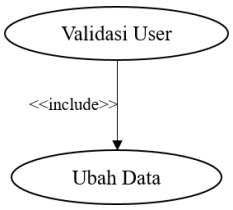
Tabel 2.4 Simbol *Use Case Diagram*

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 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, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Asosiasi / *Association* ———	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor
Ekstensi / *Extend*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tnapa *use case* tambahan itu; mirip

Tabel 2.4 Lanjutan

<<extend>> 	dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, misal  Arah panah mengarah pada *use case* yang ditambahkan; biasanya *use case* yang menjadi *extend*-nya merupakan jenis yang sama dengan *use case* yang menjadi induknya
Generalisasi / *Generalization* 	Hubungan generalisasi dan spesialisasi (umum - khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya:  Arah panah mengarah pada *use case* yang menjadi generalisasinya (umum)
Menggunakan / *Include* / *Uses*	Relasi *use case* tambahan ke sebuah *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.

Tabel 2.4 Lanjutan

<<include>> <<uses>>	Ada dua sudut pandang yang cukup besar mengenai *include* di *use case*: *include* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan, misal pada kasus berikut:  *Include* berarti *use case* yang akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan, misal pada kasus berikut:  Kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

Sumber: Rosa dan Shalahuddin (2016:156-158)

2.3.1.2. *Activity Diagram*

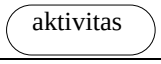
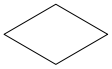
Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktvitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem (Rosa dan Shalahuddin, 2016:161)

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut (Rosa dan Shalahuddin, 2016:161-162):

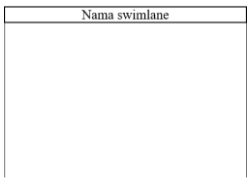
1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggao memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

Berikut adalah simbol-simbol yang ada pada diagram aktivitas:

Tabel 2.5 Simbol *Activity Diagram*

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

Tabel 2.5 Lanjutan

 Atau 	

Sumber: Rosa dan Shalahuddin (2016:162-163)

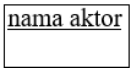
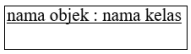
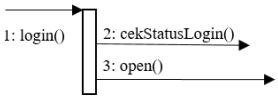
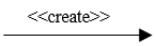
2.3.1.3. *Sequence Diagram*

Diagram sekuen menggambarkan kelakuan objek *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansikan menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case* (Rosa dan Shalahuddin, 2016:165).

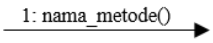
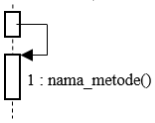
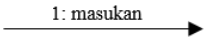
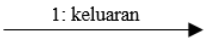
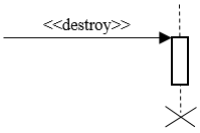
Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak (Rosa dan Shalahuddin, 2016:165).

Berikut adalah simbol-simbol yang ada pada diagram sekuen:

Tabel 2.6 Simbol *Sequence Diagram*

Simbol	Deskripsi
Aktor  Atau  Tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda diawal frase nama aktor
Garis hidup / *lifeline* 	Menyatakan kehidupan suatu objek
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, misalnya  Maka cekStatusLogin() dan open() dilakukan didalam metode login() Aktor tidak memiliki waktu aktif
Pesan tipe *create*  	Menyatakan suatu objek membuat objek lain, arah panah mengarah pada objek yang dibuat

Tabel 2.6 Lanjutan

Pesan tipe *call* 	Menyatakan suatu objek memanggil operasi / metode yang ada pada objek lain ata dirinya sendiri,  Arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi
Pesan tipe *send* 	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
Pesan tipe *return* 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe *destroy* 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*

Sumber: Rosa dan Shalahuddin (2016:165-167)

2.3.1.4. *Class Diagram*

Menurut Rosa dan Shalahuddin (2016:141-142) Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.

2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan didalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. Banyak berbagai kasus, perancangan kelas yang dibuat tidak sesuai dengan kelas-kelas yang dibuat pada perangkat lunak, sehingga tidaklah ada gunanya lagi sebuah perancangan karena apa yang dirancang dan hasil jadinya tidak sesuai (Rosa dan Shalahuddin, 2016:142).

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak dapat membuat kelas-kelas di dalam program perangkat lunak sesuai dengan perancangan diagram kelas. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut (Rosa dan Shalahuddin, 2016:142):

1. Kelas *main*

   Kelas yang memiliki funsi awal dieksekusi ketika sistem dijalankan.

2. Kelas yang menangani tampilan sistem (*view*)

   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.

3. Kelas yang diambil dari pendefinisian *use case*

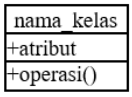
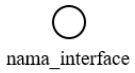
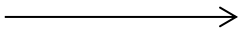
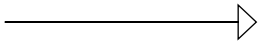
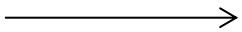
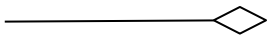
   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*, kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.

4. Kelas yang diambil dari pendefinisian data (*model*)

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

Berikut adalah simbol-simbol yang ada pada diagram kelas:

Tabel 2.7 Simbol *Class Diagram*

Simbol	Deskripsi
Kelas 	Kelas pada struktur sistem
Antarmuka / *interface* 	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi / *association*	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah / *directed association* 	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi 	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan / *dependency* 	Relasi antarkelas dengan makna kebergantungan antar kelas
Agregasi / *aggregation* 	Relasi antarkelas dengan makna semua-bagian (*whole-part*)

Sumber: Rosa dan Shalahuddin (2016:146-147)

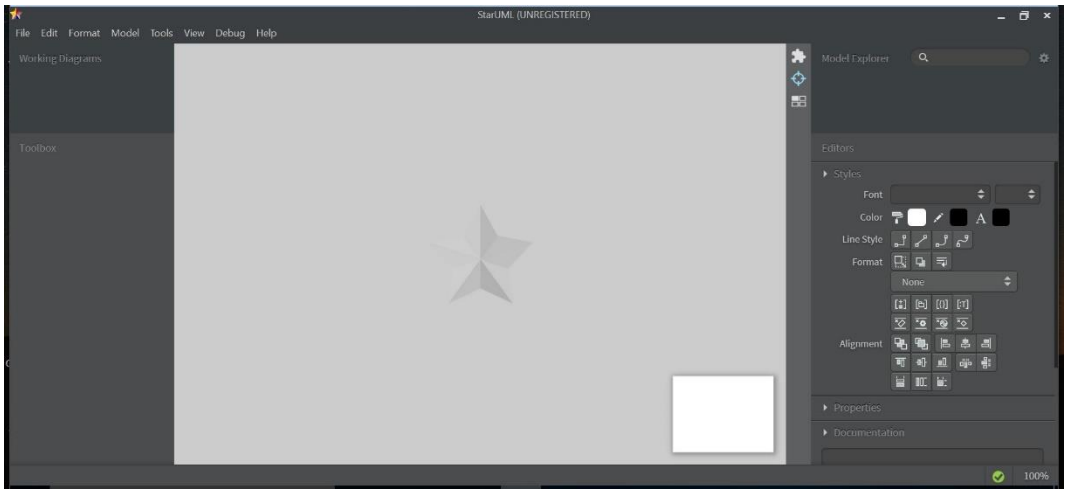
2.3.2. *StarUML*

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 (Rosa dan Shalahuddin, 2016:133).

Gambar 2.14 Logo *StarUML*

Menurut Rosa dan Shalahuddin (2016:137-138) *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 2.15 Tampilan *StarUML*

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 *upper CASE tools* yang mendukung perencanaan strategis dan pembangunan perangkat lunak (Rosa dan Shalahuddin, 2016: 122-123).

2.3.3. *PHP* (*PHP: Hypertext Preprocessor*)

Menurut Welling dan Thomson (2009: 2) *PHP: Hypertext Prepocessor* (*PHP*) adalah bahasa pemrograman *scripting server-side* yang didesain khusus untuk sebuah *web*. Di dalam halaman *HTML* (*Hyper Text Markup Language*), bisa di masukkan *Script PHP* yang ingin dijalankan (*Executed*). Kode *PHP* akan diinterpretasikan pada *web server* dan menghasilkan *HTML* atau *output* lainnya yang akan dilihat oleh pengunjung.

Gambar 2.16 Logo *PHP*

PHP adalah proyek *open source*, yang berarti memiliki akses ke kode sumber dan tidak membutuhkan biaya untuk menggunakan, mengubah, dan mendistribusikannya. Pada 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 dan Thomson, 2009: 3)

2.3.3. *HTML* (*Hyper Text Markup Language*)

HTML merupakan singkatan dari *Hyper Text Markup Language*. *HTML* bisa disebut bahasa paling dasar dan penting yang digunakan untuk menampilkan dan mengelola tampilan pada halaman *website*. Menurut sumber yang penulis kutip dari Wikipedia, *HTML* digunakan untuk menampilkan berbagai informasi didalam sebuah penjelajah *web* internet dan formatting *hypertext* sederhana yang ditulis ke dalam berkas format ASCII agar dapat meghasilkan tampilan wujud yang terintegrasi (Saputra, 2012:1).

Gambar 2.17 Logo *HTML*

Dengan kata lain, berkas yang dibuat dalam perangkat lunak pengolah kata, disimpan ke dalam format ASCII normal sehingga menjadi *homepage* dengan perintah-perintah *HTML*. *HTML* menggunakan 2 macam ekstensi file yaitu **.htm** dan **.html**. Format ekstensi berformat **.htm** awalnya hanyalah untuk mengakomodasi penggunaan *html* dalam operasi DOS (Saputra, 2012:1).

2.3.3.1. Struktur Dasar Dokumen *HTML*

Dokumen *HTML* memiliki sebuah struktur yang harus kita ikuti aturan pembuatannya. Beberapa elemen-elemen wajib yang ada pada *file html* apabila kita

ingin membangun suatu pondasi kerangka *website*. Elemen tersebut diantaranya (Saputra, 2012: 4-5):

Tabel 2.8 Syntax Struktur Dasar *HTML*

Struktur Dasar	Syntax
Elemen *HTML*	`<html> ... </html>`

Elemen *Head*	`<html>`
	`<head>`
	`</head>`
	`</html>`

Elemen *Title*	`<html>`
	`<head>`
	`<title> Tuliskan Judul disini </title>`
	`</head>`
	`</html>`

Elemen *Body*	`<html>`
	`<head>`
	`<title> Tuliskan Judul disini`
	`</title>`
	`</head>`
	`<body>`
	`Tuliskan Konten disini`
	`</body>`
	`</html>`

Sumber: Saputra (2012: 5-6)

1. Elemen *HTML*

   Elemen *HTML* merupakan *tag* dasar apabila kita ingin memulai suatu dokumen *html*. jika kita menemukan *tag* ini, berarti dapat didefinisikan bahwa dokumen ini merupakan dokumen *html*. *Tag* ini merupakan perintah wajib bagi pemprogram *web* untuk menuliskan *tag* pertama dalam dokumen *html*.

2. Elemen *Head*

   Head merupakan *tag* berikutnya setelah elemen *html*, yang berfungsi untuk menuliskan keterangan tentang dokumen *web* yang akan di tampilkan.

3. Elemen *Title*

   Elemen *Title* merupakan suatu elemen yang harus dituliskan didalam elemen *head* yang digunakan untuk memberikan judul/informasi pada *caption browser web* tentang topik dari suatu dokumen *web* yang ditampilkan pada *browser*.

4. Elemen *Body*

   Elemen *Body* merupakan bagian utama dalam dokumen *web*. Jika kita ingin menampilkan suatu teks atau informasi atau yang dikenal dengan sebutan konten, maka kita harus meletakkan teks tersebut pada elemen *body*.

2.3.3.2. Mengenal Perintah *HTML*

HTML memiliki perintah-perintah dasar yang dapat kita gunakan untuk mengatur konten *web*, diantaranya (Saputra, 2012:6-8):

Tabel 2.9 Syntax Pada Perintah *HTML*

Perintah *HTML*	Syntax
P (*Paragraph*)	`<p> ... </p>`
BR (*Line Break*)	` `
H1, H2, H3, H4, H5, H6 (*Header*)	`<Hx>...</Hx>`
B (*Bold*)	`<b> ... </b>`
I (*Italic*)	`<i> ... </i>`
U (*Underline*)	`<u> ... </u>`

Tabel 2.9 Lanjutan

PRE (*Preformated Text*)	`<pre>`
	`...`
	`</pre>`

Center	`<center>`
	`...`
	`</center>`

Basefont	`<basefont size="pixel">`

Font	`<font color="warna" face="font" size="pixel">`
	`...`
	`</font>`

HR (*Horizontal Rule*)	`<hr>`

OL (*Ordered List*)	`<ol start="number" type="A" | "a" | "I" | "i" | "1">`
	…
	`</ol>`

UL (*Unordered List*)	`<ul type="circle" | "square" | "disc">`
	…
	`</ul>`

LI (*List Item*)	`<li type="A" | "a" | "I" | "i" | "1" | "circle" | "square" | "disc">`
	…
	`</li>`

Sumber: Saputra (2012:6-8)

---

1. **P (*Paragraph*)**, berfungsi untuk mengganti paragraf yang diikuti dengan baris kosong diawal dan akhir paragraf.
2. **BR (*Line Break*)**, berfungsi untuk menggantikan baris.
3. **H1, H2, H3, H4, H5, H6 (*Header*)**, berfungsi untuk membuat *header* dengan ukuran enam jenis berbeda dan tercetak tebal.
4. **B (*Bold*)**, berfungsi untuk membuat tampilan teks tercetak tebal.
5. **I (*Italic*)**, berfungsi untuk membuat tampilan teks tercetak miring.

6. **U (*Underline*)**, berfungsi untuk membuat tampilan teks tercetak garis bawah.
7. **PRE (*Preformated Text*)**, berfungsi untuk menampilkan teks apa adanya.
8. ***Center***, berfungsi untuk menampilkan teks dengan posisi *horizontal* tengah.
9. ***BaseFont***, berfungsi untuk merubah dasar ukuran huruf dari *web browser*.
10. ***Font***, berfungsi untuk merubah jenis, ukuran, warna, dan tampilan huruf.
11. **HR (*Horizontal Rule*)**, berfungsi untuk membuat garis bawah.
12. **OL (*Ordered List*)**, berfungsi untuk membuat nomor daftar urut.
13. **UL (*Unordered List*)**, berfungsi untuk membuat daftar tanpa nomor urut (dalam format *bullet*).
14. **LI (*List Item*)**, merupakan isi pada daftar.

2.3.4. *CSS* (*Cascading Style Sheet*)

Menurut Saputra (2012:27) *CSS* merupakan bahasa pemrograman web yang didesain khusus untuk mengendalikan dan membangun berbagai komponen dalam *web* sehingga tampilan *web* lebih rapi, terstruktur dan seragam. *CSS* merupakan salah satu pemrograman wajib disamping *html* yang harus dikuasai oleh para setiap pemprogram *web*, terlebih lagi itu adalah *Web Designer*.

Gambar 2.18 Logo *CSS*

Tujuan utama dari *CSS* adalah untuk memisahkan konten utama dengan tampilan dokumen lainnya. *Web* yang menggunakan *CSS* akan lebih ringan dan mudah untuk dibuka dibandingkan dengan *web* yang tidak menggunakan *CSS*. Dengan menggunakan *CSS*, akan banyak keuntungan yang dapat kita peroleh, diantaranya (Saputra, 2012:27-29):

1. Memisahkan pembuatan dokumen (*CSS* dan *HTML*).
2. Mempermudah dan mempersingkat pembuatan dan pemeliharaan dokumen *web*.
3. Akses *web* lebih cepat saat di-*loading* (mempercepat pembacaan *HTML*).
4. Fleksibel, interaktif, tampilan lebih menarik dan nyaman dipandang.
5. Lebih kecil ukuran file sehingga *bandwith* yang digunakan juga otomatis menjadi lebih kecil.
6. Dapat digunakan pada semua *web* browser.

2.3.5. *Java Script*

Menurut Sidik dan Pohan (2009: 267) *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* (\)

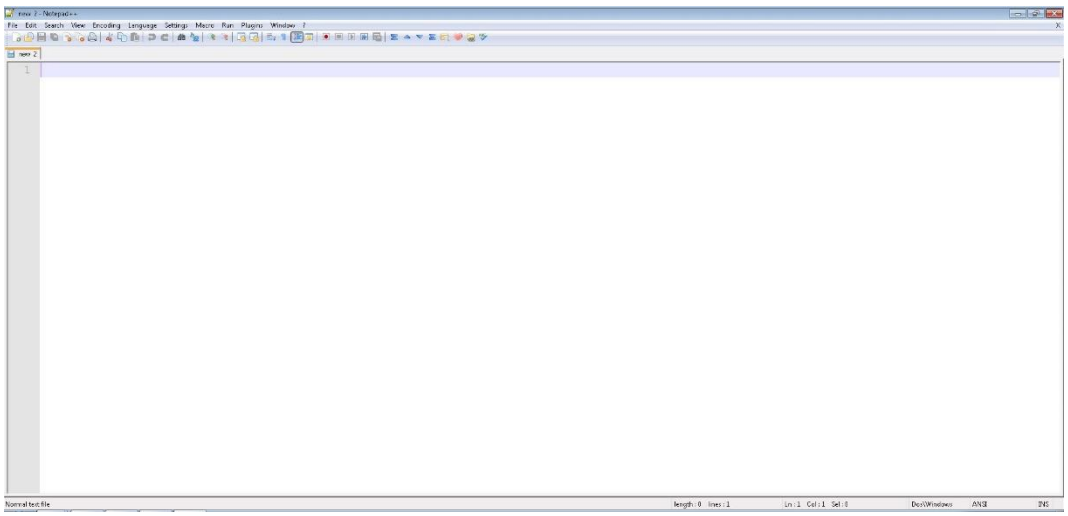
Gambar 2.19 Logo Java Script

2.3.6. *Notepad++*

Menurut Gilmore (2010: 36) *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).

Gambar 2.20 Logo Notepad++

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.



Gambar 2.21 Tampilan *Notepad++*

Salah satu kelebihan *Notepad++* adalah dukungan dasar untuk *auto-completion* dari nama fungsi yang ditawarkan sehingga akan mengurangi beberapa proses pengetikan kode.

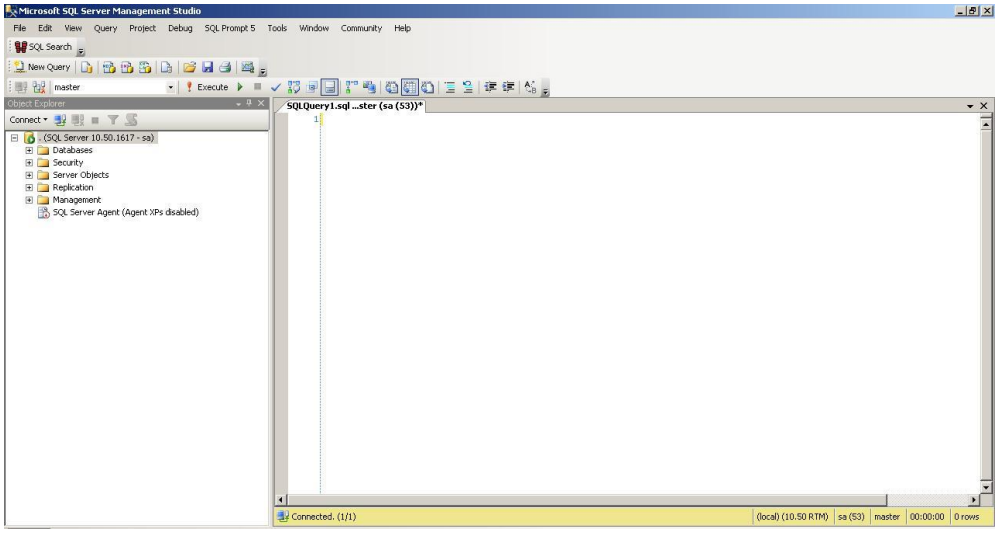
2.3.7. *Microsoft SQL Server 2008*

Database atau basis data adalah sekumpulan data yang memiliki hubungan secara logika dan diatur dengan susunan tertentu serta disimpan dalam media penyimpanan komputer. *Database* sering digunakan untuk melakukan proses

terhadap data-data tersebut untuk menghasilkan informasi tertentu. Tentu saja informasi tersebut akan anda dapatkan dari *software* pemroses *database* dengan cara anda memberikan perintah dalam bahasa tertentu yaitu *SQL* (*Structured Query Language*) (Wahana Komputer, 2010:24).

Gambar 2.22 Logo *Microsoft SQL Server*

SQL Server 2008 adalah sebuah terobosan baru dari *Microsoft* dalam bidang *database*. *SQL Server* adalah sebuah *DBMS* (*Database Management System*) yang dibuat oleh *Microsoft* untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahulunya seperti *IBM* dan *Oracle*. *SQL Server 2008* dibuat pada saat kemajuan dalam bidang *hardware* sedemikan pesat. Oleh karena itu sudah dapat dipastikan *SQL Server 2008* membawa beberapa terobosan dalam bidang pengolahan dan penyimpanan data (Wahana Komputer, 2010:2).



Gambar 2.23 Tampilan *Microsoft SQL Server*

Menurut Wahana Komputer (2010:26) *SQL* dibedakan menjadi dua jenis sub bahasa, yaitu:

1. *Data Definition Language* (*DDL*), bahasa ini digunakan untuk "membangun" struktur *database*. Contoh dari bahasa ini adalah *CREATE*, *DROP*, *ALTER*. Bahasa ini dikenakan pada *database*, tabel, kolom (*field*), dan index.
2. *Data Manipulation Language* (*DML*), jenis *SQL* ini berfungsi untuk melakukan manipulasi terhadap data yang ada seperti *record*, *field*. Contoh perintahnya adalah *DELETE*, *UPDATE*, *INSERT*, dan yang paling terkenal adalah *SELECT*.

2.3.8. *XAMPP* (*X Apache MySQL PHP Perl*)

Menurut Sidik dan Pohan (2009: 6) *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 *database server MySQL* dan dapat mendukung pemrograman *PHP*.

Gambar 2.24 Logo *XAMPP*

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*, *PHP support* (*PHP* 4 dan *PHP* 5) dan beberapa modul lainnya (Februariyanti dan Zuliarso, 2012: 129).

2.4. Penelitian Terdahulu

Sebagai bahan pertimbangan dalam penelitian ini, maka di cantumkan beberapa penelitian dari jurnal ilmiah, yaitu:

1. **Diah Krisnatuti, Tin herawati dan Nurlaili** (2011:154), Hubungan Antara Kecerdasan Emosi Dengan Kepatuhan dan Kemandirian Santri Remaja, Hasil penelitian menemukan bahwa santri berasal dari keluarga sedang (jumlah anggota keluarga 5-6 orang) dan sebagian besar santri memiliki kecerdasan emosi yang baik. Hasil uji korelasi dalam penelitian ini menunjukkan bahwa kecerdasan emosi berhubungan signifikan positif dengan keluarga. Hasil penelitian menunjukkan bahwa santri memiliki tingkat kepatuhan yang tergolong dalam kategori rendah, sedangkan kemandirian dan kecerdasan emosi tergolong dalam kategori baik. Kecerdasan emosi berhubungan signifikan dengan besar keluarga, kepatuhan, dan kemandirian.
2. **Triana Fitriastuti** (2013:111), Pengaruh Kecerdasan Emosional, Komitmen Organisasinal dan *Organizational Citizenship Behaviour* Trhadap Kinerja Karyawan, Berdasarkan hasil penelitian ditemukan, bahwa kecerdasan emosional berpengaruh positif signifikan terhadap kinerja karyawan. Peningkatan kinerja karyawan dipengaruhi oleh tinggi rendahnya tingkat

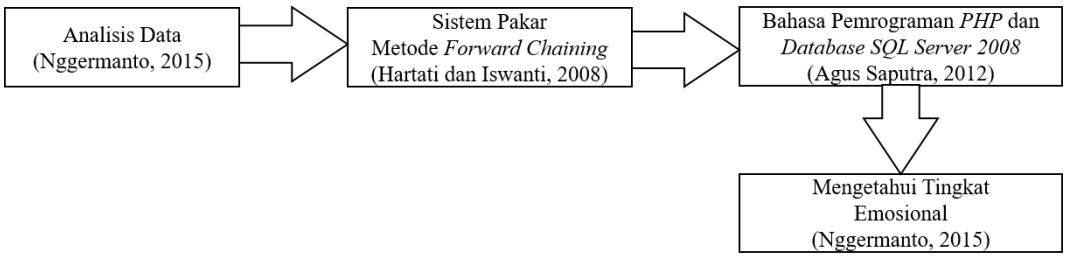
kecerdasan emosional yang dimiliki karyawan. Hal itu dikarenakan, self awareness yang semakin baik membuat karyawan akan cenderung berperilaku sesuai dengan standar organisasi, sehingga pada akhirnya akan mencapai kinerja yang lebih baik.

3. **Adhi Prastistha Silen** (2014:127), Pengaruh Kecerdasan Intelektual, Kecerdasan Emosional dan Kecerdasan Spiritual Terhadap Prestasi Akademik, Terdapat pengaruh positif dan signifikan antara Kecerdasan emosional terhadap Hasil belajar. Hasil ini menunjukkan bahwa Taruna yang memiliki ketrampilan emosi yang baik memiliki motivasi untuk terus belajar. Sedangkan, Taruna yang memiliki ketrampilan emosi yang kurang baik, akan kurang memiliki motivasi untuk belajar, sehingga dapat merusak kemampuannya untuk memusatkan perhatian pada tugas-tugas individu tersebut sebagai Taruna.
4. **Suardi Yakub, Rudi Gunawan, Jufri Halim** (2015:169), Pengaruh Kemampuan Komunikasi dan Kecerdasan Emosional Terhadap Kinerja Karyawan Pada PT. Perkebunan Nusantara I (Persero)Aceh, Kemampuan komunikasi dan kecerdasan emosional secara simultan berpengaruh positif dan signifikan terhadap kinerja karyawan pada PT. Perkebunan Nusantara I (Persero) Aceh.Hal ini berarti dengan adanya kemampuan komunikasi dan kecerdasan emosional yang ada pada diri karyawan maka akan dapat memberikan dampak yang nyata terhadap peningkatan kinerja karyawan.
5. **Irham Cahya Nugraha, Herlawati** (2016:147), Sistem Pakar Tes Minat dan Bakat Jurusan Kuliah Berbasis Android pada SMA Islam Teratai Putih Global

Bekasi, Dengan adanya aplikasi ini, dapat meningkatkan antusias masyarakat dengan ilmu psikologi yang sesungguhnya dapat memberikan banyak manfaat bagi masyakarakat khususnya siswa-siswi SMA Islam Teratai Putih Global Bekasi. Aplikasi sistem pakar tes minat bakat berbasis android ini ini dirancang dengan sederhana dan interaktif, agar muda dalam pendistribusiannya dan juga dalam penggunaannya. Dengan berbasis android pengguna lebih nyaman dan mudah dalam melakukan tes ini tanpa harus merasa tidak nyaman apabila bertatap muka langsung dengan psikolog.

2.5. Kerangka Berfikir

Kerangka berfikir merupakan model konseptual tentang bagaimana teori berhubungan dengan berbagai faktor yang telah diindetifikasi sebagai masalah penting (Sugiyono, 2013:60). Berikut ini adalah kerangka pemikiran yang mendasari penelitian ini.



Gambar 2.17 Kerangka Pemikiran

Data-data yang dibutuhkan berkaitan dengan tes *Emotional Quotient* dianalisis terlebih dahulu agar mudah dilakukan pengolahan. Data-data tersebut kemudian diolah menggunakan Sistem Pakar metode *Forward Chaining* untuk

membuat aturan (*rule*). Sistem Pakar tersebut dibuat dengan bahasa pemrograman *PHP* dan *Database Microsoft SQL Server 2008* sehingga menghasilkan sebuah Sistem Pakar untuk mengetahui tingkat emosional menggunakan metode *forward chaining*.