BAB II

LANDASAN TEORI

II.1. Sistem

II.1.1. Konsep Dasar Sistem

Sistem merupakan kumpulan dari unsur atau elemen-elemen yang saling berkaitan/berinteraksi dan saling mempengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu.

Berikut ini merupakan pengertian sistem dari beberapa ahli:

1. 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.
2. Ludwig Von Bartalanfy “Sistem merupakan seperangkat unsur yang saling terikat dalam suatu antar relasi di antara unsur-unsur tersebut dengan lingkungan.
3. Anatol Raporot “Sistem adalah suatu kumpulan kesatuan dan perangkat hubungan satu sama lain.

L. Ackof “Sistem adalah setiap kesatuan secara konseptual atau fisik yang terdiri dari bagian-bagian dalam keadaan saling tergantung satu sama lainnya (Hendra ; 2011 : 157-158).

Dari uraian di atas, sehingga dapat disimpulkan bahwa sistem adalah sekumpulan elemen yang saling terkait atau terpadu untuk mencapai tujuan tertentu.

II.1.2. Pengertian Sistem

Sistem adalah suatu kesatuan usaha yang terdiri dari bagian-bagian yang berkaitan satu sama lain yang berusaha mencapai suatu tujuan dalam suatu lingkungan kompleks.

Kata sistem berasal dari bahasa Yunani yaitu "Sistema" yang berarti suatu kesatuan yang saling bergantung dan saling bekerja sama untuk mencapai tujuan tertentu. Suatu sistem dapat terdiri dari sistem-sistem bagian lainnya atau sering disebut subsistem. Sistem suatu jaringan kerja dari beberapa prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu tujuan tertentu untuk menerima input lalu memprosesnya dan akhirnya menghasilkan output (Nasrullah dan Sudrajat ; 2015 : 2).

II.1.3. Karakteristik Sistem

Untuk memahami atau mengembangkan suatu sistem, maka perlu membedakan unsur-unsur dari sistem yang membentuknya. Berikut adalah karakteristik sistem yang dapat membedakan suatu sistem dengan sistem lainnya yaitu:

1. Batasan (*boundary*): Penggambaran dari suatu elemen atau unsur mana yang termasuk di dalam sistem dan mana yang di luar sistem.
2. Lingkungan (*environment*): Segala sesuatu di luar sistem, lingkungan yang menyediakan asumsi, kendala, dan input terhadap suatu sistem.

3. Masukan (*input*): Sumber daya (data, bahan baku, peralatan, energi) dari lingkungan yang dikonsumsi dan dimanipulasi oleh suatu sistem.
4. Keluaran (*output*): Sumber daya atau produk (informasi, laporan, dokumen, tampilan *layer computer*, barang jadi) yang disediakan untuk lingkungan sistem oleh kegiatan dalam suatu sistem.
5. Komponen (*component*): Kegiatan-kegiatan atau proses dalam sistem yang mentransformasikan *input* menjadi bentuk setengah jadi (*output*). Komponen ini bisa merupakan subsistem dari sebuah sistem.
6. Penghubung (*interface*): Tempat di mana komponen atau sistem dan lingkungannya bertemu atau berinteraksi.
7. Penyimpanan (*storage*): Area yang dikuasai dan digunakan untuk penyimpanan sementara dan tetap dari informasi, energi, bahan baku, dan sebagainya. Penyimpanan merupakan suatu media penyangga di antara komponen tersebut bekerja dengan berbagai tingkatan yang ada dan memungkinkan komponen yang berbeda dari berbagai data yang sama (Aris, dkk ; 2015 : 2).

II.2. Sistem Pakar

II.2.1. Pengenalan Sistem Pakar

Untuk memahami 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* .Berikut adalah beberapa pengertian sistem pakar (Sutojo, dkk ; 2011 : 159-160).

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.

II.2.1.1 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.2.1.2 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.2.1.3 Aturan-aturan (*Rule*)

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

II.2.1.4 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 :

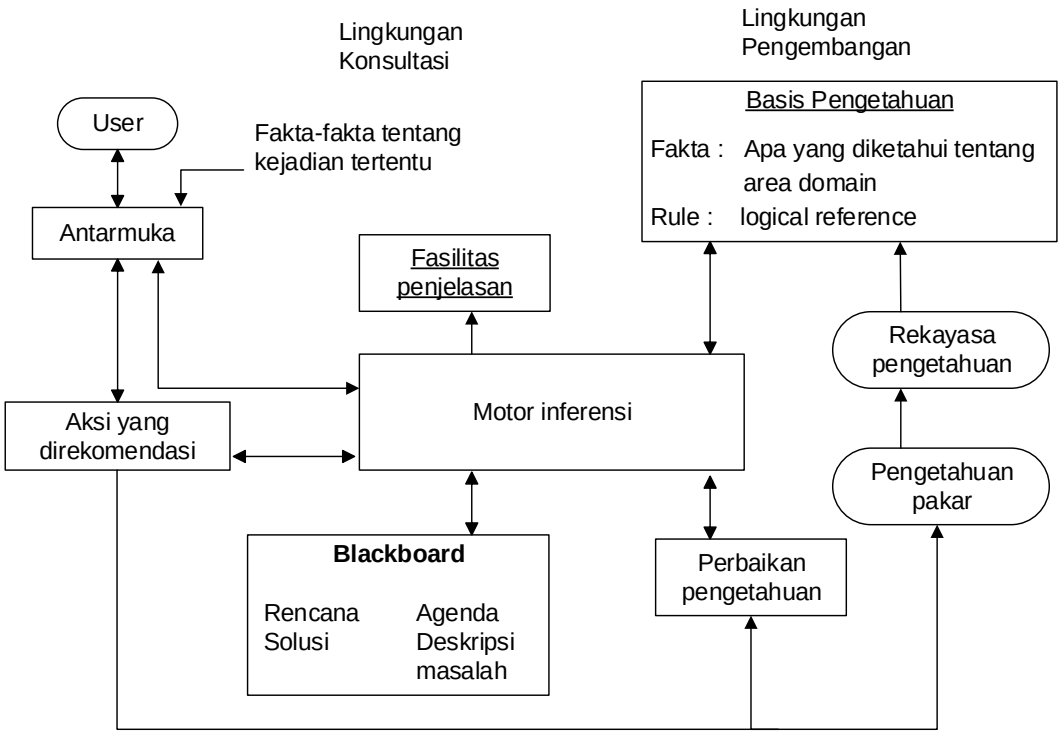
Tabel II.1 Perbandingan Antara Sistem Konvensional dengan Sistem Pakar

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 :Sutojo, dkk ; 2011 : 165)

II.2.2. 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 pengetahuan 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.1 berikut ini :



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

Penjelasan tentang gambar II.2 adalah sebagai berikut (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*)

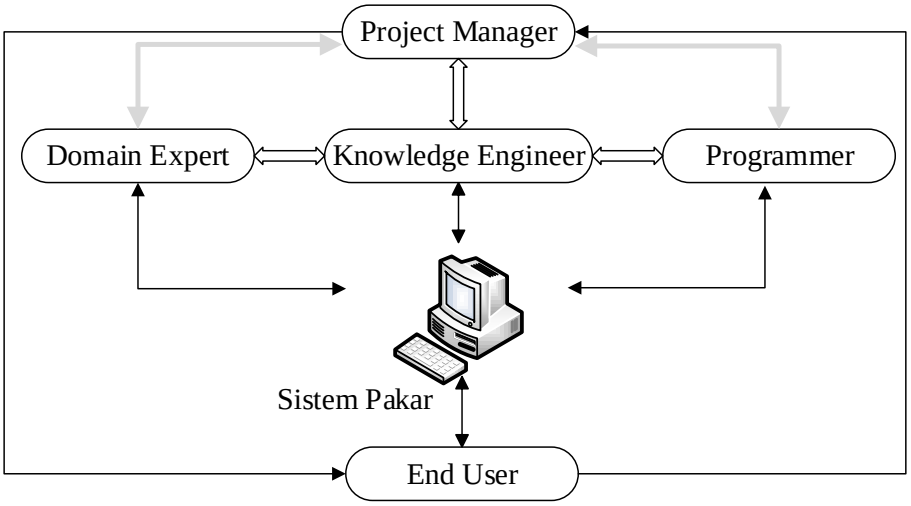
   Kemampuan memperbaiki pengetahuan (*knowledge refining system*) dari seorang pakar diperlukan untuk menganalisis pegetahuan, belajar dari kesalahan masa lalu, kemudian memperbaiki pengetahuan sehingga dapat dipakai di massa mendatang. Kemampuan evaluasi siri seperti itu diperlukan oleh program agar dapat 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.2.3. Tim Pengembang Sistem Pakar

Tim Pengembangan Sistem Pakar



Gambar II.2 Tim Pengembangan Sistem Pakar
(Sumber : Sutojo, dkk ; 2010 : 169)

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 (Sutojo, dkk ; 2011 : 170).

II.2.4. *Rule* Sebagai Teknik Represantasi Pengetahuan

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.

Secara umum,*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.*

IF (E_1 AND E_2 AND E_3 ..... AND E_n) THEN H

IF (E_1 OR E_2 OR E_3 ..... OR E_n) THEN H

Satu evidance bisa juga mempunyai hipotesis lebih dari satu.

IF E THEN (H_1 AND H_2 AND H_3 ..... AND H_n) (Sutojo, dkk ; 2011 : 170-171).

II.3. Metode *Teorema Bayes*

Teori Bayes dikemukakan oleh seorang pendeta Inggris pada tahun 1763 yang bernama Thomas Bayes.Teori Bayes ini kemudian disempurnakan oleh Laplace.Teori Bayes digunakan untuk menghitung probabilitas terjadinya suatu peristiwa berdasarkan pengaruh yang didapat dari hasil observasi. Teori Bayes merupakan kaidah yang memperbaiki atau merevisi suatu probabilitas dengan cara memanfaatkan informasi tambahan. Maksudnya, dari probabilitas awal (*prior probability*) yang belum diperbaiki yang dirumuskan berdasarkan informasi yang

tersedia saat ini, kemudian dibentuklah probabilitas berikutnya (*posterior probability*) (Hartatik dan Yasa ; 2015 :).

Bentuk teorema Bayes untuk *evidence* tunggal E dan hipotesis tunggal H adalah (Sutojo, dkk ; 2011 : 189-191) :

$$P(H|E) = \frac{P(E|H) \times P(H)}{P(E)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. (II.1)$$

dengan:

P(H | E) = probabilitas hipotesis H terjadi jika *evidence* E terjadi

P(E | H) = probailitas munculnya *evidence* E, jika hipotesis H terjadi

P(H) = probabilitas hipotesis H tanpa memandang *evidence* apa pun

P(E) = probabilitas *evidence* E tanpa memandang apa pun

Contoh:

Diketahui, p(demam) = 0,4. P(muntah) = 0,3. P(demam|muntah) = 0,75

a. Berapa nilai dari P(muntah|demam) ?
b. Berapa nilai dari P(muntah|demam) jika p(demam) = 0,1 ?

Jawab:

a. $p(muntah|demam) = \frac{p(demam|muntah) \times p(muntah)}{p(demam)} = \frac{0,75\ x\ 0,3}{0,4} = 0,56$

b. $p(muntah|demam) = \frac{p(demam|muntah) \times p(muntah)}{p(demam)} = \frac{0,75\ x\ 0,3}{0,1} = 2,75$

Tetapi nilai probabilitas harus antara 0 dan 1. Apanya yang salah?

Perhatikan bahwa: p(demam) harus lebih besar atau sama dengan p(demam muntah)

p(demam) ≥ p(demam ∩ muntah)

p(demam ∩ muntah) = p(demam|muntah) x P(muntah)

= 0,75 x 0,3

= 0,225

Jadi, p(demam) ≥ 0,225

Untuk nilai p(demam) = 0,1 tidak memnuhi syarat sehingga menghasilkan perhitungan yang salah.

Bentuk teorema Bayes untuk *evidence* tunggal E dan hipotesis ganda H_1, H_2,...., H_n adalah:

$$P(H_i|E) = \frac{P(E\,|H_i)\ x\ P(H_i)}{\sum_{k=1}^{n} P(E|H_k)\ x\ P(H_k)} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (II.2)$$

dengan:

$P(H_i \mid E)$ = probabilitas hipotesis H_i terjadi jika *evidence* E terjadi

$P(E \mid H_i)$ = probailitas munculnya *evidence* E, jika hipotesis H_i terjadi

$P(H_i)$ = probabilitas hipotesis H_i tanpa memandang *evidence* apa pun

n = jumlah hipotesis yang terjadi

Untuk *evidence* ganda E_1, E_2,....., E_m dan hipotesis ganda H_1, H_2,...., H_n adalah:

$$P(H_i|E_1E_2 \dots . E_m) = \frac{P(E_1E_2 \dots . E_m|H_i)\ x\ P(H_i)}{\sum_{k=1}^{n} P(E_1E_2 \dots . E_m|H_k)\ x\ P(H_k)} \dots\dots\dots\dots (II.3)$$

Untuk mengaplikasikan persamaan (3), maka harus diketahui probabilitas bersyarat dari semua kombinasi yang mungkin dari *evidence-evidence* untuk seluruh hipotesis.Secara praktis, hal ini tidak mungkin bisa dilakukan. Oleh karena itu, persamaan (3) diganti dengan persamaan (4):

$$P(H_i|E_1E_2 \dots E_m) = \frac{P(E_1|H_i)\ x\ P(E_2|H_i)\ x\ \dots\ x\ P(E_m|H_i)\ x\ P(H_i)}{\sum_{k=1}^{n} P(E_1|H_k)\ x\ P(E_2|H_k)\ x \dots x\ P(E_m|H_k)\ x\ P(H_k)} . (II.4)$$

II.4. 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 (Choliviana dan Yulianto ; 2013 : 55).

Sebagai satu kesatuan istilah, Basis Data (*Database*) sendiri dapat didefinisikan dalam sejumlah sudut pandang seperti:

1. Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
2. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redundansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
3. Kumpulan *file*/ tabel/ arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis (Monica, dkk ; 2015 : 67-68).

II.5. *Unified Modeling Language* (UML)

Unifed Modeling Language adalah sebuah bahasa yang diterima dan digunakan oleh developer dan *softwareanalyst* sebagai suatu bahasa yang cocok untuk merepresentasikan grafi darisuatu relasi antar entitas-entitas *software*. 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.

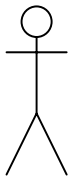
UML juga menggunakan formatur *input* dan *output* yang sudah mempunyai bentuk standar yaitu XML Metadata *Interchange* (XMI), menggunakan aplikasi dan pemodelan data yang universal, merepresentasikan dari tahap analisis ke implementasi lalu ke *deployment*yang terpadu, dan mendeskripsikan keutuhan tentang spesifiasi *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(Winata dan Setiawan ; 2013 : 37).

II.5.1. *Use CaseModel*

Use casemodel adalah teknik pemodelan untuk mendapatkan *functional requirement* dari sebuah sistem, menggambarkan interaksi antara pengguna dan sistem, menjelaskan secara naratif bagaimana sistem akan digunakan, menggunakan skenario untuk menjelaskan setiap aktivitas yang mungkin terjadi. Ada beberapa bagian didalam *use casemodel*.

Gambar II.3 *Use Case* Pada *Use Case Diagram*
(Sumber :Winata dan Setiawan ; 2013 : 38)



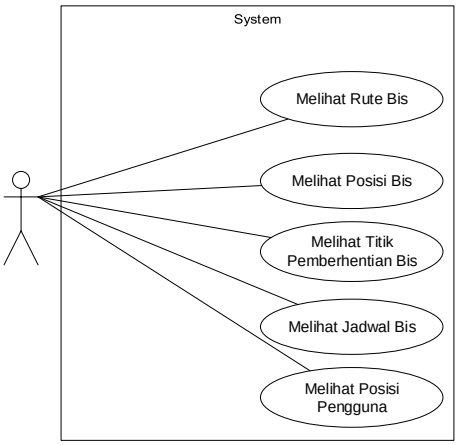
Actor

Gambar II.4. *Actor* Pada *Use Case Diagram*
(Sumber :Winata dan Setiawan ; 2013 : 38)

a. Use Case, untuk mengetahui action atau prosedur apa yang ada didalam sistem.

b. Actor, siapa saja yang terlibat dalam action tersebut.

c. Relationship, bagaimana actions saling berelasi satu sama lain didalam sistem (Winata dan Setiawan ; 2013 : 38).



Gambar II.5. *ContohUse Case Diagram*
(Sumber :Winata dan Setiawan ; 2013 : 38)

II.5.2. *Class Diagram*

Class diagram merupakan diagram paling umum yang dijumpai dalam pemodelan berbasis UML. Didalam *Class diagram* terdapat class dan interface beserta atribut-atribut dan operasinya, relasi yang terjadi antar objek, *constraint* terhadap objek-objek yang saling berhubungan dan *inheritance* untuk organisasi class yang lebih baik. *Class diagram* juga terdapat *static view* dari elemen

pembangun sistem. Pada intinya *Class diagram* mampu membantu proses pembuatan sistem dengan memanfaatkan konsep *forward* ataupun *reverse engineering* (*Rational Software Corporation*, 1997).

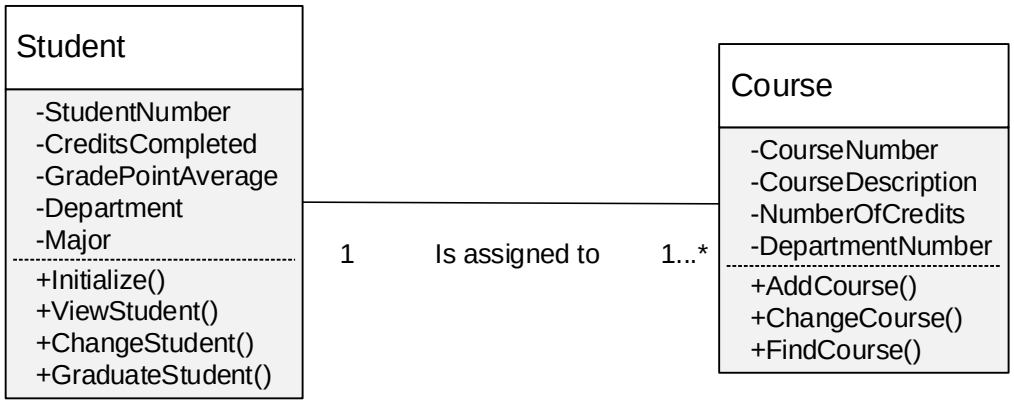
Class diagram mempunyai 2 komponen penting, yaitu:

1. *Structural*, yaitu ciri pembeda objek.
2. *Behavioral,* yaitu tingkah laku atau kegiatan yang mampu dilakukan oleh objek.

Berbagai simbol yang hadir didalam *class diagram*

1. *Class*, yang berfungsi untuk merepresentasikan tipe dari data yang dimilikinya. Class diagram dapat ditampilkan dengan menunjukkan atribut dan operasi yang dimilikinya atau hanya menunjukkan nama class-nya saja. Dapat juga kita tuliskan nama class dengan atributnya saja atau nama class dengan operasinya.
2. *Attribute*, merupakan data yang terdapat didalam class dan instance-nya dengan operator.
3. *Operation*, berfungsi untuk merepresentasikan fungsi-fungsi yang ditampilkan oleh class dan instance-nya dengan operator.
4. *Association,* digunakan untuk menunjukkan bagaimana dua class berhubungan satu sama lainnya. Association ditunjukkan dengan sebuah garis yang terletak diantara dua class. Didalam setiap association terdapat multiplicity, yaitu simbol yang mengindikasikan berapa banyak instance dari class pada ujung association yang satu dengan instance class di ujung association lainnya.

5. *Generalizations*, berfungsi untuk mengelompokkan class ke dalam hirarki inheritance.
6. *Aggregation*, merupakan bentuk khusus dari association yang merepresentasikan hubungan "part-whole". Bagian "whole" dari hubungan ini sering disebut dengan assembly atau aggregate. Class yang satu dapat dikatakan merupakan bagian dari class yang lain yang ikut membentuk class tersebut.
7. *Composition*, merupakan jenis aggregation yang lebih kuat diantara dua class yang memiliki association dimana jika *whole* ditiadakan, maka *part*-nya juga ikut ditiadakan. Berbeda dengan aggregation, *part* akan tetap bisa berdiri sendiri meskipun bagian *whole*-nya ditiadakan.
8. Penggunaan operator (+) dalam class diagram diartikan dengan public, operator (-) diartikan private, dan operator (#) diartikan protected.



Gambar II.6. Contoh *ClassDiagram*
(Sumber :Winata dan Setiawan ; 2013 : 38)

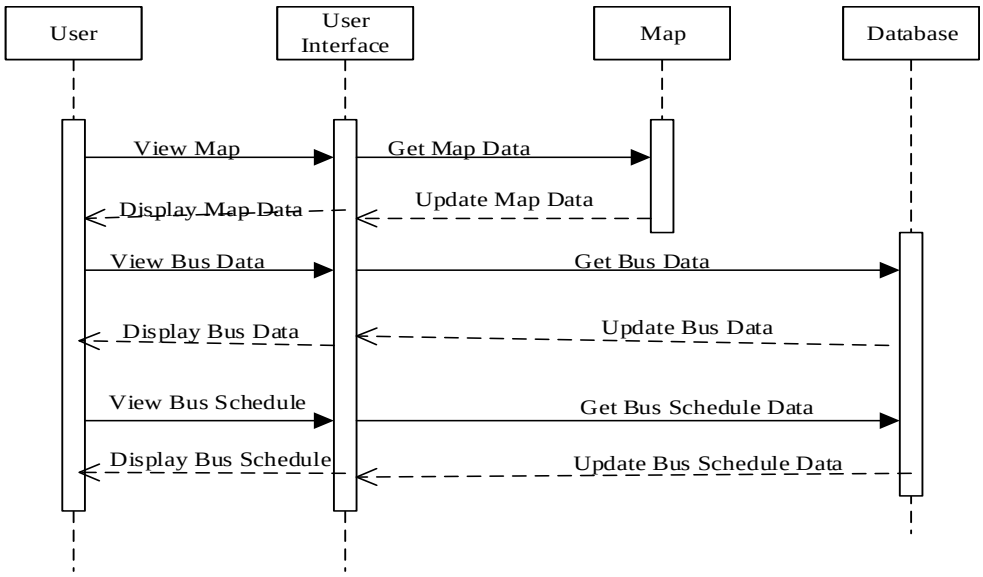
Class diagram menggambarkan karakteristik statis sistem tanpa menjelaskan proses secara mendetil. Sebuah class diagram juga memperlihatkan hubungan antar class (Winata dan Setiawan ; 2013 : 38-39).

II.5.3. Sequence Diagram

Menjelaskan interaksi obyek-obyek yang saling berkolaborasi (berhubungan), mirip dengan *activitydiagram* yaitu menggambarkan alur kejadian sebuah aktivitas tetapi lebih detil dalam menggambarkan aliran data termasuk data atau behaviour yang dikirimkan atau diterima namun kurang mampu menjelaskan detil dari sebuah algoritma.

Dalam *sequence diagram* terdapat beberapa bagian.

1. *Participant*, yaitu objek yang terkait dengan sebuah urutan proses.
2. *Lifeline*, menggambarkan daur hidup sebuah objek.
3. *Activation*, suatu titik waktu dimana sebuah objek mulai berpartisipasi dalam sebuah sequence.
4. *Time*, elemen paling penting dalam sequence diagram yang konteksnya adalah urutan, bukan durasi.
5. *Return*, suatu hasil kembalian sebuah operasi. Operasi mengembalikan hasil tetapi boleh tidak ditulis jika tidak ada perbedaan dengan Getternya.



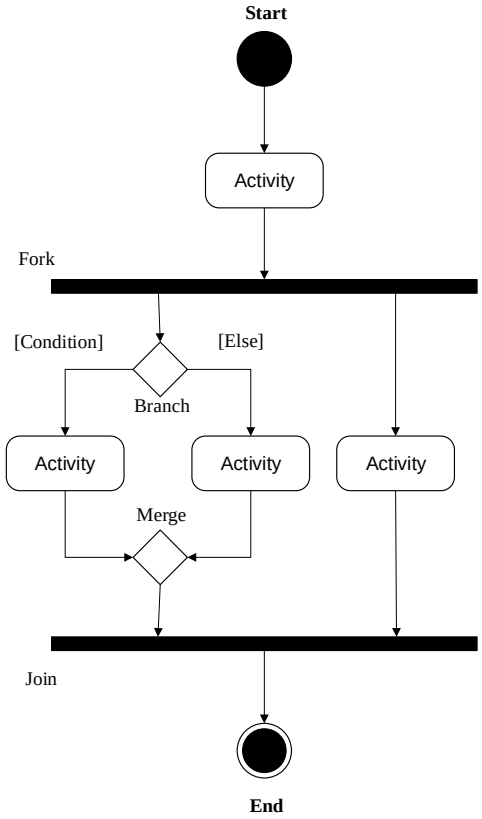
Gambar II.7. Contoh *SquenceDiagram*
(Sumber :Winata dan Setiawan ; 2013 : 38)

Dalam penggunaannya, *sequence diagram* tepat untuk memperlihatkan dengan jelas bagaimana urutan kejadian suatu proses karena didalamnya terlihat interaksi beberapa objek (Winata dan Setiawan ; 2013 : 39).

II.5.4. Activity Diagram

Teknik untuk menjelaskan *business process*, *procedural logic*, dan *work flow*. Bisa dipakai untuk menjelaskan teks use case dalam notasi grafis dengan menggunakan notasi yang mirip *flow chart*, meskipun terdapat sedikit perbedaan notasi.

1. *Nodes*, menandakan initial dan final node, final node boleh lebih dari 1.
2. *Activity*, aktivitas sistem dapat berupa aktivitas fisik juga bagi user.
3. *Flow/edge*, arah sebuah proses.
4. *Fork*, awal sebuah proses paralel.
5. *Join* akhir proses paralel.
6. *Condition*, kondisi yang dituliskan dalam bentuk teks.
7. *Decision*, implementasi if dan then.
8. *Merge*, penyatuan beberapa flow.
9. *Partition*, siapa atau apa yang menjalankan aktivitas (Winata dan Setiawan ; 2013 : 39).



Gambar II.8. Contoh *ActivityDiagram*
(Sumber : Winata dan Setiawan ; 2013 : 39)

II.6. PHP

PHP adalah bahasa *server-side scripting* yang menyatu dengan HTML untuk membuat halaman *web* yang dinamis. Karena PHP merupakan *server-side scripting* maka sintaks dan perintah-perintah PHP akan dieksekusi di *server* kemudian hasilnya dikirimkan ke *web browser* dalam format HTML. Dengan demikian kode program yang ditulis dalam PHP tidak akan terlihat oleh *user* sehingga keamanan halaman *web* lebih terjamin. PHP dirancang untuk membentuk halaman *web* dinamis, yaitu halaman *web* yang dapat membentuk

suatu tampilan berdasarkan permintaan terkini, seperti menampilkan isi basis data ke halaman *web*.

Salah satu keunggulan yang dimiliki oleh PHP adalah kemampuannya untuk melakukan koneksi ke berbagai macam sistem manajemen basis data/*Database Management System* (DBMS), sehingga dapat menciptakan suatu halaman *web* yang dinamis. PHP mempunyai koneksitas yang baik dengan beberapa DBMS antara lain *Oracle*, *Sybase*, *mSql*, *MySQL*, *Microsoft SQL Server*, *Solid*, *PostgreSQL*, *Adabas*, *FilePro*, *Velocis*, *dBase*, *Unix dbm*, dan tak terkecuali semua *database* ber-*interface* ODBC (Arief ; 2011 : 43).

II.7. MySQL

MySQL adalah salah satu jenis *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi *web* yang menggunakan *database* sebagai sumber dan pengolahan datanya. Kepopuleran *MySQL* antara lain karena *MySQL* menggunakan *SQL* sebagai bahasa dasar untuk mengakses *database*-nya sehingga mudah untuk digunakan, kinerja *query* cepat, dan mencukupi untuk kebutuhan *database* perusahaan-perusahaan skala menengah-kecil. *MySQL* juga bersifat *open source* dan *free* pada berbagai *platform* (kecuali pada *windows* yang bersifat *shareware*). *MySQL* didistribusikan dengan lisensi *open source* GPL (*General Public License*) mulai versi 3.23, pada bulan Juni 2000.

MySQL merupakan *database* yang pertama kali didukung oleh bahasa pemrograman *script* untuk internet(PHP dan Perl).*MySQL* dan PHP dianggap sebagai pasangan pengembangan aplikasi *web* yang ideal. *MySQL* lebih sering

digunakan untuk membangun aplikasi berbasis *web*, umumnya pengembangan aplikasinya menggunakan bahasa pemrograman *script* PHP (Arief ; 2011 : 151).