BAB II

TINJAUAN PUSTAKA

II.1. Konsep Sistem Pakar

II.1.1. Sistem Pakar

Sistem pakar menirukan perilaku seorang pakar dalam menangani suatu persoalan. Pada suatu kasus seorang pasien mendatangi dokter untuk memeriksa badannya yang mengalami gangguan kesehatan, maka dokter atau pakar kesehatan akan memeriksa dan melakukan diagnosa. Bila dokter cukup sibuk dan pelaksana diagnosa digantikan oleh sebuah sistem pakar, maka sistem pakar diharapkan dapat membantu memahami dan menganalisa keadaan pasien dan menemukan penyakit yang diderita pasien itu. Sistem pakar diharapkan juga untuk menghasilkan dugaan atau hasil diagnosa yang sama dengan diagnosa yang dilakukan oleh seorang ahli. Tujuan utama sistem pakar bukan untuk menggantikan kedudukan seorang ahli maupun pakar, tetapi untuk memasyarakatkan pengetahuan dan pengalaman pakar-pakar yang ahli di bidangnya (Jurnal Teknologi dan Informatika : Andri Saputra ; 2011 : 202-206).

II.1.2. Tujuan Sistem Pakar

Tujuan utama sistem pakar bukan untuk menggantikan kedudukan seorang ahli maupun pakar, tetapi untuk memasyarakatkan pengetahuan dan

pengalaman pakar-pakar yang ahli di bidangnya. Sistem pakar disusun oleh dua bagian utama, yaitu :

1. Lingkungan pengembangan (*development environment*), digunakan untuk memasukkan pengetahuan pakar ke dalam lingkungan sistem pakar.
2. Lingkungan konsultasi (*consultation environment*), digunakan oleh pengguna yang bukan pakar guna memperoleh pengetahuan pakar (Jurnal Teknologi dan Informatika : Andri Saputra ; 2011 : 202-206).

II.1.3. Komponen Sistem Pakar

Sistem pakar memiliki 2 komponen utama yaitu basis pengetahuan dan mesin inferensi. Basis pengetahuan merupakan tempat penyimpanan pengetahuan dalam memori komputer, dimana pengetahuan ini diambil dari pengetahuan pakar. komponen-komponen sistem pakar adalah seperti di bawah ini :

1. Antarmuka (*User Interface*)

   User Interface merupakan mekanisme yang digunakan oleh pengguna dan sistem pakar untuk berkomunikasi. Antarmuka menerima informasi dari pemakai dan mengubahnya kedalam bentuk yang dapat diterima oleh sistem.

2. Basis Pengetahuan

   Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah. Komponen sistem pakar ini disusun atas dua elemen dasar, yaitu fakta dan aturan.

3. Akuisisi Pengetahuan (*Knowledge Acquisition*)

   Akuisisi pengetahuan adalah akumulasi, transfer dan transformasi keahlian dalam menyelesaikan masalah dari sumber pengetahuan kedalam program komputer.

4. Mesin Inferensi

   Komponen ini mengandung mekanisme pola pikir dan penalaran yang digunakan oleh pakar dalam menyelesaikan suatu masalah.

5. *Workplace*

   Workplace merupakan area dari sekumpulan memori kerja (*working memory*). *Workplace* digunakan untuk merekam hasil-hasil antara dan kesimpulan yang dicapai.

6. Fasilitas Penjelasan

   Fasilitas penjelasan adalah komponen tambahan yang akan meningkatkan kemampuan sistem pakar.

7. Perbaikan Pengetahuan

   Pakar memiliki kemampuan untuk menganalisis dan meningkatkan kinerjanya serta kemampuan untuk belajar dari kinerjanya (Jurnal Teknologi dan Informatika : Andri Saputra ; 2011 : 202-206).

II.2. Implementasi Metode

II.2.1. Metode Hill Climbing

Metode *Hill Climbing Search* adalah salah satu metode pencarian dalam menentukan diagnosa penyakit yang singkat dengan memperkecil jumlah keadaan yang disinggahinya tanpa harus mengecek pada node sesudahnya. Metode *Hill Climbing* merupakan variasi dari metode DFS. Dengan metode ini, eksplorasi terhadap keputusan dilakukan dengan cara DFS dengan mencari *path* yang bertujuan menurunkan *cost* untuk menuju kepada goal/kesimpulan. Dalam prosedur *Hill Climbing*, fungsi uji dikombinasikan dengan fungsi heuristik. Dimana fungsi heuristik disini menggunakan ukuran keyakinan dari pasien (*user*) untuk menentukan diagnosa awal penyakitnya atau tes yang berupa fungsi heuristik ini akan menunjukkan seberapa baiknya nilai terkaan (estimasi) yang diambil terhadap keadaan-keadaan lainnya yang mungkin. Cara kerjanya adalah menentukan langkah berikutnya dengan menempatkan node yang akan muncul sedekat mungkin dengan sasarannya dengan menggunakan fungsi heuristic, yaitu berupa bobot yang didapat dari dokter ahli pada gejala dengan angka presentase dari 1 hingga 100 (Suci Oktaviana ; 2012 : 3).

II.3. Bahasa Pemrograman

II.3.1. Visual Basic 2010

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai jenis komputer dan berbagai sistem operasi termasuk telepon genggam. Java dikembangkan oleh Sun Microsystem dan dirilis tahun 1995. Java merupakan

suatu teknologi perangkat lunak yang digolongkan multi platform. Selain itu, Java juga merupakan suatu platform yang memiliki virtual machine dan library yang diperlukan untuk menulis dan menjalankan suatu program.

Bahasa pemrograman java pertama lahir dari The Green Project, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan versi yang dinamakan Oak. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, serta Sembilan pemrograman lainnya dari Sun Microsystem. Salah satu hasil proyek ini adalah mascot Duke yang dibuat oleh Joe Palrang (Wahana Komputer ; 2010 : 1).

II.4.1. MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa inggris; *database management system*) data DBMS yang *multithread*, MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Tidak seperti Apache yang merupakan *software* yang dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial swedia yaitu MySQL AB. MySQL AB memegang penuh hak cipta hampir atas semua kode sumbernya (Achmad Solichin ; 2010 : 2)

II.4. Model Perancangan Sistem

II.5.1. UML (*Unified Modelling Language)*

Menurut Sri Dharwiyanti (2013) *Unified Modelling Language* (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

1. *Use Case* Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan system untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

2. *Class* Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

3. *Statechart* Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

4. *Activity* Diagram

Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas

5. *Sequence* Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah

yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output*

tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message.

6. *Collaboration* Diagram

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*.

Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama.

7. *Component* Diagram

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

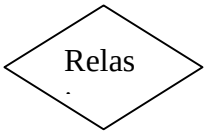
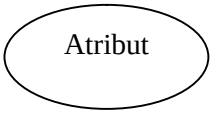
8. *Deployment* Diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya.

II.5.2. ERD

Entity relationship (ER) data model didasarkan pada persepsi terhadap dunia nyata yang tersusun atas kumpulan objek-objek dasar yang disebut entitas dan hubungan antarobjek. Entitas adalah sesuatu atau objek dalam dunia nyata yang dapat dibedakan dari objek lain. Misal: mahasiswa, dan matakuliah. Entitas digambarkan dalam basis data dengan kumpulan atribut. Misalnya: nim, nama, alamat, dan kota. Relasi adalah hubungan antara beberapa entitas. Misalnya: relasi menghubungkan mahasiswa dengan mata kuliah yang diambilnya. Struktur logis (skema database) dapat ditunjukkan secara grafis dengan diagram ER yang dibentuk dari komponen-komponen berikut :

Tabel II.1. Simbol ERD

Notasi	Keterangan
	Entitas, adalah suatu objek yang dapat didentifikasi dalam lingkungan pemaki.
	Relasi, menunjukan adanya hubungan di antara sejumlah entitas yang berbeda.
	Atribut, berfungsi mendeskripsikan karakter entitas (atribut yang berfungsi sebagai key diberi garis bawah).
	Garis, sebagai penghubung antara relasi dengan entitas, relasi dan entitas dengan atribut.

(Sumber : Jurnal Teknologi dan Informatika ; D. Tri Octafian ; 2010 : 151)

II.5.3. Normalisasi

Normalisasi adalah proses mengubah relasi dari bentuk tidak normal menjadi bentuk normal atau proses untuk mengidentifikasi dan menghilangkan anomali. Proses ini dilakukan dengan memecah sebuah relasi menjadi beberapa relasi lain yang lebih kecil, relasi yang dihasilkan memiliki jumlah atribut lebih sedikit. Tiga bentuk normal yaitu bentuk normal pertama (1NF), bentuk normal kedua (2NF), bentuk normal ketiga (3NF). Tetapi dalam perkembangan muncul bentuk-bentuk normal yang baru. Definisi tentang bentuk normal sebagai berikut:

1. Tahap tidak normal

   Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti format tertentu, dapat saja tidak lengkap dan terduplikasi. Data dikumpulkan apa adanya sesuai keadaanya.

Tabel II.2. Tidak Normal

No_mhs	Nama	Prg_Studi	Kode_mk	Nama_mk	SKS	Kd_Dsn	Dosen
0231	Cahyo	1 komputer	PAM211	Kalkulus Lanjut 1	3	MT002	Yasir
			PAAM261	Prg Terstruktur 1	3	IK003	Kamal

Sumber : Tawar ; 2011 : 7

2. Tahap normal tahap pertama (1” *Normal Form*)

   Sebuah table disebut 1NF jika :

   a. Tidak ada baris yang duplikat dalam tabel tersebut.

   b. Masing-masing cell bernilai tunggal

Tabel II.3. Normalisasi 1 NF

MHS(No_mhs, Nama, Prg_Studi)

No mhs	Nama	Prg Studi
0231	Cahyo	I Komp.
0232	Hoho	Statistik
0233	Budi	Matematika

DAFTAR MK(No mhs, Kode mk, Nama Mk, SKS, Kd Dsn, Dosen)

No mhs	Kode mk	Nama mk	SKS	Kd Dsn	Dosen
0231	PAM211	Kalkulus Lanjut I	3	MT002	Yasir
0231	PAAM261	Prg. Terstruktur I	3	IK003	Kamal
0231	PAM367	Simulasi	3	IK003	Jack
0232	PAM333	Prg. Linier	3	MT003	Andri
0232	PAM241	Met. Statistik I	3	ST002	Fendi
0232	PAM345	Analisis Data	3	ST003	Hasbi
0233	PAM337	Fungsi Khas	3	MT001	Jaya
0233	PAM522	Topologi	3	MT003	Andri

Sumber : Tawar ; 2011 : 8

3. Tahap normal tahap kedua (2nd *normal form*)

   Bentuk normal kedua (2NF) terpenuhi jika pada sebuah tabel semua atribut yang tidak termasuk dalam primary key memiliki ketergantungan fungsional pada primary key secara utuh.

Tabel II.4. Normalisasi 2NF

AMBIL (No_mhs, Kode_mk)

No_mhs	Kode_mk
0231	PAM211
0231	PAAM261
0231	PAM367
0232	PAM333
0232	PAM241
0232	PAM345
0233	PAM337
0233	PAM322
0233	PAM432

PENGAJAR (Kode_mk, Nama_mk, SKS, Kd_Dsn, Dosen)

Kode_mk	Nama_mk	SKS	Kd_Dsn	Dosen
PAM211	Kalkulus Lanjut I	3	MT002	Yuni
PAAM261	Prg. Terstruktur I	3	IK005	Kamal
PAM367	Simulasi	3	IK002	Jack
PAM333	Prg. Linier	3	MT003	Andri
PAM241	Met. Statistik I	3	ST002	Dendi

Sumber : Tawar ; 2011 : 9

4. Tahap normal tahap ketiga (3rd *normal form*)

   Sebuah tabel dikatakan memenuhi bentuk normal ketiga (3NF), jika untuk setiap ketergantungan fungsional dengan notasi X Y Z, dimana Y mewakili semua atribut tunggal di dalam tabel yang tidak ada di dalam X, maka :

   a. X haruslah superkey pada tabel tersebut.
   b. Atau Y merupakan bagian dari primary key pada tabel tersebut.

Tabel II.5. Normalisasi 3NF

KULIAH(Kode_mk, Nama_mk, SKS, Kd_Dsn)

Kode mk	Nama mk	SKS	Kd Dsn
PAM211	Kalkulus Lanjut I	3	MT002
PAAM261	Prg. Terstruktur I	3	IK003
PAM367	Simulasi	3	TK002
PAM333	Prg. Linier	3	MT003
PAM241	Met. Statistik I	3	ST002
PAM345	Analisis Data	3	ST003
PAM337	Fungsi Khas	3	MT001
PAM322	Topologi	3	MT003
PAM432	Teori Optimasi	3	MT004

Sumber : Tawar ; 2011 : 10

5. Boyce Code Normal Form (BCNF)

   a. Memenuhi 1st NF

   b. Relasi harus bergantung fungsi pada atribut superkey

Tabel II.6. Normalisasi BCNF

NILAI(No_mhs, No_Rkng, Kd_Mk, Nilai)

No mhs	Kode mk	Nilai
0231	PAM211	A
0231	PAAM261	B
0231	PAM367	A
0232	PAM333	C
0232	PAM241	A
0233	PAM345	B
0233	PAM337	A
0235	PAM322	B
0237	PAM432	B

REKENING(No_mhs, No_Rkng)

No_mhs	No_Rkng
0231	88681
0232	88682
0233	88683
0235	88685
0237	88687

Sumber : Tawar ; 2011 : 12

6. Tahap Normal Tahap Keempat dan Kelima

Penerapan aturan normalisasi sampai bentuk normal ketiga sudah memadai untuk menghasilkan tabel berkualitas baik. Namun demikian, terdapat pula bentuk normal keempat (4NF) dan kelima (5NF). Bentuk Normal keempat berkaitan dengan sifat ketergantungan banyak nilai (*multivalued dependency*) pada suatu tabel yang merupakan pengembangan dari ketergantungan fungsional. Adapun bentuk normal tahap kelima merupakan nama lain dari *Project Join Normal Form* (PJNF)

Tabel II.7. Normalisasi 4NF

BAHASA(No_Mhs, Prg_Studi, Bhs_Asing)

No_mhs	Prg_Studi	Bhs_Asing
0232	Komputer	Inggris
0232	Akuntasnsi	Jerman
0232	Komputer	Jerman
0232	Akuntasnsi	Inggris
0236	Statistik	Perancis
0236	Hukum	Belanda
0236	Statistik	Belanda
0236	Hukum	Perancis

BAHASA2(No_Mhs, Prg_Studi, Bhs_Asing)

No_mhs	Prg_Studi	Bhs_Asing
0232	Komputer	Inggris
0232	Akuntasnsi	Jerman

Sumber : Tawar ; 2011 : 13