BAB II

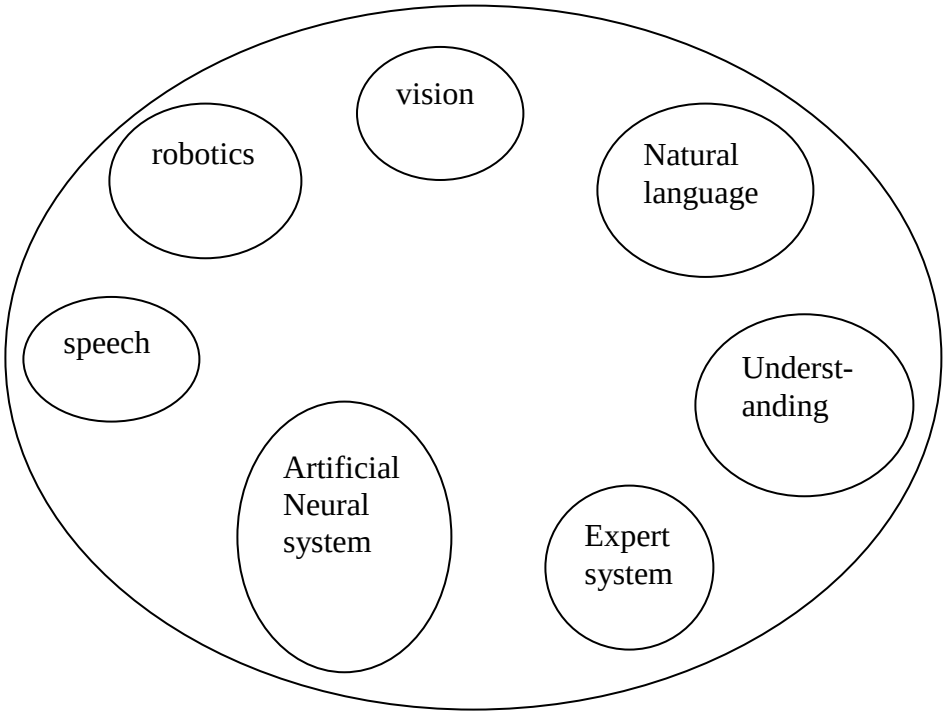
TINJAUAN PUSTAKA

II.1. Sistem Pakar

Sistem pakar adalah sistem berbasis komputer yang menggunakan pengetahuan, fakta, dan tehnik penalaran dalam memecahkan masalah yang biasanya hanya dapat dipecahkan oleh seorang pakar dalam bidang tersebut. Sistem pakar memberikan nilai tambahan pada teknologi untuk membantu dalam menangani era informasi yang semakin canggih. (Jurnal : Aplikasi sistem pakar dioagnosa penyakit ginjal metode dempster shafer, Aprilia Sulistyohati, dkk : 2008 : E-1)

Artificial intelligence (Al) memiliki beberpa dominan masalah atau arena. Bidang sistem pakar merupakan penyelesaian pendekatan yang sangat berhasil dan bagus untuk permasalahan Al klasik dari pemrograman *intelligent* (cerdas). Sistem pakar *(expert system)* merupaka solusi Al bagi masalah pemrograman pintar (*intelligent*). Professor Edward Feigenbaum dari Stanford University yang merupakan pionir dalam teknologi sistem pakar mendefenisikan sistem pakar sebagai sebuah program komputer pintar *(intelligent computer program)* yang memanfaatkan pengetahuan (*knowledge*) dan prosedur inferensi *(inference procedure)* untuk memecahkan masalah yang cukup sulit hingga membutuhkan keahlian khusus dari manusia. Dengan kata lain, sistem pakar adalah sistem komputer yang ditujukan untuk meniru semua aspek (*emulates*) kemampuan pengambilan keputusan *(decision*

making) seorang pakar. Sistem pakar memanfaatkan secara maksimal pengetahuan khusus selayaknya seorang pakar untuk memecahkan masalah. (Rika Rosnelly:2012:2)



Gambar II.1 Area dari Artificial Intelligent (Al)

(Sumber : Rika Rosnelly:2012:3)

II.1.1. Kelebihan Sistem Pakar

Sistem pakar memiliki beberapa fitur menarik yang merupakan kelebihannya , seperti :

1. Meningkatkan ketersediaan *(increased availibilty)*. Kepakaran atau keahlian menjadi tersedia dalam system computer. Dapat dikatakan bahwa sistem pakar merupakan produksi kepakaran secara masal *(massproduction).*
2. Mengurangi biaya *(reduced cost)*. Biaya yang diperlukan untuk menyediakan keahlian per satu orang user menjadi berkurang.
3. Mengurangi bahaya *(reduce danger).* Sistem pakar dapat di gunakan dilingkungan yang mungkin berbahaya bagi manusia.
4. Permanen (*permanence*). Sistem pakar dan mengetahui yang terdapat didalam bersifat lebih permanen dibandingkan manusia yang dapat merasa lelah, bosan, dan pengetahuannya hilang saat sang pakar menghilang.
5. Keahlian multiple *(multiple expertise).* Pengetahuan dari beberapa pakar dapat dimuat ke dalam sistem dan bekerja secara simultan dan kontinyu menyelesaikan suatu masalah setiap saat. Tingkat dan keahlian atau pengetahuan yang digabungkan dari beberapa pakar dapat melebihi pengetahuan suatu seorang pakar.
6. Meningkatkan kehandalan *(increased reliability).* Sistem pakar meningkatkan kepercayaan dengan memberikan hasil yang benar sebagai alternative pendapat dari seorang pakar atau sebagai penengah jika trjadi konflik antara beberapa pakar . namun hal tersebut tidak berlaku jika system dibuat oleh salah seorang pakar, sehingga akan selalu sama dengan pendapat pakar tersebut kecuali jika sang pakar melakukan kesalahan yang mungkin terjadi pada saat tertekan atau stress.

7. Penjelasan (*explanation*). Sistem pakar dapat menjelaskan detail proses penalaran (*reasoning*) yang dilakukan hingga mencapai suatu kesimpulan. Seorang pakar mungkin saja terlalu lelah, tidak bersedia atau tidak mampu melaksanakannya setiap waktu. Hal ini akan meningkatkan tingkat kepercayaan bahwa kesimpulan yang dihasilkan adlah benar.
8. Respon yang cepat *(fast response).* Respon yang cepat atau *real time* diperlukan dari beberapa aplikasi. Meskipun bergantung pada *hardware* dan *software* yang digunakan, namun sistem pakar memberiakn respon yang lebih cepat dibandingkan seorang pakar.
9. Stabil, tidak emosional, dan memberikan respon yang lengkap setiap saat *(steady, unemotional and complete response at all times).* Karakteristik ini diperlukan pada situasi *real-time* dan keadaan darurat (*emergency*) ketika seorang pakar mungkin tidak berada pada kondisi puncak disebabkan oleh stress atau kelelahan.
10. Pembimbing pintar *(intelligent tutor).* Sistem pakar dapat berperan sebagai *intelligent tutor* dengan memberikan kesempatan pada user untuk menjalankan contoh program dan menjelaskan proses *reasoning* yang dilakukan.
11. Basis data cerdas *(intelligent database).* Sistem pakar dapat digunakan untuk mengakses basis data secara cerdas. (Rika Rosnelly:2012:5-6)

II.1.2 Elemen Manusia Pada Sistem Pakar

Sistem pakar tidak terlepas dari elemen manusia yang terkait di dalamnya personil yang terkait dengan sistem pakar ada 4 :

1. Pakar (*expert*)

   Pakar adalah seorang individu yang memiliki pengetahuan khusus, pemahamana, pengalaman, dan metode – metode yang digunakan untuk memecahkan persoalan dalam bidang tertentu.

2. Pembangunan / pembuatan pengetahuan

   Pembangunan pengetahuan memiliki tugas utama menerjemahkan dan mempresentasikan pengetahuan yang diperoleh dari pakar, baik berupa pengalaman pakar dalam menyelesaikan masalah maupun sumber terdokumentasi lainnya ke dalam bentuk yang bisa diterima oleh sistem pakar.

3. Pembangunan / Pembuatan Sistem

   Pembangunan sistem adalah orang yang bertugas untuk merancang antarmuka pemakai sistem pakar, merancang pengetahuan yang sudah diterjemahkan oleh pembangunan pengetahuan ke dalam bentuk yang sesuai dan dapat diterima oleh sistem pakar dan mengimplementasikannya ke dalam mesin inferensi.

4. Pengguna

   Banyak sistem berbasis computer mempunyai susunan pengguna tunggal. (Rika Rosnelly:2012:9-12)

II.2. Teori *Dempster Shafer*

Ada berbagai macam penalaran dengan model yang lengkap dan sangat konsisten, tetapi pada kenyataan banyak permasalahan yang tidak dapat terselesaikan secara lengkap dan konsisten. Ketidakkonsistenan yang tersebut adalah akibat adanya penambahan fakta baru. Permasalahan yang seperti itu disebut dengan penalaran *non monotonis*. Untuk mengatasi ketidakkonsistenan tersebut maka dapat menggunakan penalaran dengan teori *Dempster Shafer*.

Secara umum teori dempster shafer ditulis dalam suatu interval :

[*Belief, Plausibility*]

1. Belif (*Bel*) adalah ukuran kekuatan *evidence* dalam mendukung suatu himpunan proposisi. Jika bernilai 0 maka mengindikasikan bahwa tidak ada *evidence*, dan jika bernilai 1 maka ada kepastian.
2. *Plausibility* (P1) dinotasikan sebagai :

   P1(s) – 1- Bel (⁻s)

   Plausibility juga bernilai 0 sampai 1. Jika yakin akan ⁻s, maka dapat dikatak bahwa Bel(⁻s)=1, dan P1(⁻s)=0.

Pada teori *dempster shafer* dikenal dengan adanya *frame of discrement* yang dinotasikan dengan Θ. *Frame* 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 mendefenisikan elemen –

elemen Θ saja, namun juga semua subsetnya. Sehinggan jika Θ berisi n elemen, maka subset Θ adalah 2". Jumlah semua m dalam subset Θ sama dengan 1. Apabilah tidak ada informasi untuk memilih hipotesis , maka nilai :

M{ Θ} = 1,0

Apabila diketahui X adalah subset dari Θ, dengan m1 sebagai fungsi densitasnya, dan Y juga merupaka subset dari Θ dengan m2 sebagai fungsi densitasnya, maka dengan bentuk fungsi kombinasi m1 dan m2 sebagai m3, yaitu :

$$M3(Z) = \frac{\Sigma x\eta y = m1\,(X).\ m2(Y)}{1 - \Sigma x\eta y = \Theta\ m1\,(X).\ m2(Y)}$$

(Jurnal : Aplikasi sistem pakar dioagnosa penyakit ginjal metode dempster shafer, Aprilia Sulistyohati, dkk : 2008 : E-2)

II.3. Tes IQ Anak

IQ adalah skor yang diperoleh dari sebuah alat tes kecerdasa IQ (Intelegence Quotient) yang hanya memberikan sedikit indikasi mengenai taraf kecerdasan seseorang dan tidak menggambarkan kecerdasan seseorang secara keseluruhan. (Jurnal : simulasi tes IQ berbasis web, Muhammad Sahal : 2008:4)

II.3.1 Keseimbangan Otak Kiri dan Kanan dalam Kecerdasan

Masyarakat sering kali menilai IQ(intelligence quotient) disamakan dengan intelegensi atau kecakapan. Pada hal, IQ hanya mengukur sebagian kecil dari kecakapan. Justru anak yang cerdas itu adalah anak yang bisa bereaksi secara logis dan berguna terhadap apa yang dialami dilingkungannya.

IQ merupakan angka yang dipakai untuk menggambarkan kapasitas berfikir seorang dibandingkan dengan dengan rata – rata orang lain.(Burhan Elfanany:2013:28)

Umumnya, Tes IQ yang digunakan berdasarkan standart Wechsler. Berikut skornya dari yang tertinggi hingga yang terendah :

1. Sangat cerdas : 128 keatas
2. Cerdas : 120 – 127
3. Di atas rata – rata : 111 – 119
4. Rata – rata : 91 – 110
5. Di bawah rata – rata : 79 -90
6. Terbalakangan mental : 65 kebawah

(Burhan Elfanany:2013:53)

II.3.2. Langkah – langkah Meningkatkan Kecerdasan Anak

Setiap orang tua pasti mendambakan anaknya memiliki otak yang cerdas dan pintar. Namun tahukah Anda bahwa kepintaran banyak dipengaruhi oleh lingkungan sianak? Selain dengan belajar disekolah, ada beberapa factor lain yang dapat meningkatkan tingkat kecerdasan sianak.

Pakar dokter menyatakan, bayi dalam kandungan usia 3 bulan sudah mempunyai perasaan, 4 bulan sudah mampu merasakan suara dari luar, suara dari luar ini terus merangsang organ indera anak dalam kandungan dan mendorong pertumbuhannya, mempunyai peran yang penting bagi pertumbuhan intelegensi.

Para pakar menyatakan, “anak – anak pada rentang usia 4 – 13 tahun, karena belum banyak mengecap asam garam dunia, hatinya masih murni, merupakan masa dengan daya ingat yang paling kuat selama hidupnya”. Jika pada masa keemasan ingatan ini memperoleh pendidikan yang baik, akan sangat bermanfaat bagi sepanjang hidupnya.

Beberapa cara yang bisa dilakukan untuk membuat anak menjadi lebih pintar adalah :

1. Pemberian ASI

   ASI merupakan makanan otak yang paling dasar. Penelitian secara konsisten terus menunjukan berbagai macam keuntungan asi yang berhubungan dengan pertumbuhan bayi.

2. Bermain permainan yang berfikir

   Catur, teka – teki silang dan Sudoku selain menyenangkan juga mendukung strategi berfikir anak – anak, bagaimana cara menyelesaikan masalah, dan membuat keputusan yang kompleks.

3. Bermain musik

   Bermain musik selain menyenangkan juga bisa merangsang pertumbuhan otak kanan.

4. Membiasakan berolah raga

   Para peneliti di universitas lllinois menunjukkan hubunga yang kuat antara kebugaran dan prestasi akademik di antara anak – anak sekolah dasar. Semakin

bugar badan sang anak maka kemampuan dalam menerima pelajaran juga meningkat.

5. Menyingkirkan makana siap saji / *junk food*

   Mengurangi asupan gula, lemak trans dari makanan siap saji dan menggantinya dengan makanan bergizi tinggi yang baik untuk perkembangan mental anak usia dini serta berfungsi dalam perkembangan motorik anak pada usia 1-2 tahun pertama.

6. Memberikan sarapan yang sehat

   Para peneliti meyakinkan bahwa mengonsumsi sarapan yang sehat akan meningkatkan memori dan konsentrasi anak dalam belajar.

7. Tingkatkan kesehatan

   Tim peneliti dari universitas of lllionois telah membuktikan hubungan antara kesehatan dan pelajaran anak di sekolah

8. Permainan

   Memang banyak games yang bisa membuat pemainnya menjadi brutal, nyentrik ataupun malas berpikir anak yang bermain games lebih berkemampuan dalam menemukan petunjuk dalam belajar.

9. Memupuk rasa ingin tahu

   Para pakar mengungkapkan, ketika orang tua mendorong anak untuk mempunyaipemikiran sendiri, sesungguhnya adalah sedang meng-arahkan pada pentingnya menuntut pengetahuan.

10. Membaca

Sejalan dengan kemajuan teknologi, banyak orang yang mengabaikan pentingnya membaca. Membaca merupakan cara meningkatkan *intelegence quotient* yang paling langsung dan efektif. (Burhan Elfanany:2013:65 - 69)

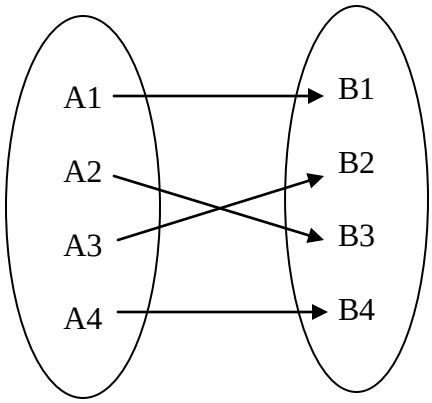
II.4. MYSQL

MYSQL diciptakan dinegara Swedia oleh perusahaan MYSQL AB. Perangkat lunak ini tersebar luas secara gratis karena memiliki lisensi GNU (*General Public License*). Sampai sekarang, yang bisa sinkronisasi dengan MYSQL, seperti C, C++, C#, bahas pemograman Eiffel, bahasa pemograman Smalltalk, bahasa pemograman Java, bahasa pemograman Lisp,Perl,PHP, bahasa pemograman Pyton, Ruby, Realbasic, dan tcl. **(**Andre Adelheid, dkk 2012:3)

II.4.1. Jenis Hubungan

Dalam banyak *literature*, jenis hubungan antara antara dua tipe entitas dinyatakan dengan istilah *one-to-one, one-to-many, many-to-one*, dan *many-to-many*. Dengan mengasumsikan bahwa terdapat dua buah tipe entitas bernama A dan B, penjelasan masing – masing jenis hubungan tersebut adalah seperti berikut :

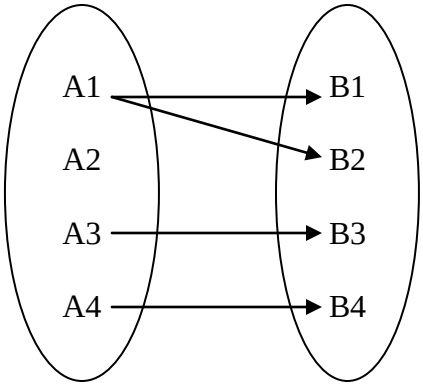
1. Hubunga *one-to-one* (1:1) menyatakan bahwa setiap entitas A paling banyak berpasangan dengan satu entitas pada entitas B. begitu sebaliknya. Berikut contoh hubungan one-to-one :



Gambar : II.2 Hubungan *one-to-one*

Sumber (Abdul Khadir : 2009 : 47)

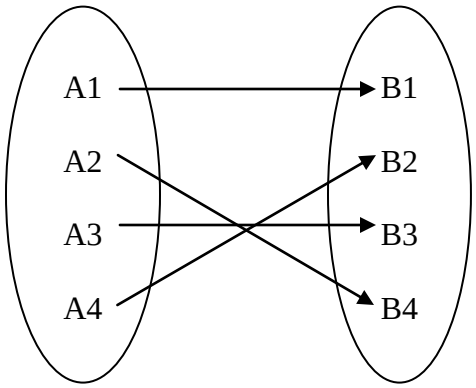
2. Hubungan *one-to-many* (1:M) menyatakan bahwa setiap entitas A bisa berpasangan dengan banyak pada entitas B, sedangkan setiap entitas B hanya bisa berpasangan pada entitas B, terlihat pada gambar dibawah ini :



Gambar : II.3 Hubungan *one-to-many*

Sumber (Abdul Khadir : 2009 : 47)

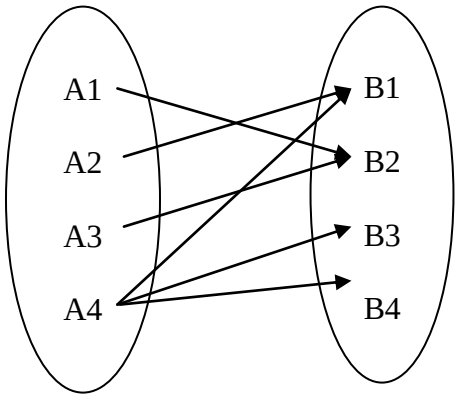
3. Hubungan *many-to-one* (M:1) menyatakan pada setiap entitas A paling banyak berpasangan dengan satu entitas pada entitas B dan setiap entitas B bisa berpasangan dengan banyak pada entitas A, terlihat pada gambar dibawah :



Gambar : II.4 Hubungan *many-to-one*

Sumber (Abdul Khadir : 2009 : 47)

4. Hubungan *many-to-many* (M:N) menyatakan bahwa setiap entitas pada suatu tipe entitas A bisa berpasangan dengan banyak entitas pada tipe entitas B dan begitu sebaliknya, terlihat pada gambar dibawah ini :



Gambar : II.5 Hubungan *many-to-many*

Sumber (Abdul Khadir : 2009 : 47)

(Abdul Khadir : 2009 : 46-47)

II.4.2. Normalisasi

Normalisasi adalah suatu proses yang digunakan untuk menentukan pengelompokkan atribut – atribut dalam sebuah relasi sehingga diperoleh relasi yang berstruktur baik. (Abdul Khadir : 2009 : 116)

Berikut adalah bentuk normal dalam normalisasi ada beberapa bentu yaitu sebagai berikut :

1. Bentuk normal pertama (1NF)

   Bentuk normal pertama (1NF) adalah suatu keadaan yang membuat setiap perpotongan baris dan kolom dalam relasi hanya berisi satu nilai. Untuk membentuk relasi agar berada dalam bentuk normal pertama, perlu lengkah untuk menghilangkan atribut – atribut yang bernilai ganda.

2. Bentuk normal kedua (2NF)

   Bentuk normal kedua (2NF) adalah suatu bentuk yang menyatakan bahwa relasi harus sudah berada dalam bentuk normal pertama dan tidak mengandung dependensi persial.

3. Bentuk normal ketiga (3NF)

   Bentuk normal ketiga (3NF) adalah suatu keadaan yang mnyatakan bahwa relasi harus sudah berada dalam bentuk normal kedua dan tidak mengandung *dependensi transitif*.

4. Bentuk normal *Boyce-Codd*

   Bentuk normal *Boyce-Codd* adalah suatu keadaan yang menyaratkan bahwa setiap determinan (penentu) dalam suatu relasi berkedudukan sebagai kunci kandidat.

5. Bentuk normal keempat (4NF)

   Bentuk normal keempat (4NF) adalah suatu keadaan yang menyaratkan relasi berada pada BCNF dan tidak mengandung lebih dari satudepedensi bernilai-banyak yang bersifat independen.

6. Bentuk normal kelima (5NF)

   Bentuk normal kelima (5NF) adalah suatu keadaan yang membuat suatu relasi yang telah memenuhi bentuk normal keempat yang tidak tidak dapat didekomposisi menjadi relasi – relasi pecahnya tersebut tidak sama dengan kunci kandidat relasi. (Abdul Khadir : 2009 : 130-144)

II.4.3. Anomali

Anomali adalah masalah yang timbul dalam relasi ketika terjadi operasi pemutakhiran data dalam relasi. Masalah yang terjadi misalnya pengubahan data yang membuat ketidakkonsistenan data atau bahkan membuat sesuatu data menjadi hilang. Jenis – jenis anolami :

1. Anomali penyisipan

   Anomali penyisipan adalah masalh yang terjadi ketika suatu baris disisipkan kedalam relasi. Anomaly ini terjadi dikarenakan kunci primernya tidak bernilai (bernilai NULL).

2. Anomali pengubahan

   Anomali pengubahan adalah masalah yang timbul ketika data dalam relasi diubah.

3. Anomali penghapusan

   Anomali penghapusan adalah masalah yang timbul ketika suatu baris dalam relasi dihapus. (Abdul Khadir : 2009 : 119-121)

II.5 Kamus Data (*Data Dictionary*)

Seperti halnya dengan kamus bahasa yang berfungsi menjelaskan lebih detail suatu kata maupun kalimat, kamus data yang digunakan dalam analisis struktur dan desain sistem informasi juga merupakan suatu katalog yang menjelaskan lebih detail tentang *data flow diagram* yang mencakup proses, *data flow* dan *data store*. Kamus data dapat digunakan pada metodologi berorientasi data dengan menjelaskan lebih detail lagi hubungan entitas, seperti atribut-atribut suatu entitas. Pada metodologi objek, kamus data dapat menjelaskan lebih detail atribut maupun metode atau *service* suatu objek. Apabila didefenisikan, kamus data adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan kamus data, sistem analis dapat mendefenisikan data yang mengalir pada sistem dengan lengkap.

Kamus data dibuat dan digunakan baik pada tahap analisis maupun pada tahap perancangan sistem. Pada tahap analisis, kamus data digunakan sebagai alat komunikasi antara sistem analis dengan *user* tentang data yang mengalir pada sistem tersebut serta informasi yang dibutuhkan oleh pemakai sistem (*user*). Sedangkan pada tahap perancangan sistem, kamus data digunakan untuk merancang *input*, *output* / laporan dan database. Kamus data dan komponen-komponen lainnya yang dikumpulkan pada saat analisis sistem sangat dibutuhkan dalam perancangan sistem.

Selain dapat digunakan untuk menjelaskan suatu model sistem, kamus data juga berfungsi untuk menghindari penggunaan kata-kata yang sama, karena kamus data disusun menurut abjad.

Kamus data dibuat berdasarkan arus data yang ada pada data flow diagram. Arus data yang ada di DFD bersifat global dan hanya menunjukkan nama arus datanya saja. Keterangan lebih lanjut tentang struktur dari suatu arus data di DFD dapat dilihat pada kamus data. Kamus data harus dapat mencerminkan keterangan yang jelas tentang data yang dicatatnya. Untuk keperluan ini, maka kamus data harus memuat hal-hal sebagai berikut

1. Arus Data

   Arus data menunjukkan dari mana data mengalir dan kemana data akan menuju. Keterangan arus data ini perlu dicatat di kamus data untuk memudahkan mencari arus data di dalam *Data Flow Diagram* (DFD).

2. Nama Arus Data

   Karena kamus data dibuat berdasarkan arus data yang mengalir di data flow diagram, maka nama dari arus data juga harus dicatat di kamus data, sehingga mereka yang membaca DFD dan memerlukan penjelasan lebih lanjut tentang suatu arus data tertentu di *data flow diagram* dapat langsung mencarinya dengan mudah di kamus data.

3. Tipe Data

   Telah diketahui bahwa arus data dapat mengalir dari hasil suatu proses ke proses yang lainnya. Data yang mengalir ini biasanya dalam bentuk laporan serta

dokumen hasil cetakan komputer. Dengan demikian bentuk dari data yang mengalir dapat berupa dokumen dasar atau formulir, dokumen hasil cetakan komputer, laporan tercetak, tampilan layar di monitor, variabel, parameter dan field-field. Bentuk data seperti ini perlu dicatat di kamus data.

4. Struktur Data

   Struktur data menunjukkan arus data yang dicatat pada kamus data yang terdiri dari item-item atau elemen-elemen data.

5. Alias

   Alias atau nama lain dari data juga harus dituliskan. Alias perlu ditulis karena data yang sama mempunyai nama yang berbeda untuk orang atau departemen lainnya.

6. Volume

   Volume yang perlu dicatat di dalam kamus data adalah volume rata-rata dan volume puncak dari arus data. Volume rata-rata menunjukkan banyaknya arus data yang mengalir dalam satu periode tertentu, sementara volume puncak menunjukkan volume yang terbanyak.

7. Periode

   Periode ini menunjukkan kapan terjadinya arus data. Periode perlu dicatat di kamus data karena dapat digunakan untuk mengidentifikasikan kapan input data harus dimasukkan kedalam sistem, kapan proses program harus dilakukan dan kapan laporan-laporan harus dihasilkan.

8. Penjelasan

   Untuk lebih memperjelas makna dari arus data yang dicatat di kamus data, maka bagian penjelasan dapat diisi dengan keterangan-keterangan tentang arus data tersebut.

II.6. UML (*Unified Modelling Language*)

UML (*Unified Modelling Language*) adalah sebuah bahasa yang telah menjadi standart dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menngunakan *class* dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa – bahasa beriorintasi objek seperti C ++, Java, C# atau VB.NET. walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahas lainnya, UML mendefenisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefenisikan bagaimana bentuk – bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya : Grady Booch

OOD (*Object-Oriented Design*), Jim Rumbaugh OMT(*Object Modeling Tecnique*), dan Ivar Jacobson OOSE(*Object-Oriented Software Engerineering*). (Yuni Sugiarti:2013:34)

Unified Modeling Language (UML) bisa digunakan untuk :

1. Menggambarkan batasan sistem dan fungsi – fungsi sistem secara umum, dibuat dengan *use case* dan *actor*.
2. Menngambarkan kegiatan atau proses bisnis yang dilaksanakan secara umum, dibuat dengan *interaction diagrams*.
3. Menggambarkan refresentasi struktur statik sebuah sistem dalam bentuk *class diagrams.*
4. Membuat model bihavior yang menngambarkan kebiasaan atau sifat sebuah sistem denagn *state tansition diagrams.* (Yuni Sugiarti:2013:36)

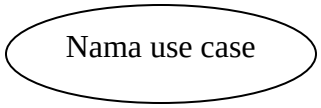
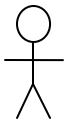
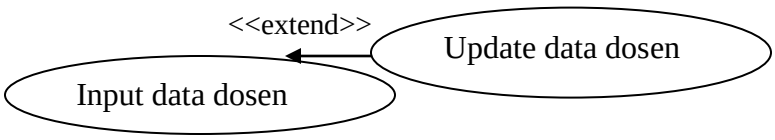
II.6.1.*Use Case Diagram*

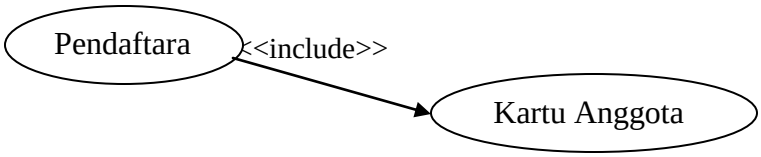
Use Case Diagram adalah menentukan kebutuhan. Terdapat 2 kebutuhan yaitu kebutuhan fungsional dan kebutuhan nonfungsional :

1. Kebutuhan fungsional adalah kebutuhan pengguna sehari – hari yang akan dimiliki oleh sistem, dimana kebutuhan ini akan digunakan oleh pengguna.
2. Kebutuhan nonfungsional adalah kebutuhan yang memperhatikan hal – hal berikut yaitu performansi, kemudahan dalam menggunakan sistem, kehandalan sistem, keamanan sistem, keuangan, legalitas, dan operasional.

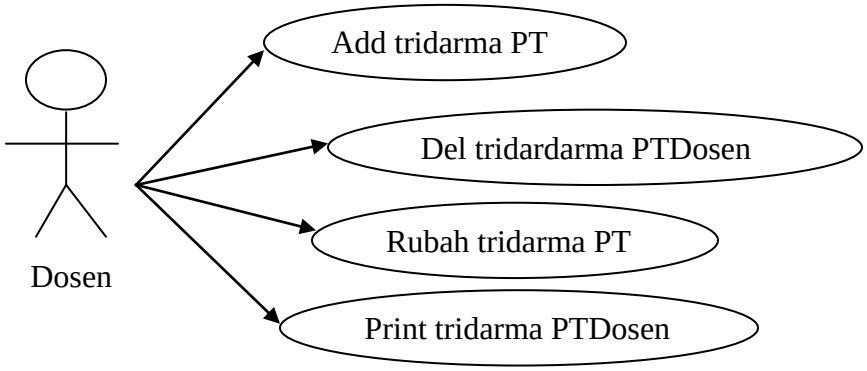
Berikut simbol – simbol yang digunakan di dalam *use case* :

Tabel II.1 Simbol – simbol *use case*

Simbol	Deskripsi
Use Case 	Fungsionalitas yang disediakan dalam sebuah unit – unit yang saling bertukar pesan antara unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal fase nama use case.
Aktor  nama aktor	Orang, proses, atau sistem lain yang berinteraksi dangan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda diawal frase nama aktor.
Asosiasi / association	Komunikasi antara aktor dan use case yang berpartisifasi pada use case atau use case memiliki interaksi dengan aktor.
Extend  <<extend>>	Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walau tampa use tambahan itu, mirip dengan prinsip inheritance pada pemorograman beriorintasi objek, biasanya use case tambahan memiliki memiliki nama depan yang sama dengan use case yang ditambahkan, arah panah menunjuk pada use case yang diju. Contoh : 
Include <<include>>	Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan

	fungsinnya atau sebagai syarat dijalankan use case ini, ada dua sudut pandang yang cukup besar mengenai include di use case, include berarti use case yang ditambahkan akan selalu dipangging saat use case tambahan dijalankan, contoh : 

Sumber (Yuni Sugiarti 2013:42)



Gambar : II.6 contoh *use case* diagram

Sumber (Yuni Sugiarti 2013:45)

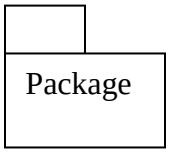
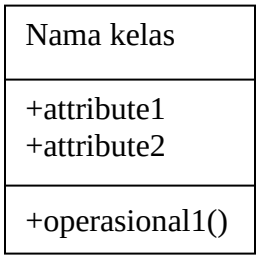
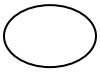
II.6.2 *Class Diagram*

Class Diagram menggarbarkan stuktur sistem dari segi pendefenisian kelas – kelas yang dibuat untuk membangun sistem. Adapun bagian dari *class diagram* adalah :

1. Abtrasi kelas yaitu menemukan hal – hal mendasar pada suatu objek dan mengabaikan hal – hal yang sifatnya isidental. Objek adalah instansiasi dari sebuah kelas.
2. Atribut adalah karakteristik yang dimiliki suatu objek dalam kelas
3. Operasi adalah fungsi transformasi yang mungkin dapat yang diaplikasikan ke suatu objek dalam kelas.
4. Multiplisitas / *multiplicity* menunjukan jumlah suatu objek yang bisa berhubungan dengan objek yang lain.

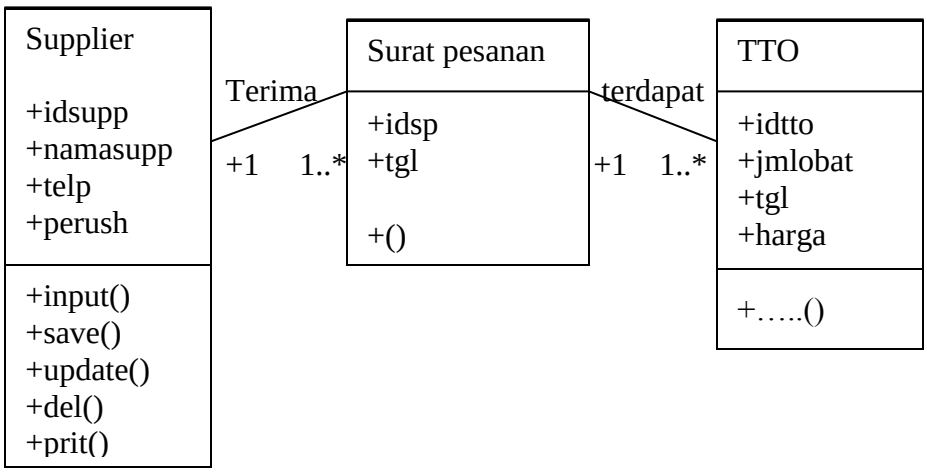
Berikut simbol – simbol yang ada di dalam diagram kelas :

Tabel II.2 simbol – simbol Diagram kelas

Simbol	**Deskripsi**
Package 	Package merupakan sebuah bungkusan dari satu atau lebih kelas
Operasi 	Kelas pada struktur sistem
Antarmuka / interface  Interface	Sama dengan konsep interface dalam pemograman beriorintasi objek
Asosiasi	Relasi antar kelas dengan makna umum, asosiasi

1 1..*	biasanya juga disertai dengan multiplicity
Asosiasi berarah / directed asosiasi	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity
Generalisasi	Relasi antar kelas dengan makna generelasi – spesialisasi (umum khusus)
Kebergantungan / defedency	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi	Relasi antar kelas dengan mekanisme bagian (*whole part*)

Sumber (Yuni Sugiarti 2013:59)



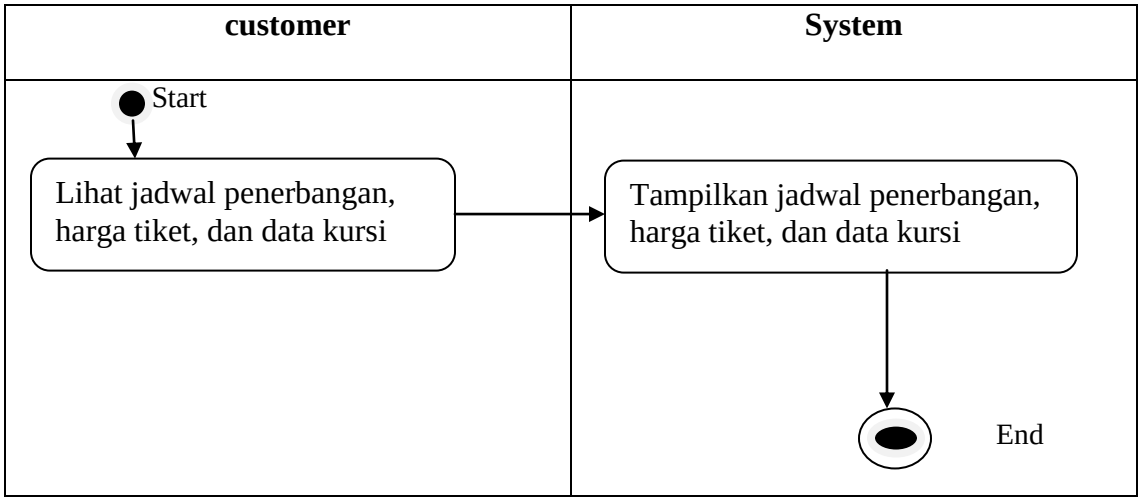
Gambar : II.7 contoh 1 *class diagram*

Sumber (Yuni Sugiarti 2013:60)

II.6.3 Diagram Aktivitas

Diagram aktivitas atau *activity diagram* menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas mendukung perilaku paralel. Contoh studi kasus sistem informasi penjualan tiket online :

Tabel : II.3 Studi kasus Diagram Aktivitas



Sumber (Yuni Sugiarti 2013:78)

II.7. PHP

PHP atau *Hypertext Preprocessor* merupakan bahasa berbentuk script yang ditempatkan dalam server dan dieksekusi dalam server untuk selanjutnya ditransfer dan dibaca oleh client. Php juga bisa disisipkan dalam bahasa HTML.

PHP pertama kali diciptakan oleh seorang pria berkewarganegaraan Denmark yang bernama Rasmus Lerdorf pada tahun 1995. Banyak programmer yang tertarik untuk mengembangkan php karena bersifat open source. Pada awal peluncurannya, php hanya dibuat untuk diintegrasikan dengan web server apache. Namun sekarang, php juga dapat bekerja dengan web server seperti PWS (*personal Web Server*), IIS (*Internet Information Server*), dan Xitama.

PHP sendiri tidak lepas dari database MYSQL. Oleh karna itu, dalam membuat sebuah website dengan bahasa pemrograman php, kita membutuhkan web server.(Andrea Adelhei, dkk:2012:2)

II.7.1 Mengenal Folder, Mengakses, serta menjalan Webserver & Localhost

Web server sudah berada dalam computer Anda. Sekarang akan kami jalankan beberapa menu dan beberapa nama folder untuk bisa diakses localhost.

1. Coba ketikkan localhost pada url browser Anda, tekan Enter.
2. Sedangkan untuk mengakses pembuatan MYSQL database, Anda bisa mengarahkan URL ke http://localhost//phpmyadmin/
3. Folder wampserver pada my computer > localdisk(c) > wamp.
4. Untuk meletakkan file php agar bisa diakses oleh browser do localhost adalah pada folder www.

II.7.2 Menciptakan Database di Localhost

Anda bisa menciptakan database yang ingin digunakan melelui phpmyadmin. Cara- caranya adalah sebagai berikut :

1. Buka http://localhost/phpmyadmin/ pada browser Anda, klik menu database.
2. Pada kolom create new database, isikan nama database yang ingin dibuat, klik tombol create.
3. Jika berhasil, akan keluar notifikasi yang memberitahukan bahwa pembuatan database baru berhasil dan nama database baru akan muncul dikiri bersama dengan nama – nama database lainnya.

II.7.3 Syntax PHP

Bentuk dasar scrip PHP dimulai dari kode : <?php dan diakhiri dengan :?>

Bisa juga seperti ini

```
<?
?>
```

Apabila digabungkan dengan script HTML maka terjadi :

```
< HTML>
<head></head>
<body>
<?php
?>
</body>
```

```
</html>
```

Sebagi latihan pertama, kita akan membuat latihan hello word.

```
<html>
<head></head>
<body>
<?php
?>
</body>
</html>
```

Simpan script di atas dalam folder www dan beri nama latihan.php kemudian buka browser Anda ketik http://localhost/latihan.php pada address bar, tekan enter. Pada tampilan browser Anda akan muncul pesan “hello word”. Selamat Anda sudah berhasil membuat script dasar php. (Andrea Adelheid, dkk:2012:9-14)