SISTEM BERBASIS ATURAN UNTUK PERHITUNGAN LEMBUR KARYAWAN

(STUDI KASUS DI PT. YUNIKO PLASTIK INDONESIA)

Oleh:

Nama : Yoko

NIM : 00.41010.0042

Program : S1 (Strata Satu)

Jurusan : Sistem Informasi

SEKOLAH TINGGI

MANAJEMEN INFORMATIKA & TEKNIK KOMPUTER

SURABAYA

2006

SISTEM BERBASIS ATURAN UNTUK PERHITUNGAN LEMBUR KARYAWAN

(STUDI KASUS DI PT. YUNIKO PLASTIK INDONESIA)

SKRIPSI

Diajukan sebagai salah satu syarat untuk menyelesaikan

Program Sarjana Komputer

Oleh:

Nama : Yoko

NIM : 00.41010.0042

Program : S1 (Strata Satu)

Jurusan : Sistem Informasi

SEKOLAH TINGGI

MANAJEMEN INFORMATIKA & TEKNIK KOMPUTER

SURABAYA

2006

SISTEM BERBASIS ATURAN UNTUK PERHITUNGAN LEMBUR KARYAWAN

(STUDI KASUS DI PT. YUNIKO PLASTIK INDONESIA)

Disusun Oleh :

Nama : Yoko

NIM : 00.41010.0042

Surabaya, September 2006

Telah diperiksa, diuji dan disetujui:

Pembimbing I	Pembimbing II
Akhmad Saikhu,S.Si,MT	Titik Lusiani, M.Kom
NIDN 0718077101	NIDN 074077401

Mengetahui:

Wakil Ketua Bidang Akademik

Drs. Antok Supryanto, M. MT
NIDN 0726106201

SEKOLAH TINGGI MANAJEMEN INFORMATIKA & TEKNIK KOMPUTER SURABAYA

Mengirim bulu angsa sepanjang 1000km

Mendapat hadiah janganlah dilihat dari

nilainya tetapi lihatlah dari

pengorbanannya

Ku persembahkan kepada

Papa & Mama tercinta

Shinta terkasih

Emma tersayang

ABSTRAKSI

Perhitungan lembur pada umumnya di setiap perusahaan masih secara manual, perhitungan tersebut berdasarkan absensi yang dilakukan oleh karyawan secara manual juga setiap harinya, yaitu dengan menggunakan mesin ceklok. Hal ini menyebabkan banyak sekali kesalahan dalam perhitungan lembur karyawan.

Dalam perkembangan ilmu komputer, *Artificial Intelelligence* (AI) menjadi salah satu hal yang diperhatikan oleh para ahli. AI adalah studi khusus dimana tujuannya untuk membuat komputer berpikir dan bertindak seperti manusia.

Salah satu contoh pengembangan AI dalam ilmu komputer adalah sistem berbasis aturan (*rule base system*). Di dalam penyusunan sistem berbasis aturan terdapat *rule-rule* yang akan digunakan untuk menghasilkan suatu konklusi

Dengan membuat suatu sistem untuk perhitungan lembur karyawan. Diharapkan sistem yang baru ini pihak perusahaan dapat melakukan perhitungan lembur karyawannya dengan lebih akurat dan effisien sehingga dapat mencegah kerugian bagi perusahaan.

Pada implementasi program/aplikasi ini dapat melakukan perhitungan lembur dengan cepat dan lebih akurat berdasarkan *rule-rule* yang telah digunakan. Dengan adanya penggunaan aplikasi perhitungan lembur menggunakan sistem berbasis aturan dengan metode *forward chaining* ini maka perhitungan lembur menjadi jauh lebih cepat dan terhindar dari kesalahan.

Kata Kunci: *Sistem Berbasis Aturan, Perhitungan Lembur*

KATA PENGANTAR

Puji syukur kepada Tuhan YME atas segala rahmat-Nya, sehingga penulis dapat menyelesaikan Tugas Akhir tentang perhitungan lembur mengunakan sistem berbasis aturan yang merupakan syarat untuk menyelesaikan Program Studi Strata Satu di Sekolah Tinggi Manajemen Informatika & Teknik Komputer Surabaya (STIKOM).

Banyak pihak yang telah membantu penulis dalam menyelesaikan Tugas Akhir ini. Pada kesempatan ini penulis ingin mengucapkan terima kasih kepada:

1. Ibu Titik dan Bapak Saikhu sebagai Dosen Pembimbing atas segala bimbingannya.
2. Heri yang telah membantu dalam pembuatan progam.
3. Mama dan Papa yang menyemangatiku.
4. Kekasihku tercinta Shinta yang bandel, tapi baik hati.

Semoga Tuhan YME memberi pahala kepada semua yang telah membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari penulisan Tugas Akhir ini masih jauh dari sempurna. Dengan senang hati penulis akan menerima segala kritik dan saran yang membangun bagi Tugas Akhir ini. Semoga Tugas Akhir ini berguna bagi pihak yang membutuhkan, terutama buat perkembangan Ilmu, khusunya bidang komputer.

Surabaya, September 2006

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

Halaman

DAFTAR LAMPIRAN

Halaman

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Perhitungan lembur karyawan merupakan hal penting bagi perusahaan sebagai contoh PT. YUNIKO PLASTIK INDONESIA yang selama ini menggunakan sistem pencatatan absen *manual* (mesin ceklok). Perhitungan lembur pada perusahaan ini masih sering mengalami kesalahan dalam perhitungan lembur karyawan. Dengan sistem manual proses absensi menjadi tidak effisien kadang terdapat kesalahan tulis, sering juga terjadi kasus kehilangan kartu absen.

Oleh karena itu dibutuhkan suatu sistem yang dapat melakukan pencatatan absensi dan perhitungan lembur yang lebih baik sehingga kesalahan-kesalahan yang dapat merugikan perusahaan dapat diperbaiki.

Berdasarkan permasalahan di atas maka dibuat sistem berbasis aturan untuk perhitungan lembur, dengan memanfaatkan data dan *rule* yang ada. Sistem ini dapat melakukan pengambilan keputusan yang akurat untuk perhitungan lembur karyawan.

Dengan implementasi sistem berbasis aturan ini maka diharapkan dapat meningkatkan kinerja perusahaan. Sistem ini mempunyai proses verifikasi, sehingga *rule* dinyatakan benar, sesuai aturan. Dengan menggunakan sistem ini maka diharapkan dapat melakukan perhitungan lembur dengan effisien dan efektif.

1.2. Perumusan Masalah

Berdasarkan latar belakang di atas, dapat dirumuskan masalah yaitu bagaimana membuat suatu sistem berbasis aturan yang dapat melakukan perhitungan lembur dengan effisien dan efektif.

1.3. Pembatasan Masalah

Batasan masalah dari pembuatan sistem ini adalah sebagai berikut:

1. Data yang digunakan dalam sistem ini mempunyai periode jangka waktu satu tahun, yaitu periode tahun 2005.
2. Metode yang digunakan dalam mencari uang lembur adalah *forward chaining*.
3. Proses verifikasi untuk *rule* yang digunakan adalah:
   a. *Redundant Rules* yaitu jika dua *rule* atau lebih mempunyai *premise* dan *conclusion* yang sama.
   b. *Conflicting Rules* yaitu jika dua *rule* atau lebih mempunyai *premise* yang sama, tetapi mempunyai *conclusion* yang berlawanan.
   c. *Subsumed Rules* yaitu jika dua *rule* atau lebih salah satu *rule* mempunyai *premise* lebih banyak, tetapi mempunyai *conclusion* yang sama.
   d. *Circular Rules* yaitu suatu keadaan dimana terjadinya proses perulangan dari suatu *rule*. Hal ini dikarenakan suatu *premise* dari salah satu *rule* merupakan *conclusion* dari *rule* yang lain, atau kebalikannya.
4. Parameter yang digunakan untuk perhitungan lembur antara lain:
   a. Jenis Lembur yang dibedakan berdasarkan status karyawan, yaitu karyawan tetap atau kontrak, untuk karyawan tetap mempunyai upah lembur yang berbeda, tergantung dari besarnya gaji pokok. Upah lembur karyawan kontrak ditentukan oleh perusahaan berdasarkan dari

perhitungan UMR yang telah ditetapkan oleh Departemen Tenaga Kerja Dan Transmigrasi (KEPMEN NO. 102 TH 2004).

b. Status yang dimiliki mempengaruhi besarnya uang lembur, status dibedakan menjadi 2, pertama pegawai tetap terdiri dari manager, kepala produksi, manager, teknisi dan sopir. Bagian bahan dan produksi termasuk dalam pegawai kontrak.

1.4. Tujuan

Tujuan dari sistem ini adalah membuat suatu sistem berbasis aturan yang dapat melakukan proses perhitungan lembur karyawan yang efektif dan effisien.

1.5. Sistematika Penulisan

Sistematika yang digunakan dalam pembuatan laporan ini terdiri dari beberapa *bab* dan *subbab* seperti dibawah ini.

BAB I : PENDAHULUAN

Bab ini membahas latar belakang masalah, perumusan masalah, batasan masalah, tujuan serta sistematika penulisan.

BAB II : LANDASAN TEORI

Bab ini membahas teori yang dipergunakan dalam membantu memecahkan masalah serta teori ilmu yang terkait.

BAB III : PERANCANGAN SISTEM

Bab ini berisi penjelasan tentang perancangan aturan upah lembur yang berisi:

1. Perancangan Diagram Blok

2. Perancangan Diagram Ketergantungan
3. Perancangan *Decicion Table*
4. Perancangan Reduksi
5. Perancangan *Rule Base*

Setelah perancangan aturan terdapat desain arsitektur dan perancangan proses yang menampilkan diagram alir, dan yag terakhir terdapat desain struktur tabel.

BAB IV : IMPLEMENTASI DAN EVALUASI

Bab ini berisi penjelasan tentang implementasi dan evaluasi dari sistem yang dibuat, berupa gambar proses aplikasi yang terjadi pada sistem.

BAB V : PENUTUP

Bab ini berisikan kesimpulan dan saran yang diambil sesuai dengan hasil pembahasan. Kesimpulan merupakan rangkuman singkat dari hasil seluruh pembahasan masalah dan saran berisi mengenai harapan dan kemungkinan lebih lanjut dari hasil pembahasan masalah.

BAB II

LANDASAN TEORI

Adapun landasan teori atau kajian pustaka dalam membangun sistem ini antara lain:

2.1 Sistem Penggajian PT. YUNIKO PLASTIK INDONESIA

Berdasarkan hasil survey sistem penggajian pada perusahaan ini sampai sekarang masih menggunakan sistem ceklok. Tiap karyawan tetap memiliki perhitungan yang berbeda untuk banyaknya upah lembur yang diterima per jamnya. Besarnya upah lembur ditentukan oleh gaji pokok yang diterima dibagi dengan parameter yang telah ditentukan. Akibat perbedaan gaji pokok pada karyawan tetap maka sering terjadi kesalahan dalam perhitungan gaji karyawan.

Dibutuhkan suatu sistem yang dapat menunjang kebutuhan perusahaan dalam mengatur absensi serta melakukan perhitungan lembur yang cepat dan akurat.

2.2 Lembur

Upah lembur adalah upah yang diberikan perusahaan kepada karyawan yang telah melakukan pekerjaan di luar jam kerja. Semakin besar kuantitas pekerjaan yang dimiliki oleh suatu perusahaan maka jam kerja yang dibutuhkan oleh seorang karyawan akan semakin bertambah, dan untuk memenuhi targetnya seorang karyawan akan melakukan lembur kerja. Lembur yang dilakukan karyawan ini tentu akan diberikan kompensasi oleh perusahaan yaitu berupa upah lembur.

2.3 Sistem Berbasis Aturan

Sistem berbasis aturan adalah sebuah sistem komputer yang dapat membantu manusia dalam memecahkan suatu permasalahan. Sistem ini biasanya digunakan untuk mediagnosa masalah dan memecahkan sebuah permasalahan serta mengambil suatu keputusan.

Sistem berbasis aturan merupakan pengembangan dari sistem berbasis pengetahuan (*Knowlegde Base System*) yang merupakan bagian dari sistem pakar (*Expert Sistem*) kecerdasan buatan yang menggabungkan pengetahuan dan penelusuran data untuk memecahkan masalah yang secara normal memerlukan keahlian manusia. Sedangkan definisi sistem pakar adalah suatu disiplin ilmu yang mentransformasi *knowledge* (berupa fakta, teori, pemikiran, prosedur) dari manusia kepada komputer sehingga komputer mempunyai kemampuan untuk berfikir, menalar, dan mengambil keputusan berdasarkan pengalaman.

Sistem berbasis aturan merupakan suatu sistem pakar yang menggunakan aturan-aturan untuk menyajikan pengetahuannya. Dengan demikian sistem berbasis aturan dapat dikatakan sebagai suatu perangkat lunak yang menyajikan keahlian pakar dalam bentuk aturan-aturan pada suatu domain tertentu untuk menyelesaikan suatu permasalahan.

2.4 Komponen Utama Sistem Berbasis Aturan

Komponen-komponen sistem berbasis aturan secara umum sebagai berikut:

1. Basis Pengetahuan

Basis pengetahuan yaitu pengetahuan yang menjadi dasar bagi pembuatan aturan-aturan dalam sistem, yang mencakup aturan-aturan itu sendiri,

fakta-fakta yang terkait, serta atribut-atributnya. Perbedaan antara fakta dan aturan dapat dilihat pada contoh berikut:

R1:	IF *p* AND *q* THEN *r*	→	aturan (*rule*)
	p, *q*, *r*	→	fakta, di mana
	p dan *q*	→	premis
	r	→	konklusi

2. Mekanisme Inferensi

Mekanisme inferensi berfungsi untuk mensimulasikan strategi penyelesaian masalah dari seorang pakar. Sebuah konklusi akan dicapai dengan menjalankan suatu aturan tertentu pada fakta yang ada. Untuk contoh di atas R1 akan menghasilkan konklusi *r* dari fakta *p* dan *q*. Dengan mekanisme inferensi sistem tahu kapan harus mengambil *rule* dan kapan memberikan pertanyaan kepada user.

3. Komponen Penjelas

Komponen penjelas berfungsi untuk menjelaskan strategi penyelesaian masalah bagi user yang meliputi:

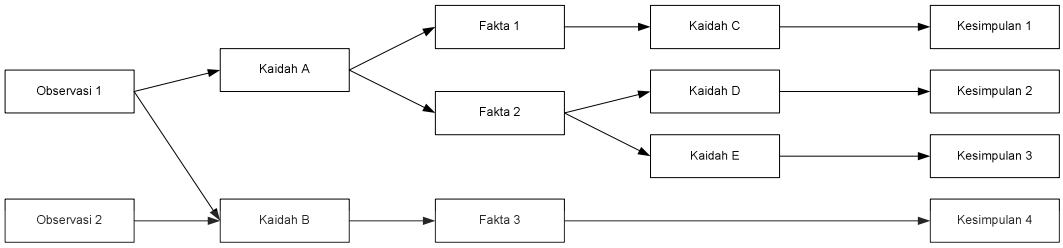
a. Pertanyaan apa yang diajukan pada pemakai dan jika diperlukan mengapa mengajukan pertanyaan tersebut.
b. Alasan bagaimana sistem tersebut memperoleh hasil demikian.
c. Karakteristik apa yang dimiliki tiap-tiap obyek.

2.5 Forward Chaining

Metode *forward chaining* adalah suatu metode dari mesin inferensi untuk memulai penalaran atau pelacakan suatu data dari fakta-fakta yang ada menuju suatu kesimpulan. Dilakukan pengecekan pada *rule* terlebih dahulu

apakah data yang diobservasi memenuhi syarat, jika memenuhi maka *rule-rule* tersebut akan dieksekusi. Proses berjalan maju dari fakta-fakta untuk menemukan solusi dari sebuah permasalahan. Hasil konklusi dari *rule* dapat digunakan untuk disesuaikan dengan *premise* dari *rule* lain.

Untuk lebih jelasnya dapat kita lihat gambar dibawah ini yang menggambarkan metode *forward chaining*:



Gambar 2.1 Metode *Forward Chaining*

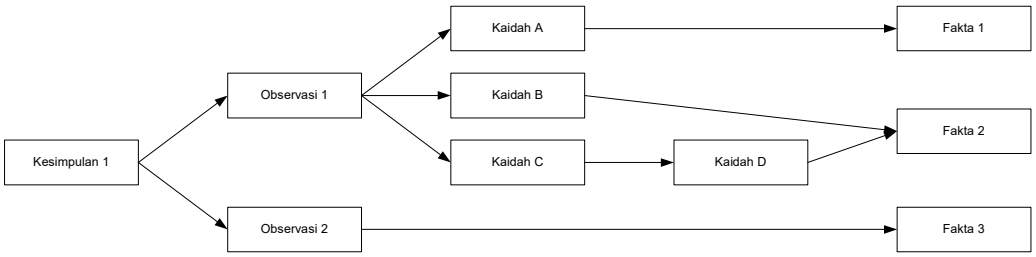
Gambar 2.1 memperlihatkan bahwa pangkalan kaidah terdiri dari 5 buah yaitu kaidah A, kaidah B, kaidah C, kaidah D, dan kaidah E. Sedangkan basis data terdiri dari fakta yang sudah diketahui yaitu fakta 1, fakta 2, fakta 3. Melalui observasi 1 mulai melacak kaidah untuk mencari premis dan menguji semua kaidah secara berurutan. Tahap pertama pada observasi 1 melacak kaidah A dan kaidah B. Mesin inferensi mulai melakukan pelacakan, pencocokan kaidah A dalam 1basis pengetahuan terhadap informasi yang ada di dalam basis data, yaitu fakta 1 dan fakta 2, jika pelacakan pada kaidah C yang kemudian menghasilkan kesimpulan 1, dan seterusnya.

2.6 Backward Chaining

Metode *backward chaining* merupakan kebalikan dari metode *forward chaining* atau disebut penalaran mundur yaitu suatu metode yang digunakan

dalam *inference engine* untuk melakukan pelacakan atau penalaran dari sekumpulan hipotesa menuju fakta-fakta yang mendukung kesimpulan tersebut.

Untuk lebih jelasnya dapat dilihat pada gambar 2.2 metode *Backward Chaining* dibawah ini:



Gambar 2.2 Metode *Backward Chaining*

Penelusuran *Backward chaining* berawal dari tujuan atau *goals*, observasi dilakukan untuk mencari informasi yang memenuhi kesimpulan 1 yang disebut juga *goal*. Dimulai dengan memberitahu sistem, bahwa ingin membuktikan keadaan *goal*. Motor inferensi melihat pangkalan data yaitu fakta untuk dicocokkan dengan pangkalan kaidah, pelacakan pada kaidah A, Kaidah B, Kaidah C dan Kaidah D menghasilkan fakta 1, fakta 2, dan fakta 3.

2.7 Verifikasi

Salah satu tujuan dari verifikasi adalah untuk memastikan adanya kesesuaian antara sistem yang dibuat dengan sistem yang telah ada. Definisi verifikasi adalah “Demonstrasi dari konsistensi dan kesempurnaan dari sistem“ (*Adrion*,1982:411) dan “Membangun sistem yang benar” (*O’Keefe*,1987:411).

Dua tahap untuk membuat basis pengetahuan verifikasi, sebagai berikut:

1. Memeriksa pelaksanaan suatu sistem secara spesifik.
2. Memeriksa konsistensi dan kelengkapan dari basis pengetahuan.

Beberapa *rule* yang harus diteliti dan diperhatikan dalam suatu basis pengetahuan (berdasarkan buku "*The Engineering of Knowledge-Based systems: Theory and Practice"*, Gonzalez, Avelino J. & Dankel, Douglas D., 1993)

1. Redundant rules

Dikatakan *redundant rule* jika dua *rule* atau lebih mempunyai *premise* dan *conclusion* yang sama.

Contoh:

Rule 1 : If Persentase Absen = A;100;90
Keaktifan = A;100;90
Kinerja = A;100;75
Then Status Kontrak = Kontrak Dapat Diperpanjang

Rule 2 : If Persentase Absen = A;100;90
Keaktifan = A;100;90
Kinerja = B;74;50
Then Status Kontrak = Kontrak Dapat Diperpanjang

2. Conflicting rules

Conflicting rules terjadi ketika dua buah *rule* atau lebih mempunyai *premise* yang sama, tetapi mempunyai *conclusion* yang berlawanan.

Contoh:

Rule 1 : If Persentase Absen = A;100;90
Keaktifan = A;100;90
Kinerja = A;100;75
Then Status Kontrak = Kontrak Dapat Diperpanjang

Rule 2 : If Persentase Absen = A;100;90

Keaktifan = A;100;90

Kinerja = A;100;75

Then Status Kontrak = Kontrak Tidak Dapat Diperpanjang

3. Subsumed rules

Suatu *rule* dapat dikatakan *subsumed* jika *rule* tersebut mempunyai *constraints* yang lebih atau kurang tetapi mempunyai *conclusion* yang sama.

Contoh:

Rule 1 : If Persentase Absen = A;100;90

Keaktifan = A;100;90

Kinerja = A;100;90

Then Status Kontrak = Kontrak Dapat Diperpanjang

Rule 2 : If Persentase Absen = A;100;90

Keaktifan = A;100;90

Then Status Kontrak = Kontrak Dapat Diperpanjang

4. Circular rules

Circular rules ialah suatu keadaan dimana terjadinya proses perulangan dari suatu *rule*. Ini dikarenakan suatu *premise* dari salah satu *rule* merupakan *conclusion* dari *rule* yang lain, atau kebalikannya.

Contoh:

Rule 1 : If Persentase Absen = A;100;90

Then Status Kontrak = Kontrak Dapat Diperpanjang

Rule 2 : If Status Kontrak = Kontrak Dapat Diperpanjang

Then Persentase Absen = A;100;90

5. Unnecessary IF conditions

Unnecessary IF terjadi ketika dua *rule* atau lebih mempunyai *conclusion* yang sama, tetapi salah satu dari *rule* tersebut mempunyai *premise* yang tidak perlu dikondisikan dalam *rule* karena tidak mempunyai pengaruh apapun.

Contoh:

Rule 1 : If Persentase Absen = A;100;90
Keaktifan = A;100;90
Kinerja = A;100;75
Then Status Kontrak = Kontrak Dapat Diperpanjang

Rule 2 : If Persentase Absen = A;100;90
Keaktifan = A;100;90
Kinerja = B;74;50
Then Status Kontrak = Kontrak Dapat Diperpanjang

Kedua rule di atas dapat digabungkan menjadi rule seperti di bawah ini.

Rule 3 : If Persentase Absen = Rajin
Keaktifan = Aktif
Then Status Kontrak = Kontrak Dapat Diperpanjang

6. Dead-end rules

Dead-end rules adalah suatu *rule* yang konklusinya tidak diperlukan oleh *rule-rul*e lainnya.

Contoh:

Rule 1 : If Persentase Absen = A;100;90
Keaktifan = A;100;90

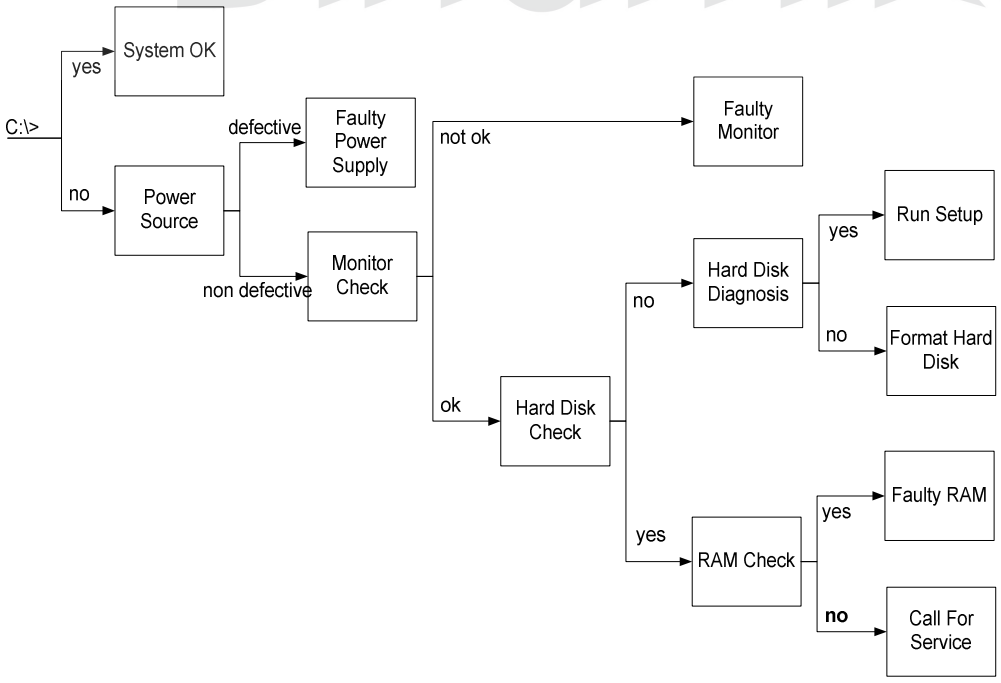
Kinerja = A;100;75

Then Status Kontrak = Tidak Ada Kontrak

2.8 Diagram Blok

Langkah awal yang dilakukan dalam menerjemahkan suatu bidang ilmu ke dalam sistem berbasis aturan yaitu melalui diagram blok (*Block Diagram*). Diagram blok merupakan susunan dari aturan-aturan yang terdapat di dalam sebuah bidang ilmu.

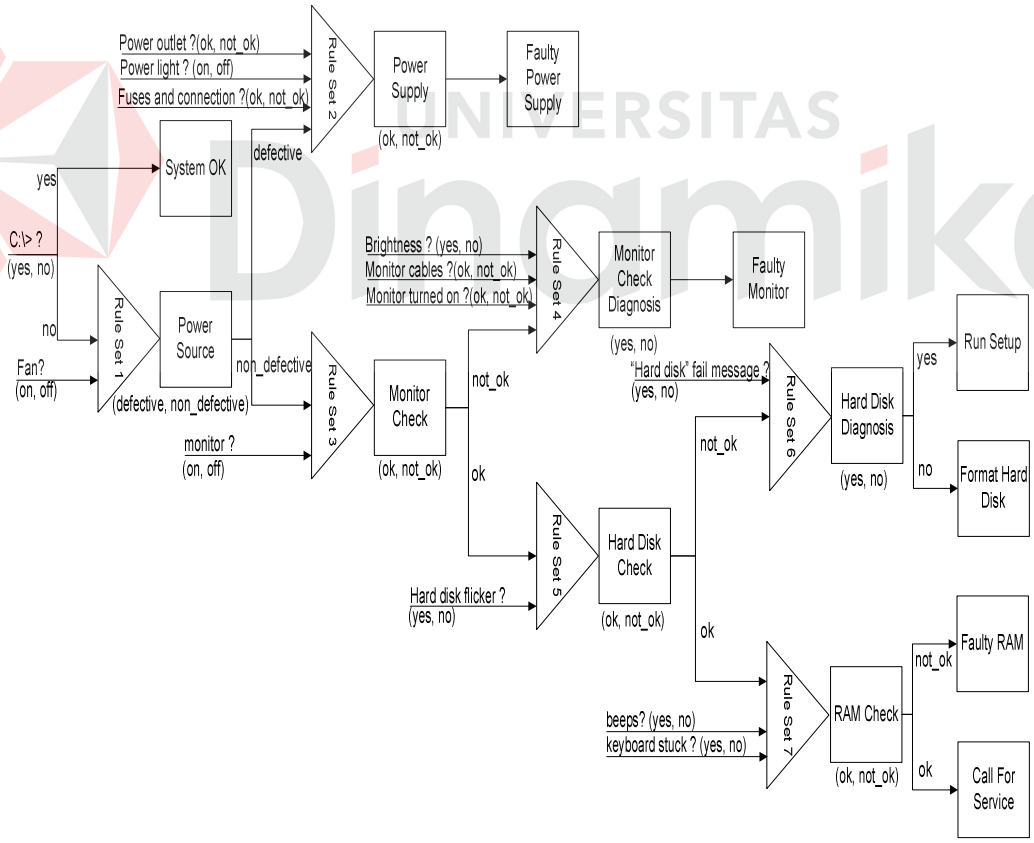
Dengan membuat diagram blok di dalam sistem berbasis aturan, dapat mengetahui urutan kerja sistem dalam mencari keputusan. Dari sini pula dapat diputuskan metode penelusuran yang akan digunakan, apakah metode *forward chaining* atau *backward chaining*. Gambar 2.3 merupakan salah satu contoh kasus diagnosa komputer yang digambarkan dalam diagram blok menggunakan metode *forward chaining*.



Gambar 2.3 Contoh Diagram Blok

2.9 Diagram Ketergantungan

Setelah mengetahui urutan kerja sistem dalam mencari keputusan dari diagram blok, langkah selanjutnya yaitu kita membuat diagram ketergantungan (*dependency diagram*). Diagram ketergantungan di dalam sistem berbasis aturan berfungsi untuk menunjukkan hubungan atau ketergantungan antara inputan jawaban, aturan-aturan (*rule*), nilai-nilai dan direkomendasikan untuk sistem berbasis pengetahuan [*Dologite,* 1993:22]. Dari blok diagram di atas apabila diteruskan menjadi diagram ketergantungan akan terlihat seperti gambar 2.4 di bawah ini:



Gambar 2.4 Contoh Diagram Ketergantungan

BAB III

PERANCANGAN SISTEM

Bab ini membahas tentang perancangan basis aturan sistem, meliputi perancangan aturan lembur yang dilengkapi dengan *block diagram*, *dependency diagram, decision table* dan proses reduksi beserta penjelasan parameternya. Pada bagian akhir terdapat desain arsitektur, diagram alir sistem, dan struktur tabel.

3.1 Perancangan Aturan Upah Lembur

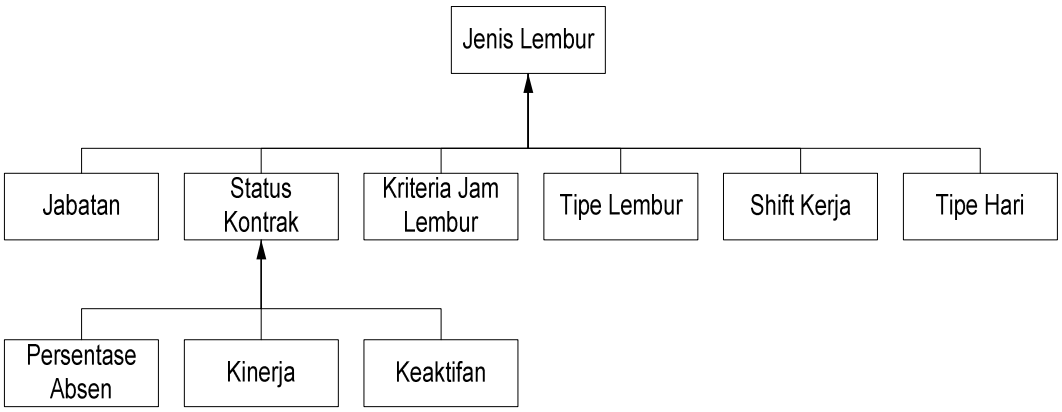
3.1.1 Perancangan Diagram Blok

Perancangan diagram blok diambil dari parameter upah lembur yang terdapat pada perusahaan, yang dibagi menjadi empat bagian yaitu lembur per bulan, jam kerja, status dan jabatan. Jam kerja pada perusahaan ini ada tiga macam, jenis shift untuk karyawan produksi dan karyawan bagian bahan, bagian teknisi mempunyai jam kerja setengah hari (12 jam), sedangkan sopir dan kepala produksi bekerja full time mulai pagi sampai malam, atau sampai tidak ada keperluan lain di kantor, jika dibutuhkan maka harus menunggu di kantor sampai urusan kantor selesai.

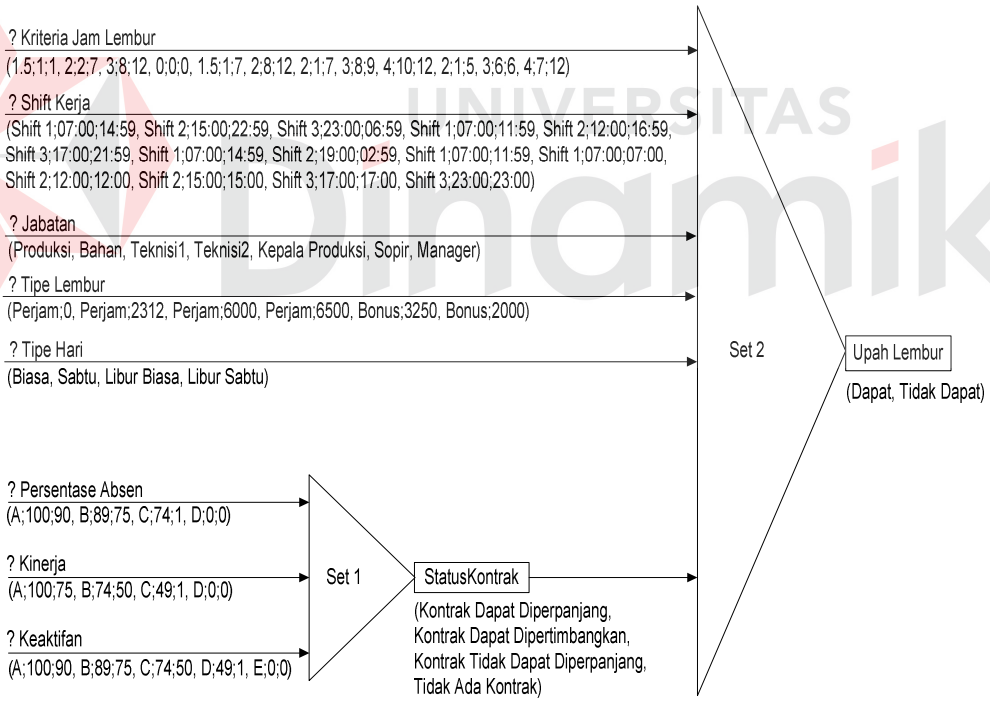
Status karyawan dibedakan menjadi dua yaitu kontrak dan tetap. Jumlah jam absensi per bulan diperhitungkan sebagai salah satu bahan pertimbangan bahwa karyawan tersebut memungkinkan untuk dikontrak lagi atau tidak (khusus untuk karyawan kontrak).

Pada Gambar 3.1 dijelaskan bahwa kriteria upah lembur pada level 1 terdiri dari parameter Status, Jabatan, Kriteria Jam Lembur, Tipe Lembur, Tipe

Hari dan Shift Kerja. Sedangkan pada level 2 Status dibagi menjadi tiga penilaian yaitu Persentase Absen, Keaktifan dan Kinerja.



Gambar 3.1 Diagram Blok Upah Lembur



Gambar 3.2 Diagram Ketergantungan Upah Lembur

3.1.2 Perancangan Diagram Ketergantungan

Diagram ketergantungan dibuat untuk menentukan faktor yang mempengaruhi dalam perhitungan upah lembur. Untuk lebih jelasnya dapat dilihat

pada Gambar 3.2 yang memberikan penjelasan bahwa hasil perhitungan lembur berdasarkan parameter Shift Kerja, Kriteria Lembur, Tipe Lembur, Tipe Hari, Status dan Jabatan dengan penjabaran tiap parameter pada sub parameter.

Berikut keterangan dari parameter – parameter yang digunakan pada diagram ketergantungan (*Sequence Diagram*) pada Gambar 3.2:

1. Upah Lembur : Kesimpulan akhir dari seluruh proses, hasil dari proses perhitungan upah lembur berdasarkan parameter–parameter yang ada.
2. Status Kontrak : Parameter Status ini digunakan untuk menentukan apakah seorang karyawan kontrak dapat dikontrak lagi. Beberapa sub parameter yang digunakan untuk melakukan penilaian tersebut adalah:

   a. Persentase Absen

      Perhitungan absen setiap bulan dibagi dengan jumlah hari aktif akan menghasilkan persentase absen yang dapat mempengaruhi status karyawan.

   b. Kinerja

      Persentase hasil kerja juga sangat menentukan status karyawan dalam perusahaan, nilai kinerja ini diinput secara manual oleh manager dengan pertimbangan pendapat dari kepala produksi selaku pengawas.

   c. Keaktifan

      Nilai keaktifan didapat dari berapa hari seorang

karyawan masuk terlambat ataupun pulang lebih cepat dibagi dengan jumlah hari kerja dalam satu bulan.

3. Kriteria Lembur : Pada perusahaan ini upah lembur yang diberikan meningkat setiap jam tertentu, karyawan produksi pada jam pertama upah lembur dikalikan dengan variabel 1.5, jam ke-2 sampai ke-7 dikalikan 2, jam ke-8 sampai ke-12 dikalikan 3. Sedangkan upah lembur perjamnya adalah Rp. 2.312,- sesuai ketentuan Departemen Tenaga Kerja, maksimal jam lembur per hari adalah 12 jam. Upah lembur perjam untuk karyawan teknisi besarnya berbeda-beda tergantung dari kebijakan perusahaan, variabel yang digunakan pada jam pertama sampai ke-7 adalah 1,5 jam selanjutnya sampai jam ke-12 dikalikan 2. Ketentuan lain diatur dalam diagram ketergantungan.

4. Jabatan : Jabatan sebenarnya dibagi menjadi dua, kontrak dan tetap. Yang termasuk karyawan kontrak adalah bagian Produksi dan bagian Bahan sedangkan Kepala Produksi, Teknisi, Sopir, dan Manager adalah karyawan tetap. Pada saat ini jenis karyawan tidak begitu banyak, jumlahnya pun masih relative sedikit.

5. Tipe Lembur : Jenis upah lembur ada 2 macam, upah lembur per jam dan bonus. Sesuai kriteria lembur maka upah lembur per jam dihitung berdasarkan variabel-variabel tertentu, sedangkan bonus diberikan untuk karyawan yang bekerja

pada jam malam (Shift 3).

6. Shift Kerja : Jam kerja pada perusahaan ini dibagi menjadi empat macam, yaitu Shift Biasa, Shift Khusus, Shift untuk Teknisi dan Shift Normal. Shift yang diperuntukkan karyawan kontrak (produksi dan bahan), dibagi menjadi 3 dengan perincian sebagai berikut:

   a. Shift 1 (07.00 – 15.00).

   b. Shift 2 (15.00 – 23.00).

   c. Shift 3 (23.00 – 07.00).

   Teknisi mempunyai jam kerja yang agak lama yaitu 12 jam atau setengah hari (*half day*). Jam kerjanya dibagi 2, pagi dan malam. Jam kerja yang sebenarnya adalah 8 jam (7 jam kerja, 1 jam istirahat), sedangkan 4 jam sisanya dihitung lembur.

   a. Shift 1 (07.00 – 19.00).

   b. Shift 2 (19.00 – 07.00).

   Manager, Sopir dan Kepala Produksi mempunyai jam kerja paling lama mereka harus tetap berada di perusahaan selama dibutuhkan. Jam kerjanya juga 8 jam (7 jam kerja, 1 jam istirahat), selebihnya tidak dihitung lembur.

7. Tipe Hari : Perhitungan lembur untuk hari biasa, sabtu, minggu dan hari libur tidak sama. Sedangkan hari libur ada dua macam yang pertama hari libur yang jatuh pada hari

biasa dan hari libur yang jatuh pada hari dengan jam kerja terpendek atau hari sabtu.

3.1.3 Perancangan Rule

Perancangan ini dibagi menjadi 3 bagian penting yaitu pembuatan *decision* tables(tabel keputusan) yang dibuat secara lengkap, perancangan *reduced decision tables* (reduksi terhadap tabel keputusan) dilakukan untuk melakukan pengurangan terhadap jumlah rule yang banyak sehingga *rule-rule* yang tidak digunakan dihapus dari tabel, dan terakhir adalah penulisan aturan-aturan IF-THEN digunakan untuk menghasilkan *file knowledge base system*.

Berikut penjelasan selengkapnya akan diulas di dalam subbab berikut :

A. Decicion Tables

Decision table dibuat untuk menunjukkan antar hubungan nilai-nilai pada hasil fase antara atau rekomendasi akhir *knowledge base system*.

Tabel 3.1 *Decision Table Rule* Set 1

Step 1 : *Plan*

Kondisi :	Persentase Absen	(A;90;100/B;75;89/C;1;74/D;0;0)	= 4
	Kinerja	(A;75;100/B;50;74/C;1;54/D;0;0)	= 4
	Keaktifan	(A;90;100/B;75;89/C;50;74/D;1;49/E;0;0)	= 5
Baris : 4 x 4 x 5 = 80			

Step 2 : *Completed Decision Table*

Tabel ini mempunyai kondisi yang banyak yaitu 80 kondisi yang berbeda–beda untuk menghasilkan keputusan (dalam hal ini Status Kontrak), untuk lebih jelasnya tabel ini dapat dilihat pada lampiran.

Pada Table 3.1 berikut menunjukkan salah satu contoh perancangan *decision table* untuk *rule* set 1 yaitu parameter Status Kontrak yang berdasarkan pada perancangan *dependency diagram* mempunyai 3 kondisi yaitu Persentase Absen, Kinerja dan Keaktifan. Jumlah kondisi total didapat dari mengalikan nilai-nilai yang dimiliki setiap kondisi. Dalam Tabel 3.1, rencana *decision table* adalah untuk rangkaian aturan akhir yang terkait dengan beberapa kondisi, yang masing-masing dapat memiliki sejumlah nilai berbeda. Persentase Absen, kondisi pertama memiliki 4 nilai yaitu A;90;100, B;75;89, C;1;74 dan D;0;0. Kinerja dibedakan menjadi 4 yaitu A;75;100, B;50;74, C;1;54 dan D;0;0. Keaktifan ada 5 yaitu A;90;100, B;75;89, C;50;74, D;1;49 dan E;0;0. Angka-angka yang tercantum dalam setiap nilai diatas menunjukkan *range* atau batasan persentase nilai dari karyawan selama satu bulan, baik dalam hal absensi, waktu terlambat dan hasil penilaian kinerja karyawan oleh Kepala Produksi.

B. Reduced Decicion Tables

Pada sistem ini proses perancangan reduksi untuk setiap *decision table* dilakukan secara *manual*. Perancangan reduksi berdasarkan *decision table* pada Tabel 3.1 menghasilkan parameter seperti pada Tabel 3.2 berikut :

Tabel 3.2 Reduksi *Decision Table Rule Set* 1

Step 1 : *Plan*

Kondisi :	Persentase Absen	(A;90;100/B;75;89/C;1;74/D;0;0)	= 4
	Kinerja	(A;75;100/B;50;74/C;1;54/D;0;0)	= 4
	Keaktifan	(A;90;100/B;75;89/C;50;74/D;1;49/E;0;0)	= 5

Baris : 4 x 4 x 5 = 80

Step 2 : *Completed Decision Table*

Seperti yang telah dijelaskan pada sub bab sebelumnya tabel ini mempunyai kondisi yang banyak, 80 kondisi yang memungkinkan untuk melakukan perhitungan upah lembur.

Step 3 : *Reduced Decision Table*

Setelah mengalami reduksi maka dari 80 kondisi menjadi 37 kondisi yang dapat diimplementasikan dalam sistem berbasis aturan untuk perhitungan upah lembur ini, tabel secara lengkap akan diuraikan pada bagian lampiran.

C. Penulisan IF-THEN Rule

Pada pengembangan *rule base* telah direpresentasikan dalam bentuk blok diagram yang kemudian diimplementasikan dalam bentuk list aturan (*rule*), yaitu struktur berbasis pengetahuan. *Rule base* pada sistem ini menghasilkan *file* KBS, yaitu *file* text berupa himpunan aturan (*rule*), sedangkan untuk premis dan konklusi disimpan dalam tabel *rule*. *Rule* pada dasarnya terdiri dari dua bagian pokok yaitu bagian *IF* (premis atau kondisi) dan bagian *THEN* (konklusi atau kesimpulan). Pemilihan representasi pengetahuan dengan *rule base* didasarkan alasan sebagai berikut:

a. Pengembangan sistem pakar menggunakan *rule base*.

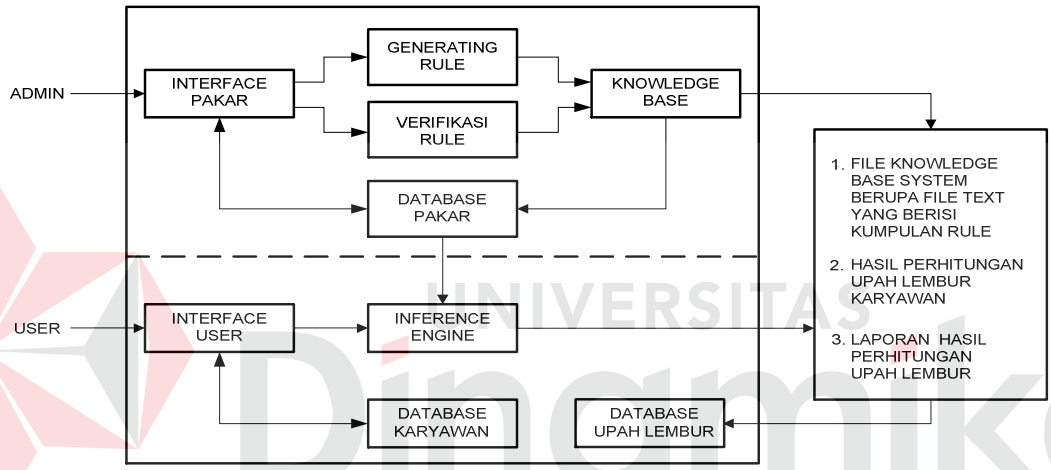
b. *Rule base* dapat dengan mudah dilakukan perubahan, seperti penambahan, penghapusan, dan perubahan aturan.

Contoh berikut merupakan struktur basis pengetahuan yang sebelumnya telah dirancang menggunakan blok diagram.

Contoh *Rule*: *IF* Shift Kerja = Shift 1 *AND* Kriteria Lembur = 1.5;1;1 AND Jenis Lembur = Per Jam;2312 AND Jabatan = Produksi AND Status = Kontrak Dapat Diperpanjang AND Tipe Hari = Biasa *THEN* Upah Lembur = Dapat

3.2 Desain Arsitektur

Desain Arsitektur seperti terlihat pada Gambar 3.3, menggambarkan hubungan antara elemen-elemen utama.



Gambar 3.3 Gambar Desain Arsitektur untuk Perhitungan Upah Lembur

Berikut penjelasan dari desain arsitektur :

1. *Interface* Pakar

Media yang digunakan oleh seorang pakar untuk mengembangkan sistem. Di mana dalam merancang sistem dengan menentukan upah lembur dan *generating rule* yang menghasilkan file text KBS, yaitu berupa himpunan aturan (*rule*).

2. *Generating Rule*

Proses *generating rule* dijalankan untuk membentuk *rule-rule* dari desain pakar melalui *Treeview*.

3. *Database* Pakar

Digunakan untuk mengembangkan basis pengetahuan apabila pakar ingin menambah, mengubah, atau menghapus *rule*. Penamaan *database* pakar sesuai dengan keinginan pakar. Dalam sistem ini *database* pakar disimpan dalam *file* YUNICOLembur.mdb yang terdiri dari:

a. Tabel *Rule* digunakan untuk menyimpan data parameter berdasarkan *Treeview* yang telah dibuat.
b. Tabel *RuleAnswer* digunakan untuk menyimpan *option-option/value* dari tabel *Rule*.
c. Tabel *RuleCon* digunakan untuk menyimpan data premis dan konklusi berdasarkan hasil reduksi.
a. Tabel *FullPath* digunakan untuk menyimpan path dari sistem berbasis aturan yang digunakan untuk menampilkan tree.

4. *Interface* Admin

Sebagai media oleh admin untuk melihat dan berinteraksi dengan sistem. Membuat, menghapus atau memperbaiki *rule* dan melakukan seting terhadap *user* pakar.

5. *Database* Karyawan

Digunakan untuk menyimpan data lembur karyawan yang diinput oleh admin. *Database* karyawan disimpan dalam *file* YUNICOData.mdb yang terdiri dari:

a. Tabel karyawan digunakan untuk menyimpan data karyawan.
b. Tabel absensi digunakan untuk menyimpan data absensi karyawan.

6. *Database* Hasil

Database hasil dari perhitungan lembur disimpan di dalam *file* YUNICOSave.mdb, di dalam *database* ini terdapat 3 tabel untuk melakukan proses *save* data, baik data lembur maupun data *user* yang berhak mengakses aplikasi.

a. Tabel *result* digunakan untuk menyimpan data yang diinput oleh admin dari hasil perhitungan lembur.
b. Tabel *dailyresult* digunakan untuk menyimpan data lembur secara rinci perhari yang diinput oleh admin atau user dari hasil perhitungan lembur.
c. Tabel admin digunakan untuk menyimpan data *login* admin dan *user*. Jenis pengguna aplikasi ada 3 yaitu admin yang mempunyai authorisasi penuh terhadap aplikasi, *user* hanya mampu melakukan perhitungan lembur dan karyawan hanya dapat melihat laporan lembur milik mereka sendiri.

7. *Inference Engine*

Mekanisme inferensi yang digunakan adalah *Forward Chaining* karena sistem lebih dulu mengetahui fakta-fakta yang ada, kemudian mencari kesimpulan sementara sampai akhirnya berhenti setelah menghasilkan sebuah kesimpulan akhir (perhitungan upah lembur karyawan).

Proses *forward chaining* diperlukan dalam mencari solusi berdasarkan *goal* upah lembur dan *rule base* yang ada dalam *working memory*. *Goal* pada proses *forward chaining* adalah hasil perhitungan upah lembur karyawan.

8. *Knowledge base*

Kumpulan fakta dan aturan (*rule*) serta *working memory* yang merupakan fakta yang diperoleh oleh sistem selama proses berlangsung, dan aturan tentang

perhitungan upah lembur karyawan. *Knowledge base* pada sistem ini, disimpan dalam tabel *rule*.

9. *Output*

a. *Output* dari desain pakar adalah *database* dengan nama tabel *result* dan *file text* yang berisi himpunan aturan.

b. *Output* dari desain admin adalah hasil akhir dari proses *inference* yaitu hasil perhitungan upah lembur dan laporan hasil upah lembur karyawan dalam periode tertentu (per bulan).

3.3 Perancangan Proses

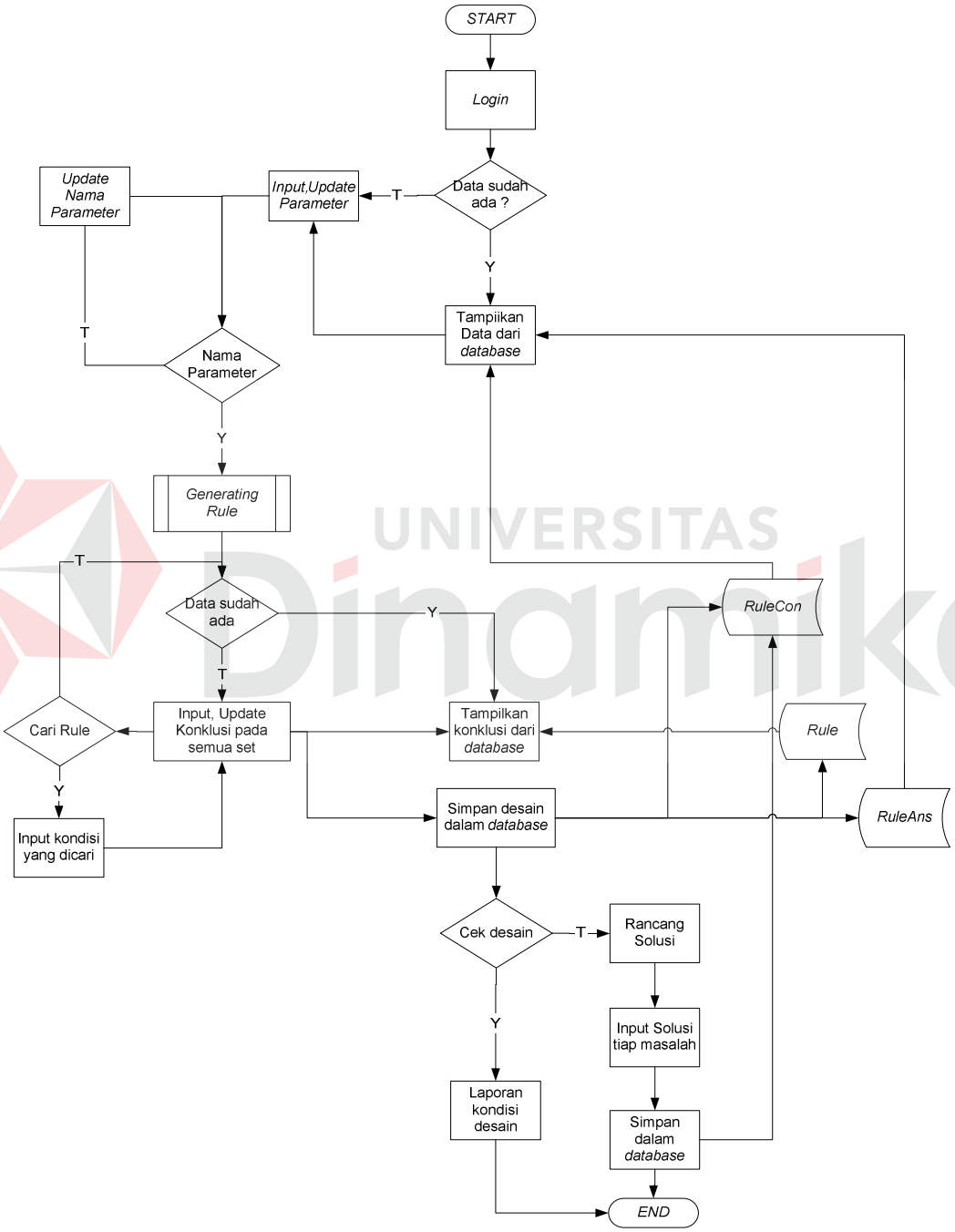
Perancangan proses dalam sistem ini adalah sebagai berikut :

1. Diagram Alir Sistem untuk Desain Pakar.
2. Diagram Alir Sistem Proses *Generating Rule*
3. Diagram Alir Sistem Proses *Inference Engine*.

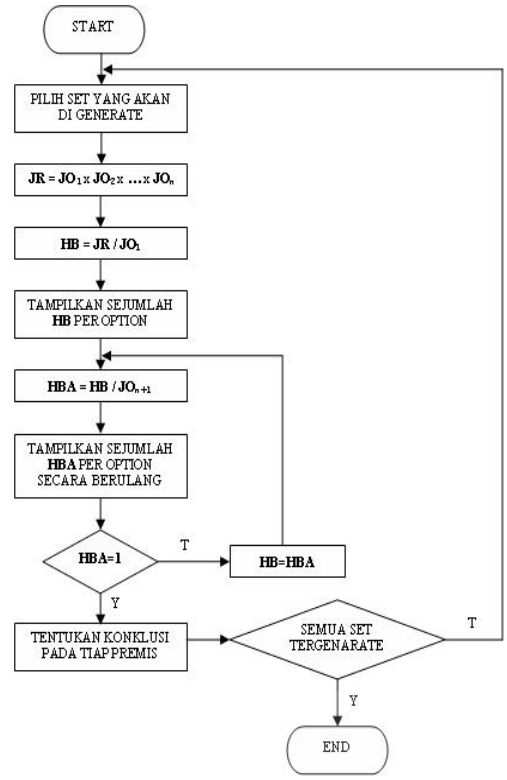
Pada Gambar 3.4 menggambarkan desain diagram alir pakar untuk mendefinisikan aturan-aturan yang akan digunakan oleh *user* dalam proses perhitungan upah lembur.

Diagram alir dari *generating rule* dapat dilihat dalam Gambar 3.5. Diawali dengan menentukan *set* yang melalui proses *generate* terlebih dahulu. Kemudian dicari Jumlah *rule* yang akan terbentuk dari *set* tersebut dengan cara mengalikan jumlah-jumlah *option* pada tiap-tiap *node*/*subset*. Setelah didapat jumlah *rule* maka jumlah *rule* tersebut akan dibagi dengan jumlah *option* pada *node*/*subset* pertama maka akan didapat nilai dari hasil bagi yang mana nilai hasil bagi ini akan menjadi jumlah per *option* yang akan ditampilkan. Untuk mendapatkan jumlah per *option* dari *node*/subset selanjutnya, Hasil Bagi harus

dibagi dengan jumlah *option*-nya sehingga mendapat nilai Hasil Bagi Akhir. Langkah ini dilakukan hingga hasil bagi akhir mencapai nilai 1 yang berarti nilai hasil bagi akhir sama dengan jumlah *option* dari *node*/level akhir.



Gambar 3.4 Diagram Alir Sistem Desain Pakar



Keterangan :
JR = Jumlah Rule
JO = Jumlah Option
HB = Hasil Bagi
HBA = Hasil Bagi Akhir
n = Jumlah node/level pada set

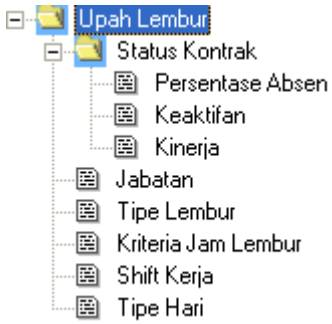
Gambar 3.5 Diagram Alir Sistem Proses *Generating Rule*

Setelah semua *rule* terbentuk kemudian pakar dapat menentukan konklusi bagi tiap-tiap *rule*. Proses berhenti apabila semua *set* yang terbentuk dari desain pakar telah melalui proses *generate*.

Proses-proses penting yang terdapat dalam *generating rule* tersebut yaitu:

A. Mendapatkan Jumlah Rule Per Set

Jumlah *rule-rule* yang akan terbentuk dari proses *generating rule* didapat dari hasil perkalian *option* tiap-tiap *node*/*subset* yang terdapat didalam suatu *set*, sebagai gambaran dapat dilihat desain pakar untuk penyakit tropik dalam bentuk *Treeview* pada Gambar 3.6



Gambar 3.6 Desain Upah Lembur dalam Bentuk *Treeview*

Pada Gambar 3.6. terdapat set Status yang memiliki *child* persentase absen, keaktifan dan kinerja. Jadi untuk mendapatkan jumlah *rule* dari *set* status adalah mengalikan jumlah *option* dari persentase absen, keaktifan dan kinerja (4 x 5 x 4) sehingga didapat 80 *rule*.

Untuk set yang memiliki *child* berupa *node* dan subset seperti set upah lembur yang memiliki *child* status, jabatan, jenis lembur, kriteria lembur dan shift kerja berupa *node*. Maka untuk mendapatkan jumlah rule bagi set upah lembur yaitu dengan mengalikan jumlah *option* dari tiap subset (bukan jumlah *rule* hasil perkalian dari *node-node* yang terdapat dalam *subset*) dan *node*. Jadi jumlah *rule* dari set upah lembur adalah jumlah *option subset* status kontrak (4 *option*) dikalikan jumlah *option node* jabatan (7 *option*) dikalikan jumlah *option node* jenis lembur (6 *option*) dikalikan jumlah *option node* kriteria jam lembur (12 *option*) dikalikan jumlah *option node* shift kerja (14 *option*) dikalikan jumlah *option node* tipe hari (4 *option*) maka didapatkan sejumlah 112.896 *rule*.

B. Menampilkan Rule-rule

Agar seluruh *rule* yang ditampilkan dapat memberikan kombinasi dari semua premis maka jumlah *rule* dari set yang ditampilkan akan dibagi dengan

jumlah *option* pada *node*/subset yang pertama, maka akan didapat nilai dari hasil bagi. Hasil bagi ini akan menjadi jumlah per *option* dari *node*/subset yang akan ditampilkan. Untuk mendapatkan jumlah per *option* dari *node*/subset yang akan ditampilkan selanjutnya, hasil bagi harus dibagi dengan jumlah *option* dari *node*/subset selanjutnya sehingga mendapat nilai Hasil Bagi Akhir. Langkah ini dilakukan hingga hasil bagi akhir mencapai nilai 1 yang berarti nilai hasil bagi akhir sama dengan jumlah *option* dari *node*/level akhir.

Sebagai gambaran dari kombinasi premis yang lengkap seperti dalam Gambar 3.7 yang menampilkan *rule-rule* yang terdapat dalam set status.

Terlihat dalam gambar 3.7 jumlah *rule* dari set status adalah 80 *rule*. Maka jumlah *rule* tersebut dibagikan dengan jumlah *option* premis pertama dari Status yaitu persentase absen yaitu 4 *option*, sehingga dalam menampilkan tiap-tiap *option* dari premis persentase absen dikenakan perulangan sebanyak 20 kali (80 *rule* dibagi 4 *option*).

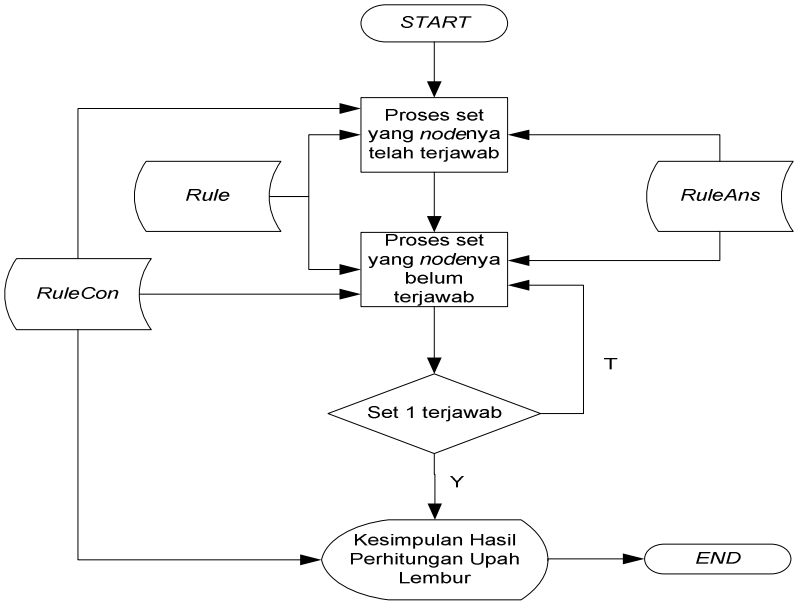
Hasil bagi dari jumlah *rule* dengan jumlah *option* premis pertama tadi akan dibagi kembali dengan jumlah *option* dari premis kedua yaitu keaktifan yang memiliki 5 *option* sehingga tiap-tiap *option* dikenakan perulangan sebanyak 4 kali (20 dibagi 4 *option*). Hasil bagi dari jumlah *option* premis kedua tadi akan dibagi kembali dengan jumlah *option* dari premis ketiga yaitu kinerja yang memiliki 4 *option* sehingga tiap-tiap *option* dikenakan perulangan sebanyak 1 kali (4 dibagi 4 *option*). Apabila hasil bagi sudah mencapai angka 1 maka proses akan dihentikan, karena ini berarti proses penampilan telah mencapai premis terakhir.

No	Status Kontrak	Persentase Absen	Keaktifan	Kinerja
1	Kontrak Dapat Diperpanjang	A;100;90	A;100;90	A;100;75
2	Kontrak Dapat Diperpanjang	A;100;90	A;100;90	B;74;50
3	Kontrak Dapat Dipertimbangkan	A;100;90	A;100;90	C;49;1
4	Kontrak Dapat Diperpanjang	A;100;90	B;89;75	A;100;75
5	Kontrak Dapat Diperpanjang	A;100;90	B;89;75	B;74;50
6	Kontrak Dapat Dipertimbangkan	A;100;90	B;89;75	C;49;1
7	Kontrak Dapat Diperpanjang	A;100;90	C;74;50	A;100;75
8	Kontrak Dapat Dipertimbangkan	A;100;90	C;74;50	B;74;50
9	Kontrak Tidak Dapat Diperpanjang	A;100;90	C;74;50	C;49;1
10	Kontrak Dapat Dipertimbangkan	A;100;90	D;49;1	A;100;75
11	Kontrak Dapat Dipertimbangkan	A;100;90	D;49;1	B;74;50
12	Kontrak Tidak Dapat Diperpanjang	A;100;90	D;49;1	C;49;1
13	Kontrak Dapat Diperpanjang	B;89;75	A;100;90	A;100;75
14	Kontrak Dapat Dipertimbangkan	B;89;75	A;100;90	B;74;50
15	Kontrak Dapat Dipertimbangkan	B;89;75	A;100;90	C;49;1
16	Kontrak Dapat Dipertimbangkan	B;89;75	B;89;75	A;100;75
17	Kontrak Dapat Dipertimbangkan	B;89;75	B;89;75	B;74;50
18	Kontrak Dapat Dipertimbangkan	B;89;75	B;89;75	C;49;1
19	Kontrak Tidak Dapat Diperpanjang	B;89;75	C;74;50	A;100;75
20	Kontrak Tidak Dapat Diperpanjang	B;89;75	C;74;50	B;74;50
21	Kontrak Tidak Dapat Diperpanjang	B;89;75	C;74;50	C;49;1
22	Kontrak Tidak Dapat Diperpanjang	B;89;75	D;49;1	A;100;75
23	Kontrak Tidak Dapat Diperpanjang	B;89;75	D;49;1	B;74;50
24	Kontrak Tidak Dapat Diperpanjang	B;89;75	D;49;1	C;49;1
25	Kontrak Tidak Dapat Diperpanjang	C;74;1	A;100;90	A;100;75
26	Kontrak Tidak Dapat Diperpanjang	C;74;1	A;100;90	B;74;50
27	Kontrak Tidak Dapat Diperpanjang	C;74;1	A;100;90	C;49;1
28	Kontrak Tidak Dapat Diperpanjang	C;74;1	B;89;75	A;100;75
29	Kontrak Tidak Dapat Diperpanjang	C;74;1	B;89;75	B;74;50
30	Kontrak Tidak Dapat Diperpanjang	C;74;1	B;89;75	C;49;1
31	Kontrak Tidak Dapat Diperpanjang	C;74;1	C;74;50	A;100;75
32	Kontrak Tidak Dapat Diperpanjang	C;74;1	C;74;50	B;74;50
33	Kontrak Tidak Dapat Diperpanjang	C;74;1	C;74;50	C;49;1
34	Kontrak Tidak Dapat Diperpanjang	C;74;1	D;49;1	A;100;75
35	Kontrak Tidak Dapat Diperpanjang	C;74;1	D;49;1	B;74;50
36	Kontrak Tidak Dapat Diperpanjang	C;74;1	D;49;1	C;49;1
37	Tidak Ada Kontrak	D;0;0	E;0;0	D;0;0

Gambar 3.7 Tampilan *Rule-rule* pada *Set* Status Kontrak.

C. Mengisi Konklusi Dari Tiap-tiap Rule

Apabila *generating rule* telah dilakukan dan telah menghasilkan *rule-rule* maka pakar perlu memberikan *konklusi* pada tiap-tiap *rule* agar ketika *user* menggunakan sistem untuk berkonsultasi akan mendapatkan kesimpulan yang sesuai dengan fakta-fakta yang diinput.



Gambar 3.8 Diagram Alir Sistem Proses *Inference Engine*

Isian konklusi dari tiap-tiap set didapatkan dari *option-opt*ion yang telah diinput sebelumnya melalui desain pakar dalam *Treeview*.

Pada Gambar 3.8 menjelaskan proses *inference engine* dengan menggunakan metode *forward chaining* yaitu penelusuran dari data-data yang ada untuk mencapai suatu konklusi.

3.4 Struktur Table

Agar sistem berbasis aturan ini dapat menampung bermacam permasalahan dan bidang ilmu maka sistem dirancang untuk dapat membuat *database* baru dan membuka *database* yang sudah ada dimana *database* tersebut menyimpan data desain dan KBS yang dibuat oleh pakar. Tabel-tabel yang terdapat didalamnya yaitu tabel Rule untuk menyimpan data dari tiap parameter yang terdapat dalam desain *Treeview*, tabel RuleAnswer untuk menyimpan *option-option* dari tiap-tiap parameter dan tabel RuleCon yang menyimpan *rule-rule* hasil *generating*.

1. Nama Tabel : Rule

   Primary Key : NoLevel, IdRule, IdRuleParent

   Foreign Key : -

   Keterangan : Tabel Rule digunakan untuk menyimpan data-data dari parameter.

Tabel 3.3 Tabel Rule

NO	Field Name	Type	Length	Description
1	NoLevel	Number	Integer	Index Level
2	IdRule	Text	50	Nama Parameter
3	IdRuleParent	Text	50	Nama Parent Parameter
4	Set	Number	Integer	Set Node
5	NoRule	Number	Integer	Nomer Urut Rule
6	Question	Text	50	Pertanyaan

2. Nama Tabel : RuleAnswer

   Primary Key : IdRule, NoAns

   Foreign Key : IdRule

   Keterangan : Tabel RuleAnswer digunakan untuk menyimpan *option-option* dari tiap parameter.

Tabel 3.4 Tabel RuleAnswer

NO	Field Name	Type	Length	Description
1	IdRule	Text	50	Nama Parameter
2	NoAns	Number	Integer	Id Jawaban
3	Answer	Text	50	Jawaban

3. Nama Tabel : RuleCon

   Primary Key : IdRule, NoAns, IdRuleChild, NoAnsChild, No

   Foreign Key : IdRule

Keterangan : Tabel RuleCon digunakan untuk menyimpan *rule-rule* hasil generating.

Tabel 3.5 Tabel RuleCon

NO	Field Name	Type	Length	Description
1	IdRule	Text	50	Nama Parameter
2	NoAns	Number	Integer	Id Jawaban
3	IdRuleChild	Text	50	Nama Parameter Child
4	NoAnsChild	Number	Integer	Id Jawaban Child
5	No	Number	Integer	No Urut

4. Nama Tabel : Karyawan

Primary Key : NIK

Foreign Key : -

Keterangan : Tabel Karyawan digunakan untuk menyimpan data karyawan yang aktif maupun tidak aktif.

Tabel 3.6 Tabel Karyawan

NO	Field Name	Type	Length	Description
1	NIK	Text	10	Nomor induk karyawan
2	Nama	Text	50	Nama karyawan
3	Jabatan	Text	50	Jabtan yang dimiliki

5. Nama Tabel : admin

Primary Key : User, NIK

Foreign Key : NIK

Keterangan : Tabel User untuk menyimpan data user yang berhak atau dapat menjalankan aplikasi ini.

Tabel 3.7 Tabel User

NO	Field Name	Type	Length	Description
1	User	Text	50	Nama user
2	Pass	Text	50	Pasword user
3	NIK	Text	50	Nik user
4	Nama	Text	10	Nama Asli user
5	Jabatan	Text	50	Jabatan user
6	HakAkses	Text	50	Jenis user

6. Nama Tabel : FullPath

   Primary Key : -

   Foreign Key : -

   Keterangan : Tabel FullPath untuk menyimpan path yang digunakan untuk menampilkan susunan tree dalam aplikasi.

Tabel 3.8 Tabel FullPath

NO	Field Name	Type	Length	Description
1	fullpath	Text	50	Path yang digunakan untuk menampilkan treeview

7. Nama Tabel : Absensi

   Primary Key : IdRule, NoAns, IdRuleChild, NoAnsChild, No

   Foreign Key : -

   Keterangan : Untuk menyimpan data absensi dari mesin ceklok.

Tabel 3.9 Tabel Absensi

NO	Nama Field	Type	Lebar	PK	Keterangan
1	NIK	Text	10	PK	Nik karyawan
2	Tanggal	Date/Time	8	PK	Tanggal kerja
3	JamMasuk	Date/Time	8		Jam masuk karyawan
4	JamKeluar	Date/Time	8		Jam pulang karyawan
5	Status	Text	50		Keterangan masuk/ijin
6	Keterangan	Text	50		Penjelasan ijin
7	Hari Libur	Text	50		Hari yang dibuat khusus

8. Nama Tabel : Result

Primary Key : NIK, Tanggal

Foreign Key : NIK

Keterangan : Tabel Result untuk menyimpan hasil perhitungan upah lembur dan keterangan kontrak.

Tabel 3.10 Tabel Result

NO	Field Name	Type	Length	Description
1	NIK	Text	10	Nik karyawan
2	Nama	Text	50	Nama karyawan
3	Bulan	Text	10	Bulan karyawan yang sedang aktif
4	Tahun	Text	4	Tahun kerja karyawan
5	UpahLembur	Currency	10	Gaji lembur karyawan
6	Status	Text	50	Status kontrak karyawan

9. Nama Tabel : dailyResult

Primary Key : NIK

Foreign Key : -

Keterangan : Tabel Result untuk menyimpan hasil perhitungan upah lembur dan keterangan kontrak.

Tabel 3.11 Tabel Result

NO	Field Name	Type	Length	Description
1	NIK	Text	10	Nik karyawan
2	Nama	Text	50	Nama karyawan
3	Jabatan	Text	10	Jabatan karyawan yg sedang aktif
4	Tanggal	Date	8	Tanggal Lembur
5	Shift	Text	4	Shift pada tanggal yg tercantum
6	JamLembur	Integer	5	Jumlah jam lembur
7	UpahLembur	Text	10	Total upah lembur per hari
8	KetTerlambat	Text	50	Keterangan terlambat
9	KetPulang	Text	50	Keterangan Pulang lebih cepat

BAB IV

IMPLEMENTASI DAN EVALUASI

4.1 Implementasi

Dalam tahap ini dijelaskan mengenai implementasi perangkat lunak yang dibangun, dikembangkan menggunakan pemrograman Microsoft Visual Basic 6.0 dengan *Microsoft Access* 2002 sebagai *database*.

Aplikasi g*enerating rule* merupakan aplikasi yang terdapat didalam *interface* pakar, namun untuk dapat melakukan uji coba terhadap *rule base system* yang dihasilkan dari aplikasi *generating rule* tersebut, maka dibangun pula *interface user* untuk dapat mengevaluasi apakah *rule base system* yang dihasilkan telah sesuai dengan yang diharapkan. Untuk itu akan digunakan data pasti yaitu data-data absensi karyawan selama satu tahun.

4.1.1 Kebutuhan Sistem

Agar dapat berjalan sesuai dengan yang diharapkan Aplikasi Perhitungan Lembur untuk Sistem Berbasis Aturan ini memerlukan *Software* sebagai berikut :

a. Sistem Operasi Windows 98/Me/2000/Xp
b. Aplikasi program adalah Microsoft Visual Basic 6.0
c. *Database* untuk mengolah data adalah Microsoft Access 2002

Dan *Hardware Minimum* yang digunakan yaitu:

a. Processor Pentium II atau lebih
b. Memory 64 Mb
c. Harddisk 10 Gb
d. Vga 32 Mb

e. Monitor

f. Keyboard dan Mouse

4.1.2 Instalasi Program dan Pengaturan Sistem

Pengembangan aplikasi *Generating Rule* untuk Sistem Berbasis Aturan ini membutuhkan perangkat lunak yang sudah terinstalasi, adapun tahapan-tahapan instalasi dan pengaturan sistem, yaitu :

a. Install sistem operasi *Windows* 98/Me/2000/Xp.

b. Install aplikasi program *Microsoft Visual Basic* 6.0.

c. Install aplikasi *database Microsoft Access* 2002.

d. Install aplikasi laporan *Crystal Report* 8.5.

4.1.3 Penjelasan Pemakaian Program

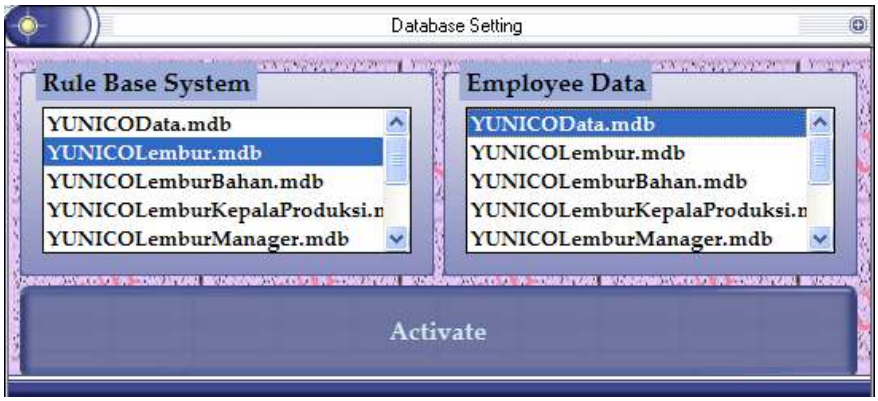
Didalam Aplikasi *Generating Rule* untuk sistem berbasis aturan ini terdapat beberapa *interface,* dimana tiap-tiap *interface* tersebut memiliki peran masing-masing didalam sistem ini, *interface-interface* tersebut yaitu:

1. Tampilan Database

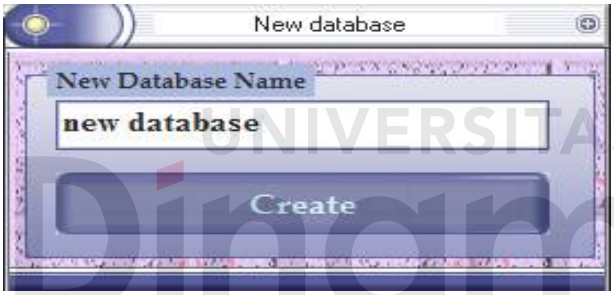
Form database ada dua, yang pertama form *set database* seprti pada gambar 4.1 untuk melakukan set *database* pada data pakar dan karyawan. Data pakar berisi tabel-tabel yang mengatur *rule* dan kondisi dari sistem berbasis aturan, database pakar ini juga berisi hasil dari proses sistem berbasis aturan. Data karyawan berisi tabel absensi dan tabel karyawan.

Form yang kedua adalah *form create database* yang dapat dilihat pada gambar 4.2, *form* ini berfungsi untuk membuat *database* baru dari sistem berbasis aturan(pakar). Database yang dihasilkan berupa file *access* (.mdb). Pada saat

pembuatan *database* tabel-tabel yang diperlukan untuk pembuatan sistem berbasis aturan sekaligus juga dibuat.



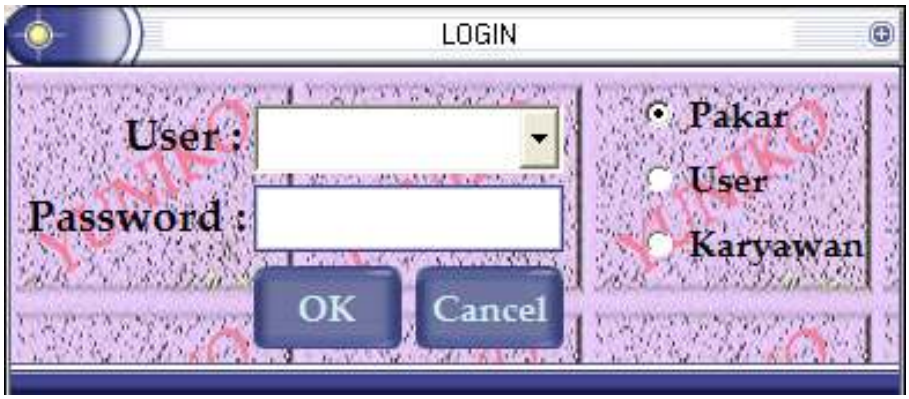
Gambar 4.1 *Set Database*



Gambar 4.2 Buat *Database*

2. Tampilan Login

Tampilan *login* pada sistem ini, dapat dilihat pada Gambar 4.3.

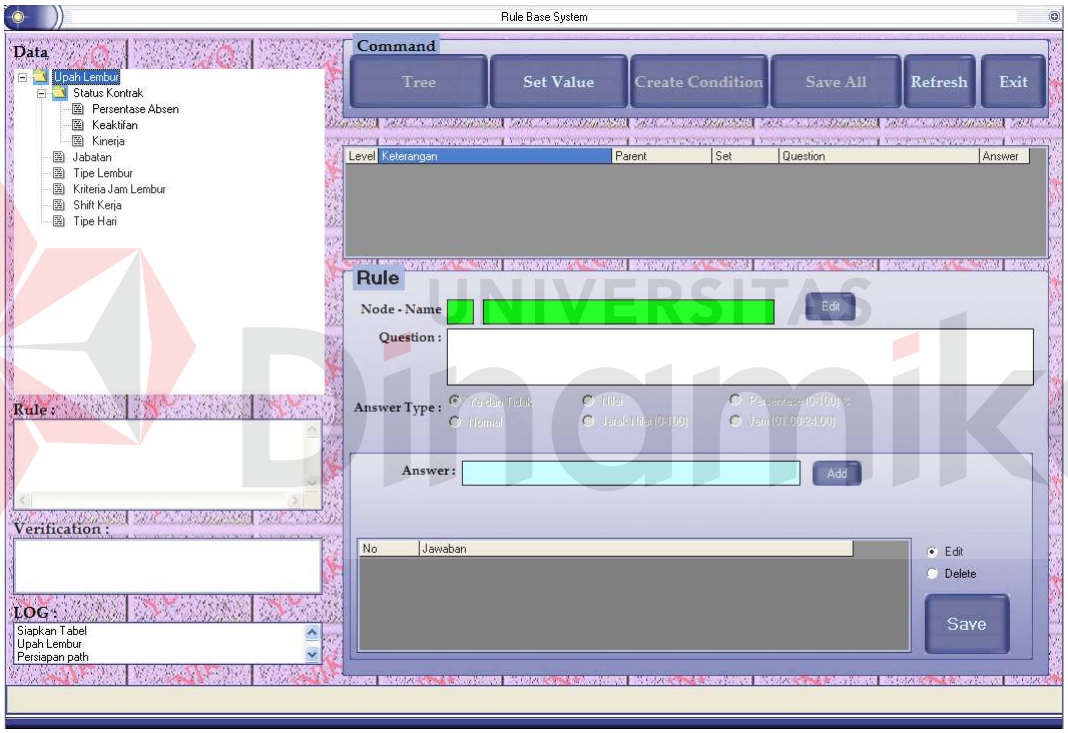


Gambar 4.3 Tampilan Login

Tampilan *login* pada Gambar 4.3 ini digunakan sebagai sekuriti data pada *interface* pakar agar parameter-parameter yang terdapat di dalam hanya dapat dimaintain oleh seorang pakar.

3. Tampilan Pakar untuk Aturan TreeView

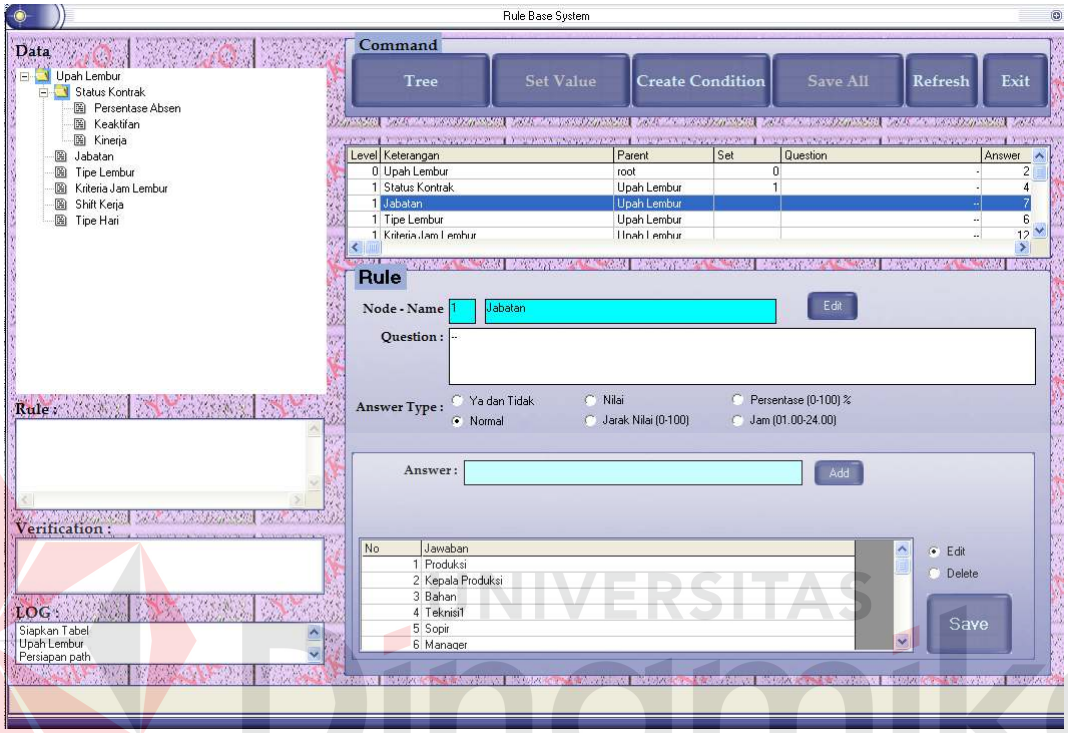
Langkah awal membuat desain *TreeView* adalah dengan membuat *rancangan node* baru terlebih dahulu, yang digunakan dalam proses penentuan parameter dan pembuatan *rule base system*, seperti Gambar 4.4.



Gambar 4.4 Desain *TreeView*

Penentuan parameter pada Gambar 4.5 dapat diubah dan dihapus sesuai dengan kebutuhan. Setiap *node parent* harus memiliki satu buah pertanyaan dan beberapa buah jawaban (*option*). *Node child* hanya memiliki beberapa jawaban saja. Jawaban dapat berupa kata-kata, nilai angka, persentase, jangkauan waktu dan jangakauan nilai. Berbagai macam jenis jawaban ini diperlukan dalam

pencarian upah lembur karyawan melalui *rule base system*. Pertanyaan dan jawaban yang telah dibuat dapat diperbaiki ataupun dihapus jika terdapat kesalahan.



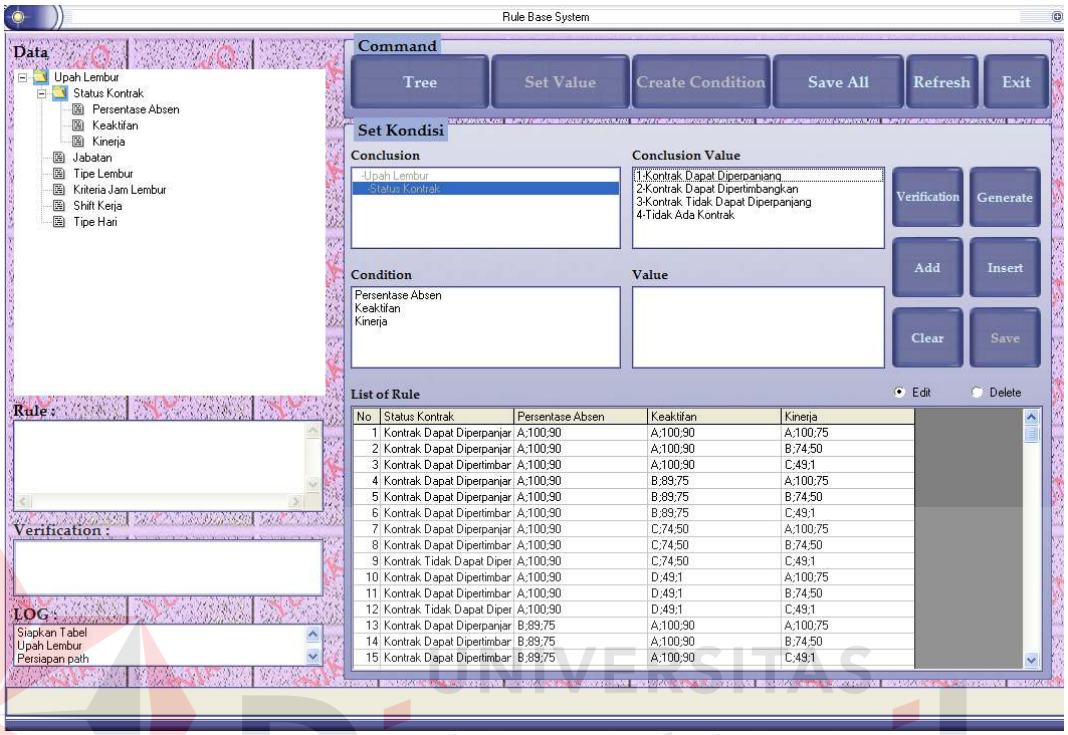
Gambar 4.5 Desain Pertanyaan

3. Tampilan Pakar untuk Generating Rule

Pada tampilan berikut adalah *list* aturan yang dihasilkan dengan melakukan *generate* dari desain *Treeview*. *Rule-rule* yang ditampilkan terbagi dalam tiap-tiap *set* seperti pada Gambar 4.6.

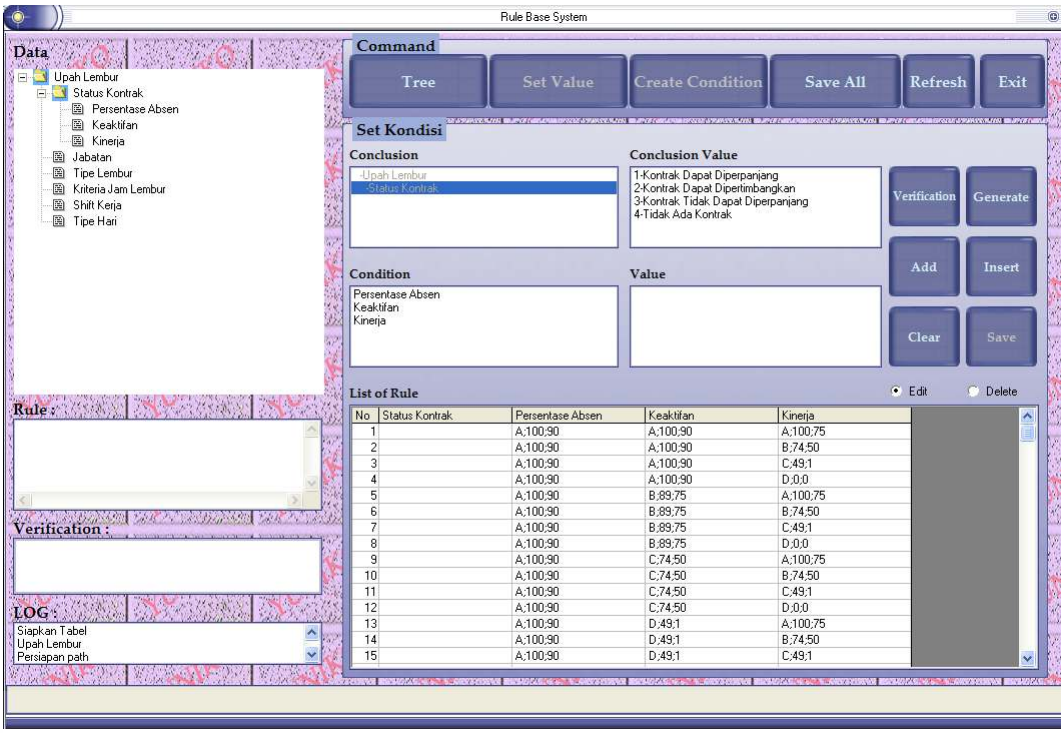
Selain ditampilkan *rule-rule* hasil dari *generate*, pada Gambar 4.6 juga ditampilkan informasi bagaimana kondisi dari *set* tersebut. G*enerating rule* dari desain *Treeview* mejadi kumpulan aturan-aturan yang digunakan dalam proses selanjutnya adalah menentukan aturan-aturan mana yang dipakai dalam

knowledge base system sekaligus menentukan konklusi dari aturan tersebut, seperti Gambar 4.6.



Gambar 4.6 Menentukan *Set* Kondisi

Pada gambar 4.6, pakar dapat mencari dan menampilkan *rule* lain selain yang ada dalam *set* saat itu. Disediakan juga *tool* untuk menambah dan menghapus kondisi yang salah atau tidak diperlukan. Jika terdapat kesalan pada salah satu aturan dapat dilakukan proses perbaikan agar tidak perlu menghapus atau mengganti data, sehingga urutan dari aturan dapat dipertahankan. Setelah pakar menentukan konklusi dari semua *rule-rule* yang diperlukan maka proses selanjutnya yaitu *save,* kemudian pakar akan mendapat pesan data telah tersimpan, jika terdapat kesalahan pada proses verifikasi atau terdapat data yang kosong maka sistem akan menampilkan pesan *error* dan proses *save* dibatalkan.



Gambar 4.7 *Generating Rule*

Selain itu disediakan juga proses *generatig rule*, proses ini akan menghasilkan *rule* secara otomatis sesuai dengan jumlah yang dihitung melalui proses *generating rule*.

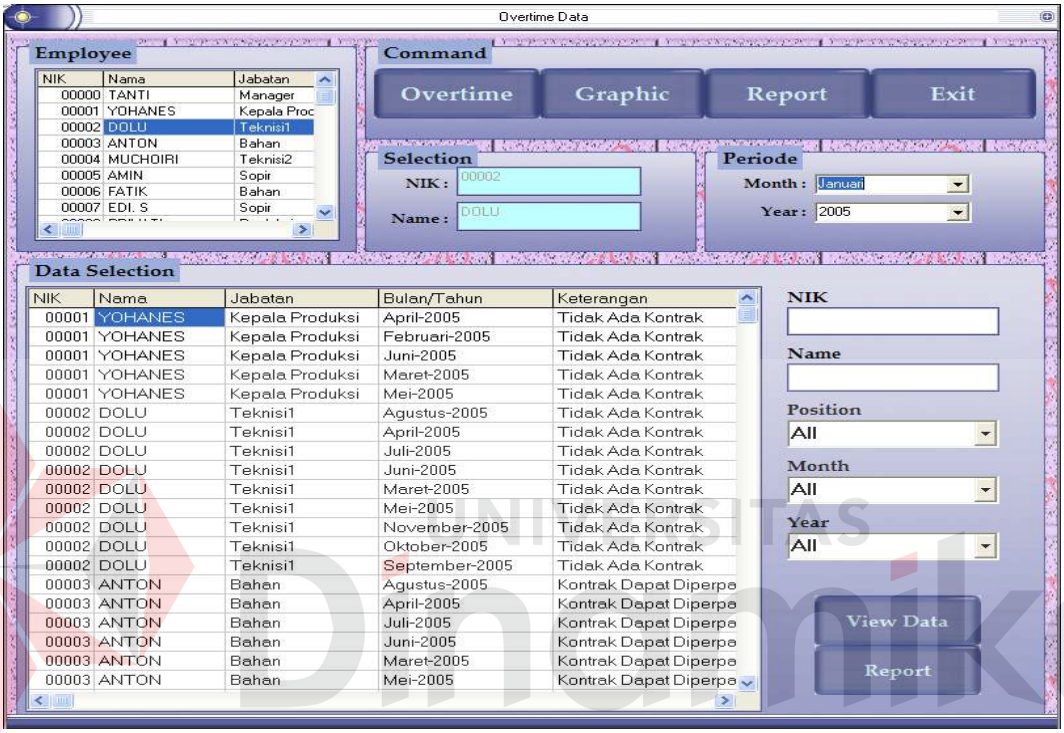
4. Tampilan Karyawan

Pada tampilan ini adalah proses perhitugan upah lembur yang dilakukan oleh *rule base system*, berdasarkan aturan yang telah ditentukan oleh pakar. Sebelum sistem melakukan perhitungan upah lembur maka pakar harus memilih data karyawan yang akan diproses lebih lanjut pada form perhitungan upah lembur, seperti pada Gambar 4.8.

Pakar juga dapat menampilkan data karyawan yang telah diproses melalui *form* perhitungan upah lembur pada bagian bawah *form* data karyawan.

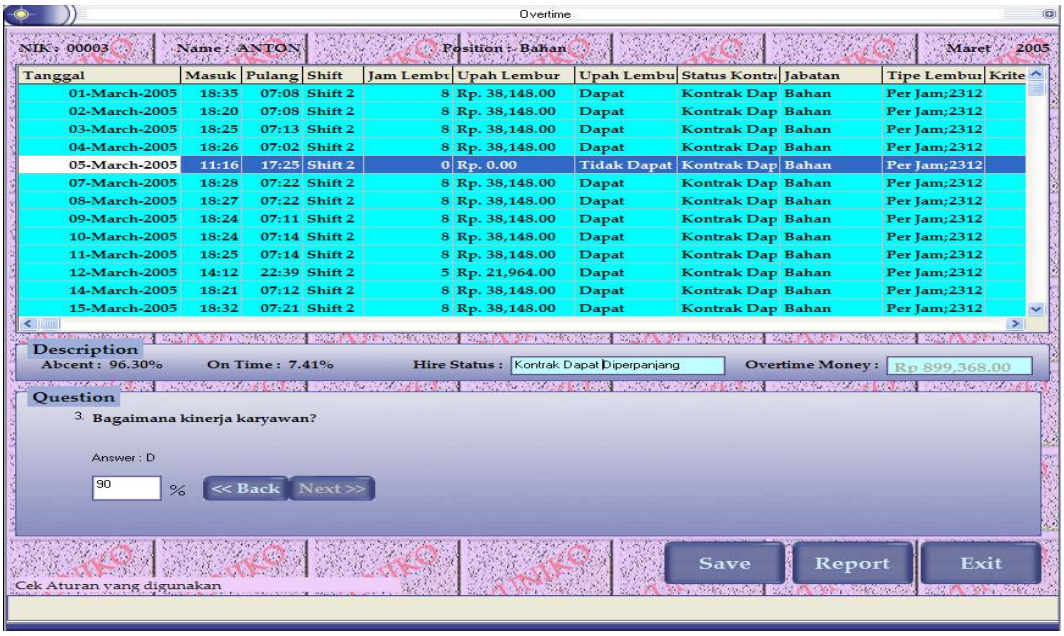
Proses selanjutnya adalah *proses* perhitungan upah lembur. Pada proses ini pakar menjawab semua pertanyaan yang diajukan oleh sistem, sebagai salah

satu bagian tambahan dari proses ini. Pada Gambar 4.9 pakar harus menjawab pertanyaan dengan lengkap untuk menetukan statu kontrak karyawan pada bulan depan. Hasil proses *input* jawaban akan ditampung pada *textbox* konfirmasi seperti pada Gambar 4.9.



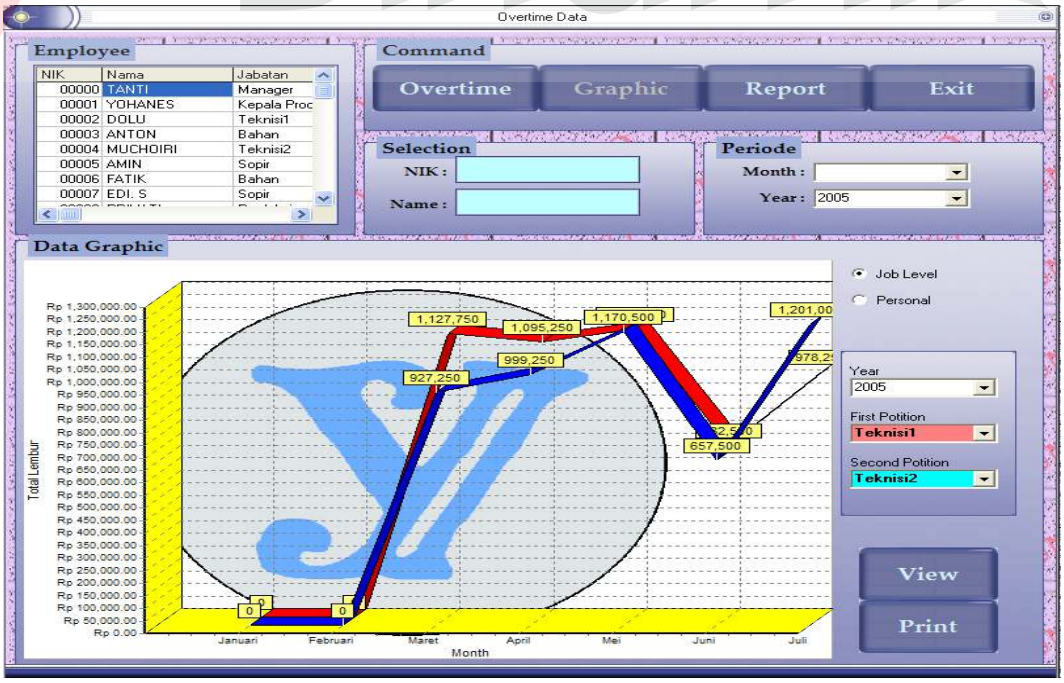
Gambar 4.8 *Select* Data Karyawan

Proses perhitungan upah lembur akan dilakukan secara otomatis oleh sistem sesuai dengan aturan yang telah dibuat pada *rule base system.* Tahap pertama dalam proses ini yaitu mencari shift dari jam kerja karyawan, setelah diketahui jenis shift-nya maka perhitungan jam lembur diproses sesuai dengan aturan yang telah dibuat. Langkah terakhir yaitu perhitungan upah lembur karyawan berdasarkan jam lembur dikalikan dengan nilai yang telah dicantumkan pada *list* aturan.



Gambar 4.9 *Input* Jawaban Pertanyaan

Gambar 4.10 menampilkan grafik data dari hasil perhitungan upah lembur karyawan. Grafik ini juga dapat digunakan untuk membandingkan antar jabatan(misalnya Produksi dan Bahan).



Gambar 4.10 *View Graphic*

User ada tiga macam yang pertama adalah pakar yang bertugas memaintain data aturan dalam sistem, kedua adalah user yang melakukan perhitungan lembur dan membuat laporan dan yang terakhir adalah karyawan. Karyawan hanya dapat melihat laporan hasil perhitungan lemburnya masing-masing. Gambar 4.11 menunjukan form pembuatan user, setiap orang hanya dapat memiliki satu user saja.

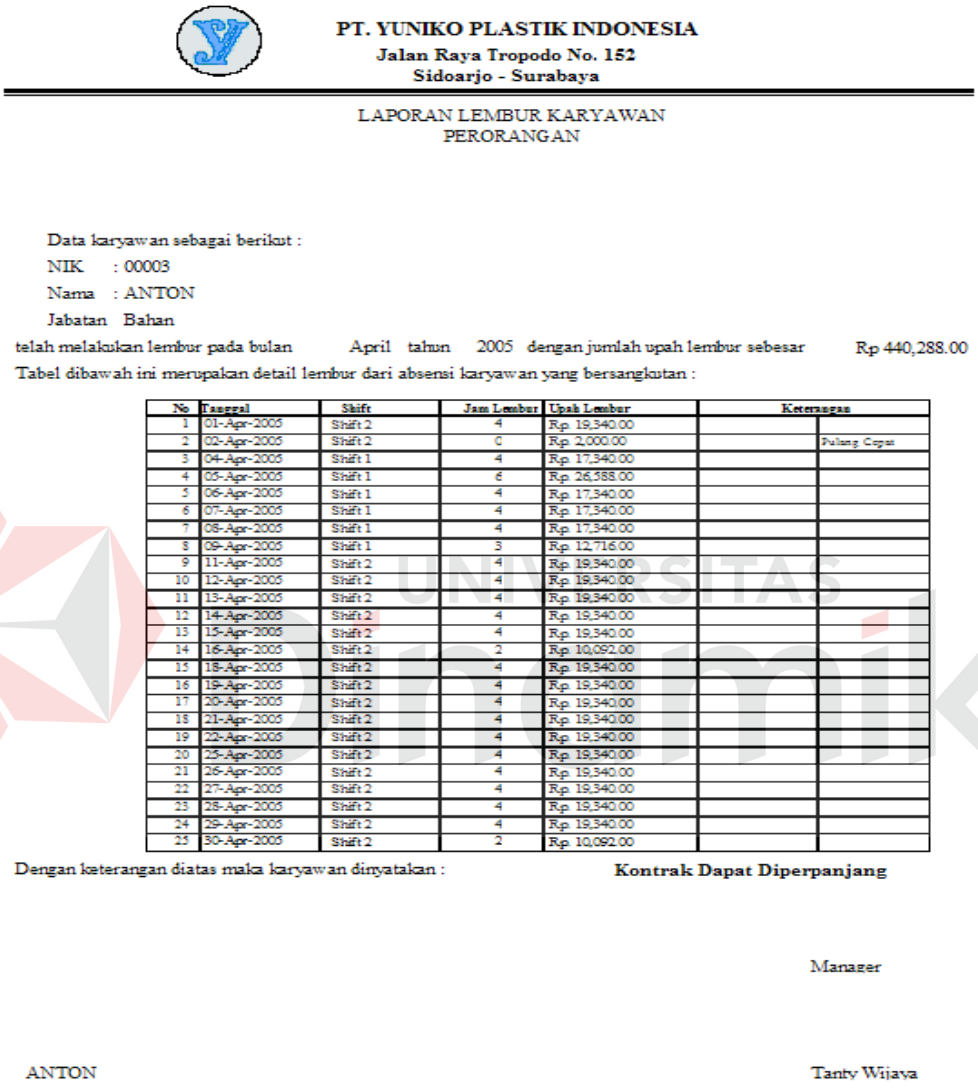
Gambar 4.11 *Maintenance User*

5. Laporan Hasil Konsultasi *User*

Pada Gambar 4.11 merupakan tampilan untuk laporan hasil perhitungan upah lembur karyawan.

Laporan pada Gambar 4.11 digunakan untuk mengetahui hasil dari perhitungan upah lembur dengan jelas dan dapat disertakan pada saat proses penggajian karyawan setiap akhir bulan.

Pada laporan ini disertakan *data-data* perhari dari karyawan agar karyawan dapat membaca dan mempelajari *data-data* pribadinya dalam kurun waktu satu bulan yang lalu.



PT. YUNIKO PLASTIK INDONESIA
Jalan Raya Tropodo No. 152
Sidoarjo - Surabaya

LAPORAN LEMBUR KARYAWAN
PERORANGAN

Data karyawan sebagai berikut :
NIK : 00003
Nama : ANTON
Jabatan Bahan
telah melakukan lembur pada bulan April tahun 2005 dengan jumlah upah lembur sebesar Rp 440,288.00
Tabel dibawah ini merupakan detail lembur dari absensi karyawan yang bersangkutan :

No	Tanggal	Shift	Jam Lembur	Upah Lembur	Keterangan	
1	01-Apr-2005	Shift 2	4	Rp 19,340.00		
2	02-Apr-2005	Shift 2	0	Rp 2,000.00		Pulang Cepat
3	04-Apr-2005	Shift 1	4	Rp 17,340.00		
4	05-Apr-2005	Shift 1	6	Rp 26,588.00		
5	06-Apr-2005	Shift 1	4	Rp 17,340.00		
6	07-Apr-2005	Shift 1	4	Rp 17,340.00		
7	08-Apr-2005	Shift 1	4	Rp 17,340.00		
8	09-Apr-2005	Shift 1	3	Rp 12,716.00		
9	11-Apr-2005	Shift 2	4	Rp 19,340.00		
10	12-Apr-2005	Shift 2	4	Rp 19,340.00		
11	13-Apr-2005	Shift 2	4	Rp 19,340.00		
12	14-Apr-2005	Shift 2	4	Rp 19,340.00		
13	15-Apr-2005	Shift 2	4	Rp 19,340.00		
14	16-Apr-2005	Shift 2	2	Rp 10,092.00		
15	18-Apr-2005	Shift 2	4	Rp 19,340.00		
16	19-Apr-2005	Shift 2	4	Rp 19,340.00		
17	20-Apr-2005	Shift 2	4	Rp 19,340.00		
18	21-Apr-2005	Shift 2	4	Rp 19,340.00		
19	22-Apr-2005	Shift 2	4	Rp 19,340.00		
20	25-Apr-2005	Shift 2	4	Rp 19,340.00		
21	26-Apr-2005	Shift 2	4	Rp 19,340.00		
22	27-Apr-2005	Shift 2	4	Rp 19,340.00		
23	28-Apr-2005	Shift 2	4	Rp 19,340.00		
24	29-Apr-2005	Shift 2	4	Rp 19,340.00		
25	30-Apr-2005	Shift 2	2	Rp 10,092.00		

Dengan keterangan diatas maka karyawan dinyatakan : **Kontrak Dapat Diperpanjang**

Manager

ANTON

Tanty Wijaya

Gambar 4.12 Laporan Hasil Perhitungan Upah Lembur

Laporan Gambar 4.13 mencetak data-data upah lembur karyawan secara keseluruhan tergantung dari kebutuhan *administrator*.

4.2 Evaluasi

Untuk pengujian logika dan evaluasi hasil akhir pada proses *generating rule* yaitu kecepatan proses *generating rule* dipengaruhi oleh banyaknya jumlah *rule* yang harus ditampilkan. Data yang didapatkan pada saat mengevaluasi sistem ini yaitu:

1. Sistem ini dapat menunjukan tingkat kerajinan dan keaktifan dari seorang karyawan.
2. Sistem ini dapat melakukan perhitungan upah lembur secara akurat dan lebih cepat dibandingkan dengan perhitungan *manual*.
3. Kecepatan proses *generating rule* ketika menampilkan *rule* sebanyak 112.896 *rule* memerlukan waktu 120 detik.
4. Kecepatan proses ketika menampilkan *rule* sebanyak 344 *rule* memerlukan waktu 10 detik.
5. Proses untuk menghitung upah lembur karyawan per bulan rata-rata 3 detik. waktu 3 detik ini masih cukup lama untuk proses komputer tetapi jauh lebih cepat dibandingkan sistem manual.
6. Kecepatan proses bergantung pada kemampuan komputer yang digunakan, memori RAM dan kapasitas prosesor juga berpengaruh.
7. Dari beberapa percobaan pembuatan rule secara manual terajadi beberapa kasus *Redundant rule* dan *Conflicting rule*, dua proses *verifikasi* inilah yang sering muncul karena kesalahan pakar. *Subsumed rule* dan *Circular rule* jarang sekali terjadi, hampir tidak ada kecuali pada waktu pakar melakukan tes terhadap *verifikasi*.

BAB V

PENUTUP

5.1 Kesimpulan

Secara umum proses pengembangan aplikasi Perhitungan Upah Lembur dengan *rule base system* ini, telah berfungsi sebagaimana yang diharapkan. Kesimpulan yang dapat diambil dari sistem ini sebagai berikut:

1. Dengan g*enerating rule* atau pembuatan *rule* secara *manual* maka aturan perhitungan upah lembur menjadi lebih transparan, dengan adanya laporan perhitungan upah lembur ini maka efektivitas dan effisiensi seorang karyawan dapat dinilai berdasarkan dari data yang ada.
2. Dengan adanya sistem ini maka kesalahan dalam perhitungan lembur dapat minimalkan. Dibandingkan dengan proses *manual* yang terkesan lambat dan banyak kesalahan pencatatan maka sistem berbasis aturan ini akan sangat membantu.

5.2 Saran

Adapun saran-saran untuk pengembangkan sistem ini adalah sistem ini diharapkan dapat melakukan perhitungan gaji karyawan dan tunjangan-tunjangan yang lain.

DAFTAR PUSTAKA

Gonzalez, Avelino J., 1993, *The Engineering of Knowledge-Based systems: Theory and Practice*, Prentice Hall, Englewood Cliffs, New Jersey.

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

Dan W. Patterson, 1990, *Introduction to Artificial Intelligence and Expert Systems,* Prentice Hall, Englewood Cliffs, New Jersey ."

John Durkin, 1994, *Expert Systems Desin and Development,* Prentice Hall Interntional, Inc., Englewood Cliffs, New Jersey ."

Buku Panduan Peraturan Perusahaan tentang "Kesepakatan Kerja Bersama.", 2005, Surabaya.

DEPNAKERTRANS, 2004, *KEPMEN NO. 102 TH 2004,* 21 Agustus 2006. URL : http://www.nakertrans.go.id/perundangan/kepmen/kepmen_102_2004.php.