BAB II

TINJAUAN PUSTAKA

II.1. Kecerdasan Buatan

Kecerdasan buatan berasal dari bahasa Inggris *"Artificial Intelligence"* atau disingkat AI, yaitu *intelligence* adalah kata sifat yang berarti cerdas, sedangkan *artificial* artinya buatan. Kecerdasan buatan yang dimaksud di sini merujuk pada mesin yang mampu berpikir, menimbang tindakan yang akan diambil, dan mampu mengambil keputusan seperti yang dilakukan oleh manusia berikut adalah beberapa definisi kecerdasan buatan yang telah didefinisikan oleh beberapa ahli.

Alan Turing, ahli matematika berkebangsaan Inggris yang dijuluki bapak komputer modern dan pembongkar sandi Nazi dalam era Perang Dunia II 1950, menetapkan definisi *Artificial Intelligent :* "Jika komputer tidak dapat dibedakan dengan manusia saat berbincang mealui terminal komputer, maka bisa dikatakan komputer itu cerdas, mempunyai kecerdasan".

Menurut Winston dan Prendergast (1984), tujuan dari kecerdasan buatan adalah :

1. Membuat mesin menjadi lebih pintar (tujuan utama)
2. Memahami apa itu kecerdasan (tujuan ilmiah)
3. Membuat mesin lebih bermanfaat (tujuan *entrepreneurial)*

Berdasarkan definisi ini, maka kecerdasan buatan menawarkan media maupun uji teori tentang keccerdasan. Teori-teori ini nantinya dapat

dinyatakan dalam bahasa pemrograman dan eksekusinya dapat dibuktikan pada komputer nyata (T. Sutojo dkk, 2011 : 1-3).

II.2. Sistem Pakar

II.2.1. Pengenalan Sistem Pakar

Untuk mengetahui aplikasi sistem pakar, selain memahami definisinya, kita juga harus mengetahui tujuan dari sistem pakar, komponen-komponennya, semua domain, dan contoh-contoh aplikasinya, *stakeholders*, dan alasan digunakannya sistem ini. Sistem pakar merupakan cabang dari *Artificial Intelligence* (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 diagnosis penyakit, DENDRAL untuk mengindentifikasi struktur molekul campuran yang tidak dikenal, XCON dan XSEL untuk membantu konfigurasi sistem komputer besar, SOPHIE untuk analisis sirkuit elektronik, *Prospector* digunakan dibidang 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 muncul karena untuk memecahkan masalah, sistem pakar menggunakan pengetahuan seorang pakar yang dimasukkan kedalam 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.2.2. Defenisi Sistem Pakar

Definisi Sistem Pakar selalu berkembang, bertambah dan bervariasi. Hal ini telihat dari banyaknya definisi Sistem Pakar yang telah beredar. Selain itu, Sistem Pakar juga merupakan suatu kajian ilmu yang relatif baru, digunakan oleh berbagai bidang disiplin ilmu, dan berkembang dengan cepat. Berikut ini adalah beberapa definisi Sistem Pakar (T.Sutojo, dkk ; 2011 : 160).

1. Turban (2001), mendefinisikan Sistem Pakar adalah sebuah sistem yang mengggunakan pengetahuan manusia dimana pengetahuan tersebut dimasukkan ke dalam sebuah komputer dan kemudian digunakan untuk menyelesaikan masalah-masalah yang biasanya membutuhkan kepakaran atau keahlian manusia.
2. Jackson (1999), Sistem pakar adalah program komputer yang mempresentasikan dan melakukan penalaran dengan pengetahuan beberapa pakar untuk memecahkan masalah atau memberikan saran.
3. Luger dan Stubblefield (1993), mendefinisikan Sistem Pakar adalah program yang berbasis pengetahuan yang menyediakan solusi “kualiatas pakar” kepada masalah-masalah dalam bidang domain yang spesifik.

Dari defenisi-definisi tersebut diatas, diambil satu buah definisi yang dapat mewakili Sistem Pakar secara umum yaitu sistem yang berusaha mengadopsi

pengetahuan manusia ke komputer agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli. Atau dengan kata lain sistem pakar adalah sistem yang di desain dan diimplementasikan dengan bantuan bahasa pemrograman tertentu untuk dapat menyelesaikan masalah seperti yang dilakukan oleh para ahli.

II.2.3. Konsep Dasar Sistem Pakar

II.2.3.1 Kepakaran (*Expertise*)

Menurut T.Sutojo, dkk (2011 : 163), Kepakaran merupakan suatu pengetahuan yang diperoleh dari pelatihan membaca, dan pengalaman. Kepakaran inilah yang memungkinkan para ahli dapat mengambil keputusan lebih cepat dan lebih baik dari pada orang yang bukan pakar, kepakaran itu sendiri meliputi pengetahuan tentang :

1. Fakta-fakta tentang bidang permasalahan tertentu.
2. Teori-teori tentang bidang permasalahan tertentu.
3. Aturan-aturan dan prosedur-prosedur menurut bidang permasalahan umumnya.
4. Aturan *heuristic* yang harus dikerjakan dalam situasi tertentu.
5. Strategi global untuk memecahkan permasalahan.
6. Pengetahuan tentang pengetahuan (*meta knowledge*).

II.2.3.2 Pakar (*Expert*)

Menurut T.Sutojo, dkk (2011 : 163), pakar adalah seseorang yang seseorang yang mempunyai pengetahuan, pengalaman, dan metode khusus, serta mampu menerapkannya untuk memecahan masalah atau memebari nasehat. Seorang pakar juga harus mampu menjelaskan dan mempelajari hal-hal baru yang berkaitan dengan topik permasalahan. Seorang pakar harus mampu melakukan kegiatan-kegiatan berikut ini :

1. Mengenali dan menformulasi permasalahan.
2. Memecahkan permasalahan secara cepat dan tepat.
3. Menerangkan pemecahannya.
4. Belajar dari pengalaman.
5. Merestrukturisasi pengetahuan.
6. Memecahkan aturan-aturan.
7. Menentukan relevansi.

II.2.3.3 Pemindahan Kepakaran (*Transferring Expertise*)

Menurut T.Sutojo, dkk (2011 : 164), Tujuan dari sistem pakar adalah memindahkan kepakaran dari seorang pakar ke dalam komputer, kemudian di transfer kepada orang lain yang bukan pacar. Proses ini melibatkan empat kegiatan, yaitu :

1. Akuisisi pengetahuan (dari pakar atau sumber lain).
2. Representasi pengetahuan (pada komputer).
3. Inferensi pengetahuan.

4. Pemindahan pengetahuan ke pengguna.

II.2.3.4 Inferensi (*Inferencing*)

Menurut T.Sutojo, dkk (2011 : 164), Referensi adalah sebuah prosedur (program) yang mempunyai kemampuan dalam melakukan penalaran. Inferensi ditampilkan pada suatu komponen yang disebut mesin inferensi yang mencakup prosedur-prosedur yang mengenai pemecahan masalah. Tugas mesin inferensi adalah mengambil kesimpulan berdasarkan basis pengetahuan yang dimilikinya.

II.2.3.5 Aturan-aturan (*Rule*)

Kebanyakan *software* sistem pakar komersial adalah sistem yang berbasis *rule (rule-based systems),* yaitu pengetahuan disimpan terutama dalam bentuk *rule,* sebagai prosedur-prosedur pemecahan masalah.

II.2.3.6 Kemampuan Menjelaskan (*Explanation Capability*)

Menurut T.Sutojo, dkk (2011 : 165), Sistem pakar adalah kemampuan untuk menjelaskan saran atau rekomendasi yang diberikan. Penjelasan yang dilakukan dalam subsistem yang disebut subsistem penjelasan (*Explanation*).

Karakterisitik dan kemampuan yang dimiliki oleh sistem pakar berbeda dengan sistem konvensional. Perbedaan ini dapat ditunjukkan pada Table II.1 berikut ini :

Tabel II.1 Perbandingan Antara Sistem Konvensional dengan Sistem Pakar

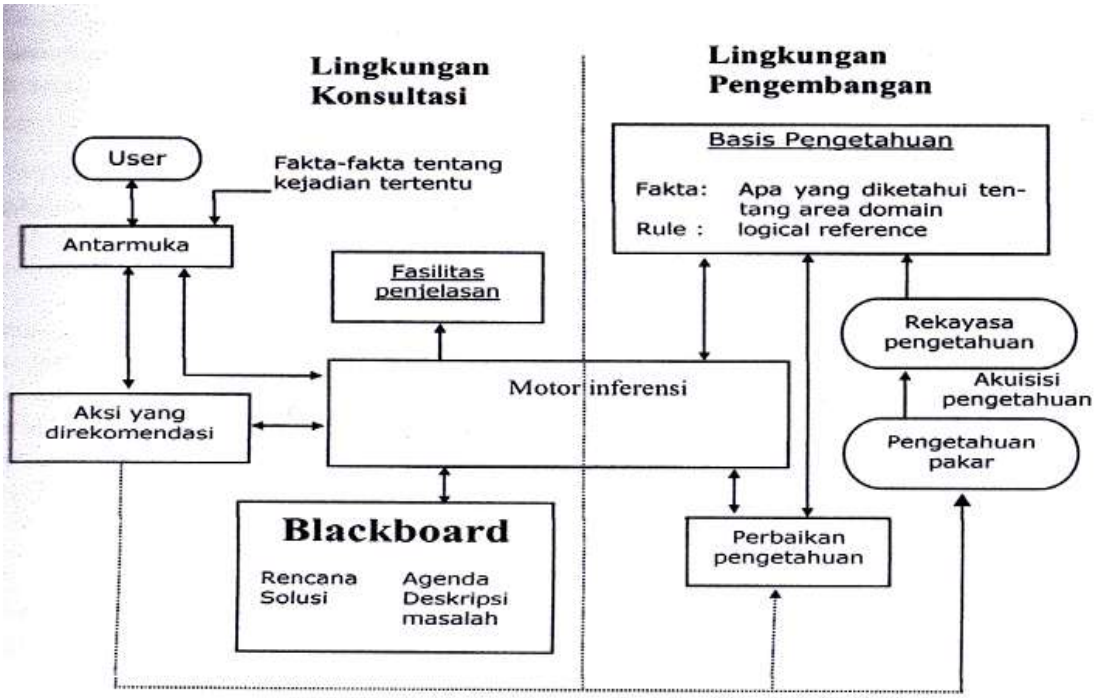
Sistem Konvensial	Sistem Pakar
Informasi dan pemrosesannya biasanya diagabungkan dalam satu program	Basis pengetahuan dipisahkan mekanisme referensi.
Program tidak membuat kesalahan (yang membuat kesalahan : *user* atau pengguna)	Program dapat berbuat kesalahan.
Biasanya tidak menjelaskan mengapa data masukkan diperlukan atau bagaimana output dihasilkan.	Penjelasan merupakan bagian terpenting dari semua sistem pakar.
Perubahan program sangat menyulitkan.	Perubahan dalam aturan-aturan mudah untuk dilakukan.
Sistem hanya bisa beroperasi setelah lengkap atau selesai.	Sistem dapat beroperasi hanya dengan aturan-aturan yang sedikit (sebagai prototipe awal).
Eksekusi dilakukan langkah demi langkah (algoritmik).	Eksekusi dilakukan dengan menggunakan *heuristic* dan logika pada seluruh basis pengetahuan.
Perlu informasi lengkap agar bisa beroperasi.	Dapat beroperasi dengan informasi yang tidak lengkap atau mengandung ketidakpastian.
Manipulasi efektif dari basis data yang besar.	Manipulasi efektif dari basis pengetahuan yang besar.
Menggunakan data.	Menggunakan pengetahuan.
Tujuan utama : efisiensi.	Tujuan utama : efektivitas
Mudah berurusan dengan data kuantitatif.	Mudah berurusan dengan data kualitatif.
Menangkap, menambah, dan mendistribusikan akses ke data numeric atau informasi.	Menangkap, menambah, dan mendistribusikan akses ke pertimbangan dan pengetahuan.

Sumber : T.Sutojo, dkk (2011 : 165)

II.2.4. 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 komponen-komponen yang penting dalam sebuah sistem pakar dapat dilihat pada gambar II.4 berikut ini :



Gambar II.1. Komponen-komponen yang penting dalam sebuah sistem pakar
Sumber : T.Sutojo, dkk (2010 : 167)

Penjelasan tentang gambar II.2 adalah sebagai berikut (T.Sutojo, dkk ; 2011 : 167-169) :

1. Akuisisi pengetahuan

   Susbsitem ini digunakan unutk memasukkan pengetahuan dari seorang pakar dengan cara merekayasa pengetahuan agar bisa diproses oleh komputer dan menaruhnya kedalam basis pengetahuan dengan format tertentu. Sumber-sumber pengetahuan bisa diambil dari pakar, buku, dokumen, multimedia, basis data, laporan riset khusus, dan informasi yang terdapat di web.

2. Basis Pengetahuan (*Knowledge Base*)

   Basis pengetahuan mengandung pengetahuan yang diperlukan untuk memahami, memformulasikan, dan menyelesaikan masalah.

3. Mesin Inferensi (*Inference Engine*)

   Mesin inferensi adalah sebuah program yang berfungsi untuk memandu proses penalaran terhadap suatu kondisi berdasarkan pada basis pengetahuan yang ada,, memanipulasi dan mengarahkan kaidah, model, dan fakta yang disimpan dalam basis pengetahuan untuk mencapai solusi dan kesimpulan.

4. Daerah Kerja (*Blackboard*)

   Untuk merekam hasi sementara yang akan dijadikan sebagai keputusan dan untuk menjelaskan sebuah masalah yang akan terjadi, sistem pakar membutuhkan *Blackboard*, yaitu area pada memori yang berfungsi sebagai basis data.

5. Antarmuka Pengguna (*User Interface*)

   Digunakan sebagai media komunikasi anatar pengguna dan sistem pakar. Komunikasi ini yang paling bagus bila disajikan dalam bahasa alami (*natural language*) dan dilengkapi dengan grafik, menu, dan formulir elektronik.

6. Subsistem Penjelasan (*Explanation Subsystem / Justifier*)

   Berfungsi memberikan penjelasan kepada pengguna, bagaimana atau kesimpulan dapat diambil. Kemampuan seperti saat ini sangat penting bagi pengguna untuk mengetahui proses pemindahan keahlian pakar maupun dalam pemecahan masalah.

7. Sistem Perbaikan Pengetahuan (*Knowledge Refining System*)

   Kemampuan memperbaiki pengetahuan (*knowledge refining system*) dari seorang pakar diperlukan untuk menganalisis pegetahuan, belajar dari kesalahan masa lalu, kemudian memperbaiki pengetahuan sehingga dapat dipakai di massa mendatang. Kemampuan evaluasi siri seperti itu diperlukan oleh program agar dapat menganalisis alasan-alasan kesuksesan dan kegagalan dalam mengambil kesimpulan. Denga cara ini basis pengetahuan yang lebih baik dan penalaran yang leih efektif akan dihasilkan.

8. Pengguna (*User*)

   Pada umumnya pengguna sistem pakar bukanlah seorang pakar (*non-expert*) yang membutuhkan solusi, saran, atau pelatihan (training) dari berbagai permasalahan yang ada.

II.2.5. Tim Pengembang Sistem Pakar

Menurut T.Sutojo, dkk (2011 : 170), tim pengembang sistem pakar adalah sebagai berikut :

1. *Domain Expert* adalah pengetahuan dan kemampuan seorang pakar untuk menyelesaikan masalah terbatas pada keahliannya saja.

2. *Knowledge Engineer* (Perekayasa Pengetahuan) adalah orang yang mampu mendesain, membangun, dan menguji sebuah sistem pakar.
3. *Programmer* adalah orang yang membuat program sistem pakar, mengkode *domain* pengetahuan agar dapat dimengerti oleh komputer.
4. *Project Manager* adalah pemimpin dalam tim pengembangan sistem pakar.
5. *End-User* adalah orang yang menggunakan sistem pakar.

II.2.6. *Rule* Sebagai Teknik Represantasi Pengetahuan

Menurut T.Sutojo, dkk (2011 : 170), Setiap rule terdiri dari 2 bagian, yaitu bagian *IF* disebut *evidence* (fakta-fakta) dan bagian *THEN* disebut dengan hipotesis atau kesimpulan.

Syntax Rule adalah IF E THEN H.

E : *Evidence* (fakta-fakta yang ada).

H : Hipotesis atau kesimpulan yang dihasilkan.

Rule mempunyai *evidence* lebih dari satu yang dihubungkan oleh kata penghubung *AND* atau *OR*, atau kombinasi keduanya. Tetapi sebaiknya biasakan menghindari penggunaan *AND* dan *OR* secara sekaligus dalam satu *rule.*

II.3. Penyakit Kanker Lidah

Kanker lidah atau yang juga sering disebut sebagai kanker mulut, adalah jenis kanker ganas yang ciri utamanya diawali dengan gejala sariawan menahun yang tidak kunjung sembuh. Penyebab yang paling sering memicu munculnya gejala kanker lidah adalah gesekan yang terjadi pada area lidah dalam jangka

panjang. Dengan adanya gesekan ini akan mengakibatkan sariawan berkepanjangan, yang menyebabkan hygienists mulut menjadi buruk dan memicu kanker lidah. Kebiasaan merokok dengan kandungan zat karsinogenik, dan kebiasaan mengkonsumsi alkohol yang mengandung senyawa pembantu penyerapan zat karsinogenik juga merupakan faktor pemacu tumbuhnya kanker lidah. Hampir 80 % kanker lidah terletak pada 2/3 anterior lidah (umumnya pada tepi lateral dan bawah lidah) dan dalam jumlah sedikit pada posterior lidah (Daftary,1992; Tambunan,1993; Pinborg,1986). Gejala pada penderita tergantung pada lokasi kanker tersebut. Bila terletak pada bagian 2/3 anterior lidah, keluhan utamanya adalah timbulnya suatu massa yang seringkali terasa tidak sakit. Bila timbul pada 1/3 posterior, kanker tersebut selalutidakdiketahui oleh penderita dan rasa sakit yang dialami biasanya dihubungkandengan rasa sakit tenggorokan. Kanker yang terletak 2/3 anterior lidah lebih dapat dideteksi dini daripadarang terletak pada 1/3 posterior lidah. Kadang-kadang metastase limph node regional mungkin merupakan indikasi pertama dari kanker kecil pada lidah :Pinborg,1986).

Pada stadium awal, secara klinis kanker lidah dapatbermanifestasi dalam berbagai bentuk, dapat berupabercak leukoplakia, penebalan, perkembangan eksofitik atau endofitik bentuk ulkus. Tetapisebagian besar dalam bentuk ulkus:Daftary,1992). Lama-kelamaanulkus ini akan mengalami infiltrasi lebih dalam jangan tepi yang mengalamiindurasi (Pinborg,1986). Umumnya tidak menimbulkan rasa sakit kecuali ada infeksi sekunder. (Taufiqurrohman, et al, 2014)

II.4. 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.8)

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.9)

   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 :

$$m\{\theta\} = 1{,}0.$$

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 (Jurnal SAINTIKOM Vol. 12, No. 1, Muhammad Dahria, dkk., Januari 2013 : 4) :

$$m_3(Z) = \frac{\sum X \cap Y = Z\, m_1(X).m_2(Y)}{1 - \sum X \cap Y = \emptyset\, m1\,(X).m2(Y)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. (\mathrm{II}.10)$$

Studi kasus penerapan *Damster Shafer* pada diagnosa gangguan Autisme :

Pada Contoh dibawah ini, akan di cari persentase kemungkinan dari gangguan gangguan Autisme dengan menggunakan perhitungan pada table dibawah ini :

Tabel II.2. Contoh Gejala Pilihan Pasien

No.	Kode	Gejala	Bobot
1	GP16	Ekspresi muka kurang hidup	0,7
2	GP18	Menolak untuk dipeluk	0,8

(Sumber : Jurnal SAINTIKOM Vol. 12, No. 1, Muhammad Dahria, dkk., Januari 2013 : 4)

Maka untuk menghitung nilai Dempster Shafer(DS) gangguan **Autisme** yang dipilih dengan menggunakan nilai *believe* yang telah ditentukan pada setiap gejala.

Pl(Θ) = 1 – Bel

Dimana nilai bel (*believe*) merupakan nilai bobot yang diinput oleh pakar, maka untuk mencari nilai dari kedua gejala diatas, terlebih dahulu dicari nilai dari Θ, seperti yang dibawah ini.

Gejala 1 : ekspresi muka kurang hidup

(GP16)

Maka : GP16(bel) = 0.7

GP16(Θ) = 1-0,7

= 0.3

Gejala 2 : menolak untuk dipeluk (GP18)

Maka : GP18 (bel) = 0.8

GP18(Θ) = 1- 0,8

= 0,2

Maka untuk mencari nilai dari GPn, digunakan rumus :

$$m_3(Z) = \frac{\sum X \cap Y = Z \; m_1(X).m_2(Y)}{1 - \sum X \cap Y = \emptyset \; m1\,(X).m2(Y)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. (II.10)$$

Makan nilai GPn dari gejala diatas adalah:

0.7*0.8

1-(0.3*0.2)

= 0.56 / 1-0.06

= 0.56 / 0,94

= 0,59

Maka nilai densitas dari kedua gejala tersebut adalah 0,59. Dengan nilai densitas 0,59 maka pasien memiliki eviden yang cukup kuat mengalami gangguan autisme.

II.5. Pengertian *Database*

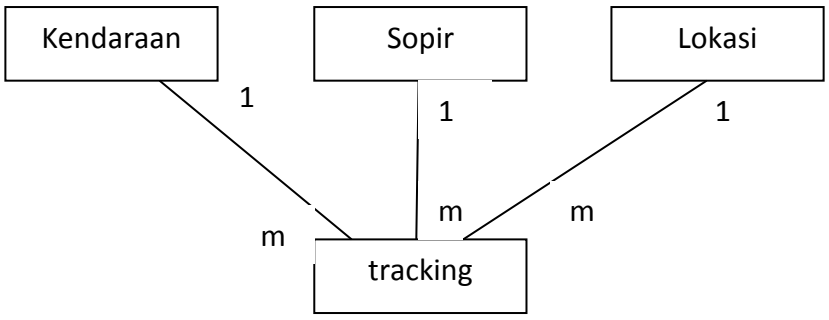
Database sering didefinisikan sebagai kumpulan data yang terkait. Secara teknis, yang berada dalam sebuah database adalah sekumpulan tabel atau objek lain (indeks, view, dan lain-lain). Tujuan utama pembuatan database adalah untuk memudahkan dalam mengakses data (Eka Choliviana dan Lies Yulianto, 2013).

Database adalah sekumpulan data yang berisi informasi mengenai satu atau beberapa object. Data dalam database tersebut biasanya disimpan dalam tabel yang saling berhubungan antara satu dengan yang lain (IJNS Volume 2 No 3 – Dani Ainur Rivai, Sukadi, Juli 2013 :15).

II.6. ERD

Model data yang digunakan dalam penelitian iniadalah *Entity Relationship Diagram* (ERD) sebagai sarana untuk menggambarkan hubungan antar data di dalam sistem. Dimaksudkan untuk komponen-komponen himpunan suatu entitas dan himpunan relasi yang menggambarkan fakta yang digunakan sebagai kebutuhan pembuatan sistem. Pada gambar di bawah ini merupakan rancangan ERD dari Rancangan Sistem Penalaran Berbasis Kasus untuk Mendiagnosa Penyakit Gigi dan Mulut (Seminar Nasional Teknologi Informasi dan Multimedia, Fryda Fatmawayati, 2015).

Dibawah ini adalah contoh ERD pada sistem informasi pemantauan posisi kendaraan dinas Unsri menggunakan teknologi GPS (Jurnal Sistem Informasi (JSI), VOL. 5, NO. 2, Ahmad Rifai, Oktober 2013) :



Gambar II.2. ERD
(Sumber : Ahmad Rifai, 2013)

II.7. Kamus Data

Menurut Roger. S. Pressman [3], kamus data adalah sebuah daftar yang terorganisasi dari elemen data yang berhubungan dengan system, dengan definisi yang tegar an teliti, sehingga pemakai dan analisis system akan memiliki pemahaman yang umum mengenai input, output, dan komponen penyimpan dan bahkan kalkulasi intermediate. (Jurnal Ilmiah Komputer dan Informatika (KOMPUTA) Vol. 2, No. 2, Oktober 2013 : 30).

Contoh dari kamus data sistem informasi pemantauan posisi kendaraan dinas Unsri menggunakan teknologi GPS.

Kendaraan : @id_kendaraan + jenis_kendaraan + merk + no_plat

Sopir : @id_sopir + nama + alamat

Lokasi : @id_lokasi + koordinat + nama_lokasi

Tracking : @id_tracking + #id_kendaraan + #id_sopir + # id_lokasi + tgl + jam (Jurnal Sistem Informasi (JSI), VOL. 5, NO. 2, Oktober 2013).

II.8. 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.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*. (Jurnal Ilmiah Mikrotek Vol. 1, No.1, Rika Yunitarini, 2013 :47).

II.10. *Microsoft Visual Basic 2010*

Microsoft Visual Basic.NET (VB.NET) adalah suatu pengembangan aplikasi bahasa pemrograman berbasis Visual Basic dan merupakan bahasa pemrograman terbaru buatan Microsoft setelah Microsoft Visual Basic 6.0. Pengembangan yang signifikan dari VB.NET ialah kemampuannya memanfaatkan platform NET, sehingga pengguna dapat membuat aplikasi Windows, aplikasi konsol, pustaka kelas, layanan NT, aplikasi web form, dan XML Web Service, yang secara keseluruhan memungkinkan integrasi tanpa batas dengan bahasa pemrograman lain sehingga berpeluang untuk berintegrasi dengan web. (Journal Speed, Volume 7 No 2, Widiana Mulyani, Bambang Eka Pratama, 2015 : 16).

Beberapa keunggulan lainnya yang dimiliki VB.NET, seperti memiliki penanganan debug yang baik sehingga pembangun aplikasi dapat mengetahui kesalahan kode yang terjadi secara cepat dan memiliki Windows form design yang memungkinkan pembangun/developer memperoleh aplikasi desktop dalam waktu singkat. VB.NET memiliki Interface Development Environment (IDE) yang lebih lengkap dan mudah bagi user pemula untuk mencari komponen atau objek yang kita inginkan, seperti menempelkan kontrol-kontrol yang terdapat pada toolbox, mampu memformat secara otomatis ukuran textbox, serta mengatur property dari masing-masing kontrol. VB.NET juga memiliki .NET Framework.

Microsoft .NET ialah sebuahplatform untuk membangun, menjalankan, dan meningkatkan generasi lanjut dari aplikasi terdistribusi, memperluas klien, server dan serviceservice. (Journal Speed, Volume 7 No 2, Widiana Mulyani, Bambang Eka Pratama, 2015 : 16).

II.11. *Unified Modelling Language* (UML)

Menurut Henderi (2012:152), *Unified Modelling Language* (UML) merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem software yang terkait dengan objek. UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek karena UML menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem membuat *blue print* atas visinya dalam bentuk yang baku. UML berfungsi sebagai jembatan dalam mengkomunikasikan beberapa aspek dalam sistem melalui jumlah elemen grafis yang bisa dikombinasikan menjadi diagram (JURNAL SISFOTEK GLOBAL, Vol. 5 No. 2, Rosana Junita Sirait, et al., September 2015).

Menurut Rama (2008:111), "*Unified Modeling Language* (UML) adalah suatu bahasa permodelan untuk menyebutkan, memvisualisasikan, membuat dan mendokumentasikan sistem informasi." Menurut Henderi (2007:4), "*Unified Modeling Language* (UML) adalah sebuah bahasa pemodelan yang telah menjadi standar dalam industri *software* untuk visualisasi, merancang, dan mendokumentasikan sistem perangkat lunak." Bahasa pemodelan UML lebih cocok untuk pembuatan perangkat lunak dalam bahasa pemrograman berorientasi

objek (C+, Java, VB.NET), namun demikian tetap dapat digunakan pada bahasa pemrograman prosedural.

Berdasarkan beberapa pendapat dikemukakan diatas dapat ditarik kesimpulan bahwa "*Unified Modeling Language* (UML) adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk menvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis OO (*Object Oriented*). (Seminar Nasional Teknologi Informasi dan Multimedia, Aris, et al., 2015 ; 12).

II.11.1.Fungsi *Unified Modelling Language* (UML)

Unified Modeling Language (UML) biasa digunakan untuk (Seminar Nasional Teknologi Informasi dan Multimedia, Aris, et al., 2015 ; 12) :

1. Menggambarkan batasan sitem dan fungsi -fungsi sistem secara umum, dibuat dengan *use case* dan *actor*.
2. Menggambarkan kegiatan atau proses bisnis yang dilaksanakan secara umum, dibuat dengan *interaction diagrams*.
3. Menggambarkan representasi struktur *static* sebuah sistem dalam betuk *class diagrams*.
4. Membuat model behavior "yang menggambarkan kebiasaan atau sifat sebuah sistem" dengan *state transition diagrams*.
5. Menyatakan arsitektur implementasi fisik menggunakan *component and development.*
6. Menyampaikan atau memperluas *fungsionality* dengan *stereotypes*.

II.11.2.Diagram-Diagram *Unified Modelling Language* (UML)

Adapun jenis-jenis dari diagram UML adalah sebagai berikut :

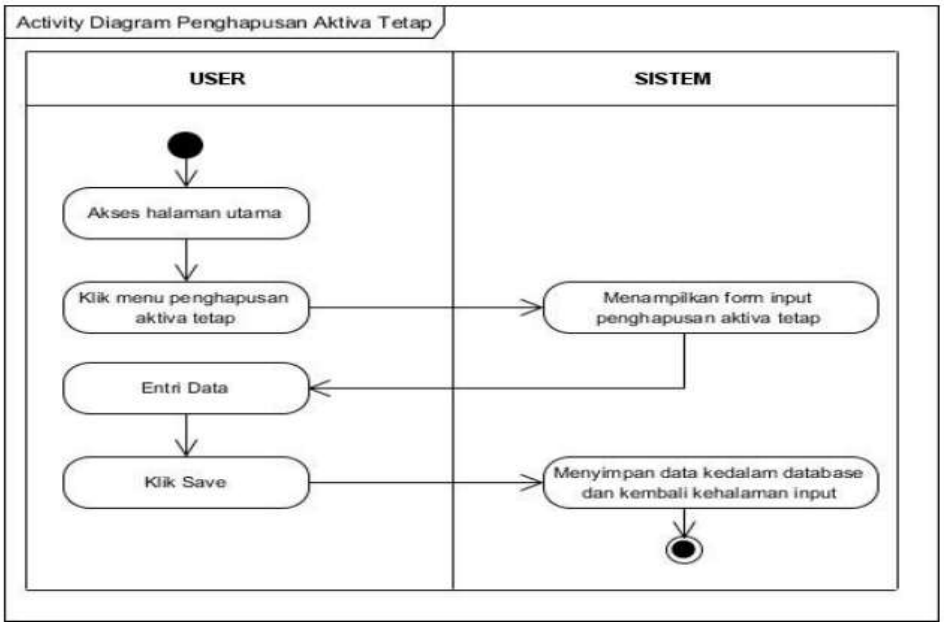
1. *Use Case* Diagram

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 (Jurnal Online ICT STMIK IKMI – Vol. 13 - No. 1, Achmad Hamzah, Dadang Sudrajat, 2015 : 6).

Gambar II.3. *Use Case* Diagram
(Sumber : Rosana Junita Sirait, et al., 2015)

2. *Activity* Diagram

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 (Jurnal Online ICT STMIK IKMI – Vol. 13 - No. 1, Achmad Hamzah, Dadang Sudrajat, Juli 2015 : 6).

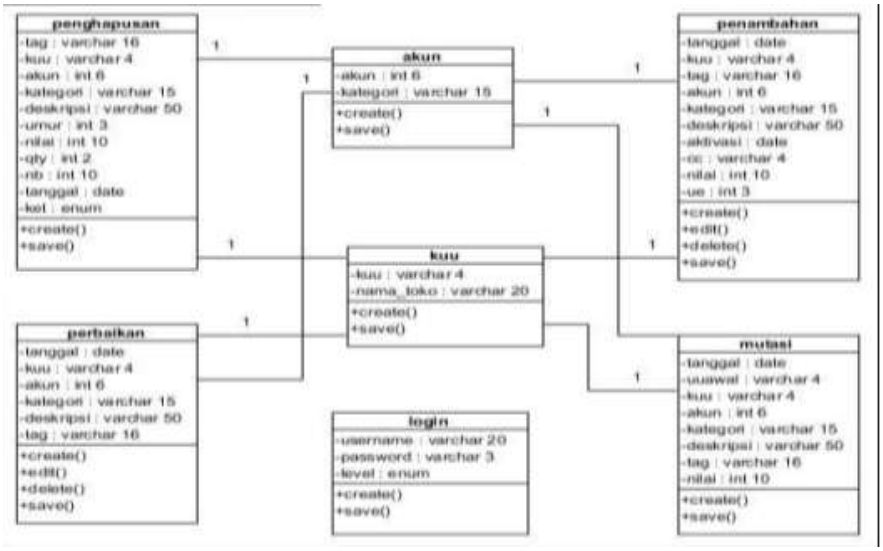


Gambar II.4. *Activity* Diagram
(Sumber : Rosana Junita Sirait, et al., 2015)

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

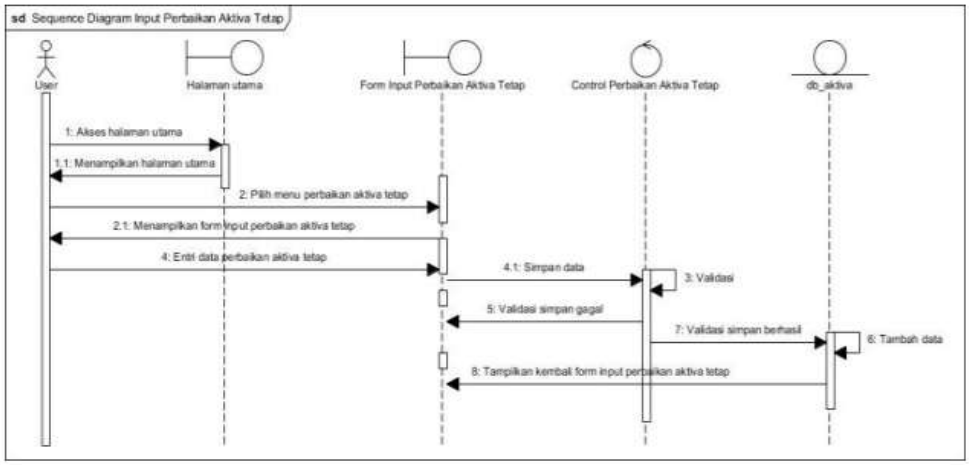
vocabulary dari sistem, pemodelan kerjasama, atau pemodelan skema. *Class* diagram dapat digunakan untuk membangun sistem yang dapat dieksekusi melalui teknik *forward and reverse*, selain untuk penggambaran, menspesifikasikan, dan pendokumentasian struktur model (Jurnal Online ICT STMIK IKMI – Vol. 13 - No. 1, Achmad Hamzah, Dadang Sudrajat, Juli 2015 : 6).



Gambar II.5. *Class* Diagram
(Sumber : Rosana Junita Sirait, et al., 2015)

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.6. *Sequence* Diagram
(Sumber : Rosana Junita Sirait, et al., 2015)