BAB II

LANDASAN TEORI

2.1 Sistem Pakar

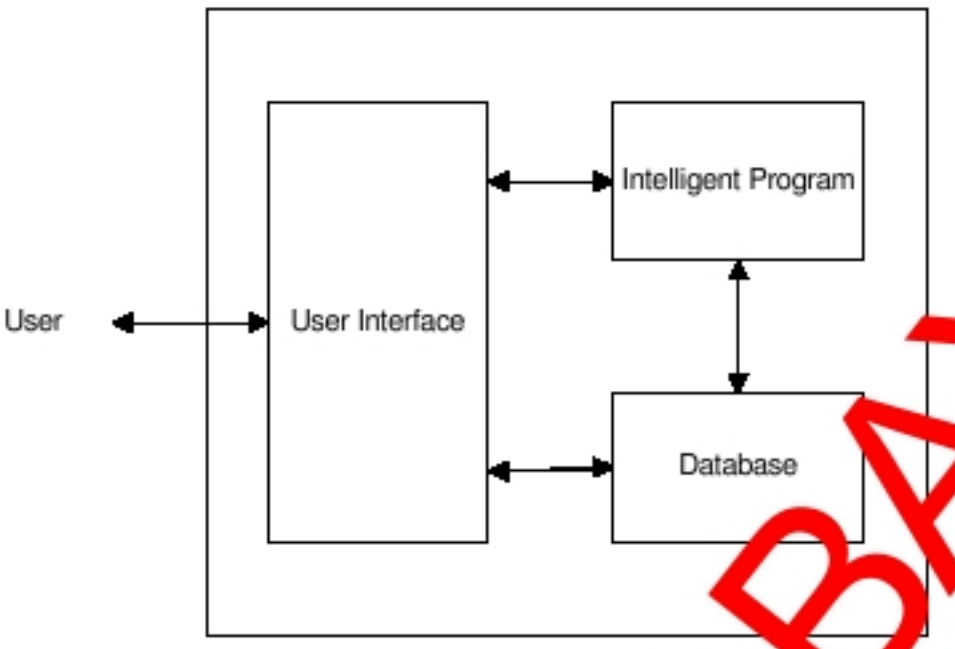
Sistem pakar yang juga dikenal sebagai sistem berbasis pengetahuan (knowledge base system) adalah sistem komputer yang dirancang untuk mewakili pengetahuan dari pakar manusia untuk memecahkan masalah sesuai dengan fakta yang ada berdasarkan definisi masalah yang menggunakan aturan yang tepat.

Sistem pakar dapat mengumpulkan dan menyimpan pengetahuan tersebut. Kemudian digunakan oleh siapa saja yang memerlukannya. Sistem pakar bukan mengganti kedudukan seorang pakar, melainkan hanya untuk memasyarakatkan pengetahuan dan pengalaman para pakar.

Didalam menyelesaikan suatu masalah sistem pakar akan mengajukan berbagai pertanyaan kepada pemakai (user) dalam rangka pengumpulan informasi sampai sistem pakar itu dapat memberikan suatu penyelesaian sangat ideal bagi seseorang yang harus memilih serangkaian alternatif terbaik dari alternatif yang ada.

2.2 Komponen Sistem Pakar

Sistem pakar terdiri dari beberapa komponen utama. Komponen tersebut bisa dilihat seperti gambar dibawah ini :



Gambar 2.1 Komponen Sistem Pakar

2.2.1 User Interface

Merupakan bagian software yang menyediakan sarana untuk user agar bisa berkomunikasi dengan system. User interface menyediakan bagi user kemudahan dalam berkomunikasi dengan program pintar. User interface akan mengajukan pertanyaan-pertanyaan, dalam itu juga menyediakan menu pilihan untuk memasukkan informasi awal dalam knowledge base. Kemudian ia juga menyediakan saran komunikasi jawaban atau solusi bila masalahnya sudah ditemukan. Setiap komunikasi antara dan selama proses pemecahan masalah dikendalikan oleh user interface.

2.2.2 Intelligent Program

1. The Knowledge Base

Knowledge base adalah data atau pengetahuan yang diperlukan untuk membuat suatu keputusan diman didalam knowledge base tersebut memuat fakta-

fakta dan juga teknik dalam menerangkan masalah yang menjelaskan bagaimana fakta-fakta tersebut cocok yang stu dengan yang lain dalam urutan yang logis

2. Inference Engine

Inference Engine adalah bagian dari sistem pakar yang melakukan penalaran dengan menggunakan isi rulebase berdasarkan urutan tertentu. Selama konsultasi, inference engine menguji aturan-aturan dari rulebase satu demi satu dan saat kondisi aturan itu benar tindakan tertentu diambil dan jika saat kondisi aturan itu salah akan dikesampingkan.

- **Backward Chaining**

Backward Chaining adalah metode penelusuran yang dimulai dari hasil/kesimpulan untuk mencari kondisi yang berkaitan dengan hasil tersebut. Metode ini banyak digunakan dalam persoalan diagnosa. Secara umum, backward chaining memiliki ciri :

* diawali dari hasil kesimpulan
* menelusuri kondisi-kondisi (premis) yang menyusun kesimpulan tersebut

Contoh persoalan yang diselesaikan dengan backward chaining adalah :

Jika kamu ingin memetik buah itu, gunakanlah tangan

Memetik buah merupakan hasil, dan menggunakan tangga adalah kondisi untuk mencapai hasil tersebut.

- **Forward Chaining**

Forward Chaining adalah metode penelusuran yang dimulai dari data untuk mencari kesimpulan. Metode ini biasanya digunakan dalam persoalan

analisa, desain, diagnosa dan konstruksi. Secara umum, forward chaining memiliki ciri :

* diawali dengan kumpulan premis
* menghasilkan suatu kesimpulan

Contoh persoalan yang diselesaikan dengan forward chaining ada

Jika tekanan darahmu tinggi, berarti kamu sakit

Tekanan darah tinggi merupakan data, dan sakit adalah ke

2.3 Verifikasi Sistem Pakar

Proses yang mana menjamin bahwa suatu sistem bebas dari error selama implementasi.

2.3.1 Redundant Rules

Dua rule dianggap *redu* jika mempunyai premis yang identik atau sama dan konklusi yang sama pula. Contohnya :

RULE 1 : IF Kelem an tinggi AND
Temperatur panas
THEN Terjadi guntur

RULE 2 : IF Temperatur panas AND
Kelembapan tinggi
THEN Terjadi guntur

Kedua rule tersebut berkonklusi sama yaitu " Terjadi guntur " maka rule tersebut dianggap *redundant*.

2.3.2 Conflicting Rules

Conflicting rules terjadi apabila premis dari 2 rule sama (identik) namun konklusinya berbeda. Misalkan :

RULE 5 : IF Temperatur panas AND

Kelembapan tinggi

THEN Akan terjadi matahari terbit

RULE 6 : IF Temperatur panas AND

Kelembapan tinggi

THEN Tidak akan terjadi matahari terbit

2.3.3 Subsumed Rules

Suatu rule dikatakan *subsumed* dengan rule yang lain jika rule tersebut mempunyai constraints lebih pada premis yang mana mempunyai konklusi yang sama. Misalkan :

RULE 7 : IF Temperatur panas AND

Kelembapan tinggi AND

Barometer menunjukkan rendah

THEN Akan terjadi guntur

RULE 8 : IF Temperatur panas AND

Kelembapan tinggi

THEN Akan terjadi guntur

Dapat dilihat RULE 7 teringkas oleh RULE 8 karena Rule awal mempunyai lebih dari satu constraint dari pada rule akhir.

2.3.4 Circular Rules

Suatu keadaan dimana terjadinya proses perulangan dari suatu rule. Ini dikarenakan suatu premis dari salah satu rule merupakan konklusi dari rule yang lain atau kebalikannya. Misalnya :

RULE 9 : IF X dan Y adalah saudara

THEN X dan Y mempunyai orang tua yang sama

RULE 10 : IF X dan Y mempunyai orang tua yang sama

THEN X dan Y adalah saudara

2.3.5 Unnecessary IF Conditions

Dikatakan *unnecessary rules* apabila 2 rule dengan konklusi yang sama mempunyai premis yang hampir sama. Premis dari rule-rule hampir sama kecuali satu dari tiap rule merupakan kebalikan (kontradiksi). Contoh :

RULE 12 : IF Pasien berbintik merah muda AND

Pasien demam

THEN Pasien menderita penyakit campak

RULE 13 : IF Pasien berbintik merah muda AND

Pasien tidak demam

THEN Pasien menderita penyakit campak

Jika premis yang kedua dari tiap rule benar-benar tidak diperlukan, dua rule tersebut bisa dibentuk menjadi single rule.

RULE 14 : IF Pasien berbintik merah muda

THEN Pasien menderita penyakit campak

Conflicting rules tidak selalu mudah untuk diperbaiki. Sering suatu conflict tidak berarti unnecessary IF conditions tapi hanya missing premis (premis yang hilang) dalam rule atau ekspresi yang salah dalam pengetahuan dalam rule

2.3.6 Dead-end Rules

Dead-end rules dalam forward chaining adalah rule yang tidak mempunyai pengaruh terhadap konklusi dan tidak digunakan oleh rule yang lain. Misalkan :

RULE 15 : IF Pengukur gas menunjukkan kosong

THEN Tangki gas kosong

Tapi jika konklusi "tangki gas kosong" bukan goal dari sistem dan fakta ini tidak digunakan oleh rule yang lain maka rule ini dianggap sebuah *dead-end rule.*

Dalam situasi yang hampir sama untuk contoh RULE 15 dalam backward chaining sistem, jika menggunakan input dari pengukur gas mobil menunjukkan berapa banyak gas yang tersisa dalam gas mobil, dianggap sebuah *dead-end rule* jika premis dalam rule tidak mempunyai input atau input tidak pernah sesuai dengan spesifikasi pada premis. Dengan kata lain, bagaimanapun tipe rule ini bisa dilacak, tidak akan pernah mempengaruhi konklusi. Itu juga menunjukkan indikasi bahwa rule tersebut mungkin tidak perlu, mungkin ada rule yang hilang atau nilai dari spesifik rule salah.

2.3.7 Missing Rules

Missing rules terjadi oleh fakta-fakta yang tidak digunakan dalam proses inference, konklusi tidak mempengaruhi oleh rule yang lain atau prosedur atau suatu kesalahan dalam melindungi semua nilai beberapa input. Untuk ilustrasi

masalah ini adalah RULE 15 merupakan diagnosa automobile. Rule ini bisa merupakan hasil dari suatu *missing rule* yang mana digunakan output dari dead-end rule untuk mencapai konklusi yang lain. Misalkan sistem ini mempunyai input untuk banyaknya gas yang tersisa dalam tangki, tetapi tidak digunakan "ukuran gas" untuk membawa ke konklusi yang lain. Bisa dipertimbangkan dalam situasi ini bahwa satu rule hilang (missing rule) yang mengikat masalah dengan diawali mesin automobile ke banyaknya gas ketika tangki kosong.

2.3.8 Unreachable Rules

Ini adalah jenis sintatic error yang mana berbeda untuk forward dan backward chaining. Unreachable rule merupakan kebalikan dari dead-end rule.

Pada forward chaining, unreachable rule mempunyai satu premis yang tidak akan pernah sesuai dengan sistem dalam berbagai keadaan, salah satunya untuk rule yang hilang atau untuk kekurangan dari input data. Ini sama dengan dead-end rule dalam backward chaining.

Dalam backward chaining keadaannya berbeda. Akibat dari unreachable rule tidak sesuai dengan arti goal dalam suatu sistem atau hipotesis. Hal ini membuat RULE 15 tidak dapat dilacak dalam backward chaining, yang mana sama dengan dead-end rule dalam forward chaining.

2.4 Certainty Factor (CF)

Certainty Factor (faktor kepastian) digunakan untuk membantu menangani situasi atau keadaan yang tidak pasti. Knowledge dalam system pakar menggunakan Certainty Factor (CF) untuk mengekspresikan rule, dengan format :

IF EVIDENCE

THEN HYPOTHESIS (CF)

Certainty Factor identik dengan "confidence" atau "truth factor". "confidence" atau "belief" bukanlah probabilitas, merupakan pengukuran kepercayaan atau kuantisasi dari penelitian / kecenderungan seorang pakar, yang menunjukkan kredibilitas dari konklusi dengan evidence berupa premis dalam rule.

Factor kepastian menunjukkan seberapa besar tingkat keyakinan bahwa fakta itu benar, berdasarkan pengalaman fakta yang ada.

Hubungan antara IF dengan probalitas adalah :

1. Measure of Belief (MB[h,e]) bernilai 0 s/d 1

$$MB[h,e] = \frac{p[h|e] - p[h]}{1 - p[h]} \quad = 1 \text{ jika } p[h] = 1$$

2. Measure of Disbelief (MD[h,e]) bernilai 0 s/d 1

$$MD[h,e] = \frac{p[h|e] - p[h]}{p[h]} \quad = 1 \quad \text{jika } p[h] = 0$$

3. Certainty Factor (CF) bernilai –1 s/d 1

$$CF = \frac{MB - MD}{1 - \min(MB,MD)}.$$

Propagasi dari certainty factor adalah :

$CF_{revised}$ (CF_{old}, CF_{new}) =

Bila kedua CF_{old} dan $CF_{new} \geq 0$

$= CF_{old} + CF_{new} (1 - CF_{old})$

Bila kedua CF_{old} dan $CF_{new} < 0$

$$= -CF_{revised}\,(-CF_{old}, -CF_{new})$$

Bila salah satu CF_{old} dan CF_{new} bernilai negatif

$$= \frac{CF_{old} + CF_{new}}{1 - \min(|CF_{old}|, |CF_{new}|)}.$$

Jika nilai evidence dalam sebuah rule tidak benar-benar yakin, maka

1. If A Then Q (CF = …)

   $CF_{combination} = CF_{evidence} * CF_{hypothesis}$

2. If B And

   C And

   D

   Then Q (CF = …)

   $CF_{combination} = CF_{evidence(min)} * CF_{hypothesis}$

3. If E Or

   F Or

   G

   Then Q (CF = …)

   $CF_{combination} = CF_{evidence(max)} * CF_{hypothesis}$