2. TEORI PENUNJANG

Penyusunan bab dua ini bertujuan untuk membantu dalam melakukan proses perancangan dan pembuatan *inference engine* dan *user interface* maupun untuk pembuatan *knowledgebase* dan *rule* dari *software* Aplikasi Pengaruh *Feng-shui* terhadap harga jual rumah pada *property agent* X. Bab ini berisikan teori yang berhubungan dengan pembuatan *software* Aplikasi Pengaruh *Feng-shui* terhadap harga jual rumah pada *property agent* X, seperti pengertian kecerdasan buatan (*Artificial Intelegence*); pengertian tentang sistem pakar (*Expert System*); pengertian tentang *knowledge base*; pengertian tentang *dependency diagram*; pengertian *decision table*; pengertian *IF-THEN rules*; pengertian *user interface*; pengertian *inference engine*; pengertian metode *forward* dan *backward chainning*; pengertian *uncertainty*; serta penjelasan tentang *feng-shui* beserta beberapa faktor-faktor yang dapat mempengaruhi harga jual rumah. Penjelasan dari teori-teori tersebut akan diuraikan pada sub-bab berikut ini.

2.1. Sistem Pakar (*Expert System*)

Sistem Pakar *(expert system)* adalah suatu metode *artificial intelegence* yang berguna untuk meniru cara berpikir dan penalaran seorang ahli dalam mengambil keputusan berdasarkan situasi yang ada. Sehingga seorang *user* dapat melakukan konsultasi kepada komputer, seolah-olah *user* tersebut berkonsultasi kepada seorang ahli. Contohnya adalah program aplikasi yang mampu meniru seorang ahli *feng-shui* dalam memprediksi harga jual sebuah rumah berdasarkan *feng-shui* pihak pembeli dan *feng-shui* rumah.

Software Aplikasi Pengaruh *Feng-shui* terhadap harga jual rumah pada *property agent* X menggunakan sistem pakar (*expert system)* sebagai dasar pembuatannya. Penjelasan lebih lanjut mengenai sistem pakar (*expert system)* dijelaskan lebih lanjut dalam sub-bab berikut.

2.1.1. Penjelasan Sistem Pakar (*Expert System*)

Sistem pakar (*expert system)* memiliki 10 karakteristik yang harus dipenuhi dalam perancangannya. Kesepuluh karakteristik *expert system* tersebut adalah :

a. Mendukung proses pengambilan keputusan, menitik beratkan pada *management perception.*
b. Adanya *human interface* dimana manusia (*user*) tetap mengontrol proses pengambilan keputusan.
c. Mendukung pengambilan keputusan uintuk membahas masalah-masalah terstruktur, semi terstruktur, dan tidak terstruktur.
d. Menggunakan model-model matematis dan statistik yang sesuai.
e. Interaktif, memiliki kapabilitas dialog untuk memperoleh informasi sesuai dengan kebutuhan.
f. *Output* ditujukan untuk semua orang secara umum.
g. Modularitas, memiliki subsistem-subsistem yang terintegrasi sedemikian rupa sehingga dapat berfungsi sebagai kesatuan sistem.
h. Membutuhkan struktur data komprehensif yang dapat melayani kebutuhan informasi seluruh tingkatan manajemen.
i. *User friendly* dan fleksibel, yaitu mudah untuk digunakan oleh *user* dan memungkinkan keleluasaan *user* untuk memilih atau mengembangkan pendekatan-pendekatan baru.
j. Kemampuan sistem beradaptasi secara cepat, dimana pengambilan keputusan dapat menghadapi masalah-masalah baru.

Sistem pakar (*expert system)* menggunakan basis pengetahuan (*knowledge base*) sebagai dasar pemikirannya. *Knowledge base* tersebut terdiri dari heuristik dan sejumlah *rule-rule* atau aturan-aturan yang tersusun secara sistematis dan spesifik, juga relasi antara data dan aturan/*rule* dalam pengambilan kesimpulan. *Knowledge base* tersebut disimpan dalam sebuah *database* atau basis data pada suatu tempat penyimpanan data. Sedangkan sebagai otak atau pusat pemrosesannya adalah *inference engine*, yaitu suatu rancangan aplikasi yang berfungsi untuk memberikan pertanyaan dan menerima *input* dari *user,* kemudian melakukan proses logika sesuai dengan *knowledge base* yang tersedia, untuk

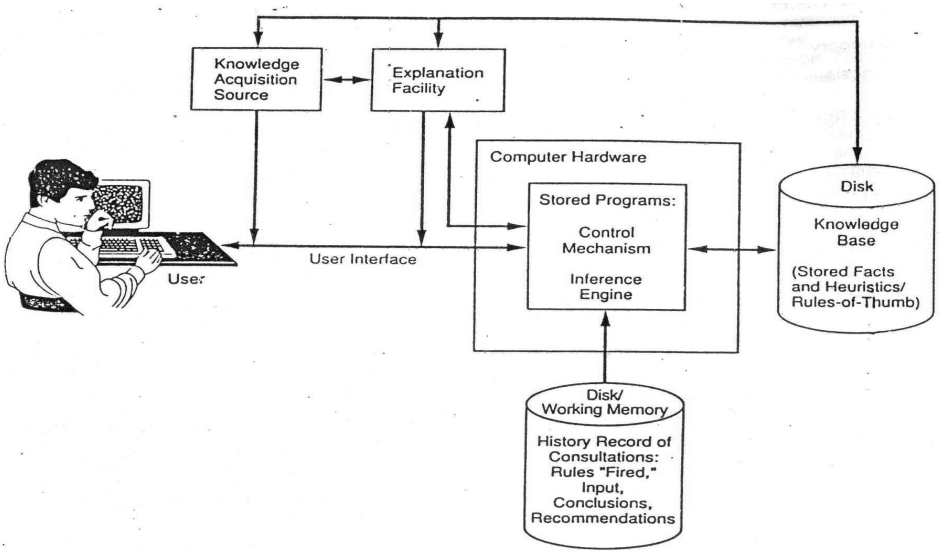
selanjutnya menghasilkan *output* berupa suatu kesimpulan atau bisa juga berupa keputusan/*decision* sebagai hasil akhir konsultasi.

Knowledge Acquisition Source berfungsi sebagai penterjemah dari *knowledge base* menjadi sebuah bahasa yang dapat dimengerti oleh *user.* Bagian ini diperlukan karena *knowledge base* yang disimpan dalam sebuah *database*, disimpan dalam format tertentu, sedemikian rupa sehingga *user* sulit mengartikannya.

Disk/Working memory adalah sejumlah modul *memory* yang menyimpan informasi sementara dari suatu proses konsultasi. Setiap proses baru dijalankan, *memory* tersebut akan di set ke kondisi awal. Dalam menjalani proses, *memory* tersebut menyimpan informasi *state* dari *rule-rule* yang dipakai dalam *knowledge base*.

Explanation facility menyimpan data-data historis dari sebuah proses konsultasi, yaitu *rule-rule* mana saja yang berperan dalam suatu proses pengambilan keputusan. Data-data historis tersebut dapat dijadikan bahan evaluasi dari hasil kerja sebuah *knowledge based system.*

User memasukkan *input* dan menerima *output* melalui sebuah *interface* atau tampilan, yaitu sebagai sarana penghubung interaksi antara *user* dengan sistem. Bagan Sistem Pakar (*Expert System*) secara umum dapat dilihat pada Gambar 2.1.



Gambar 2.1 Bagan Sistem Pakar (*Expert System*) secara umum
Sumber: Dologite, D.G. Developing Knowledge-Based Systems Using VP-Expert. New York : Macmillan Publishing Company, 1993.

Langkah-langkah perancangan sistem pakar (*expert system)* sebaiknya mengikuti urutan sebagai berikut :

- Menentukan batasan-batasan atau bidang konsentrasi dari sebuah *expert system* yang akan dirancang.
- Memilih jenis keputusan apa yang akan diambil.
- Membuat sebuah *dependency diagram*.
- Membuat *decision table*.
- Menuliskan *IF-THEN rules*.
- Merancang *user interface*.

2.1.1.1. *Knowledge Base*

Knowledge base adalah suatu struktur data yang menyimpan informasi data, *rule*, relasi antara data dengan *rule* dalam pengambilan kesimpulan. *Knowledge base* dapat dikatakan sebagai kumpulan informasi dan pengalaman seorang ahli pada suatu bidang tertentu. *Knowledge base* terdiri dari fakta seperti halnya buku teori dan heuristik yang merupakan langkah-langkah seorang ahli dalam mengambil keputusan.

Setiap kasus yang ditemui, diproses berdasarkan pengalaman yang pernah tercatat, baru dapat diambil kesimpulan. Apabila terjadi suatu kasus yang belum pernah dialami, maka sistem pakar tidak mampu mengambil kesimpulan dari kasus tersebut. Untuk itu diperlukan penambahan data baru yang dapat dijadikan acuan yang melengkapi *knowledge base*. Jadi, suatu *knowledge base* dapat dikatakan baik apabila memiliki sejumlah *rule* yang mampu digunakan dalam setiap kemungkinan kasus dalam ruang lingkup tertentu. Dalam perancangannya, perancang harus mampu menggali sebanyak mungkin informasi dari narasumbernya, seorang ahli dan memindahkannya ke dalam *knowledge base*.

Berdasarkan dari *knowledge base* yang dimilikinya, sistem pakar dapat digolongkan menjadi 5 tingkatan, yaitu :

a. Tingkat 1 :
    - Menggunakan *internal knowledge base.*
    - Dapat melakukan penambahan data dan membuat laporan data yang ada pada *knowledge base.*

- Menggunakan metode *backward chaining* atau metode *forward chaining* untuk menyelesaikan suatu kasus.
- Mampu menampilkan hasil kerja dari *inference engine* melalui *user interface.*

b. Tingkat 2 :
- Menggunakan *external knowledge base.*
- Mampu memilih metode *backward chaining* atau *forward chaining* yang akan digunakan untuk menyelesaikan suatu kasus.
- Mampu melakukan penghitungan matematis.

c. Tingkat 3 :
- Mampu melakukan ekspor atau impor suatu *knowledge base.*
- Mampu melakukan penghitungan matematis secara dinamis.

d. Tingkat 4 :
- Dapat dioperasikan pada berbagai *operating system.*

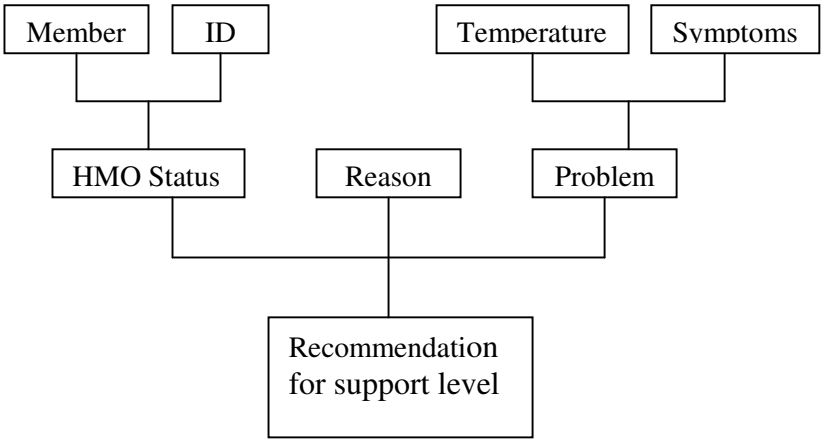
e. Tingkat 5 :
- Mampu merubah atau menambah isi *knowledge base* secara otomatis dan mempelajari suatu pola baru dari beberapa kasus yang pernah dialaminya.

2.1.1.2. *Dependency Diagram*

Dependency diagram menunjukkan hubungan antar faktor-faktor kritis, pertanyaan yang di-*input*-kan, *rules*, *values*, dan rekomendasi yang dibuat oleh *knowledge based system.*

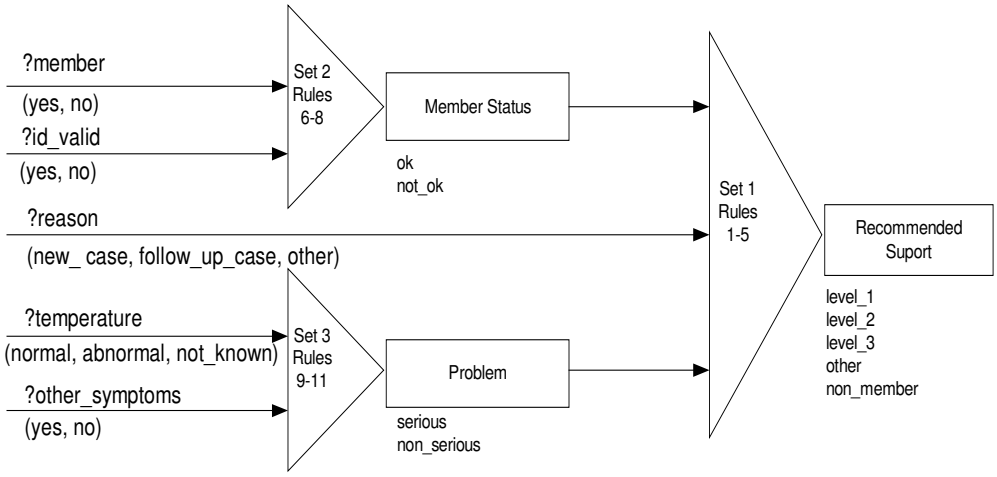
Untuk mempermudah penjelasan, dapat diambil sebuah contoh kasus tentang sebuah organisasi kesehatan hewan. Organisasi tersebut ingin membuat sebuah sistem pakar yang berguna untuk menentukan jenis perawatan bagi seekor hewan berdasarkan beberapa kondisi tertentu. Seekor hewan dapat menjadi anggota oraganisasi tersebut dan mendapat perawatan khusus, sedangkan hewan yang bukan merupakan anggota organisasi tersebut akan mendapat perawatan biasa. Perawatan khusus hanya bagi anggota organisasi, perawatan khusus tersebut dibagi menjadi tiga level tergantung pada jenis kasusnya, dan tingkat keseriusan problemnya. Untuk anggota dengan problem yang serius, diberikan perawatan level 1. Untuk anggota dengan kasus baru dan problem yang tidak

serius, diberikan perawatan level 2. Untuk anggota dengan kasus lanjutan dan problem yang tidak serius, diberikan perawatan level 3. Perawatan lain-lain diberikan pada anggota dengan kasus lain-lain. Dari persoalan di atas, dapat disusun sebuah diagram blok mengenai pengambilan keputusan berdasarkan situasi yang ada, sebagai prototipe awal pembuatan *knowledge based system*.



Gambar 2.2. Diagram Blok sebagai prototipe pembuatan *knowledgebased system.*
Sumber: Dologite, D.G. Developing Knowledge-Based Systems Using VP-Expert. New York : Macmillan Publishing Company, 1993.

Dari Gambar 2.2., dapat dibentuk sebuah *dependency diagram*. Untuk tiap faktor kritis yang pada diagram blok tadi, dapat dibuat sebuah diagram berbentuk segitiga dan sebuah diagram berbentuk persegi panjang di sebelahnya. Faktor kritis adalah faktor-faktor yang memberikan sebuah *value* kepada proses pengambilan keputusan *intermediate* maupun pengambilan keputusan final. Dalam kasus ini, faktor-faktor kritis tersebut adalah *member_status, reason, problem.* Karena faktor *reason* berhubungan langsung dengan pengambilan keputusan final, maka diagram *intermediate*-nya dapat dihilangkan. Pada setiap segitiga, dapat digambarkan garis panah menuju segitiga tersebut. Dan pada garis tersebut dituliskan variabel-variabel yang menentukan hasil dari faktor kritis beserta kemungkinan *values* yang dimiliki variabel tersebut. Pada setiap persegi panjang yang melambangkan faktor kritis dan rekomendasi, dapat juga dituliskan kemungkinan *values* yang dimiliki faktor kritis tersebut. Hasilnya dapat dilihat pada Gambar 2.3.



Gambar 2.3. *Dependency diagram*
Sumber: Dologite, D.G. Developing Knowledge-Based Systems Using VP-Expert. New York : Macmillan Publishing Company, 1993.

2.1.1.3. *Decision Table*

Decision table diperlukan untuk menunjukkan inter-relasi antara *value* dari hasil proses pengambilan keputusan *intermediate* maupun keputusan final sebuah *knowledge based system*.

Dalam membuat *decision table*, langkah pertama adalah dengan merencanakan jumlah baris yang diperlukan dalam tabel dengan cara mendaftar setiap faktor-faktor atau kondisi-kondisi yang pada *dependency diagram* dinotasikan dalam bentuk notasi segitiga. Setelah semua kondisi dan mendata *value* yang dimilikinya, jumlah baris yang diperlukan dapat diketahui. Jumlah baris yang diperlukan adalah hasil perkalian *value* yang ada dari setiap kondisi.

Contoh dari perencanaan *decision table* adalah sebagai berikut :

Kondisi :	Jumlah value
Member_status (ok,not_ok)	= 2
Reason (new_case, follow_up_case, information_other)	= 3
Problem (serious, non_serious)	= 2

Jumlah baris = 2 x 3 x 2 = 12 ←

Langkah kedua adalah membuat tabel, dimulai dengan membuat tabel kosong berisi nama kolom dan nomor baris. Nomor baris menjadi nomor *rule*

pada *knowledge base* dan nama kolom berisi nama kondisi. Perlu juga ditambahkan satu kolom pada tabel yang berisi kemungkinan-kemungkinan yang bisa terjadi dari kombinasi antar *value*.

Setelah itu setiap *value* diisikan ke dalam tabel, untuk membantu dapat dibuat garis pemisah. Pengisian *value* diatur sedemikian rupa sehingga *value* yang sama berkumpul pada paruh bagian tertentu. Pada kolom sebelahnya, *value* yang sama berkumpul dan berulang pada paruh bagian dari kolom sebelumnya, dan seterusnya sehingga setiap baris tidak ada kombinasi *value* yang sama.

Tabel 2.1. *Decision table* yang telah diisi dengan lengkap

Rule		Member Status	Reason	Problem	Concluding Recommendation for Support Level
A	1	ok	new_case	serious	level_1
A	2	ok	new_case	non_serious	level_2
A	3	ok	follow_up_case	serious	level_1
A	4	ok	follow_up_case	non_serious	level_3
A	5	ok	information_other	serious	information_other
A	6	ok	information_other	non_serious	information_other
A	7	not_ok	new_case	serious	non_member
A	8	not_ok	new_case	non_serious	non_member
A	9	not_ok	follow_up_case	serious	non_member
A	10	not_ok	follow_up_case	non_serious	non_member
A	11	not_ok	information_other	serious	non_member
A	12	not_ok	information_other	non_serious	non_member

Sumber: Dologite, D.G. Developing Knowledge-Based Systems Using VP-Expert. New York : Macmillan Publishing Company, 1993.

Untuk mengevaluasi, dapat diambil contoh pada baris pertama, dimana *rule* pertama dapat dibaca sebagai berikut :

‘ *If member status is ok and reason is a new_case and problem is serious then the person should be assigned level_1 support.*’

Langkah ketiga adalah mereduksi *decision table* tersebut dengan menggabungkan beberapa *rule* yang memiliki kesamaan pada kondisi dan hasil

kesimpulan akhirnya. Misalkan pada *rule* ke-7 sampai ke-12, dapat digabung menjadi satu *rule* saja, yaitu :

'*If member status is not_ok then the person should be assigned as non_member support'* .

Pada *rule* ke-7 sampai ke-12 tersebut, dapat dilihat bahwa hasil akhir dapat ditentukan hanya dari satu kondisi saja, yaitu *member_status*, tanpa memperdulikan kondisi yang lainnya.

Tabel 2.2. *Decision table* tereduksi

Rule	Member Status	Reason	Problem	Concluding Recommendation for Support Level
B 1	ok	new_case	serious	level_1
B 2	ok	new_case	non_serious	level_2
B 3	ok	follow_up_case	serious	level_1
B 4	ok	follow_up_case	non_serious	level_3
B 5	ok	information_other	-	information_other
B 6	not_ok	-	-	non_member

Sumber: Dologite, D.G. Developing Knowledge-Based Systems Using VP-Expert. New York : Macmillan Publishing Company, 1993.

Setelah terbentuk *decision table* yang tereduksi, dapat dilihat bahwa *decison table* tersebut memiliki pola yang sama seperti yang dinotasikan dalam *dependency diagram*.

2.1.1.4. *IF-THEN Rules*

Dari *decision table* yang telah tereduksi, setiap barisnya dapat dikonversikan menjadi *IF-THEN rule*. Setiap baris pada *decision table* tereduksi akan membentuk satu set *rule* final.

Struktur dan *syntax* penulisan *rule* adalah sebagai berikut :

- *RULE* label : label berisi nama *rule* tersebut.
- *IF* : Sebagai penanda awal kondisi pada sebuah *rule*.
- *THEN* : Sebagai penanda awal kesimpulan pada sebuah *rule*.
- *ELSE* : Sebagai penanda awal alternatif kesimpulan pada sebuah *rule*, bersifat opsional, jadi boleh tidak ada.

Operator yang dapat digunakan pada *IF-THEN rule* adalah :

- *AND* : Semua kondisi yang dihubungkan oleh operator ini harus bernilai benar, agar kondisi keseluruhan *rule* tersebut bernilai benar. Bila ada satu kondisi yang bernilai salah, keseluruhan *rule* tersebut bernilai salah.
- *OR* : Bila semua kondisi yang dihubungkan oleh operator ini harus bernilai salah, maka kondisi keseluruhan *rule* tersebut bernilai salah. Bila ada satu kondisi atau lebih yang bernilai benar, keseluruhan *rule* tersebut bernilai benar.

Sebagai contoh, diambil *rule* B1 pada *decison table* tereduksi diatas. Karena ada *rule* lain, yaitu *rule* B3 yang memiliki kesimpulan sama seperti *rule* B1, dapat digabungkan menjadi dalam satu *IF-THEN rule*, dengan menghubungkan kondisi *reason* dengan operator *or* menjadi seperti berikut ini :

RULE 1
IF member_status = ok and
reason = new_case or
reason = follow_up_case and
problem = serious
THEN support = level_1;

Bila kondisi dari sebuah *rule* adalah benar, maka kesimpulannya (klausa di belakang *THEN*) akan diambil sebagai kesimpulan baik kesimpulan *intermediate* atau kesimpulan final.

2.1.1.5. *User Interface*

User interface adalah apa yang ditampilkan di hadapan seorang *user*, jadi melalui *user interface*-lah, *user* dapat melakukan interaksi dengan sistem, yaitu dengan memasukkan *input* dan menerima *output*. Karena itu, perancangan *user interface* haruslah memenuhi prinsip *user friendly*, yaitu mudah digunakan oleh *user* dan sejelas mungkin agar tidak terjadi adanya salah interpretasi.

2.1.2. *Inference Engine*

Inference engine bertindak sebagai pengambil kesimpulan dan mengkontrol mekanisme dari *expert system*. Karena itu *inference engine* merupakan bagian terpenting dari *expert* s*ystem*, karena sangat berperan penting dalam menentukan efektivitas dan efisiensi *expert system*. *Inference engine* melakukan proses pengambilan kesimpulan (baik keputusan *intermediate* maupun keputusan final) berdasarkan suatu kumpulan *rules* tertentu, untuk sekumpulan fakta-fakta yang spesifik, pada suatu situasi tertentu.

Metode yang paling umum digunakan dalam pengambilan keputusan pada *expert* sy*stem* adalah modus ponen. Dengan teori dasar yang sederhana saja, yaitu apabila premisnya benar, maka kesimpulannya benar. Tetapi sebaliknya apabila kesimpulannya benar, maka premisnya belum tentu benar.

Biasanya modus tersebut dipresentasikan dalam notasi A → B.

Contoh :

If seseorang yang memiliki *shio* tikus,

Then rumah yang cocok adalah rumah yang menghadap selatan.

Hal tersebut menyatakan bila seseorang yang memiliki rumah yang menghadap selatan maka orang tersebut pasti memiliki *shio* tikus. Tetapi apabila dibalik, menjadi seperti berikut :

If seseorang yang memiliki rumah yang menghadap selatan,

Then orang tersebut memiliki *shio* tikus.

Pernyataan tersebut menjadi salah, karena ada *shio* lain yang cocok memiliki rumah yang menghadap selatan, seperti *shio* ular.

Inference engine juga bertugas untuk menentukan keputusan mana yang diambil dalam keadaan-keadaan tertentu seperti berikut ini :

- Bagaimana memulai proses pengambilan keputusan
- *Rule* mana yang akan dijadikan kesimpulan (*Fired*) apabila terdapat lebih dari satu *rule* yang dipenuhi (*Trigerred*).
- Langkah-langkah yang diambil ketika proses pengambilan keputusan sedang berlangsung.

Seperti halnya *knowledge base, inference engine* juga terdiri dari *rule-rule* dan fakta-fakta, bedanya apabila di dalam *knowledge base*, *rule-rule* dan

fakta-fakta yang ada berisi sebuah domain yang spesifik dari sifat-sifat seorang ahli. Sedangkan *rule-rule* dan fakta-fakta pada *inference eng*ine, berisi tentang kontrol secara umum dan strategi pencarian kesimpulan yang dilakukan oleh *expert system.* Kedua jenis *rule-rule* dan fakta-fakta ini disimpan secara terpisah dalam *expert system.*

Pemisahan kedua jenis *rule-rule* dan fakta-fakta tersebut memiliki beberapa keuntungan, yaitu pertama, memudahkan untuk melakukan perubahan dalam *knowledge base* tanpa berimbas banyak pada *inference engine*, dan sebaliknya. Kedua, berguna untuk pengembangan dan komponen-komponen lainnya yang terdapat pada *expert system* (terkecuali *knowledge base*).

Tujuan utama dari *expert system* adalah membentuk dan mengeluarkan saran (alternatif saran adalah beberapa saran yang telah diprogramkan sebelumnya). Untuk menyelesaikan tugas ini, *expert system* harus melakukan pencarian solusi yang mana adalah tanggung jawab dari *inference engine* untuk melakukannya secara efisien dan efektif. Seringkali dalam proses pencarian, *user* dihadapkan pada sejumlah alternatif (solusi yang potensial) dan berbagai jenis batasan (*constraints*).

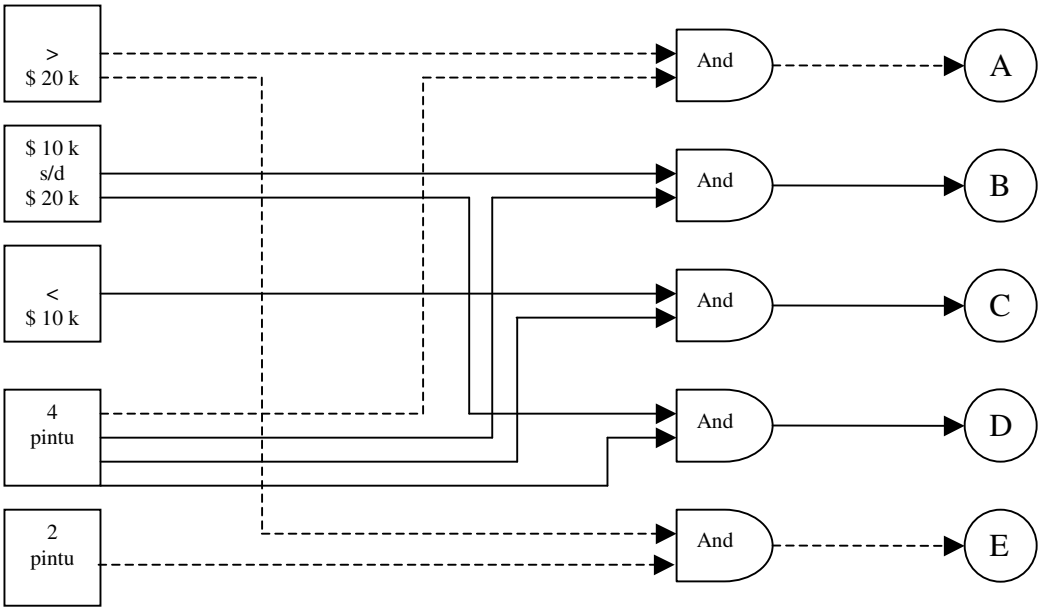
Contoh kasus :

Dalam mengambil keputusan untuk membeli sebuah mobil, akan dihadapkan pada batasan-batasan sebagai berikut :

- Batasan dana, harus membuat batasan harga mobil yang akan beli agar lebih murah atau paling tidak sama dengan sejumlah harga.
- Lokasi penjual mobil tersebut, apakah berada di dalam kota atau luar kota, untuk mempermudah mutasi mobil tersebut.
- Fasilitas mobil tersebut, disesuaikan dengan kebutuhan, apakah mobil tersebut untuk keluarga, angkutan dan lainnya.
- Waktu, pengambilan keputusan harus dilakukan secepatnya dalam suatu jangka waktu tertentu.

Batasan-batasan tersebut dapat dijadikan *filter* dalam memilih mobil yang akan dibeli dan dapat ditambahkan faktor-faktor lain yang merupakan kecenderungan dalam memilih suatu mobil, seperti merek, tahun pembuatan,

warna dan sebagainya. Cara melakukan pengambilan keputusan untuk membeli mobil dalam kasus di atas dapat dilakukan dengan metode pencarian *forward chaining* dengan eliminasi. Metode pencarian *forward chaining* akan dijelaskan lebih lanjut dalam sub-bab berikutnya. Memulai proses pencarian dengan menyeleksi tipe dan merk mobil, warna, fasilitas, harga yang disenangi menurut batasan yang telah dibuat sebelumnya. Sampai pada akhirnya diperoleh sejumlah kecil mobil yang potensial menjadi pilihan terakhir untuk dibeli. Dalam sudut pandang *expert system*, proses eliminasi untuk menyeleksi mobil tersebut mengurangi besarnya *inference network* yang bersangkutan. Hasilnya adalah pengurangan jumlah *search requirement* (syarat-syarat yang harus dipenuhi dalam proses pencarian).



Gambar 2.4. *Inference network* untuk pemilihan mobil
Sumber: Ignizio, James P. Introduction to Expert System : The Development and Implementation of Rule-Based Expert System. Singapore : McGraw-Hill, 1991.

Kesimpulan dari *inference network* pada Gambar 2.4. berupa lingkaran yang berada di sebelah kanan, yang mewakili pilihan mobil. Apabila ingin membatasi pilihan dengan mencari mobil yang harganya dibawah $20.000, dan berpintu 4, maka ukuran *inference network* dapat diperkecil dengan mengeliminasi alur yang berupa garis terputus-putus termasuk menghilangkan

pernyataan, kesimpulan dan hubungan logikal yang terhubung dengan alur tersebut.

Pemilihan mobil yang akan dibeli dapat juga dilakukan dengan pendekatan yang lain. Pertama-tama dipilih sebuah mobil dan ditentukan apakah mobil tersebut cocok dengan batasan-batasan yang telah ditentukan. Metode tersebut dinamakan metode *backward chaining*. Metode pencarian *backward chaining* juga akan dijelaskan lebih lanjut dalam sub-bab berikutnya.

Contohnya adalah apabila ingin membeli sebuah mobil Porsche Boxter, dilakukan pencocokkan dengan batasan yang dibuat, ternyata mobil tersebut memiliki kriteria yang tidak memenuhi batasan yang dibuat, karena mobil tersebut berharga lebih dari $20.000 dan berpintu dua. Kemudian dicari kemungkinan lainnya, yaitu mobil Porsche Boxter, dicocokkan dengan batasan yang telah dibuat, ternyata mobil tersebut juga memiliki kriteria yang tidak memenuhi batasan yang dibuat, karena mobil tersebut berharga lebih dari $20.000 meskipun mobil tersebut berpintu empat. Kemudian dicari kemungkinan lainnya, yaitu mobil Toyota Kijang, lalu dicocokkan dengan batasan yang dibuat, ternyata mobil tersebut memiliki kriteria yang memenuhi batasan yang dibuat, karena mobil tersebut berharga kurang dari $20.000 dan mobil tersebut berpintu empat. Maka mobil Toyota Kijang menjadi pilihan utama dalam membeli mobil. Meskipun demikian tidak tertutup kemungkinan untuk membeli Isuzu Panther, Mitsubishi Kuda dan beberapa mobil lainnya yang juga memiliki kriteria yang memenuhi batasan yang telah dibuat.

Apabila kasus di atas diselesaikan dengan metode *forward chaining*, maka akan dilakukan pencarian sebanyak mungkin informasi mengenai mobil yang dijual.

Contohnya adalah bila diterima informasi ada 6 buah mobil yang dijual, yaitu mobil Isuzu Panther, Toyota Kijang, Mitsubushi Kuda, Volkswagen, Porsche Boxter dan Mercedes Benz. Mula-mula ambil suatu kriteria yang diinginkan. Misalkan kriteria yang diambil adalah berpintu empat, maka dari keenam mobil tadi dilakukan eliminasi terhadap pilihan mobil yang berpintu dua sehingga pilihan mobil yang ada hanyalah mobil yang berpintu empat saja. Dengan melakukan eliminasi tadi, dapat Porsche Boxter dan Volkswagen

dihilangkan, sehingga pilihan menyempit menjadi antara Toyota Kijang, Isuzu Panther, Mitsubihsi Kuda dan Mercedes Benz. Kemudian diambil kriteria berikutnya, yaitu harganya tidak boleh melebihi $20.000. Kemudian kembali dilakukan eliminasi terhadap pilihan mobil yang berharga lebih dari $20.000 sehingga pilihan mobil yang ada hanyalah mobil yang berharga kurang dari $20.000 saja. Dengan melakukan eliminasi tadi, Mercedes Benz dapat dihilangkan, sehingga pilihan menyempit menjadi tinggal Toyota Kijang, Isuzu Panther dan Mitsubishi Kuda. Karena kriteria utama telah dipenuhi oleh ketiga mobil tersebut, maka boleh ditetapkan kriteria-kriteria lain yang tidak terlalu penting dan tergantung pada selera, misalnya, warna tahun pembuatan dan bahan bakar. Cara *forward chaining* tadi dapat juga diterapkan dengan urutan pemilihan kriteria yang berbeda, yaitu misalnya melakukan eliminasi terhadap mobil yang berharga lebih dari $20.000 terlebih dulu, setelah itu baru dilakukan eliminasi terhadap mobil yang tidak memiliki empat pintu. Maka hasil yang diperoleh akan tetap sama saja.

Dua strategi pencarian tersebut umum digunakan dalam *expert system*. Metode *forward chaining* berangkat dari kiri ke kanan, yaitu dari premis menuju kepada kesimpulan akhir, sering disebut *data driven* (yaitu, pencarian dikendalikan oleh data yang diberikan).

Metode *backward chaining* berangkat dari kanan ke kiri, yaitu dari dari sebuah kesimpulan sementara, mundur menuju premis awal, untuk menentukan apakah kesimpulan sementara tersebut didukung oleh data yang ada, sering disebut *goal driven* (yaitu, pencarian dikendalikan oleh tujuan yang diberikan). Kedua metode tersebut, baik metode *forward chaining* dan metode *backward chaining* memberikan hasil kesimpulan yang sama, tapi efisiensi dan efektifitas dari keduanya tergantung dari keadaan problem yang dihadapi, dan dari ukuran *inference network* dari problem yang bersangkutan.

Apabila memiliki banyak premis dan banyak kesimpulan, maka metode *forward chaining* lebih baik digunakan. Sedangkan apabila terdapat banyak premis dan sedikit kesimpulan, maka metode *backward chaining* lebih baik digunakan.

2.1.2.1. Metode *Forward Chaining*

Metode *forward chaining* adalah suatu metode pengambilan keputusan yang umum digunakan dalam *expert system*. Proses pencarian dengan metode *forward chaining,* berangkat dari kiri ke kanan, yaitu dari premis menuju kepada kesimpulan akhir, metode ini sering disebut *data driven* (yaitu pencarian dikendalikan oleh data yang diberikan).

Pada metode *forward chaining,* dapat dilakukan 2 cara dalam melakukan pencarian.

Pertama, dengan memasukkan semua data yang tersedia ke dalam *expert system* pada satu kesempatan dalam sesi konsultasi. Cara ini banyak berguna pada *expert system* yang termasuk dalam proses terautomatisasi dan menerima data langsung dari komputer yang menyimpan *database*, atau dari satu set sensor.

Kedua, dengan hanya memberikan elemen spesifik dari data yang diperoleh selama sesi konsultasi kepada *expert system*. Cara ini mengurangi jumlah data-data yang diminta, sehingga data-data yang diminta hanyalah data-data yang benar-benar dibutuhkan oleh *expert system* dalam mengambil kesimpulan.

Algoritma untuk metode *forward chaining* adalah sebagai berikut :

1. Inisialisasi. Dapat dibuat tiga tabel kosong, yaitu tabel *Working Memory*, tabel *Attribute Queue*, dan tabel *Rule/Premis Status*. Tabel *Working Memory* berguna untuk menyimpan setiap *input,* yaitu semua fakta yang disimpulkan selama proses konsultasi. Tabel *Attribute Queue* berguna untuk menyimpan semua atribut dari *value* yang sedang diperiksa. *Atribut* pada awal tabel adalah atribut yang sedang menjalani proses pemeriksaan. Tabel *Rule/Premis Status* menyimpan status dari *rule-rule* yang ada yaitu *Active, Marked, Unmarked, Discarded, Triggered, Fired*. Status setiap premis pada saat inisialisasi adalah bebas (*Free*). Status setiap *rule* pada saat inisialisasi adalah tidak bertanda (*Unmarked*) dan aktif (*Active*). Notasi yang digunakan untuk merepresentasikan keadaan suatu *rule* dan premis adalah sebagai berikut :

   A = *Active Rule*
   D = *Discarded Rule*
   U = *Unmarked Rule*

M = *Marked Rule*
TD = *Triggered Rule*
FD = *Fired Rule*
FR = *Free Clause*
FA = *False Clause*
TU = *True Clause*

2. Memulai proses pengambilan keputusan. Sebuah *value* dari sebuah atribut premis diambil, dimana atribut tersebut tidak boleh ada pada klausa kesimpulan. Atribut ini disimpan pada bagian teratas tabel *Attribute Queue*. Juga simpan atribut ini beserta *value*nya pada bagian terbawah tabel *Working Memory*.
3. Penelitian satu persatu *rule* yang ada untuk memeriksa ada tidaknya kesamaan. Periksa tabel *Rule/Premis Status*, jika tidak ada *rule* yang statusnya '*Active'*, pencarian dihentikan. Bila ada, dilakukan penelitian bagian klausa premis *rule* yang statusnya '*Active'* untuk mencocokkan klausa premis yang sesuai dengan *value* dari atribut pada bagian teratas tabel *Attribute Queue*. Simpan perubahan status klausa premis dari sekumpulan *rule* yang statusnya *Active.* Pada tabel *Rule/Premis Status* diberikan tanda FA (*False Clause*) pada status klausa premis yang bernilai salah dan tanda TU (*True Clause*) pada status klausa premis yang bernilai benar. Periksalah status *Rule* pada tabel *Rule/Premis Status.*
    a. Bila ada premis dari sebuah *rule* yang bernilai salah, maka diberi tanda D (*Discarded*) pada *rule* tersebut untuk menunjukkan bahwa *rule* tersebut bernilai salah dan tidak dipakai lagi. Langkah tersebut dilakukan pada setiap setiap *rule* yang memiliki premis yang bernilai salah.
    b. Bila ada sebuah *rule* yang semua premisnya dari bernilai benar, diberi tanda TD (*Triggered*) pada *rule status*. Kemudian dilanjutkan ke langkah 3c.
    c. Bila tidak ada *rule* yang statusnya TD (*Triggered*), dilanjutkan ke langkah ke 5, bila ada satu atau lebih *rule* yang statusnya TD (*Triggered*), dilanjutkan ke langkah ke 4.

4. *Rule firing*, atau menyatakan *rule* tersebut benar dan mengambil klausa kesimpulan *rule* tersebut sebagai kesimpulan akhir. Coretlah atribut pada bagian teratas tabel *Attribute Queue,* kemudian status *rule* tersebut diganti dan ditempatkan atribut kesimpulan pada bagian terbawah tabel *Attribute Queue* dan *value*-nya pada tabel *Working Memory.*
5. Status antrian. Bagian teratas tabel *Attribute Queue* dicoret .
6. Menandai *rule*. Telitilah kumpulan *rule active* untuk mencari rule yang statusnya U (*Unmarked*) dan A (*Active*). Bila tidak ditemukan, pencarian dihentikan. Bila ada, *rule* pertama yang ditemui ditandai dengan M (*Marked)*.
7. *Query*. Pada *rule* yang baru saja diberi tanda M (*Marked*), ditanyakan pada *user* untuk memperoleh *input*. Apabila *user* memberikan jawaban, dilanjutkan ke langkah 8. Sedangkan bila *user* tidak memberikan jawaban atau bila atribut *rule* yang ditanyakan tersebut tidak memerlukan jawaban dari *user*, dilanjutkan langkah ini pada setiap klausa premis pada *rule* yang diberi tanda M (*Marked*) tersebut. Apabila setiap klausa premis pada *rule* yang diberi tanda M (*Marked*) tersebut telah diperiksa, kembali ke langkah 6.
8. Menghilangkan tanda M (*Marked*) pada *rule*. atribut dan nomor *rule* diletakkan pada bagian teratas tabel *Attribute Queue*. Atribut ini diletakkan juga beserta *value*-nya pada bagian terbawah tabel *Working Memory*. pada *rule* yang baru saja diberi tanda M (*Marked)* diberikan tanda U (*Unmarked*), dan kembali ke langkah 3.

Untuk memperjelas algoritma *metode forward chaining*, dapat diambil contoh sebagai berikut ini :

Dalam menentukan jenis sebuah pesawat terbang yang disimpan pada suatu hangar, diperlukan *rule-rule* sebagai berikut :

Rule 1: *If engine type is propeller*
Then plane is C130

Rule 2: *If engine type is jet*
and wing position is low
Then plane is B747p

Rule 3: *If engine type is jet*
and wing position is high
and bulges are none
Then plane is C5A

Rule 4: *If engine type is jet*
and wing position is high
and bulges are aft wing
Then plane is C141

Berikut adalah contoh tabel yang disusun untuk menentukan jenis pesawat dari jenis mesin, posisi mesin dan tonjolan yang dimiliki berdasarkan contoh soal diatas.

Tabel 2.3. Inisialisasi tabel untuk metode *forward chaining*

Rule / Premise Table			
Rule Number	**Rule Status**	**Premise Clause Number**	**Premise Clause Status**
(1)	A, U	(1)-1	FR
(2)	A, U	(2)-1	FR
		(2)-2	FR
(3)	A, U	(3)-1	FR
		(3)-2	FR
		(3)-3	FR
(4)	A, U	(4)-1	FR
		(4)-2	FR
		(4)-3	FR
Attribute - Queue Table (attribute listing)		**Working Memory Table (attribute = value)**	

Sumber: Ignizio, James P. Introduction to Expert System : The Development and Implementation of Rule-Based Expert System. Singapore : McGraw-Hill, 1991.

Tabel 2.4. Memberikan sebuah *value* dari sebuah atribut premis.

Rule / Premise Table			
Rule Number	**Rule Status**	**Premise Clause Number**	**Premise Clause Status**
(1)	A, U	(1)-1	FR
(2)	A, U	(2)-1	FR
		(2)-2	FR
(3)	A, U	(3)-1	FR
		(3)-2	FR
		(3)-3	FR
(4)	A, U	(4)-1	FR
		(4)-2	FR
		(4)-3	FR
Attribute - Queue Table (attribute listing)		**Working Memory Table (attribute = value)**	
Engine Type		Engine Type =	jet

Sumber: Ignizio, James P. Introduction to Expert System : The Development and Implementation of Rule-Based Expert System. Singapore : McGraw-Hill, 1991.

Tabel 2.5. Memeriksa kebenaran dari klausa premis

Rule / Premise Table			
Rule Number	**Rule Status**	**Premise Clause Number**	**Premise Clause Status**
(1)	~~A~~, U, D	(1)-1	~~FR~~, FA
(2)	A, U	(2)-1	~~FR~~, TU
		(2)-2	FR
(3)	A, U	(3)-1	~~FR~~, TU
		(3)-2	FR
		(3)-3	FR
(4)	A, U	(4)-1	~~FR~~, TU
		(4)-2	FR
		(4)-3	FR
Attribute - Queue Table (attribute listing)		**Working Memory Table (attribute = value)**	
Engine Type		Engine Type =	jet

Sumber: Ignizio, James P. Introduction to Expert System : The Development and Implementation of Rule-Based Expert System. Singapore : McGraw-Hill, 1991.

Tabel 2.6. Memberikan tanda M (*Marked*) pada *rule* pertama yang statusnya U (*Unmarked*) dan A (*Active*).

Rule / Premise Table			
Rule Number	**Rule Status**	**Premise Clause Number**	**Premise Clause Status**
(1)	~~A~~, U, D	(1)-1	~~FR~~, FA
(2)	A, ~~U~~, M	(2)-1	~~FR~~, TU
		(2)-2	FR
(3)	A, U	(3)-1	~~FR~~, TU
		(3)-2	FR
		(3)-3	FR
(4)	A, U	(4)-1	~~FR~~, TU
		(4)-2	FR
		(4)-3	FR

Attribute - Queue Table (attribute listing)		Working Memory Table (attribute = value)	
~~Engine Type~~		Engine Type =	jet

Sumber: Ignizio, James P. Introduction to Expert System : The Development and Implementation of Rule-Based Expert System. Singapore : McGraw-Hill, 1991.

Tabel 2.7. Menghilangkan tanda M (*Marked*) pada *rule* dan kembali ke langkah ke 3.

Rule / Premise Table			
Rule Number	**Rule Status**	**Premise Clause Number**	**Premise Clause Status**
(1)	~~A~~, U, D	(1)-1	~~FR~~, FA
(2)	~~A~~, ~~U~~, ~~M~~, U, D	(2)-1	~~FR~~, TU
		(2)-2	~~FR~~, FA
(3)	A, U	(3)-1	~~FR~~, TU
		(3)-2	~~FR~~, TU
		(3)-3	FR
(4)	A, U	(4)-1	~~FR~~, TU
		(4)-2	~~FR~~, TU
		(4)-3	FR

Attribute - Queue Table (attribute listing)		Working Memory Table (attribute = value)	
~~Engine Type~~		Engine Type =	jet
Wing Position (rule 2)		Wing Position =	high

Sumber: Ignizio, James P. Introduction to Expert System : The Development and Implementation of Rule-Based Expert System. Singapore : McGraw-Hill, 1991.

Tabel 2.8. Mengulangi proses untuk atribut kedua.

Rule / Premise Table			
Rule Number	**Rule Status**	**Premise Clause Number**	**Premise Clause Status**
(1)	~~A~~, U, D	(1)-1	~~FR~~, FA
(2)	~~A~~, ~~U~~, ~~M~~, U, D	(2)-1	~~FR~~, TU
		(2)-2	~~FR~~, FA
(3)	A, ~~U~~, M	(3)-1	~~FR~~, TU
		(3)-2	~~FR~~, TU
		(3)-3	FR
(4)	A, U	(4)-1	~~FR~~, TU
		(4)-2	~~FR~~, TU
		(4)-3	FR
Attribute - Queue Table (attribute listing)		**Working Memory Table (attribute = value)**	
~~Engine Type~~		Engine Type =	jet
~~Wing Position (rule 2)~~		Wing Position =	high

Sumber: Ignizio, James P. Introduction to Expert System : The Development and Implementation of Rule-Based Expert System. Singapore : McGraw-Hill, 1991.

Tabel 2.9. Memulai pemeriksaan untuk atribut ketiga.

Rule / Premise Table			
Rule Number	**Rule Status**	**Premise Clause Number**	**Premise Clause Status**
(1)	~~A~~, U, D	(1)-1	~~FR~~, FA
(2)	~~A~~, ~~U~~, ~~M~~, U, D	(2)-1	~~FR~~, TU
		(2)-2	~~FR~~, FA
(3)	A, ~~U~~, ~~M~~, U	(3)-1	~~FR~~, TU
		(3)-2	~~FR~~, TU
		(3)-3	FR
(4)	A, U	(4)-1	~~FR~~, TU
		(4)-2	~~FR~~, TU
		(4)-3	FR
Attribute - Queue Table (attribute listing)		**Working Memory Table (attribute = value)**	
~~Engine Type~~		Engine Type =	jet
~~Wing Position (rule 2)~~		Wing Position =	high
Bulges (rule 3)		Bulges =	None

Sumber: Ignizio, James P. Introduction to Expert System : The Development and Implementation of Rule-Based Expert System. Singapore : McGraw-Hill, 1991.

Tabel 2.10. Pada *rule* 3, statusnya TD (*Triggered*) bila status semua klausa premisnya adalah benar, TU (*True Clause*)

Rule / Premise Table			
Rule Number	**Rule Status**	**Premise Clause Number**	**Premise Clause Status**
(1)	~~A~~, U, D	(1)-1	~~FR~~, FA
(2)	~~A~~, ~~U~~, ~~M~~, U, D	(2)-1	~~FR~~, TU
		(2)-2	~~FR~~, FA
(3)	A, ~~U~~, ~~M~~, U, TD	(3)-1	~~FR~~, TU
		(3)-2	~~FR~~, TU
		(3)-3	~~FR~~, TU
(4)	~~A~~, U, D	(4)-1	~~FR~~, TU
		(4)-2	~~FR~~, TU
		(4)-3	~~FR~~, FA

Attribute - Queue Table (attribute listing)		Working Memory Table (attribute = value)	
~~Engine Type~~		Engine Type =	jet
~~Wing Position (rule 2)~~		Wing Position =	high
Bulges (rule 3)		Bulges =	none
Plane (rule 3)			

Sumber: Ignizio, James P. Introduction to Expert System : The Development and Implementation of Rule-Based Expert System. Singapore : McGraw-Hill, 1991.

Tabel 2.11. Bentuk final sebuah tabel yang telah memberikan kesimpulan akhir.

Rule / Premise Table			
Rule Number	**Rule Status**	**Premise Clause Number**	**Premise Clause Status**
(1)	~~A~~, U, D	(1)-1	~~FR~~, FA
(2)	~~A~~, ~~U~~, ~~M~~, U, D	(2)-1	~~FR~~, TU
		(2)-2	~~FR~~, FA
(3)	~~A~~, ~~U~~, ~~M~~, U, ~~TD~~, FD	(3)-1	~~FR~~, TU
		(3)-2	~~FR~~, TU
		(3)-3	~~FR~~, TU
(4)	~~A~~, U, D	(4)-1	~~FR~~, TU
		(4)-2	~~FR~~, TU
		(4)-3	~~FR~~, FA

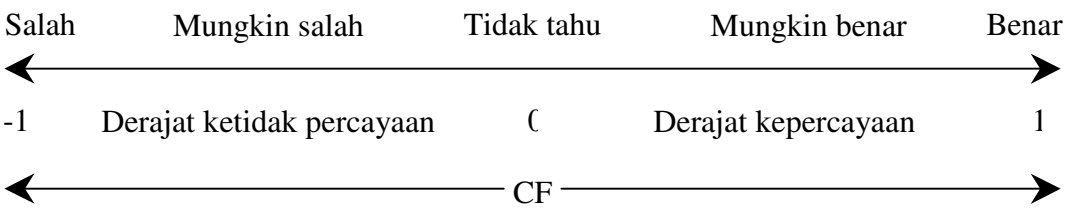
Attribute - Queue Table (attribute listing)		Working Memory Table (attribute = value)	
~~Engine Type~~		Engine Type =	jet
~~Wing Position (rule 2)~~		Wing Position =	high
~~Bulges (rule 3)~~		Bulges =	none
Plane (rule 3)		Planes =	C5A

Sumber: Ignizio, James P. Introduction to Expert System : The Development and Implementation of Rule-Based Expert System. Singapore : McGraw-Hill, 1991.

2.1.2.2. *Certainty Factor*

Pakar sering kali membuat keputusan saat memecahkan suatu permasalahan. Informasi dari suatu permasalahan bisa dugaan atau tidak lengkap, dan beberapa *knowledge* untuk menafsirkan informasi itu mungkin tidak dapat dipercaya. Namun pakar telah belajar untuk beradaptasi dengan situasi dan terus mempertimbangkan suatu permasalahan dengan cerdas. Situasi membuat pengembang sistem pakar mencari teknik untuk mengelola pertimbangan yang tidak tepat. Contoh pertanyaan berikut ini adalah mengenai permasalahan *feng-shui*: Apakah calon pembeli cocok membeli rumah yang menghadap timur? Bagaimana kita mengharapkan *user* menjawab pertanyaan ini? Dalam hal ini *user* tidak cukup hanya dengan menjawab benar atau salah, karena pertanyaan itu bersifat subyektif dan memerlukan *user* untuk membuat keputusan.

Kita memilih untuk mengijinkan *user* memberikan angka untuk menjawab, angka itu merefleksikan tingkat keyakinan dari jawabannya. Contohnya kita menggunakan standar skala probabilitas dari 0 sampai 1. Untuk keterangan yang tidak pasti menggunakan *certainty factor* (CF) untuk menggambarkan tingkat keyakinan terhadap permasalahan yang sedang dihadapi. Angka yang diberikan dalam jangkauan antara -1 (pasti tidak benar) sampai +1 (pasti benar). Nilai positif menggambarkan derajat kepercayaan, sedangkan nilai negatif menyatakan derajat ketidak percayaan. Contoh: Ahli *feng-shui* menunjukkan bahwa beberapa keterangan mungkin benar, maka ahli *feng-shui* memberikan nilai CF=0.7. Jangkauan nilai CF dapat dilihat pada gambar 2.5.



Gambar 2.5. Jangkauan nilai CF

Sumber: John Durkin, *Expert System Design And Development*, (Prentice Hall, New Jersey, 1994), p. 339

Berikut ini adalah beberapa *certainty factor*:

a.) CF untuk *Single Premise Rules*

IF E *THEN* H CF(*RULE*)

CF(H, E) = CF(E) * CF(*RULE*)

Contoh:

IF Ada awan gelap E

THEN Hujan akan turun H

(CF = 0.8)

Diasumsikan CF(E) = 0.5

maka $\Rightarrow$ CF(H,E) = 0.5 * 0.8 = 0.4

"Hujan mungkin akan turun"

b.) CF untuk *Multiple Premise Rules*

- *Conjunctive Rules*

  IF E_1 *AND* E_2 *AND* ... *THEN* H CF(*RULE*)

  CF(H, E_1 *AND* E_2 *AND* ...) = *min*{CF(E_1|E')} * CF(*RULE*)

  Contoh:

  IF Langit gelap

  AND Angin bertiup kencang

  THEN Hujan akan turun

  (CF = 0.8)

  Diasumsikan CF(Langit gelap) = 1 dan CF(Angin bertiup kencang) = 0.7

  Maka $\Rightarrow$ CF(Hujan akan turun) = *min*{1, 0.7} * 0.8 = 0.56

  "Hujan mungkin akan turun"

- *Disjunctive Rules*

  IF E_1 *OR* E_2 *OR* ... *THEN* H CF(*RULE*)

  CF(H, E_1 *OR* E_2 *OR* ...) = *max*{CF(E_1|E')} * CF(*RULE*)

  Contoh:

  IF Langit gelap

  OR Angin bertiup kencang

THEN Hujan akan turun

(CF = 0.9)

Diasumsikan CF(Langit gelap) = 1 dan CF(Angin bertiup kencang) = 0.7

Maka $\Rightarrow$ CF(Hujan akan turun) = $max\{1, 0.7\} * 0.9 = 0.9$

"Hujan kemungkinan besar akan turun"

2.2 *Feng-shui*

Feng-shui berasal dari dua kata yaitu *feng* yang berarti angin dan *shui* yang berarti air. *Feng-shui* adalah ilmu untuk hidup dalam harmoni dengan tanah yang ditempati oleh seseorang agar dengan demikian dapat memperoleh sebesar-besarnya dari sumber vital pertanahan dan juga menciptakan kedamaian dan kesejahteraan (Skinner, 2002, p.8). *Feng-shui* merupakan persilangan antara seni dan ilmu pengetahuan. Tujuannya adalah untuk mengatur bangunan-bangunan, ruangan-ruangan, dan perabotan rumah dengan cara yang paling menguntungkan guna mencapai keselarasan yang maksimum (Rossbach, 1994, p.1). *Feng-shui* menurut Lilian Too adalah seni hidup menuju keharmonisan dengan alam. Melalui penataan yang sesuai dengan aturan *feng-shui* seseorang akan mendapat keberuntungan, ketenangan, dan kemakmuran (Lilian Too, 1995, p.21). Sedangkan Lin Yun menanggapi bahwa *feng-shui* adalah bagaimana keadaan lingkungan sekitar mempengaruhi hidup kita (Lin Yun, 2000, p.15).

2.2.1 Metode dalam *Feng-shui*

Menurut Jenie Kumala Dewi dan Herman Wilianto, *feng-shui* secara umum bisa dipelajari dengan menggunakan dua metode pendekatan, yaitu pendekatan indrawi dan pendekatan rumusan (2003, p.14).

a. Pendekatan Indrawi

Pendekatan dalam metode indrawi lebih menekankan pada orientasi pengamatan secara fisik melalui pandangan manusia. Metode ini dikenal sebagai *feng-shui* metode bentuk. Metode bentuk ini biasa digunakan untuk mencari lokasi atau *site* untuk rumah. Metode ini berupaya untuk mencari lahan yang baik berdasarkan kondisi alam

dengan keberadaan gunung, sungai, jalan, dan elemen eksterior lainnya.

b. Pendekatan Rumusan

Pendekatan metode *feng-shui* yang lain adalah pendekatan rumusan yaitu pendekatan *feng-shui* yang lebih menekankan pada kecermatan penghitungan dimensi-dimensi *feng-shui* yang harus diukur.

2.2.2 Faktor-faktor yang Berpengaruh Pada *Feng-shui*

Menurut Herlianto (1996, p.19), *feng-shui* tentang keberuntungan manusia dipengaruhi oleh beberapa faktor penentu yaitu :

a. Perputaran matahari, bulan, dan bintang

Dikaitkan dengan sifat-sifat binatang dalam *horoskop* China yang biasa disebut *shio*, dilambangkan dengan 12 macam binatang ditentukan dari tahun kelahiran.

b. Unsur-unsur alam

Unsur-unsur alam ini seperti kayu, api, tanah, logam, dan air.

c. Tata lingkungan alam atau *Topografy*

Biasa dikaitkan dengan empat macam bentuk *kontur* alam yang dikaitkan dengan sifat empat macam binatang.

d. Tata lingkungan buatan

Lingkungan buatan yang berkaitan seperti monumen untuk orang yang telah meninggal dunia, meriam, air mancur, kolam.

e. Tata letak bangunan

Tata letak bangunan yang berorientasi terhadap kompas.

f. Bentuk tanah

Menurut *feng-shui*, bentuk tanah yang mendatangkan keberuntungan bagi penghuninya yakni yang berbentuk menyerupai gentong, persegi atau bujursangkar, persegi panjang, lingkaran, jajaran genjang, setengah lingkaran depan, dan berlian. Ada juga bentuk tanah yang kurang baik seperti bentuk segitiga, punuk unta, bentuk L dan T.

g. Angka atau *Numerology*

Banyak kata-kata China ketika dilafalkan memiliki bunyi yang mirip. Ada beberapa angka yang memiliki bunyi pelafalan yang memiliki arti baik seperti angka 3,6,8,9 yang dianggap sebagai angka keberuntungan, sementara angka 4 merupakan angka yang dianggap sial.

2.2.3 Konsep-konsep Dasar *Feng-shui*

Menurut Lin Yun (2000, p.19-29), konsep-konsep dasar *feng-shui* adalah :

a. *Chi*

Dalam ajaran *feng-shui*, hal yang paling diutamakan adalah memelihara siklus *chi* tetap utuh dan mengalir dengan lancar. *Chi* adalah energi daya hidup yang membantu keberadaan manusia. *Chi* merupakan energi atau kekuatan yang menciptakan pegunungan dan gunung-gunung berapi, mengarahkan aliran-aliran air dan sungai-sungai, dan menentukan warna serta bentuk pohon-pohon dan tumbuhan-tumbuhan. *Chi* adalah energi *elektromagnetik* yang tidak terlihat yang menghubungkan semua hal di alam semesta. Jadi, *chi* adalah energi yang dimiliki manusia dan alam yang keduanya harus seimbang dan harmonis supaya saling menguntungkan. Hal tersebut sesuai dengan tujuan dari *feng-shui.*

b. *Tao* dan *Yin-Yang*

Tao adalah sebuah proses dan prinsip yang menghubungkan menusia dengan alam semesta. *Tao* memiliki arti sebagai jalan atau cara. *Tao* adalah memikirkan suatu cara ilmiah agar alam dapat bersanding dengan jalan hidup manusia. Dari *tao* lahir prinsip *yin-yang*. *Yin-yang* mutlak berlawanan dan keduanya sama-sama mempengaruhi segala aspek kehidupan. Orang China menghubungkan langit, bumi, dan manusia melalui *tao* serta membaginya ke dalam *dualisme* yang saling melengkapi satu sama lain. Konsep *yin-yang* menghubungkan manusia dengan linkungan.

c. Lima unsur atau lima elemen

Feng-shui mengenal lima unsur di bumi yaitu kayu, api, tanah, logam, dan air. Unsur-unsur tersebut digabung dengan warna dan arah mata angin, yaitu :

1) Kayu dilambangkan dengan warna hijau dan arah timur.
2) Api dilambangkan dengan warna merah yang merupakan warna kemujuran dan arah selatan.
3) Tanah dilambangkan dengan warna kuning sebagai pusat arah mata angin.
4) Logam dilambangkan dengan warna putih atau warna emas, warna kesucian dan arah barat.
5) Air dilambangkan dengan warna hitam dan arah utara.

Kelima unsur tersebut dapat bekerja sama dalam hubungan siklus yang produktif dan saling mendukung. Siklus produktif mengarah pada harmoni, kemakmuran, dan kebahagiaan.

2.2.4 Harga

2.2.4.1 Pengertian Harga

Dalam konteks ekonomi, biasanya harga dipandang sebagai jumlah uang yang harus dikeluarkan untuk mendapatkan sesuatu. Menurut Swastha (1984) definisi harga adalah jumlah uang (ditambah dengan beberapa barang) yang dibutuhkan untuk mendapatkan sejumlah kombinasi dari barang beserta pelayanannya (p.147).

Harga terdapat di sekeliling kita dan dalam kehidupan sehari-hari saat akan dibelinya suatu barang atau jasa. Seperti yang dikatakan Kotler (1987), harga adalah jumlah uang yang harus konsumen bayar untuk mendapatkan sebuah barang (p.64). jadi dapat disimpulkan dari kedua pengertian yang sudah ada bahwa harga adalah sejumlah uang yang harus dikorbankan oleh setiap individu untuk memperoleh barang atau jasa yang disediakan oleh penjual. Dengan demikian, dalam menetapkan harga rumah dipertimbangkan kualitas bahan yang digunakan, fasilitas umum yang ada di dekat rumah, infrastruktur pendukung, lingkungan sekitar,

keuntungan di masa yang akan datang, dan juga hal-hal lain yang mendukung seperti faktor-faktor *feng-shui* yang turut mempengaruhi.

2.2.4.2 Penetapan Harga

Menurut Swastha, terdapat tiga kemungkinan penetapan harga :

1. Penetapan harga diatas pesaing

   Cara ini dapat dilakukan agar dapat meyakinkan konsumen bahwa barang yang dijual mempunyai kualitas yang lebih baik, bentuk yang lebih menarik,dan mempunyai kelebihan-kelebihan lain dari barang sejenis yang dijual di pasaran.

2. Penetapan harga dibawah pesaing

   Kebijakan ini dipilih untuk menarik lebih banyak konsumen untuk barang yang baru diperkenalkan kepada masyarakat dan belum stabil kedudukannya di pasaran.

3. Mengikuti harga pesaing

   Cara ini dipilih untuk mempertahankan agar konsumen tidak beralih ke barang lain.

Metode penetapan harga jual rumah menggunakan rumus regresi linier berganda dengan rumus : **Y = a + b1 x X1 + b2 x X2 + ...+ bi x Xi**

Dimana **Y** = variabel *dependent* (variabel yang nilainya dipengaruhi oleh X)

a = nilai konstanta harga Y jika X = 0.

b = nilai penentu ramalan (prediksi) yang menunjukkan nilai peningkatan (+) atau penurunan (-) variabel Y

X = variabel bebas yang mempunyai nilai tertentu untuk diprediksikan.

2.2.5 Hubungan Antar Konsep

Masyarakat keturunan Tionghoa percaya bahwa rumah yang ditempati dapat memberikan pengaruh pada penghuni rumah, baik itu berpengaruh baik maupun buruk. Oleh karena itu, banyak orang menggunakan ilmu pengetahuan tentang *feng-shui* sebelum melakukan transaksi pembelian rumah. Untuk meningkatkan penjualan rumah, *developer* juga menerapkan *feng-shui* pada setiap rumah yang dibangun dan akan dijual, seperti

peniadaan angka 4 dan 13 pada nomor urut rumah. Oleh karena ini Tugas Akhir ini bertujuan untuk memprediksi harga jual sebuah rumah berdasarkan faktor-faktor *feng-shui* yang ada pada rumah tersebut dengan si pembeli rumah, selain luas rumah yang digunakan sebagai acuan harga dasar jual.