BAB II

TINJAUAN PUSTAKA

II.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.

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.

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.

Ada banyak keuntungan bila menggunakan sistem pakar, diantaranya adalah :

1. Menjadikan pengetahuan dan nasehat mudah didapat.
2. Meningkatkan *output* dan produktivitas.
3. Menyimpan kemampuan dan keahlian pakar.
4. Meningkatkan penyelesaian masalah, menerusi paduan pakar, penerangan, sistem pakar khas.
5. Meningkatkan reliabilitas.
6. Memberikan *respons* (jawaban) yang cepat.
7. Merupakan penduan yang *inteligence* (cerdas).
8. Dapat bekerja dengan informasi yang lengkap dan mengandung ketidakpastian.
9. *Intelligence database* (basis data cerdas), bahwa sistem pakar dapat digunakan untuk mengakses basis data dengan cara cerdas.

Selain keuntungan-keuntungan di atas, sistem pakar seperti sistem lainnya juga memiliki kelemahan, diantaranya adalah:

1. Masalah dalam mendapatkan pengetahuan dimana pengetahuan tidak selalu bisa didapatkan dengan mudah, kadangkala pakar dari masalah yang kita buat tidak ada, dan kalaupun ada kadang-kadang pendekatan yang dimiliki pakar berbeda-beda.
2. Untuk membuat suatu sistem pakar yang benar-benar berkualitas tinggi sangatlah sulit dan memerlukan biaya yang sangat besar untuk pengembangan dan pemeliharaannya.
3. Boleh jadi sistem tak dapat membuat keputusan.
4. Sistem pakar tidaklah 100% menguntungkan, walaupun seorang tetap tidak sempurna atau tidak selalu benar. Oleh karena itu perlu diuji ulang secara teliti sebelum digunakan. Dalam hal ini peran manusia tetap merupakan faktor dominan.

Sistem pakar merupakan program-program praktis yang menggunakan strategi *heuristic* yang dikembangkan oleh manusia untuk menyelesaikan permasalahan-permasalahan yang *spesifik* (khusus). Disebabkan oleh *keheuristikannya* dan sifatnya yang berdasarkan pada pengetahuan, maka umumnya sistem pakar bersifat :

1. Memiliki informasi yang handal, baik dalam menampilkan langkah-langkah antara maupun dalam menjawab pertanyaan-pertanyaan tentang proses penyelesaian.

2. Mudah dimodifikasi, yaitu dengan menambah atau menghapus suatu kemampuan dari basis pengetahuan.
3. Heuristik dalam menggunakan pengetahuan (yang sering kali tidak sempurna) untuk mendapatkan penyelesaiannya.
4. Dapat digunakan dalam berbagai jenis computer.
5. Memiliki kemampuan untuk belajar beradaptasi**.**

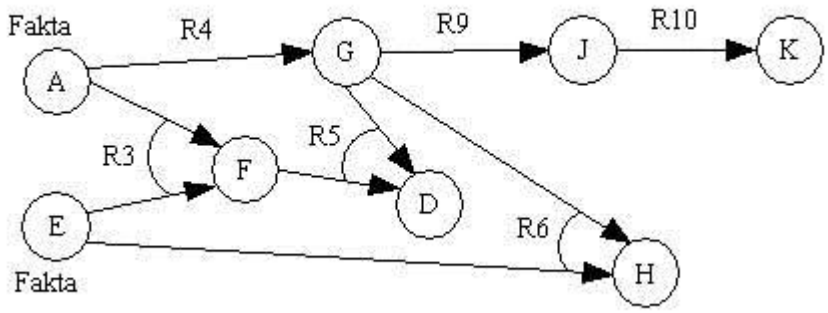
Basis pengetahuan (*knowladge base)* sebagian sistem pakar yang berisikan fakta-fakta yang menggambarkan wilayah masalah dan teknik-teknik refresentasi pengetahuan yang menggambarkan bagaimana fakta-fakta saling bersesuaian secara logis.

Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah. Komponen sistem pakar ini disusun atas dua elemen dasar, yaitu fakta dan aturan (*rule*). Fakta merupakan informasi tentang obyek dalam area permasalahan tertentu, sedangkan aturan merupakan informasi tentang cara bagaimana memperoleh fakta baru dari fakta yang telah diketahui.

Suatu perkalian inferensi yang menghubungkan suatu permasalahan dengan solusinya disebut rantai (*chain*). ada dua metode penalaran dengan *rules*, yaitu *forward chaining* atau *data-driven* dan *backward chaining* atau *goaldriven*.

1. *Forward Chaining*

   Suatu rantai yang dicari atau dilewati/dilintasi dari suatu permasalahan untuk memperoleh solusinya dengan penalaran dari fakta menuju konklusi yang terdapat dari fakta.

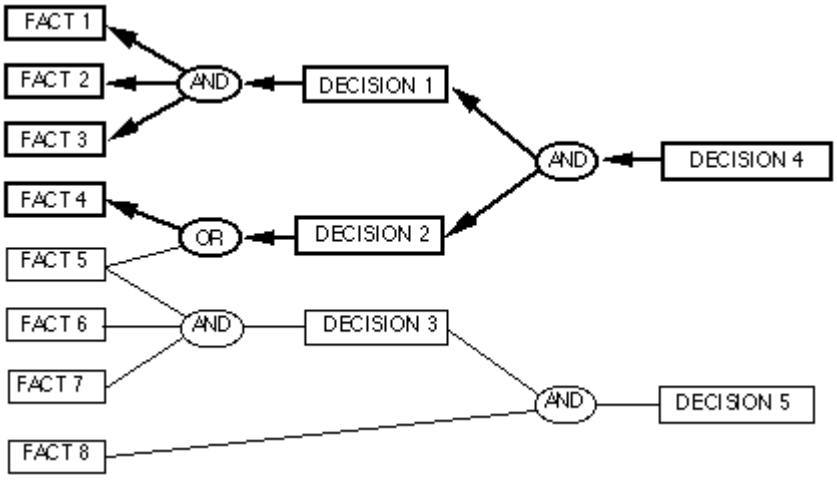


Gambar II.1. Forward Chaining

Sumber : Andri Saputra ; 2011 : 202-206

2. *Backward Chaining*

Suatu rantai yang dilintasi dari suatu hipotesa kembali ke fakta yang mendukung hipotesa tersebut, dan dalam hal tujuan yang dapat dipenuhi dengan pemenuhan sub tujuannya. (Jurnal Teknologi dan Informatika : Andri Saputra ; 2011 : 202-206).



Gambar II.2. Backward Chaining

Sumber : Andri Saputra ; 2011 : 202-206

II.2. Metode *Certainty Factor*

Certainty factor (CF) menunjukkan ukuran kepastian terhadap suatu fakta atau aturan. Faktor kepastian ini merupakan bentuk penggabungan kepercayaan dan ketidakpercayaan dalam suatu bilangan tunggal. *Certainty Theory* ini diusulkan oleh Shortliffe dan Buchanan pada tahun 1975 untuk mengakomadasi ketidakpastian pemikiran (*inexact reasoning*) seorang pakar. Teori ini berkembang bersamaan dengan pembuatan sistem pakar MYCIN. Team pengembang MYCIN mencatat bahwa tim ahli sering kali menganalisa informasi yang ada dengan ungkapan seperti misalnya: mungkin, kemungkinan besar, hampir pasti. Untuk mengakomodasi hal ini tim MYCIN menggunakan *Certainty factor* (CF) guna menggambarkan tingkat keyakinan pakar terhadap masalah yang sedang dihadapi.

Beberapa *evidence* dapat dikombinasikan untuk menentukan CF dari suatu hipotesis. Untuk sistem ini, tingkat kepastian sistem terhadap kesimpulan yang diperoleh dihitung berdasarkan nilai probabilitas penyakit karena adanya evident/gejala tertentu. Faktor kepastian bukanlah suatu peluang, tetapi merupakan ukuran tingkat kepercayaan terhadap suatu evidensi. CF mempresentasikan tingkat kepercayaan bahwa suatu evidensi adalah benar.

Certainty Theory ini diusulkan oleh Shortliffe dan Buchanan pada tahun 1975 untuk mengakomadasi ketidakpastian pemikiran (*inexact reasoning*) seorang pakar. Teori ini berkembang bersamaan dengan pembuatan sistem pakar MYCIN. Team pengembang MYCIN mencatat bahwa tim ahli sering kali menganalisa informasi yang ada dengan ungkapan seperti misalnya: mungkin, kemungkinan

besar, hampir pasti. Untuk mengakomodasi hal ini tim MYCIN menggunakan *certainty factor* (CF) guna menggambarkan tingkat keyakinan pakar terhadap masalah yang sedang dihadapi. Secara umum, rule direpresentasikan dalam bentuk sebagai berikut:

IF E1 [AND/OR] E2 [AND/O R] … En

THEN H (CF = CF)

Tabel II.1. Nilai CF Paralel

No	Nilai Certainty Factor	Kepastian Menurut Pakar
1	0	Pasti tidak (*Definitely not*)
2	0.1	Hampir pasti tidak (*Almost certainly not*)
3	0.2	Lebih mungkin tidak (*Probably not*)
4	0.3	Mungkin tidak (*Maybe not*)
5	0.4	Tidak tahu (*Unknown*)
6	0.5	Tidak Tahu (*Unknown*)
7	0.6	Mungkin (*Maybe*)
8	0.7	Lebih mungkin (*Probably*)
9	0.8	Hampir pasti (*Almost certainly*)
10	0.9	Pasti (*Definitely*)

Sumber : Bhaskara Adhi Pradhana ; 2013 : 4

II.2.1. Model Perhitungan *Certainty Factor* dengan *Rule*

Certainty Factor (CF) menunjukkan ukuran kepastian terhadap suatu fakta atau aturan. Faktor kepastian ini merupakan bentuk penggabungan kepercayaan dan ketidakpercayaan dalam suatu bilangan tunggal. Berikut notasi faktor kepastian :

CF(Pk, G) = MB(Pk,G) - MD(Pk, G)

Beberapa *evidence* dapat dikombinasikan untuk menentukan CF dari suatu hipotesis. Untuk sistem ini, tingkat kepastian sistem terhadap kesimpulan yang diperoleh dihitung berdasarkan nilai probabilitas penyakit karena adanya evident/gejala tertentu [5]. Jika ada gejala dan penyakit sebagai *hipothesis* maka tingkat kepastian diformulasikan sebagai *CF(Pk,G)*:

$$MB(Pk,G)=\begin{cases}1 & ,P(Pk)=1\\ \dfrac{\max[P(Pk|G),P(Pk)]-P(Pk)}{\max[1,0]-P(Pk)} & ,yanglain\end{cases}$$

$$MD(Pk,G)=\begin{cases}1 & ,P(Pk)=1\\ \dfrac{\min[P(Pk|G),P(Pk)]-P(Pk)}{\min[1,0]-P(Pk)} & ,yanglain\end{cases}$$

.....(1)

Sumber : Bain Khusnul, 2010 : 13

Di mana:

P(Pk) : probabilitas kerusakan Pk

G : Gejala

CF : Certainty Factor (faktor Kepastian) dalam hipotesis H yang dipengaruhi oleh evidence (fakta) E.

MB : Measure of Belief (tingkat keyakinan), merupakan ukuran kepercayaan dari hipotesis H dipengaruhi oleh evidence (fakta) E.

MD : Measure of Disbelief (tingkat ketidakyakinan) merupakan ukuran ketidakpercayaan hipotesis H dipengaruhi oleh fakta E.

H : Hipotesa atau konklusi yang dihasilkan

Jika ada kaidah lain termasuk dalam hipotesis yang sama tetapi berbeda dalam faktor kepastian, maka perhitungan faktor kepastian dari kaidah yang sama

dihitung dari penggabungan fungsi untuk faktor kepastian yang didefinisikan sebagai berikut:

$$CF_{combine}(CF_1, CF_2) = \begin{cases} CF_1 + CF_2(1 - CF_1) & kedua - duanya > 0 \\ \dfrac{CF_1 + CF_2}{1 - \min(|CF_1|, |CF_2|)} & salah\ satu < 0 \\ CF_1 + CF_2(1 - CF_1) & kedua - duanya < 0 \end{cases}$$

.....(2)

Sumber : Bain Khusnul, 2010 : 13

Di mana, $CF_{combine}$ digunakan bergantung pada apakah faktor kepastian positif atau negatif.

Evidensi tidak pasti diperoleh dengan menggali dari hasil wawancara dengan pakar. Nilai CF(Rule) didapat dari interpretasi term dari pakar menjadi nilai MD/MB tertentu. Suatu sistem yang menerapkan *inexact* reasoning, pertama-tama harus menemukan cara untuk mempresentasikan *uncertain evidence*. Pendekatan di atas mengubah bentuk formal dari suatu peluang P(E) dengan CF(E) yang berupa nilai MD/MB.

Faktor kepastian bukanlah suatu peluang, tetapi merupakan ukuran tingkat kepercayaan terhadap suatu evidensi. CF mempresentasikan tingkat kepercayaan bahwa suatu evidensi adalah benar (Bain Khusnul, 2010 : 13).

II.3. Basis Data

Database (Basis data) merupakan kumpulan data yang saling berhubungan satu dengan lainnya yang tersimpan di perangkat keras komputer dan diperlukan suatu perangkat lunak untuk memanipulasi basis data tersebut. Buku telepon,

katalog film merupakan contoh dari database. Sistem manajemen basis data (*database management sistem/DBMS*) adalah perangkat lunak yang digunakan untuk mengendalikan data, termasuk penyimpanan data, pengambilan data, keamanan data, dan integritas data. Fungsi utama DBMS adalah untuk menyediakan lingkungan yang nyaman dan efesien untuk digunakan dalam pengambilan dan penyimpanan informasi di basis data.

Penekanan file pada basis data adalah kemampuan untuk mengakses data dengan cepat dan efesien dalam menggunakan media simpanan luarnya. Faktor yang mempengaruhi hal ini adalah organisasi dari file basis data. Organisasi file basis data ini mencoba meningkatkan struktur dari data antara satu file dengan file yang lainnya, dengan menunjukkan hubungan antardata yang ada :

a. Struktur data berjenjang (*hierarchial data structure*) atau disebut juga dengan nama data pohon (*tree data structure*) menunjukkan hubungan antara data membentuk suatu jenjang seperti pohon.
b. Struktur data jaringan (*network data structure*) disebut juga dengan plex data struktur. Pada struktur data pohon tiap-tiap node tidak dapat mempunyai lebih dari satu orang tua, sedangkan struktur data jaringan ini, tiap-tiap node punya banyak orang tua.
c. Struktur data hubungan adalah meletakkan semua hubungan dalam bentuk tabel dua dimensi (Junidar ; 2012 : 19).

II.4. 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	I Komputer	PAM211	Kalkulus Lanjut I	3	MT002	Yasir
			PAAM261	Prg. Terstruktur I	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

MHS(No_mhs, Nama, Prg_Studi)

No_mhs	Nama	Prg_Studi
0231	Cahyo	I Komp.
0232	Hoho	Statistik
0233	Budi	Matematika

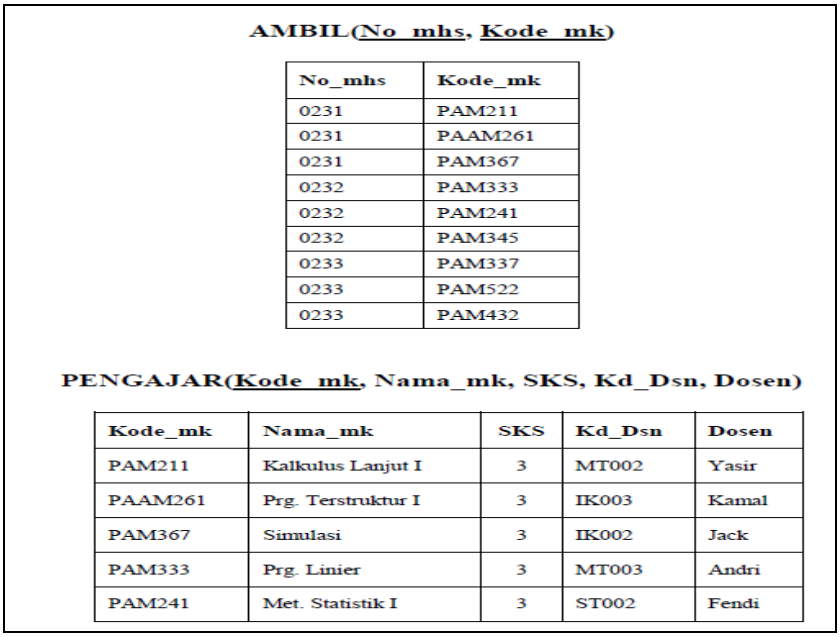
DAFTAR_MK(No_mhs, Kode_mk, Nama_Mk, SKS, Kd_Dsn, Dosen)

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	IK002	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

Gambar II.1. Normalisasi 1NF
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.



AMBIL(No_mhs, Kode_mk)

No_mhs	Kode_mk
0231	PAM211
0231	PAAM261
0231	PAM367
0232	PAM333
0232	PAM241
0232	PAM345
0233	PAM337
0233	PAM522
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	Yasir
PAAM261	Prg. Terstruktur I	3	IK003	Kamal
PAM367	Simulasi	3	IK002	Jack
PAM333	Prg. Linier	3	MT003	Andri
PAM241	Met. Statistik I	3	ST002	Fendi

Gambar II.2. Normalisasi 2NF
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.

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	IK002
PAM333	Prg. Linier	3	MT003
PAM241	Met. Statistik I	3	ST002
PAM345	Analisis Data	3	ST003
PAM337	Fungsi Khas	3	MT001
PAM522	Topologi	3	MT003
PAM432	Teori Optimasi	3	MT004

Gambar II.3. Normalisasi 3NF
Sumber : Tawar ; 2011 : 10

5. Boyce Code Normal Form (BCNF)

   a. Memenuhi 1st NF

   b. Relasi harus bergantung fungsi pada atribut superkey

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	PAM522	B
0237	PAM432	B

REKENING(No_mhs, No_Rkng)

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

Gambar II.4. Normalisasi BCNF
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)

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

Gambar II.5. Normalisasi 4NF

Sumber : Tawar ; 2011 : 13

II.5. PHP

PHP adalah kode atau skrip yang akan dieksekusi pada *server side*. Skrip PHP akan membuat suatu aplikasi daat diintegrasikan ke dalam HTML, sehingga suatu halaman web tidak lagi bersifat statis, namun menjadi bersifat dinamis. Sifat *server-side* berarti pengerjaan skrip dilakukan di *server* baru kemudian hasilnya dikirimkan ke *browser* (Deni Sutaji : 2012 ; 2).

PHP singkatan dari PHP : *Hypertext Preprocessor* yaitu bahasa pemrograman *web server-side* yang bersifat *open source*. PHP merupakan *script* yang terintegrasi dengan HTML dan berada pada server (*Server Side HTML Embedded Scripting*). PHP adalah *script* yang digunakan untuk membuat halaman website yang dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh client. Mekanisme ini menyebabkan informasi yang diterima *client* selalu yang terbaru/*up to date*. Semua *script* PHP diesekusi pada *server* di mana *script* tersebut dijalankan. (Anhar ; 2010 : 3).

II.5.1. Pengertian Wamp

WAMP merupakan salah satu web server yang bisa digunakan untuk membuat wordpress karena wordpress merupakan aplikasi yang berbasis PHP dan menggunakan database MySQL. Di dalam WAMP terdapat PHP, MySQL, Apache dan Pearl. Versi terbaru WAMP adalah versi 1.7.3. Anda dapat memilih paket *lite* untuk versi yang sederhana atau paket *basic* untuk versi yang komplit. Ada 2 tipe varian WAMP yang bisa di download, yaitu *self-extracting RAR*

Archive atau ZIP Archive. Anda disarankan untuk memilih varian *self-extracting RAR Archive* karena Lebih aman (Elcom ; 2012 : 86).

II.5.2. Pengertian Apache

Apache adalah paket aplikasi yang digunakan untuk web server yang handal dan stabil. Jika dibandingkan dengan web server lainnya, Apache masih menjadi andalan para webmaster. Perkembangan server ini sangat pesat sehingga hampir semua server web menggunakan Apache. Aplikasi apache dikenal dengan nama httpd. Jika menggunakan linux redhat/Fedora maka paket ini sudah ada dalam bentuk .RPM (Albertus Dwiyoga ; 2012 : 11).

II.6. Pengertian MySQL

Mysql pertama kali dirintis oleh seorang programmerdatabase bernama Michael Widenius, yang dapat anda hubungi di emailnya monty@analytikerna. Mysql *databaseserver* adalah RDBMS (*Relasional Database Management System*) yang dapat menangani data yang bervolume besar. meskipun begitu, tidak menuntut *resource* yang besar. Mysql adalah *database* yang paling populer di antara *database* yang lain.

MySQL adalah program *database* yang mampu mengirim dan menerima data dengan sangat cepat dan *multiuser*. MySQL memiliki dua bentuk lisensi, yaitu *freesoftware* dan *shareware*. penulis sendiri dalam menjelaskan buku ini menggunakan *database* ini untuk keperluan pribadi atau usaha tanpa harus

membeli atau membayar lisensi, yang berada di bawah lisensi GNU/GPL (*general publiclicense*) (Wahana Komputer ; 2010 : 5)

II.7. *Unified Modeling Language* (UML)

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

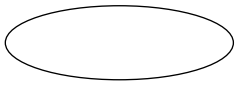
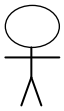
Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login 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.

Tabel II.3. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	Include, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	Extend, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Sri Dharwiyanti ; 2013 : 5)

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.

Tabel II.4. *Multiplicity Class Diagram*

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Sri Dharwiyanti ; 2013 : 6)

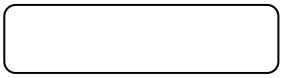
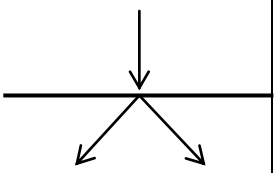
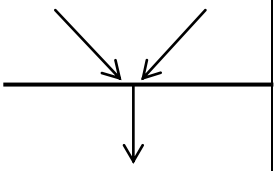
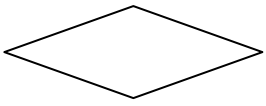
3. *Activity* Diagram

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

Tabel II.5. Simbol *Activity Diagram*

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false*.
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

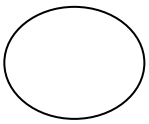
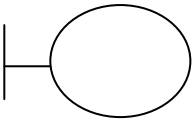
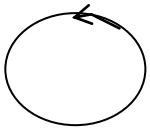
(Sumber : Sri Dharwiyanti ; 2013 : 8)

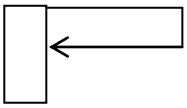
4. *Sequence* Diagram

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.

Tabel II.6. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.

	*Message*, simbol mengirim pesan antar *class*.
	*Recursive,* menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation, activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation.*

(Sumber : Sri Dharwiyanti ; 2013 : 9)