BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Sistem merupakan kumpulan dari unsur atau elemen-elemen yang saling berkaitan/berinteraksi dan saling memengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu. Menurut Jerry FithGerald, sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan dan berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu. (Asbon Hendra ; 2012 : 157)

Berdasarkan prinsip dasar secara umum, sistem terbagi dalam:

1. *Sistem Terspesialisasi* adalah sistem yang sulit diterapkan pada lingkungan yang berbeda, misalnya sistem biologi: ikan yang dipindahkan ke darat.
2. *Sistem Besar* adalah sistem yang sebagian besar sumber dayanya berfungsi melakukan perawatan harian. Misalnya dinosaurus sebagai sistem biologi menghabiskan sebagian besar masa hidupnya dengan makan.
3. *Sistem Sebagai Bagian dari Sistem Lain*. Sistem selalu merupakan bagian dari sistem yang lebih besar, dan dapat terbagi menjadi sistem yang lebih kecil.
4. *Sistem Berkembang*. Walaupun tidak berlaku bagi semua sistem, hamper semua sistem selalu berkembang. (Asbon Hendra ; 2012 : 163)

II.2. Sistem Pakar

Untuk memahami aplikasi sistem pakar, selain memahami definisinya, kita juga harus mengetahui tujuan dari sistem pakar, kompkonen-komponennya, semua domain, dan contoh-contoh aplikasinya, *stakeholders*, dan alasan digunakannya sistem ini.

Sistem pakar merupakan cabang dari *Artificial interllingence (AI)* yang cukup tua karena sistem ini mulai dikembangkan pada pertengahan 1960. Sistem pakar yang muncul pertama kali adalah *General-purpose problem solver (GPS)* yang dikembangkan oleh Newel dan Simon. Sampai saat ini sudah banyak sistem pakar yang dibuat, seperti MYCIN untuk diagnosa penyakit, DENDRAL untuk mengidentifikasi struktur molekul campuran yang tak dikenal, XCON & XSEL untuk membantu komfigurasi sistem komputer besar, SOPHIE untuk analisis sirkuit elekronik, Prospector digunakan di bidang geologi untuk membantu mencari dan menemukan deposit, FOLIO digunakan untuk membantu memberikan keputusan bagi seorang manager dalam stok dan investasi, DELTA dipakai untuk pemeliharaan lokomotif listrik diesel, dan sebagainya.

Istilah sistem pakar berasal dari istilah *knowledge-based expert system*. Istilah ini mencul karena untuk memecahkan masalah, sistem pakar menggunakan pengetahuan seorang pakar yang dimasukkan ke dalam komputer. Seseorang yang bukan pakar menggunakan sistem pakar untuk meningkatkan kemampuan pemecahan masalah, sedangkan seorang pakar menggunakan sistem pakar untuk *knowledge assistant* (T.Sutojo, dkk ; 2011 : 159-160).

II.2.1. Manfaat Sistem Pakar

Sistem pakar menjadi sangat populer karena sangat banyak kemampuan dan manfaat yang diberikannya, di antaranya (T. Sutojo dkk, 2011 : 160-161):

1. Meningkatkan produktivitas, karena sistem pakar dapat bekerja lebih cepat daripada manusia.
2. Membuat seorang yang awam bekerja seperti layaknya seorang pakar.
3. Meningkatkan kualitas, dengan memberi nasehat yang konsisten dan mengurangi kesalahan.
4. Mampu menangkap pengetahuan dan kepakaran seseorang.
5. Dapat beroperasi di lingkungan yang berbahaya.
6. Memudahkan akses pengetahuan seorang pakar.
7. Andal. Sistem pakar tidak pernah menjadi bosan dan kelelahan atau sakit.
8. Meningkatkan kapabilitas sistem komputer. Integrasi sistem pakar dengan sistem komputer lain membuat sistem lebih efektif dan mencakup lebih banyak aplikasi.
9. Mampu bekerja dengan informasi yang tidak lengkap atau tidak pasti.
10. Bisa digunakan sebagai media pelengkap dalam pelatihan. Pengguna pemula yang bekerja dengan sistem pakar akan menjadi lebih berpengalaman karena adanya fasilitas penjelas yang berfungsi sebagai guru.
11. Meningkatkan kemampuan untuk menyelesaikan masalah karena sistem pakar mengambil sumber pengetahuan dari banyak pakar.

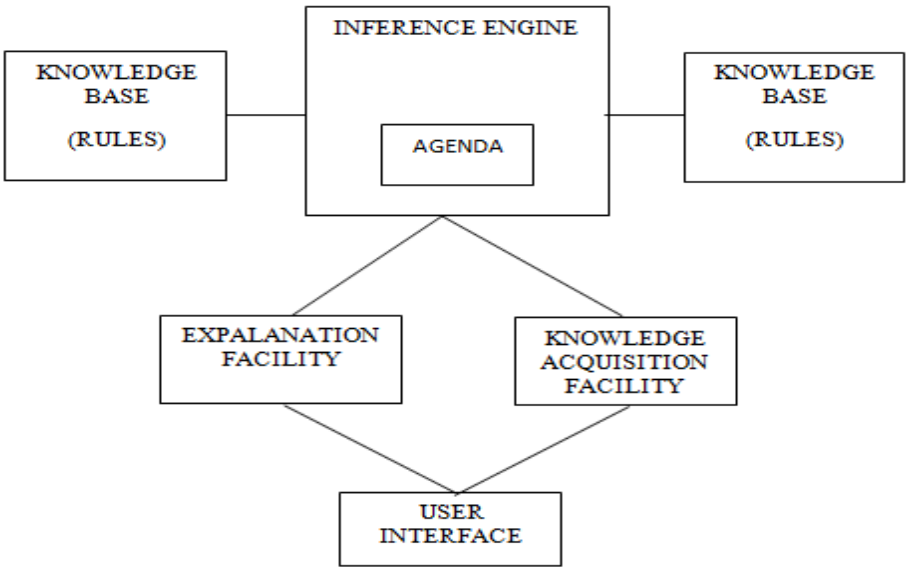
II.2.2. Ciri-Ciri Sistem Pakar

Ciri-ciri dari sistem pakar adalah sebagai berikut (T. Sutojo dkk, 2011 : 162) :

1. Terbatas pada dominan keahlian tertentu
2. Dapat memberikan penalaran untuk data-data yang tidak lengkap atau tidak pasti.
3. Dapat menjelaskan alasan-alasan dengan cara yang dapat dipahami.
4. Bekerja berdasarkan kaidah/ *rule* tertentu.
5. Mudah dimodifikasi.
6. Basis pengetahuan dan mekanisme inferensi terpisah.
7. Keluarannya bersifat anjuran.
8. Sistem dapat mengaktifkan kaidah secara sesuai, dituntun oleh dialog dengan pengguna.

II.2.3. Struktur Sistem Pakar

Menurut T.Sutojo, dkk (2011 : 166), ada 2 bagian penting dari sistem pakar, yaitu lingkungan konsultasi (*consultation environment*). Lingkungan pengembangan digunakan oleh pembuat sistem pakar untuk membangun komponen-komponennya dan memperkenalkan pengetahun kedalam *knowledge base* (basis pengetahuan). Lingkungan konsultasi digunakan oleh pengguna untuk berkonsultasi sehingga pengguna mendapatkan pengetahuan dan nasehat dari sistem pakar layaknya berkonsultasi dengan seorang pakar. Adapun struktur sistem pakar dapat dilihat pada gambar II.1.



Gambar II.1. Struktur Sistem Pakar

Sumber : (Rika Rosnelly ; 2012 : 13)

Komponen yang terdapat dalam struktur sistem pakar ini adalah (Rika Rosnelly ; 2012 : 14-15) :

1. *Knowledge Base* (Basis Pengetahuan)

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

2. *Inference Engine* (Mesin Inferensi)

   Mesin Inferensi merupakan otak dari sebuah sistem pakar dan dikenal juga dengan sebutan *control structure* (struktur kontrol) atau *rule interpreter* (dalam sistem pakar berbasis kaidah).

3. *Working Memory*

   Berguna untuk menyimpan fakta yang dihasilkan oleh *inference engine* dengan penambahan parameter berupa derajat kepercayaan atau dapat juga dikatakan sebagai global database dari fakta yang digunakan oleh rule-rule yang ada.

4. *Explanation Facility*

   Menyediakan kebenaran dari solusi yang dihasilkan kepada user (*reasoning chain*).

5. *Knowledge Acquisition Facility*

   Meliputi proses pengumpulan, pemindahan dan perubahan dari kemampuan pemecahan masalah seorang pakar atau sumber pengetahuan terdokumentasi keprogram komputer, yang bertujuan untuk memperbaiki atau mengembangkan basis pengetahuan.

6. *User Interface*

   Mekanisme untuk memberi kesempatan kepada user dan sistem pakar untuk berkomunikasi. Antar muka menerima informasi dari pemakai dan mengubahnya ke dalam bentuk yang dapat diterima oleh sistem. Selain itu antarmuka menerima informasi dari sistem dan menyajikannya kedalam bentuk yang dimengerti oleh pemakai.

II.3. Metode *Dempster Shafer*

Dempster-Shafer adalah suatu teori matematika untuk pembuktian berdasarkan *belief functions and plausible reasoning* (fungsi kepercayaan dan pemikiran yang masuk akal), yang digunakan untuk mengkombinasikan potongan informasi yang terpisah (bukti) untuk mengkalkulasi kemungkinan dari suatu peristiwa. Teori ini dikembangkan oleh Arthur P. Dempster dan Glenn Shafer. Secara umum teori Dempster-Shafer ditulis dalam suatu interval :

[*Belief,Plausibility*].....(II.1)

1) Belief (*Bel)* adalah ukuran kekuatan evidence dalam mendukung suatu himpunan proposisi. Jika bernilai 0 maka mengindikasikan bahwa tidak ada evidence, dan jika bernilai 1 menunjukkan adanya kepastian. Dimana nilai bel yaitu(0-0.9).
2) *Plausibility* (Pl) dinotasikan sebagai :

Pl(s) = 1 – Bel (-s) .....(II.2)

Plausibility juga bernilai 0 sampai 1. Jika yakin akan -s, maka dapat dikatakan bahwa Bel(-s)=1, dan Pl(-s)=0.

Pada teori *Dempster-Shafer* dikenal adanya *frame of discrement* yang dinotasikan dengan θ. Frame ini merupakan semesta pembicaraan dari sekumpulan hipotesis. Tujuannya adalah mengaitkan ukuran kepercayaan elemen-elemen θ. Tidak semua evidence secara langsung mendukung tiap-tiap elemen. Untuk itu perlu adanya probabilitas fungsi densitas (m). Nilai m tidak hanya

mendefinisikan elemen-elemen θ saja, namun juga semua subsetnya. Sehingga jika θ berisi n elemen, maka subset θ adalah 2*n* .Jumlah semua m dalam subset θ sama dengan1. Apabila tidak ada informasi apapun untuk memilih hipotesis, maka nilai :

m{θ} = 1,0.

Apabila diketahui X adalah subset dari θ, dengan m1 sebagai fungsi densitasnya, dan Y juga merupakan subset dari θ dengan m2 sebagai fungsi densitasnya, maka dapat dibentuk fungsi kombinasi m1 dan m2 sebagai m3, yaitu (Muhammad Dahria, et al., 2013) :

$$m_3(Z) = \frac{\sum X \cap Y = Z\ m_1(X).m_2(Y)}{1 - \sum X \cap Y = \emptyset\ m1\ (X).m2(Y)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.(\text{II}.3)$$

II.4. Metode *Certainty Factor*

Teori *Certainty Factor* (CF) diusulkan oleh Shortlife dan Buchanan pada 1975 untuk mengakomodasi ketidakpastian pemikiran (*inexact reasoning*) seorang pakar. Seorang pakar, (misalnya dokter) sering kali menganalisis informasi yang ada dengan ungkapan seperti “mungkin”, “kemungkinan besar”, hampir pasti”. Untuk mangakomodasi hal ini kita menggunakan *certainty factor* (CF) guna menggambarkan tingkat keyakinan pakar terhadap masalah yang sedang dihadapi (T.Sutojo, dkk ; 2011 : 194).

Ada dua cara dalam mendapatkan tingkat keyakinan (CF) dari sebuah rule yaitu (T.Sutojo, dkk ; 2011 : 194-196):

1. Metode *‘Net Belief’* yang diusulkan oleh E. H. Shortliffe B. G. Buchanan

CF(Rule) = MB[H,E] – MD[H,E] .....(II.4)

$$MB(H,E) = \begin{cases} 1 & P(H) = 1 \\ \dfrac{\max[P(H|E),P(H)] - P(H)}{\max[1,0] - P(H)} & \text{lainnya} \end{cases} \quad \text{... (II. 5)}$$

$$MD(H,E) = \begin{cases} 1 & P(H) = 0 \\ \dfrac{\min[P(H|E),P(H)] - P(H)}{\min[1,0] - P(H)} & \text{lainnya} \end{cases} \quad \text{... (II. 6)}$$

Dimana :

CF(Rule) = Faktor kepastian

MB(H,E) = *measure of belief* (ukuran kepercayaan) terhadap hipotesis H, jika diberikan *evidence* E (antara 0 dan 1)

MD(H,E) = *measure of disbelief* (ukuran ketidakpercayaan) terhadap *evidenc*e H, jika diberikan *evidence* E (antara 0 dan 1)

P(H) = Probabilitas kebenaran hipotesis H

P(H|E) = Probabilitas bahwa H benar karena fakta E

2. Dengan cara mewawancarai seorang pakar

   Nilai CF (Rule) didapat dari interpretasi “term” dari pakar, yang diubah menjadi nilai CF tertentu sesuai tabel berikut.

Tabel II.1. Nilai CF

Uncertain Term	CF
Definitely not (pasti tidak)	-1.0
Almost certainly not (hampir pasti tidak)	-0.8
Probably not (kemungkinan besar tidak)	-0.6
Maybe not (mungkin tidak)	-0.4
Unknown (tidak tahu)	-0.2 to 0.2
Maybe (mungkin)	0.4
Probably (kemungkinan besar)	0.6
Almost certainly (hampir pasti)	0.8
Definitely (pasti)	1.0

Sumber : T. Sutojo, dkk (2011 : 195-196)

II.4.1. Perhitungan *Certainty Factor* Gabungan

Secara umum, rule direpresentasikan dalam bentuk sebagai berikut (T. Sutojo, dkk ; 2011 : 196-198) :

IF E_1 AND E_2 .....AND E_n THEN H (CF Rule)

Atau

IF E_1 OR E_2 .....OR E_n THEN H (CF Rule)

Di mana :

E_1 ... E_n : Fakta-fakta (evidence) yang ada

H : Hipotesis atau konklusi yang dihasilkan

CF (Rule) : Tingkat keyakinan terjadinya hipotesis H akibat adanya fakta-fakta E_1 ... E_n

1. Rule dengan *evidence* E tunggal dan Hipotesis H tunggal

   IF E THEN H (CF rule)

   CF(H,E) = CF(E) x CF (rule).....(II.7)

   Catatan :

   Secara praktik, nilai CF rule ditentukan oleh pakar, sedangkan nilai CF(E) ditentukan oleh pengguna saat berkonsultasi dengan sistem pakar.

2. Rule dengan *evidence* E ganda dan Hipotesis H tunggal

   IF E_1 AND E_2 .....AND E_n THEN H (CF Rule)

   CF(H,E) = min[CF(E_1), CF (E_2),....., CF(E_n)] x CF (rule).....(II.8)

   IF E_1 OR E_2 .....OR E_n THEN H (CF Rule)

   CF(H,E) = max[CF(E_1), CF (E_2),....., CF(E_n)] x CF (rule).....(II.9)

3. Kombinasi dua buah rule dengan *evidence* berbeda (E_1 dan E_2), tetapi hipotesis sama.

   IF E_1 THEN H Rule 1 $CF(H,E_1) = CF_1 = C(E_1)$ x CF (Rule1)

   IF E_2 THEN H Rule 2 $CF(H,E_2) = CF_2 = C(E_2)$ x CF (Rule2)

$$CF(CF_1, CF_2) = \begin{cases} CF_1 + CF_2(1 - CF_1) \text{ jika } CF_1 > 0 \; dan \; CF_2 > 0 \\ \dfrac{CF_1 + CF_2}{1 - \min[|CF_1|, |CF_2|]} \text{ jika } CF_1 < 0 \; atau \; CF_2 < 0 \\ CF_1 + CF_2 \, x \, (1 + CF_1) \text{ jika } CF_1 < 0 \; dan \; CF_2 < 0 \end{cases} \quad \ldots (II.10)$$

II.4.2. Kelebihan dan Kekurangan Metode *Certainty Factor*

Kelebihan metode *Certainty Factor* adalah (T. Sutojo, dkk ; 2011 : 204) :

1. Metode ini cocok dipakai dalam sistem pakar yang mengandung ketidakpastian.
2. Dalam sekali proses perhitungan hanya dapat mengolah 2 data saja sehingga kekurangan data dapat terjaga.

Sedangkan kekurangan metode *Certainty Factor* adalah:

1. Pemodelan ketidakpastian yang menggunakan perhitungan metode *Certainty Factor* biasanya masih diperdebatkan.
2. Untuk data lebih dari 2 buah, harus dilakukan beberapa kali pengolahan data.

II.5. Pengertian Basis Data

Basis data dapat dipahami sebagai suatu kumpulan data terhubung (*interrelated data*) yang disimpan secara bersama-sama pada suatu media, tanpa *mengatap* satu sama lain atau tidak perlu suatu kerangkapan data (kalaupun ada maka kerangkapan data tersebut harus seminimal mungkin dan terkontrol [*controlled redudancy*]), data disimpan dengan cara-cara tertentu sehingga mudah digunakan/atau ditampilkan kembali; data dapat digunakan oleh satu atau lebih program-program aplikasi secara optimal; data disimpan tanpa mengalami ketergantungan dengan program yang akan menggunakannya; data disimpan sedemikian rupa sehingga proses penambahan, pengambilan, dan modifikasi data dapat dilakukan dengan mudah dan terkontrol (Edy Sutanta, 2011 : 29-30).

II.6. Normalisasi

Menurut Martin (1975), Normalisasi diartikan sebagai suatu teknik yang menstrukurkan/ mendekomposisi data dalam cara-cara tertentu untuk mencegah timbulnya permasalahan pengolahan data dalam basis data. Permasalahan yang dimaksud adalah berkaitan dengan penyimpangan-penyimpangan *(anomalies)* yang terjadi akibat adanya kerangkapan data dalam relasi dan in-efisiensi pengolahan (Edy Sutanta ; 2011 : 174).

Proses normalisasi menghasilkan relasi yang optimal, yaitu (Martin, 1975) : (Edy Sutanta ; 2011 : 175)

1. Memiliki struktur *record* yang konsisten secara logik;
2. Memiliki struktur *record* yang mudah untuk dimengerti;
3. Memiliki struktur *record* yang sederhana dalam pemeliharaan;
4. Memiliki struktur *record* yang mudah ditampilkan kembali untuk memenuhi kebutuhan pengguna;
5. Minimalisasi kerangkapan data guna meningkatkan kinerja sistem.

Secara berturut-turut masing-masing level normal tersebut dibahas berikut ini, dimulai dari bentuk tidak normal. (Edy Sutanta ; 2011 : 176-179)

1. Relasi bentuk tidak normal (*Un Normalized Form* / UNF)

Relasi-relasi yang dirancang tanpa mengindahkan batasan dalam defisi basis data dan karakteristik *Relational Database Management System* (RDBM) menghasilkan relasi *Un Normalized Form* (UNF). Bentuk ini harus di hindari dalam perancangan relasi dalam basis data. Relasi *Un Normalized Form* (UNF) mempunyai kriteria sebagai berikut.

a. Jika relasi mempunyai bentuk *non flat file* (dapat terjadi akibat data disimpan sesuai dengan kedatangannya, tidak memiliki struktur tertentu, terjadi duplikasi atau tidak lengkap)
b. Jika relasi membuat *set atribut* berulang *(non single values)*
c. Jika relasi membuat *atribut non atomic value*

2. Relasi bentuk normal pertama (*First Norm Form* / 1NF)

Relasi disebut juga *First Norm Form* (1NF) jika memenuhi kriteria sebagai berikut.

a. Jika seluruh atribut dalam relasi bernilai *atomic (atomic value)*
b. Jika seluruh atribut dalam relasi bernilai tunggal *(single value)*
c. Jika relasi tidak memuat set atribut berulang
d. Jika semua record mempunyai sejumlah atribut yang sama.

Permasalahan dalam *First Norm Form* (1NF) adalah sebagai berikut.

a. Tidak dapat menyisipkan informasi parsial
b. Terhapusnya informasi ketika menghapus sebuah *record*

3. Bentuk normal kedua *(Second Normal Form /* 2NF)

Relasi disebut sebagai *Second Normal Form (*2NF) jika memenuhi kriteria sebagai berikut

a. Jika memenuhi kriteria *First Norm Form (*1NF)
b. Jika semua atribut nonkunci *Functional Dependence* (FD) pada *Primary Key* (PK)

Permasalahan dalam *Second Normal Form /* 2NF adalah sebagai berikut:

a. Kerangkapan data *(data redundancy)*

b. Pembaharuan yang tidak benar dapat menimbulkan inkonsistensi data *(data inconsistency)*

c. Proses pembaharuan data tidak efisien

Kriteria tersebut mengidentifikasikan bahwa antara atribut dalam *Second Normal Form* masih mungkin mengalami *Third Norm Form*. Selain itu, relasi *Second Normal Form* (2NF) menuntut telah didefinisikan atribut *Primary Key* (PK) dalam relasi. Mengubah relasi *First Norm Form* (1NF) menjadi bentuk *Second Normal Form* (2NF) dapat dilakukan dengan mengubah struktur relasi dengan cara :

a. Identifikasikan *Functional Dependence* (FD) relasi *First Norm Form* (1NF)

b. Berdasarkan informasi tersebut, dekomposisi relasi *First Norm Form* (1NF) menjadi relasi-relasi baru sesuai *Functional Dependence* nya. Jika menggunakan diagram maka simpul-simpul yang berada pada puncak diagram ketergantungan data bertindak *Primary Key* (PK) pada relasi baru

4. Bentuk normal ketiga (*Third Norm Form* / 3NF)

Suatu relasi disebut sebagai *Third Norm Form* jika memenuhi kriteria sebagai berikut.

a. Jika memenuhi kriteria *Second Normal Form* (2NF)

b. Jika setiap atribut nonkunci tidak (TDF) *(Non Transitive Dependeny)* terhadap *Primary Key* (PK)

Permasalahan dalam *Third Norm Form* (3NF) adalah keberadaan penentu yang tidak merupakan bagian dari *Primary Key* (PK) menghasilkan duplikasi

rinci data pada atribut yang berfungi sebagai *Foreign Key* (FK) (duplikasi berbeda dengan keterangan data).

Mengubah relasi *Second Normal Form* (2NF) menjadi bentuk *Third Norm Form* (3NF) dapat dilakukan dengan mengubah struktur relasi dengan cara :

a. Identifikasi TDF relasi *Second Normal Form* (2NF)
b. Berdasarkan informasi tersebut, dekomposisi relasi *Second Normal Form* (2NF) menjadi relasi-relasi baru sesuai TDF-nya.

5. Bentuk normal *Boyce-Cood (Boyce-Codd Norm Form* / BCNF)

Bentuk normal *Boyce-Codd Norm Form* (BCNF) dikemukakan oleh R.F. Boyce dan E.F. Codd. Suatu relasi disebut sebagai *Boyce-Codd Norm Form* (BCNF) jika memenuhi kriteria sebagai berikut.

a. Jika memenuhi kriteria *Third Norm Form* (3NF)
b. Jika semua atribut penentu (determinan) merupakan CK

6. Bentuk normal keempat (*Forth Norm Form* / 4NF)

Relasi disebut sebagi *Forth Norm Form* (4NF) jika memenuhi kriteria sebagai berikut.

a. Jika memenuhi kriteria *Boyce-Codd Norm Form*.
b. Jika setiap atribut didalamnya tidak mengalami ketergantungan pada banyak nilai.

7. Bentuk normal kelima (*Fifth Norm Form* / 5NF)

Suatu relasi memenuhi kriteria *Fifth Norm Form* (5NF) jika kerelasian antar data dalam relasi tersebut tidak dapat direkonstruksi dari struktur relasi yang sederhana.

8. Bentuk normal kunci domain (*Domain Key Norm Form* / DKNF)

   Relasi disebut sebagai *Domain Key Norm Form* (DKNF) jika setiap batasan dapat disimpulkan secara sederhana dengan mengetahui sekumpulan nama atribut dan domainnya selama menggunkan sekumpulan atribut pada kuncinya.

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

Unified Modeling Language (UML) adalah sebuah bahasa yang diterima dan digunakan oleh *software developer* dan *software analyst* sebagai suatu bahasa yang cocok untuk merepresentasikan grafik dari suatu relasi antar entitas-entitas software (Gornik, 2003). Dengan menggunakan UML, tim pengembang *software* akan mempunyai banyak keuntungan, seperti memudahkan komunikasi dengan sesama anggota tim tentang *software* apa yang akan dibuat, memudahkan integrasi ke dalam area pengerjaan *software* karena bahasa ini berbasiskan *meta-models* dimana *meta-models* bisa mendefinisikan proses-proses untuk mengkonstruksikan konsep-konsep yang ada. UML juga menggunakan formatur *input* dan *output* yang sudah mempunyai bentuk standar yaitu *XML Metadata Interchange* (XMI), menggunakan aplikasi dan pemodelan data yang universal, merepresentasikan dari tahap analisis ke implementasi lalu ke *deployment* yang terpadu, dan mendeskripsikan keutuhan tentang spesifikasi software.

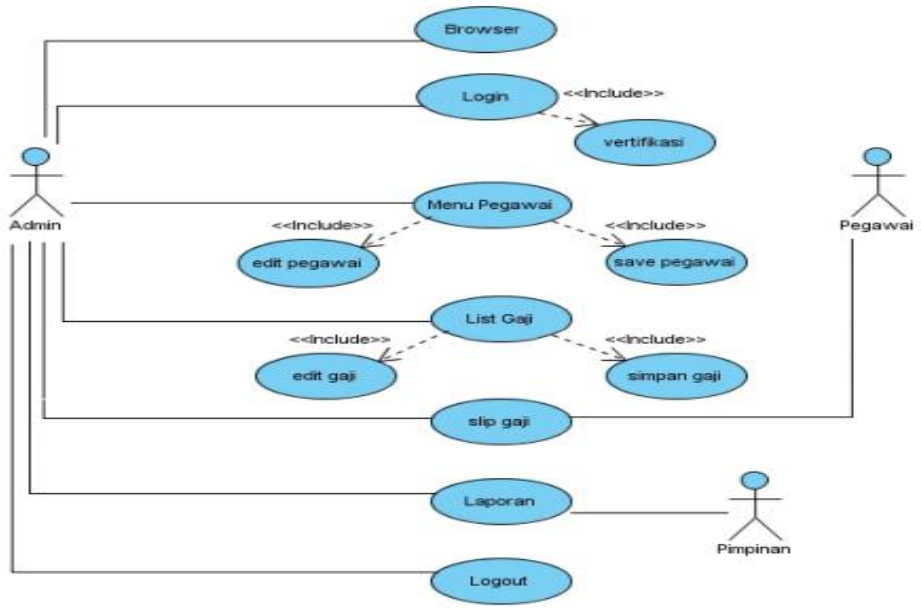
UML menyediakan kumpulan alat yang sudah terstandarisasi, yang digunakan untuk mendokumentasikan analisis dan perancangan sebuah sistem perangkat lunak. (Kendall & Kendall, 2005, p663) Peralatan utama UML adalah

diagram-diagram yang digunakan untuk membantu manusia dalam memvisualisasikan proses pengembangan sebuah sistem perangkat lunak, sama seperti penggunaan denah (*blueprint*) dalam pembuatan bangunan (Edgar Winata dan Johan Setiawan, 2013).

II.7.1. *Use Case* Diagram

Use case adalah deskripsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai (Oktafiansyah, 2012).

Suatu *use case* diagram menampilkan sekumpulan *use case* dan aktor (pelaku) dan hubungan diantara *use case* dan aktor tersebut. *Use case* diagram digunakan untuk penggambaran *use case* statik dari suatu sistem. *Use case* diagram penting dalam mengatur dan memodelkan kelakuan dari suatu sistem. *Use case* menjelaskan apa yang dilakukan sistem (atau subsistem) tetapi tidak menspesifikasi cara kerjanya. *Flow of event* digunakan untuk menspesifikasi cara kerjanya kelakuan dari *use case*. *Flow of event* menjelaskan *use case* dalam bentuk tulisan dengan sejelas-jelasnya, dintaranya bagaimana, kapan *use case* dimulai dan berakhir, ketika *use case* berinteraksi dengan aktor, objek apa yang digunakan, alur dasar dan alur alternatif. Terdapat beberapa simbol dalam menggambarkan diagram *use case*, yaitu *use cases*, aktor dan relasi (Achmad Hamzah Nasrullah dan Dadang Sudrajat, 2015).



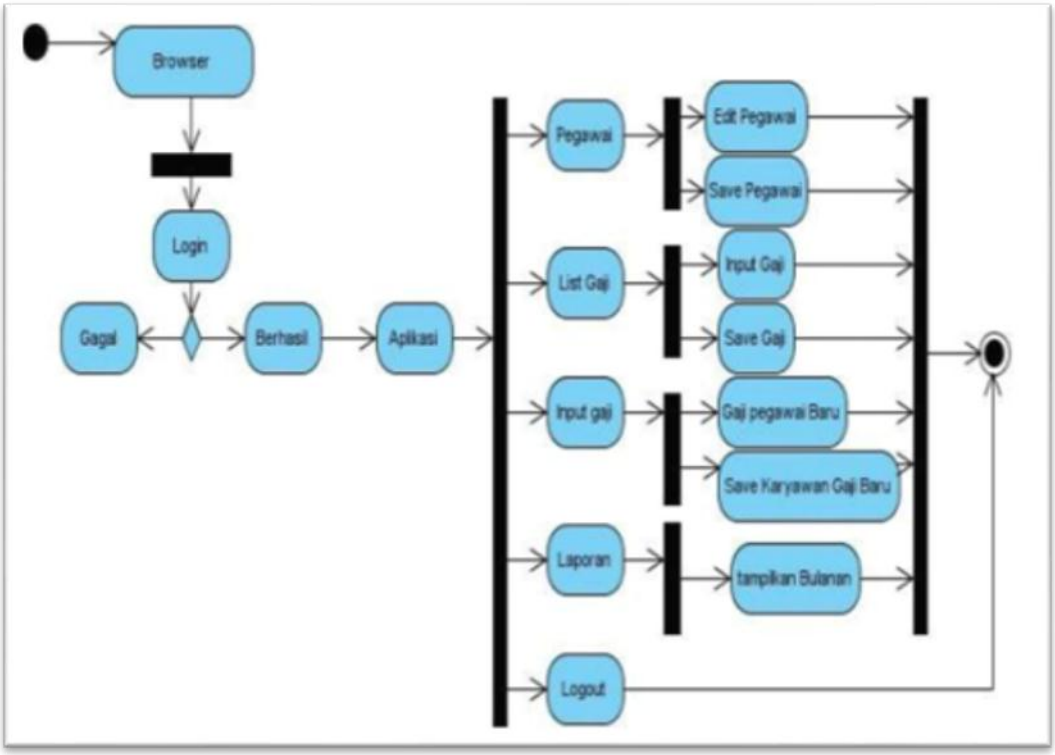
Gambar II.2. *Use Case* Diagram

(Sumber : Aris, et al., 2015)

II.7.2. *Activity* Diagram

Activity diagram adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus (Oktafiansyah, 2012).

Activity diagram memperlihatkan alur langkah demi langkah dalam suatu proses. Suatu aktivitas menunjukkan sekumpulan aksi (secara sekuensial atau bercabang dari satu aksi ke aksi lain), dan nilai yang dihasilkan atau digunakan oleh aksi-aksi yang terjadi. Activity diagram ditunjukkan untuk memodelkan fungsi dari suatu sistem dan menekankan pada alur dari kontrol didalam pelaksanaan dari suatu tindakan (Achmad Hamzah Nasrullah dan Dadang Sudrajat, 2015).

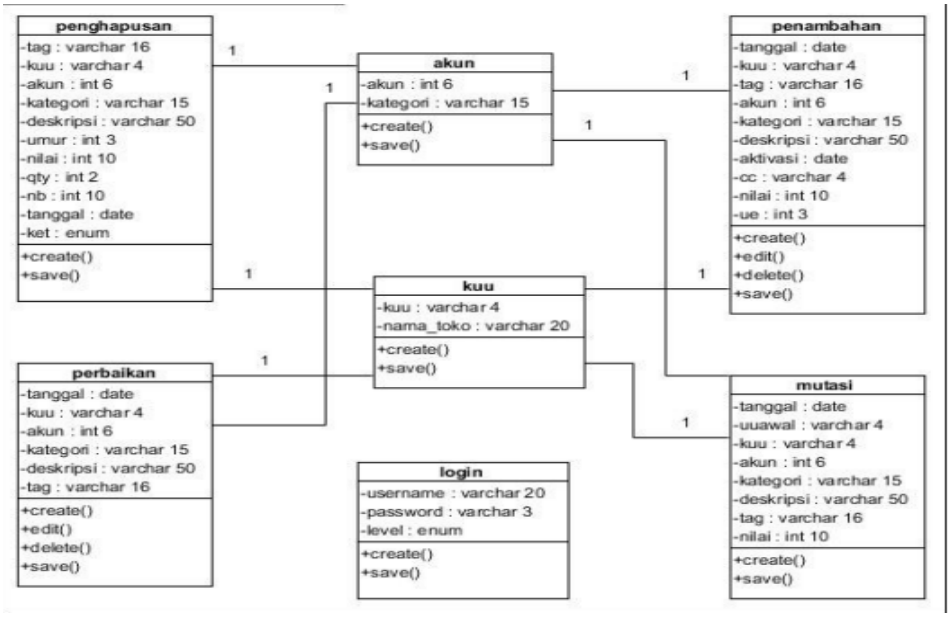


Gambar II.3. *Activity* Diagram

(Sumber : Aris, et al., 2015)

II.7.3. *Class* Diagram

Class diagram menunjukan sekumpulan kelas, antarmuka, dan kerjasama serta hubungannya. *Class* diagram digunakan untuk memodelkan perancangan statik dari gambaran sistem. Biasanya meliputi pemodelan *vocabulary* dari sistem, pemodelan kerjasama, atau pemodelan skema. *Class* diagram dapat digunakan untuk membangun sistem yang dapat dieksekusi melalui teknik *forward and reverse*, selain untuk penggambaran, menspesifikasikan, dan pendokumentasian struktur model (Achmad Hamzah Nasrullah dan Dadang Sudrajat, 2015).

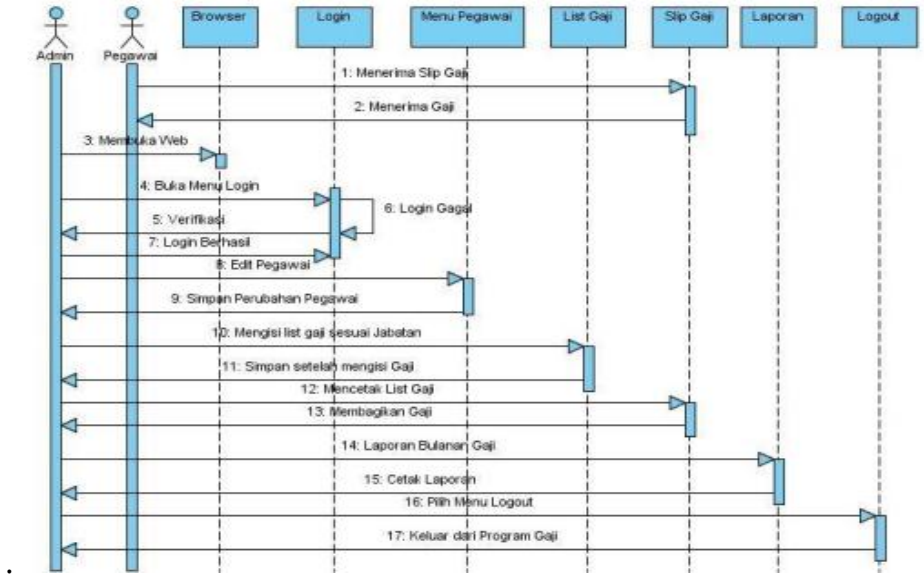


Gambar II.4. *Class* Diagram

(Sumber : Rosana Junita Sirait, et al., 2015)

II.7.4. *Sequence* Diagram

Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan jumlah contoh obyek dan *message* (pesan) yang diletakkan diantara obyek-obyek ini di dalam *use case* (Oktafiansyah, 2012).



Gambar II.5. *Sequence* Diagram

(Sumber : Aris, et al., 2015)

II.8. *Microsoft Visual Basic 2010*

Microsoft Visual Basic .NET adalah sebuah alat untuk mengembangkan dan membangun aplikasi yang bergerak di atas sistem .NET Framework, dengan menggunakan bahasa BASIC (Eka Kurniawan, 2015).

Pada akhir tahun 1999, Teknologi .NET diumumkan. Microsoft memosisikan teknologi tersebut sebagai *platform* untuk membangun XML Web *Services*. XML Web *services* memungkinkan aplikasi tipe manapun dan dapat mengambil data yang tersimpan pada server dengan tipe apapun melalui internet.

Visual Basic.NET adalah Visual Basic yang direkayasa kembali untuk digunakan pada *platform* .NET sehingga aplikasi yang dibuat menggunakan Visual Basic .NET dapat berjalan pada sistem komputer apa pun, dan dapat

mengambil data dari server dengan tipe apa pun asalkan terinstal .NET Framework. (Priyanto Hidayatullah, 2012 ; 5).

II.9. *SQL Server 2008*

SQL Server adalah sebuah *database* relasional yang dirancang untuk mendukung aplikasi dengan arsitektur *client/server* dimana *database* terdapat pada komputer pusat yang disebut *server,* dan informasi digunakan bersama–sama oleh beberapa *user* yang menjalankan aplikasi didalam komputer lokalnya yang disebut dengan *client.* Arsitektur semacam ini memberikan integritas data yang tinggi karena semua *user* bekerja dengan informasi yang sama. Melalui aturan-aturan bisnis, kendali diterapkan kepada semua *user* mengenai informasi yang ditambahkan ke dalam *database*. *Database SQL Server* dibagi kedalam beberapa komponen logikal, seperti misalnya tabel, view, dan elemen–elemen lain yang terlihat oleh *user* (Rika Yunitarini, 2013).